Д.П.Зегжда
А.М.Ивашко

Основы безопасности информационных систем

Рекомендовано УМО в области информационной безопасности в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальностям "Компьютерная безопасность" и "Комплексное обеспечение информационной безопасности автоматизированных систем".

Москва
Горячая линия–Телеком
2000

Зегжда Д.П., Ивашко А.М.

3 47 Основы безопасности информационных систем. – М.: Горячая линия – Телеком, 2000. 452 с., ил.

ISBN 5-93517-018-3

В книге изложены результаты исследований обобщающие отечественный и зарубежный опыт в области разработки нормативных, теоретических и практических положений технологии построения специальных защищенных информационных систем. Описываются основные положения базовых стандартов безопасности информационных технологий, исследуются нарушения информационной безопасности, рассматриваются формальные модели безопасности и принципы их использования в системах обработки информации, анализируются существующие архитектуры защищенных систем.

Для специалистов в области информационной безопасности, студентам вузов, обучающихся по специальностям "Компьютерная безопасность" и "Комплексное обеспечение информационной безопасности автоматизированных систем".

ББК 32.973

ISBN 5-93517-018-3

ОГЛАВЛЕНИЕ

Введение

Волна информатизации, стремительно захлестнувшая нашу страну, привела кроме прочих последствий к необходимости применения современных информационных технологий в тех областях, для которых основным требованием к автоматизированным системам обработки информации является обеспечение безопасности. Поскольку в нашей стране информационные технологии начали применяться совсем недавно, мы вынуждены догонять общемировой уровень и проходить все стадии информатизации одновременно. Например, разработка отечественных стандартов и критериев безопасности идет параллельно с применением популярных импортных средств и технологий, а также с разработкой собственных средств защиты, причем эти процессы далеко не всегда скоординированы.

Информационная безопасность — это такая область, в которой невозможно обойтись без отечественных разработок, тем более что для этого есть определенная база в виде высококвалифицированных специалистов и достижений в области теории защиты информации, особенно в сфере криптографии. С другой стороны, невозможно (да и не нужно) изолировать эту сферу от общего прогресса и отказываться от применения средств и технологий, используемых во всем мире. Другое дело, что в процессе информатизации следует соблюдать баланс между необходимостью поддержания безопасности критических систем с помощью специальных отечественных средств и естественным стремлением (а зачастую и необходимостью) использовать последние мировые достижения в области информационных технологий.

Из вышесказанного следует, что в настоящий момент перед отечественными разработчиками средств защиты стоят две противоречивые задачи:

- реализовать довольно высокие по мировым меркам требования безопасности, предъявляемые отечественными пользователями к системам обработки закрытой информации;
- сохранить в полном объеме совместимость с используемыми повсеместно незащищенными прикладными импортными средствами.

Основное противоречие состоит в том, что требование совместимости с популярными импортными средствами подразумевает их обязательное использование в составе защищенной системы, не смотря на то, что уже стал абсолютно очевидным тот факт, что эти приложения не рассчитаны на работу в составе систем, для которых безопасность имеет существенное значение. Их разработчики просто не ставили перед собой такой задачи в силу изначальной ориентации этих продуктов на массовый рынок

и на автоматизацию мелкого бизнеса. Эта проблема особенно проявляется в нашей стране, где подобные программные средства используются повсеместно, в том числе для решения таких задач, для которых они никогда не предназначались. Поэтому встраивать дополнительные средства защиты в прикладные системы обработки информации — занятие безнадежное, особенно в условиях отсутствия исходных текстов и соответствующей поддержки со стороны производителей этих систем. Таким образом, в сложившейся ситуации решение проблемы построения защищенных систем на базе популярных компонентов является более чем актуальной задачей, причем решение этой задачи требует развития новых подходов к проблеме построения защищенных систем.

Проблемами обеспечения информационной безопасности занимаются многие организации, однако среди них очень немногие проводят научные исследования в области современных технологий обеспечения безопасности. Именно такой подход развивается в Специализированном Центре Защиты Информации Санкт-Петербургского Технического Университета (СЦЗИ СПбГТУ), объединяющем научных сотрудников и преподавателей, последовательно реализующих системный подход к проблемам безопасности, базирующийся на анализе современных достижений в этой области. Результаты этой деятельности отражены в целой серии книг сотрудников Центра, посвященных различным аспектам проблемы информационной безопасности — от компьютерных вирусов до информационных атак в Internet. В СЦЗИ СПбГТУ уже в течение семи лет ведутся исследования и практические разработки в области информационной безопасности. Целью этих исследований является создание технологии построения защищенных систем обработки информации на базе разрабатываемой в СЦЗИ СПбГТУ защищенной ОС класса UNIX. Сложность поставленной цели и необходимость получения реальных практических результатов потребовали от специалистов СПбГТУ разработки нового подхода к определению самого понятия "защищенная система" и тех задач, которые необходимо решить в процессе ее построения.

Данная книга знакомит читателя с основными результатами исследований в области разработки нормативных, теоретических и практических и положений технологии построения специальных защищенных информационных систем. Представленная в этой книге постановка задачи построения защищенных информационных систем своей попыткой комплексного решения проблемы является новой для отечественной литературы, посвященной данному вопросу, и обобщает мировой опыт в области обеспечения безопасности информационных технологий и создания защищенных систем обработки информации.

Данная книга должна ответить на давно назревший, но до сих пор остающийся без конструктивного ответа вопрос, волнующий всех специалистов, занимающихся обеспечением информационной безопасности: как в сложившихся обстоятельствах обеспечить безопасность в соответствии с современными требованиями?

Ответ на данный вопрос не является очевидным в силу действия следующих факторов:

- на отечественном рынке практически отсутствуют специализированные средства работы с конфиденциальной информацией такого класса как операционные системы, СУБД, информационные системы, системы обработки документов и т. д.;
- все широко распространенные в нашей стране базовые средства — операционные системы (MS Windows-9X и NT, Novell Netware, распространенные версии Unix) и протоколы передачи информации (TCP/IP, IPX/SPX, SMB) с точки зрения безопасности не выдерживают никакой критики;
- предлагаемые на отечественном рынке отдельные средства защиты не в состоянии решить проблему в целом (а другого решения в принципе быть не может).

Действительно, ни один из популярных продуктов не рассчитан на решение задач, связанных с обеспечением безопасности, о чем свидетельствует отсутствие сертификации этих продуктов в западных странах в работоспособной конфигурации. Единственные системы общего назначения, получившие международные сертификаты безопасности, — это специальные защищенные версии ОС Unix типа **CX/SX** *Harris Computer Systems Corporation*, **HP-UX BLS** *Hewlett Packard Corporation*, **Trusted IRIX/B** *Silicon Graphics Inc.* и т. д.. Однако, возможности применения подобных систем в нашей стране ограничены дорогой эксклюзивной аппаратной платформой, да и сама возможность поставки систем такого класса также вызывает сомнение. Поэтому отечественный потребитель, нуждающийся в средствах работы с закрытой информацией, в полной мере является заложником ситуации, сложившейся на рынке программных средств, и вынужден искать частные или половинчатые решения своих проблем.

Данная книга посвящена поискам выхода из этой ситуации и пытается ответить на целый ряд вопросов, связанных с обеспечением информационной безопасности. Существующие публикации на эту тему в основном ограничиваются перечислением угроз и аналитическими обзорами. Данная книга — первое издание, содержащее позитивную информацию о том, как строить защищенные системы, которую можно применить на практике.

Авторы ставили перед собой задачу осветить основные теоретические аспекты создания подобных систем и дать ответы на следующие вопросы:

- Что представляет собой защищенная система?
- Какими должны быть требования к безопасности?
- Каким образом можно учесть негативный опыт нарушения безопасности существующих систем?
- Какие принципы могут быть положены в основу технологии создания защищенных систем обработки информации?

Авторы пытаются найти ответы на эти вопросы, опираясь на собственный практический опыт исследований в данной области и результаты разработок проводимых в Центре защиты информации.

Основными положениями авторского подхода к решению проблемы информационной безопасности являются:

- комплексный подход к созданию защищенной системы и автоматизации процессов обработки конфиденциальной информации;
- анализ и устранение причин нарушений безопасности компьютерных систем.

Разумеется, данная публикация не претендует на окончательное разрешение всех проблем информационной безопасности, но, как надеются ее авторы, прояснит ряд вопросов и позволит решить некоторые практические задачи. Основное достоинство данной книги авторы видят в том, что в ней собран под одной обложкой весь материал, послуживший авторам базой при разработке технологии создания защищенных систем.

Структура данной книги

Свою основную задачу авторы видели в том, чтобы представить читателю по возможности более полное описание основных проблем построения защищенных систем. Поэтому мы подробно остановились на требованиях и критериях стандартов информационной безопасности, анализе механизмов реализации угроз и причин их успешного осуществления, анализе опыта создания защищенных систем и архитектурах построения современных защищенных операционных систем. Теоретическая база информационной безопасности представлена объемным разделом, определившим название книги, который содержит описание моделей политик безопасности и криптографических методов защиты (идентификации, аутентификации, шифрования и контроля целостности). В качестве практического примера построения защищенной системы представлен подробный анализ одного из самых современных проектов в области защиты ОС — Trusted Mach.

Этот материал распределился по главам следующим образом:

Первая глава знакомит читателя с проблемой построения защищенных систем, предлагает определения базовых понятий, содержит постановку задачи и указывает методы ее решения.

Вторая глава содержит подробный обзор основных положений существующих стандартов безопасности информационных технологий, регламентирующих требования к защищенным системам обработки информации и критерии их оценки, и их сравнительный анализ. Сопоставление и сравнение требований и критериев стандартов позволяет понять основы информационной безопасности, основные принципы технологии создания защищенных систем, а также и место и роль всех участников этого процесса.

Третья глава представляет собой исследование нарушений информационной безопасности, их систематизацию и выявление причин, обуславливающих возможность осуществления нарушений.

В четвертой главы рассматриваются формальные модели безопасности и принципы их использования в системах обработки информации, а также основы криптографии и методы идентификации, аутентификации и контроля целостности.

Пятая глава содержит анализ существующих архитектур защищенных систем и достаточно подробный анализ архитектуры наиболее передовой защищенной операционной системы Trusted Mach.

Информация об авторах:

Дмитрий Петрович Зегжда — ведущий научный сотрудник Центра защиты информации Санкт-Петербургского государственного технического университета, кандидат технических наук, доцент кафедры Информационной безопасности компьютерных систем СПбГТУ, специалист в области информационной безопасности, защиты операционных систем и формальных моделей безопасности (dmitry@ssl.stu.neva.ru).

Андрей Михайлович Ивашко — сотрудник Федерального Агентства Правительственной Связи и Информации при президенте Российской Федерации (Ivashko_AM@gov.ru).

Глава, посвященная исследованиям нарушений безопасности (гл. 3), написана при участии *Всеволода Михайловича Кузмича.*

Обзор криптографических методов защиты информации (гл.4) выполнен *Еленой Борисовной Маховенко.*

Анализ архитектур сертифицированных систем (гл.5) проведен *Семеном Станиславовичем Кортом.*

Научная редакция:

Петр Дмитриевич Зегжда - доктор технических наук, профессор, директор Центра защиты информации Санкт-Петербургского государственного технического университета, заведующий кафедрой Информационной безопасности компьютерных систем СПбГТУ, главный редактор журнала "Проблемы информационной безопасности. Компьютерные системы" специалист в области теории информационной безопасности компьютерных систем и интеллектуальных технологий анализа и проектирования распределенных систем (zeg@ssl.stu.neva.ru).

Владимир Владимирович Платонов - сотрудник Центра защиты информации, кандидат технических наук, профессор кафедры Информационной безопасности компьютерных систем СПбГТУ, эксперт в области информационной безопасности(plato@ssl.stu.neva.ru).

Авторы будут благодарны за отзывы и пожелания, а также любые предложения о сотрудничестве в области безопасности информационных технологий, которые можно отправлять по адресу:

195251, Санкт-Петербург, Политехническая 29,
Специализированный центр защиты информации СПбГТУ.
Тел./факс: (812) 552-64-89

Internet:

WWW - www.ssl.stu.neva.ru
E-mail – dmitrv@ssl.stu.neva.ru

Наука никогда не решает вопроса,
не поставив при этом десятка новых.
Бернард Шоу

Глава 1. Что такое защищенная информационная система?

Жизнь современного общества немыслима без повсеместного применения информационных технологий. Компьютеры обслуживают банковскую систему, контролируют работу атомных реакторов, распределяют энергию, следят за расписанием поездов, управляют самолетами и космическими кораблями. Сегодня компьютерные системы и телекоммуникации определяют надежность систем обороны и безопасности страны, реализуют современные информационные технологии, обеспечивая хранение информации, ее обработку, доставку и представление потребителям.

Однако именно высочайшая степень автоматизации, к которой стремится современное общество, ставит его в зависимость от уровня безопасности используемых информационных технологий, обеспечивающих благополучие и даже жизнь множества людей. Действительно, массовое применение компьютерных систем, позволившее решить задачу автоматизации процессов обработки постоянно нарастающих объемов информации, сделало эти процессы чрезвычайно уязвимыми по отношению к агрессивным воздействиям и поставило перед потребителями информационных технологий новую проблему, — проблему информационной безопасности.

Примеры, подтверждающие актуальность этой проблемы, можно во множестве найти на страницах многочисленных изданий и еще больше на "страницах" Internet. Не будем отвлекать внимание читателя на изложение скандальных фактов нарушения безопасности, оценку принесенного ущерба и описание хитроумных уловок компьютерных нарушителей. Приведем только некоторые факты, свидетельствующие об актуальности проблемы безопасности информационных технологий. Каждые двадцать секунд в Соединенных Штатах происходит преступление с использованием программных средств. В более 80% компьютерных преступлений, расследуемых ФБР, "взломщики" проникают в атакуемую систему через глобальную сеть Internet. Последние оценки исчисляют потери от хищения или повреждения компьютерных данных в 100 млн. долларов за год, но точная статистика не поддается учету. Во многих случаях организации не

знают о том, что вторжение имело место, — информация ворується незаметно, и похитители гениально заметают свои следы.

Сложившаяся в сфере безопасности информационных технологий ситуация еще раз подтверждает давно известную неприятную особенность технического прогресса порождать в ходе решения одних проблем новые, иногда даже более сложные. Положение с безопасностью в сфере обработки информации настолько критическое, что невольно закрадывается сомнение, не является ли верным старый тезис о том, что в каждом из значительных, эпохальных достижений технического прогресса таится некий деструктивный фактор, ограничивающий сферу его применения, и даже на каком-то этапе обращающий это достижение не во благо, а во вред человечеству. Во всяком случае, сегодня одним из основных аргументов консервативных оппонентов популяризации информационных технологий служит не нашедшая на сегодняшний день своего решения проблема безопасности этих технологий, или, если точнее, их небезопасности.

Таким образом, опыт эксплуатации существующих компьютерных систем обработки информации показывает, что проблема обеспечения безопасности еще далека от своего решения, а предлагаемые производителями различных систем средства защиты сильно различаются как по решаемым задачам и используемым методам, так и по достигнутым результатам.

Все вышесказанное определяет актуальность проблемы построения защищенных систем обработки информации, решение которой следует начать с анализа причин сложившейся ситуации.

1.1. Кризис информационной безопасности — истоки и последствия

Проблема безопасности информационных технологий возникла на пересечении двух активно развивающихся и, наверное, самых передовых в плане использования технических достижений направлений — безопасности технологий и информатизации. Сама проблема безопасности, конечно, не является новой, ведь обеспечение собственной безопасности — задача первостепенной важности для любой системы независимо от ее сложности и назначения будь то социальное образование, биологический организм или система обработки информации. Однако, в условиях, когда защищаемый объект представляет собой информационную систему, или когда средства нападения имеют форму информационных воздействий, необходимо разрабатывать и применять совершенно новые технологии и методы защиты.

Не останавливаясь на социальных, правовых и экономических аспектах данной проблемы, систематизируем научные и технические предпосылки сложившейся кризисной ситуации с обеспечением безопасности информационных технологий.

1. Современные компьютеры за последние годы приобрели гигантскую вычислительную мощь, но одновременно с этим (что на первый взгляд парадоксально!) стали гораздо проще в эксплуатации. Это означает, что пользоваться ими стало намного легче, поэтому все большее количество новых, как правило, неквалифицированных людей получает доступ к компьютерам, что приводит к снижению средней квалификации пользователей. Эта тенденция существенно облегчает задачу нарушителям, т. к. в результате "персонализации" средств вычислительной техники большинство пользователей имеют личные компьютеры и осуществляют их администрирование самостоятельно. Большинство из них не в состоянии постоянно поддерживать безопасность своих систем на должном уровне, т. к. это требует соответствующих знаний, навыков, а также времени и средств. Повсеместное распространение сетевых технологий объединило отдельные машины в локальные сети, совместно использующие общие ресурсы, а применение технологий клиент-сервер и кластеризации преобразовало такие сети в распределенные вычислительные среды. Безопасность сети определяется защищенностью всех входящих в нее компьютеров и сетевого оборудования, и злоумышленнику достаточно нарушить работу только одного компонента, чтобы скомпрометировать всю сеть. Современные телекоммуникационные технологии объединили локальные компьютерные сети в глобальную информационную среду. Это привело к появлению такого уникального явления, как Internet. Именно развитие Internet вызвало всплеск интереса к проблеме информационной безопасности и поставило вопрос об обязательном наличии средств защиты у сетей и систем, подключенных к Internet, независимо от характера обрабатываемой в них информации. Дело в том, что Internet (кроме всего прочего) обеспечивает широкие возможности злоумышленникам для осуществления нарушений безопасности в глобальном масштабе. Если компьютер, который является объектом атаки, подключен к Internet, то для атакующего не имеет большого значения, где он находится — в соседней комнате или на другом континенте.

2. Прогресс в области аппаратных средств сочетается с еще более бурным развитием программного обеспечения. Как показывает практика, большинство распространенных современных программных средств, в первую очередь операционных систем, не отвечает даже минимальным требованиям безопасности, хотя их разработчики в последнее время и осуществляют определенные усилия в этом направлении. В первую оче-

редь это выражается в наличии изъянов в работе средств защиты и наличии огромного числа различных "недокументированных" возможностей. После их обнаружения многие изъяны ликвидируются с помощью обновления версий или дополнительных средств, однако то постоянство, с которым обнаруживаются все новые и новые изъяны, не может не вызывать опасений. В настоящий момент можно констатировать, что большинство систем предоставляют злоумышленникам широкие возможности для осуществления нарушений.

3. Развитие гибких и мобильных технологий обработки информации привело к тому, что практически исчезает грань между обрабатываемыми данными и исполняемыми программами за счет появления и широкого распространения виртуальных машин и интерпретаторов. Теперь любое развитое приложение от текстового процессора (MS Word) до броузера Internet (Java) не просто обрабатывает данные, а интерпретирует интегрированные в них инструкции специальных языков программирования, т. е. по сути дела является отдельной машиной с привычной фоннеймановской архитектурой, для которой можно создавать средства нападения, вирусы и т.д. Это увеличивает возможности злоумышленников по созданию средств внедрения в чужие системы и затрудняет задачу защиты подобных систем, т. к. наличие таких "вложенных" систем требует и реализации защиты для каждого уровня.

4. Несоответствие бурного развития средств обработки информации и медленной проработки теории информационной безопасности привело к появлению существенного разрыва между теоретическими моделями безопасности, оперирующими абстрактными понятиями типа объект, субъект и т. д., и реальными категориями современных информационных технологий. Кроме того, многие средства защиты, например, средства борьбы с компьютерными вирусами и системы защиты корпоративных сетей firewall на данный момент вообще не имеют системной научной базы. Такое положение является следствием отсутствия общей теории защиты информации, комплексных моделей безопасности обработки информации, описывающих механизмы действий злоумышленников в реальных системах, а также отсутствием средств, позволяющих эффективно промоделировать адекватность тех или иных решений в области безопасности. Следствием этого является то, что практически все системы защиты основаны на "латании дыр", обнаруженных в процессе эксплуатации, что предопределяет их отставание от динамично развивающихся угроз. В качестве примера можно привести распространенную практику закрытия "внезапно" обнаружившихся пробелов в системах защиты с помощью различных "заплаток" (т. н. patch, service pack и т.д.), характерную для большинства коммерческих систем. На практике отсутствие системной и

научной базы информационной безопасности проявляется уже в том, что нет даже общепринятой терминологии, которая бы адекватно воспринималась всеми специалистами в области безопасности. Поэтому зачастую теория и практика действуют в разных плоскостях.

5. Необходимость создания глобального информационного пространства и обеспечение безопасности протекающих в нем процессов потребовала разработки международных стандартов, следование которым может обеспечить необходимый уровень гарантии обеспечения защиты. В современных условиях чрезвычайно важным является стандартизация не только требований безопасности, но и обоснование их применения, а также методов подтверждения адекватности реализованных средств защиты и корректности самой реализации. Существующие национальные стандарты пытались решить эту проблему, однако эти документы не могут претендовать на то, чтобы заложить фундамент безопасных информационных технологий будущего. В России официальным стандартом в этой области являются документы, Гостехкомиссии России, представляющие собой подражание зарубежным стандартам десятилетней давности. В условиях обвальной информатизации и компьютеризации важнейших сфер отечественной экономики и государственного аппарата нашей стране крайне необходимы новые решения в этой области.

Вследствие совокупного действия всех перечисленных факторов перед разработчиками современных информационных систем, предназначенных для обработки важной информации, стоят следующие задачи, требующие немедленного и эффективного решения:

1. Обеспечение безопасности новых типов информационных ресурсов. Поскольку компьютерные системы теперь напрямую интегрированы в информационные структуры современного общества, средства защиты должны учитывать современные формы представления информации (гипертекст, мультимедиа и т. д.). Это означает, что системы защиты должны обеспечивать безопасность на уровне информационных ресурсов, а не отдельных документов, файлов или сообщений.

2. Организация доверенного взаимодействия сторон (взаимной идентификации/аутентификации) в информационном пространстве. Развитие локальных сетей и Internet диктует необходимость осуществления эффективной защиты при удаленном доступе к информации, а также взаимодействии пользователей через общедоступные сети. Причем требуется решить эту задачу в глобальном масштабе, несмотря на то, что участвующие стороны могут находиться в разных частях планеты, функционировать на различных аппаратных платформах и в разных операционных системах.

3. Защита от автоматических средств нападения. Опыт эксплуатации существующих систем показал, что сегодня от защиты требуются совершенно новые функции, а именно, механизмы, обеспечивающие безопасность системы в условиях возможного взаимодействия с ней, или появления внутри нее программ, осуществляющих деструктивные действия — компьютерных вирусов, автоматизированных средств взлома, агрессивных агентов. На первый взгляд кажется, что эта проблема решается средствами разграничения доступа, однако это не совсем так, что подтверждается известными случаями распространения компьютерных вирусов в "защищенных" системах.

4. Интеграция защиты информации в процесс автоматизации ее обработки в качестве обязательного элемента . Для того, чтобы быть востребованными современным рынком информационных систем средства безопасности не должны вступать в конфликт с существующими приложениями и сложившимися технологиями обработки информации, а, напротив, должны стать неотъемлемой частью этих средств и технологий.

Если эти задачи не будут решены, то дальнейшее распространение информационных технологий в сфере критических систем, обрабатывающих важную информацию, очень скоро окажется под угрозой. Наша страна благодаря тому, что начинает информатизацию практически "с нуля", является полигоном для применения последних достижений информационных технологий, поэтому для нас решение задачи создания защищенных информационных систем актуально как нигде в мире.

Взгляд авторов на перечисленные проблемы информационной безопасности и методы решения указанных задач и определили тематику данной книги. Свою главную задачу авторы видят в том, чтобы помочь читателю разобраться, что представляют собой современные защищенные системы и средства защиты, и предложить те принципы, которые, по мнению авторов, должны быть положены в основу технологии создания защищенных систем. Начнем с того, что определим предмет наших исследований и конечную цель разработок.

1.2. Понятие "защищенная система" — определение и свойства

Любая цель может быть достигнута, и любые задачи могут быть успешно решены только в том случае, если есть четкое определение цели и задач, решаемых на пути к ней. Без этого невозможно ни определить правильный путь достижения цели, ни выбрать оптимальные методы решения возникающих задач, и, самое главное, невозможно доказать, что цель достигнута, и задачи успешно решены. Следовательно, прежде чем при-

ступить к созданию защищенной системы обработки информации, надо предварительно дать четкий и недвусмысленный ответ на вынесенный в название главы вопрос: что представляет собой защищенная система? Конструктивное определение этого понятия должно позволить определить, что входит в состав защищенной системы, какими свойствами она должна обладать, какие задачи необходимо решить, чтобы ее построить, и какие методы наиболее эффективны для их решения.

Многие полагают, что защищенная система — это система обработки информации, в состав которой включен тот или иной набор средств защиты. С нашей точки зрения это неправильный подход, т. к. наличие средств защиты является лишь необходимым условием и не может рассматриваться в качестве критерия защищенности системы от реальных угроз, поскольку безопасность не является абсолютной характеристикой и может рассматриваться только относительно некоторой среды, в которой действуют определенные угрозы. Поэтому мы считаем, что:

защищенная система обработки информации *для определенных условий эксплуатации обеспечивает безопасность (конфиденциальность и целостность) обрабатываемой информации и поддерживает свою работоспособность в условиях воздействия на нее заданного множества угроз.*

Каким образом осуществляется защита — вопрос не принципиальный. Это определение содержит достаточное условие безопасности, позволяющее назвать систему защищенной. Кроме того, из него следует, что защищенность является качественной характеристикой системы, ее нельзя измерить в каких-либо единицах, более того, нельзя даже с однозначным результатом сравнивать уровень защиты двух систем — одна будет лучше обеспечивать безопасность обрабатываемой информации в одном случае, другая — в другом.

Безусловно, созданием защищенных систем занимаются уже достаточно давно, и на сегодняшний день у каждой группы специалистов, занимающихся проблемами безопасности информационных технологий, имеется свой взгляд на задачи средств защиты и методы достижения безопасности, и, следовательно, свое представление о том, что должна представлять собой защищенная система. Разумеется, любая точка зрения имеет право на существование и развитие, но нам кажется, что предложенное определение может объединить усилия всех специалистов в направлении конструктивной работы над созданием защищенных систем.

Взяв за основу предложенное определение, рассмотрим свойства, которыми должна обладать защищенная система.

Автоматизация процесса обработки конфиденциальной информации

Как и все автоматизированные (компьютерные) системы обработки информации, защищенные системы решают задачу автоматизации некоторого процесса обработки информации. Под процессом обработки информации мы понимаем действия, связанные с ее хранением, преобразованием и передачей. Следовательно, защищенная система — это, прежде всего, такое же средство автоматизации прикладного информационного процесса, также как и обычные, "незащищенные" информационные системы типа баз данных, систем передачи информации, поисково-информационных систем, систем обработки документов, мультимедиа и т. д.. Однако, в случае защищенной системы, даже если рассматривать ее только с этой точки зрения, существует еще один специфический аспект автоматизации — защищенные системы используются для автоматизации тех процессов, для которых безопасность играет существенную роль. Это процессы обработки т. н. конфиденциальной информации, — информации, разглашение, искажение или уничтожение которой наносит ущерб кому-либо (например, информация, представляющая собой государственную или коммерческую тайну). Причем, если подвергающийся автоматизации процесс предусматривал какие-либо меры защиты информации, то эквивалентные средства защиты должны быть реализованы и в автоматизированной информационной системе. Например, при автоматизации процесса обработки документов, содержащих конфиденциальную информацию, необходимо реализовать те правила обработки информации, которые регламентируются соответствующими стандартами. При этом потребители вправе требовать от разработчиков, чтобы в процессе автоматизации безопасность системы, по крайней мере, не ухудшилась.

Таким образом, можно сформулировать первое свойство защищенной системы обработки информации:

Защищенная система обработки информации *должна автоматизировать процесс обработки конфиденциальной информации, включая все аспекты этого процесса, связанные с обеспечением безопасности обрабатываемой информации.*

Противодействие угрозам безопасности

Кроме традиционных свойств, которыми обладают автоматизированные системы — надежности, эффективности, удобства использования и т. д., защищенная система обработки информации должна вдобавок обладать еще одним — свойством безопасности, которое является для нее самым главным.

Свойство обеспечения собственной безопасности присуще всем высокоорганизованным системам (как естественного, так и искусственного происхождения) и означает способность функционирования в условиях действия факторов, стремящихся нарушить работу системы. Эти факторы называют угрозами безопасности. Традиционно выделяют три вида угроз безопасности — угрозы конфиденциальности информации, угрозы целостности информации и угрозы отказа в обслуживании. При этом надо учитывать, что угрозы могут носить как случайный (объективный), так и целенаправленный (субъективный) характер. В случае компьютерных систем к первым относятся, например, сбои аппаратуры и изменения параметров внешней среды, т. е. факторы действующие "вслепую" и нецеленаправленно. Примером субъективных угроз являются атаки хакеров, "троянских коней", программ-вирусов и т. д. Механизм действий этих угроз коренным образом отличается, т. к. за ними стоят чья-то воля и интеллект, преследующие определенную цель. Противодействие объективным угрозам в большей степени относится к обеспечению надежности функционирования системы, в то время как противодействие субъективным угрозам представляет собой основную задачу безопасности. При этом необходимо учитывать как "традиционные" угрозы, направленные на автоматизируемый процесс обработки информации, так и те "новые" угрозы, которые появились в результате его автоматизации. Это означает, что система должна сохранить все возможности по противостоянию "унаследованным" угрозам безопасности и включить в свой состав дополнительные средства защиты, защищающие ее от новых угроз, появившихся после ее включения в пространство современных информационных технологий. Большинство проблем информационной безопасности, с которыми столкнулись пользователи автоматизированных систем, возникли именно после осуществления процесса автоматизации и являются его непосредственным следствием. Это объясняется тем, что автоматизация обработки информации неразрывно связана с отстранением человека от непосредственной работы с носителями информации, а также с невероятной гибкостью мощных компьютерных технологий, универсальность которых позволяет применять их как в целях защиты, так и для нападения. Действительно, в те времена, когда доступ к информации осуществлялся путем непосредственного контакта человека с бумажным документом, то, в общем случае, все угрозы безопасности сводились к двум основным: кража конфиденциальных документов и разглашение их содержания лицом, допущенным к работе с ними. В такой ситуации каждый человек, работающий с конфиденциальной информацией, был либо лоялен по отношению к принятым правилам работы, либо нет, т. к. непосредственно осуществлял обработку информации и действовал всегда либо в интересах одной стороны, либо в

интересах другой. Процесс автоматизации привел к тому, что пользователи компьютерных систем не могут обрабатывать информацию непосредственно, они используют программные средства, которые должны осуществлять за них те или иные действия. При такой технологии пользователь отстраняется от процесса обработки информации и неявным образом передает свои полномочия программе, которая обладает некоторой свободой в своих действиях и совсем не обязательно делает то, что хочет или предполагает пользователь. Таким образом, возник принципиально новый класс угроз — т. н. "троянские кони" (это программы, которые обладают деструктивными функциями, маскируя их под полезные), присущий исключительно автоматизированным средствам обработки информации. Кроме того, чем дальше простирается процесс автоматизации, тем больше появляется возможностей для осуществления угроз, т. к. пользователь все больше отстраняется от процесса обработки информации и растет количество, сложность и универсальность используемых средств, которые могут быть задействованы для реализации угроз. Наконец, чем сложнее система, тем больше вероятность возникновения в ней ошибок и сбоев. Все это свидетельствует о том, что проблема безопасности во многом является непосредственным следствием автоматизации процессов обработки информации.

Таким образом, чтобы быть защищенной, компьютерная система должна успешно противостоять многочисленным и разнообразным угрозам безопасности, действующим в пространстве современных информационных технологий, и главным образом тем, которые носят целенаправленный характер. Следовательно, ***защищенная система*** *— это система, которая успешно и эффективно противостоит угрозам безопасности.*

Соответствие стандартам безопасности информационных технологий

Наконец, поскольку проблема безопасности компьютерных систем изучается и прорабатывается уже достаточно давно, защищенная система должна соответствовать сложившимся требованиям и представлениям. Кроме того, необходимо обеспечить возможность сопоставления параметров и характеристик защищенных систем для того, чтобы их можно было сравнивать между собой. Для решения этой задачи и были разработаны стандарты информационной безопасности в виде соответствующих критериев и требований, регламентирующие подходы к информационной безопасности на государственном или межгосударственном уровне. Эти документы определяют понятие “защищенная система” посредством стандартизации требований и критериев безопасности, образующих шкалу оценки степени защищенности автоматизированных систем обработки информа-

ции. Наличие таких общепринятых стандартов позволяет согласовать подходы различных участников процесса создания защищенных систем (требования потребителей, технологии и методы производителей, критерии независимой экспертизы) и хотя бы качественно оценить обеспечиваемый уровень безопасности. При этом следует понимать, что все требования и критерии безопасности носят исключительно необходимый, но никак не достаточный характер, т. к. никакой сертификат соответствия стандартам не сможет защитить систему от реальных угроз.

Тем не менее, для того, чтобы существовала возможность оценить уровень безопасности, обеспечиваемый в защищенной системе, и сравнить ее возможности с другими, *защищенная система должна соответствовать требованиям и критериям стандартов информационной безопасности.*

Свойства защищенной системы обработки информации

Таким образом, под защищенной системой обработки информации предлагается понимать систему, которая обладает следующими тремя свойствами:

- осуществляет автоматизацию некоторого процесса обработки конфиденциальной информации, включая все аспекты этого процесса связанные с обеспечением безопасности обрабатываемой информации;
- успешно противостоит угрозам безопасности, действующим в определенной среде;
- соответствует требованиям и критериям стандартов информационной безопасности.

Предложенный подход к определению понятия "защищенная система" отличается от существующих в первую очередь тем, что рассматривает проблему обеспечения безопасности компьютерных систем как лежащую на стыке двух направлений: автоматизации обработки информации и общей безопасности. Это дает возможность объединить задачи автоматизации обработки конфиденциальной информации и разработки средств защиты в одну проблему создания защищенных информационных систем и процессе ее решения применять методы и технологии, разработанные как в той, так и в другой области.

На основании предложенного определения и перечисленных свойств защищенной системы сформулируем три задачи, которые необходимо и достаточно решить, для того чтобы создать защищенную систему обработки информации, а именно:

- в ходе автоматизации процесса обработки конфиденциальной информации реализовать все аспекты этого процесса, связанные с обеспечением безопасности обрабатываемой информации;

- обеспечить противодействие угрозам безопасности, действующим в среде эксплуатации защищенной системы;
- реализовать необходимые требования соответствующих стандартов информационной безопасности.

После того, как определен перечень задач, которые необходимо решить для достижения поставленной цели, можно перейти к определению методов, позволяющих решить эти задачи.

1.3. Методы создания безопасных систем обработки информации

В рамках данного раздела ограничимся тем, что рассмотрим общие методы обеспечения безопасности автоматизированных систем с точки зрения автоматизации процессов обработки конфиденциальной информации и способы противодействия угрозам безопасности, поскольку реализации требований стандартов информационной безопасности будут посвящены следующие главы данной книги.

Автоматизация процесса обработки конфиденциальной информации

Как уже указывалось, обеспечение информационной безопасности можно рассматривать как одну из составляющих процесса внедрения информационных технологий в государственных и военных учреждениях, на производстве и в бизнесе, т. е. в тех структурах, для которых проблема обеспечения конфиденциальности и целостности информации существовала всегда. Это означает, что задача обеспечения безопасности компьютерной системы никогда не решается с "нуля", т. к. само понятие информационной безопасности имеет смысл только для тех систем, в которых эта проблема существовала и до их автоматизации. До применения компьютерных технологий в любой организации, для которой безопасность информации имела определенное значение, был установлен определенный порядок работы с информацией (например, система работы с секретными документами Министерства Обороны), регламентирующий информационные потоки внутри организации и обмен информацией с внешним миром. Этот порядок включал, во-первых, схему информационных потоков внутри организации, и, во-вторых, правила управления этими потоками.

Таким образом, если основная цель внедрения информационных технологий — автоматизировать процесс обработки информации, то частная задача автоматизации процессов обработки конфиденциальной информации — обеспечить адекватную реализацию в компьютерной системе

схемы информационных потоков и правил управления ими, существовавших до применения компьютерных средств обработки информации.

Как правило применение средств автоматизации позволяет увеличить объем обрабатываемой информации и даже полностью изменить ее вид, что может повлечь за собой изменение смысла самого информационного процесса, однако это не означает, что исходные меры безопасности больше не нужны, они просто должны быть адаптированы к новой ситуации. В любом случае автоматизированная система должна, во-первых, реализовывать только те потоки информации, которые существовали до ее применения, и не создавать новых и, во-вторых, обеспечить возможности управления потоками информации в соответствии с набором правил, унаследованным от автоматизируемого процесса.

Решение этой задачи осуществляется последовательным осуществлением следующих действий:

1. Определение формального механизма, адекватно выражающего заданную схему информационных потоков и правила управления ими.

2. Построение модели безопасности, отражающей заданный порядок обработки информации, и формальное доказательство ее безопасности.

3. Реализация системы обработки информации в соответствии с предложенной моделью.

4. Доказательство адекватности допустимых в автоматизированной системе потоков информации и правил управления доступом исходной схеме информационных потоков и правил управления ими.

В ходе осуществления данной последовательности действий разработчики защищенных систем сталкиваются со следующими проблемами:

1. Схема информационных потоков и правила управления ими могут быть неполны или использовать трудно формализуемые концепции. Это затрудняет создание моделей безопасности или делает невозможным доказательство их безопасности.

2. Как правило, схема информационных потоков носит статический характер и определяет разделение информационных потоков только в штатном режиме работы. Такие аспекты как добавление и удаление компонентов в систему обычно остаются за рамками этой схемы и правил, т. к. эти операции не поддаются формализации. Соответственно, они остаются за рамками моделей безопасности, и порядок их осуществления определяется разработчиками конкретных систем в частном порядке, что приводит к потере доказательности и неадекватной реализации модели безопасности.

3. В результате автоматизации в компьютерной системе появляются новые объекты (например, служебные файлы, ресурсы и т.д.), которые не

соответствуют никаким сущностям реального мира. Соответственно, они не присутствуют в схеме информационных потоков и не учитываются правилами управления этими потоками. Однако, совершенно очевидно, что контроль доступа к подобным объектам имеет ключевое значение для безопасности всей системы в целом.

4. В ходе реализации модели безопасности могут появиться неконтролируемые потоки информации, поскольку в компьютерной системе пользователи не могут манипулировать информацией непосредственно и используют программные средства, которые могут независимо от их воли создавать нежелательные неконтролируемые информационные потоки (например, атака с помощью "троянского коня").

Только успешное разрешение этих проблем, лежащих на стыке теории информационной безопасности и автоматизации процессов обработки информации, позволит разработчикам защищенных систем обработки информации обеспечить адекватную автоматизацию существующего в реальных системам порядка обработки информации и корректно применять на практике формальные модели безопасности.

Противодействие угрозам безопасности путем устранения предпосылок их успешного осуществления

Как уже говорилось, противодействие угрозам безопасности является основной задачей защиты, и успешность ее решения определяет степень безопасности системы. Современные автоматизированные системы обработки информации предоставляют массу средств и механизмов для осуществления подобных угроз, о чем свидетельствует статистика правонарушений с использованием компьютерных средств. Однако, средства осуществления угроз безопасности выбираются не случайным образом — никто не будет пытаться вмешаться в работу системы с помощью средств, которые не смогут обеспечить нарушителю доступ к информации и преодолеть существующие системы защиты. Отсюда следует, что любая успешная реализация угрозы (будем называть ее атакой) непременно использует определенные особенности построения и функционирования системы обработки информации или недостатки средств защиты.

Эти особенности систем обработки информации исследуются уже достаточно давно и получили название "изъянов защиты" или "уязвимостей". Глава 3 данной книги опирается на результаты этих исследований и представляет собой их обобщение и развитие с точки зрения объяснения источников возникновения изъянов защиты и методов их устранения. Здесь отметим только то обстоятельство, что в большинстве случаев наличие уязвимостей определяется особенностями архитектуры и реализации средств защиты.

Поскольку с точки зрения авторов причины успеха атак на системы обработки информации предопределены свойствами самих систем (недостатками их архитектуры, ошибками реализации и неправильной эксплуатацией), то знание природы этих свойств позволяет оценить реальную способность систем противостоять угрозам безопасности. Более того, понимание недостатков существующих средств защиты, которые привели к успешным нарушениям безопасности, позволяет построить систему, лишенную этих недостатков, и соответственно защищенную по отношению к целым классам угроз.

Из полученной авторами таксономии причин успешной реализации угроз безопасности (глава 3) следует, что все механизмы осуществления атак базируются на определенных свойствах автоматизированных систем, которые как бы провоцируют появление средств нападения. Таким образом, противостояние угроз и средств защиты напоминает систему с обратной связью — новые виды атак приводят к появлению новых средств защиты, а недостатки в средствах защиты приводят к появлению новых средств нападения и т.д. Разорвать этот порочный круг бесконечного противостояния можно двумя способами: создать эффективные и безупречно надежные средства защиты от каждого типа атак, или устранить указанные недостатки автоматизированных систем, служащие источником успешной реализации угроз безопасности. Рассмотрим недостатки и преимущества того и другого метода.

Создание средств защиты от каждого вида угроз. К преимуществам данного метода следует отнести то, что средства защиты не зависят напрямую от назначения системы и не требуют модификации по мере ее развития. Недостатки такого подхода очевидны: для создания эффективной системы безопасности необходимо проанализировать все типы угроз и выработать эффективные механизмы противодействия для каждого типа. Кроме того, практика показывает, что данный путь является трудно осуществимым вследствие следующих факторов:

- множество угроз постоянно расширяется и имеет тенденцию экспоненциального роста. Это означает, что все время будут появляться новые угрозы, требующие новых мер защиты, т. к. старые против них бессильны. Появление новых средств защиты, будет приводить к появлению новых классов угроз и т.д. и т.п. .

- множество угроз растет не только количественно, но и качественно, т. к. для того, чтобы угроза состоялась, она должна принципиально отличаться от тех, на которые рассчитаны системы защиты. Это означает, что невозможно создать исчерпывающую классификацию угроз безопасности и предсказывать появление новых типов угроз.

Устранение причин, обуславливающих успешную реализацию угроз. Преимущества этого метода очевидны: он не зависит от развития угроз, т. к. ликвидирует причину, а не следствие, поэтому он более эффективен, чем создание средств защиты от каждого вида угроз. В качестве недостатков данного метода можно указать необходимость модернизации некоторых аспектов процесса проектирования и создания защищенных систем обработки информации путем применения технологий проектирования и разработки, направленных на устранение указанных причин успешной реализации угроз.

Приведенные рассуждения позволяют сделать следующий вывод: для обеспечения эффективной защиты автоматизированной системы обработки информации против современных угроз безопасности необходимо проанализировать те причины, по которым произошли известные нарушения, и устранить их.

Совокупность предложенных методов построения защищенных систем — автоматизации процессов обработки конфиденциальной информации и устранения причин появления изъянов защиты образует разрабатываемую авторами оригинальную технологию создания защищенных систем обработки информации.

1.4. Заключение к главе 1

В этой главе мы показали актуальность проблемы информационной безопасности, исследовали предпосылки сложившейся кризисной ситуации и сформулировали задачи, которые стоят перед разработчиками защищенных систем. Кроме того, мы обратили внимание читателя на те трудности, которые необходимо преодолеть для достижения поставленной цели, основной из которых, по нашему мнению, является создавшийся разрыв между теорией безопасности и практикой защиты. Вследствие того, что теория информационной безопасности часто замыкается на исследовании абстрактных моделей безопасности и методов защиты, сложилась порочная практика доработки систем до "защищенных" путем добавления в них отдельных функций защиты, эффективность которых часто не может быть обоснована. Результатом этого разрыва является отсутствие теоретически обоснованной технологии разработки защищенных систем. Распространенный в настоящее традиционный подход к построению защищенных автоматизированных систем обработки информации, заключающийся в создании средств защиты, препятствующих осуществлению определенных типов угроз, и шаблонной реализации требований стандартов, является малоэффективным и не может обеспечить надежной защиты, т. к. исходно предполагает постоянное отставание защиты от нападения.

Для того чтобы найти выход из сложившейся ситуации мы предложили определение понятия «защищенная система», которое может служить в качестве достаточного критерия безопасности системы, и перечислили свойства, которыми защищенная система должна обладать. Это позволило нам определить свой подход к технологии проектирования и разработки защищенных систем, основанный на объединении теоретических достижений и практических методов обеспечения безопасности и автоматизации процессов обработки информации. С нашей точки зрения ключом к решению проблемы безопасности является устранение причин появления изъянов защиты, обуславливающих успех существующих на данный момент угроз и провоцирующих появление новых.

Надо отметить, что предлагаемый взгляд на технологию создания защищенных автоматизированных систем обработки информации ни в чем не противоречит существующим требованиям и критериям стандартов информационной безопасности и технологиям построения информационных систем. В тоже время благодаря его нацеленности на устранение факторов, определяющих причины успешной реализации угроз, его применение позволяет достичь качественно нового уровня защиты и значительно продвинуться в решении актуальной сегодня проблемы обеспечения безопасности информационных технологий.

Данная книга посвящена теоретическим основам построения защищенных систем, и в следующих главах мы рассмотрим основные положения нормативных документов, теоретическую базу информационной безопасности, результаты исследований нарушений безопасности и примеры архитектуры защищенных систем, которые лежат в основе предложенного подхода к технологии построения защищенных систем.

Если начинают с неправильного,
то мало надежды на правильное завершение.
Конфуций

Глава 2. Обзор и сравнительный анализ стандартов информационной безопасности

(по материалам отечественных и зарубежных стандартов)

2.1. Введение

Любой проект может получить успешное завершение только в том случае, если его участники хорошо представляют, что они хотят получить в результате. Только четкое понимание цели позволит выбрать оптимальный путь ее достижения. Следовательно, прежде чем приступить к созданию защищенной системы обработки информации, надо сперва получить четкий и недвусмысленный ответ на вопрос: что представляет собой защищенная система?

Цель данной главы — выяснить, что скрывается за понятием защищенная вычислительная система, или защищенная система обработки информации, т. к. без конструктивного определения этого понятия невозможно рассматривать основные принципы функционирования подобных систем и технологию их создания.

Как уже говорилось, безопасность является качественной характеристикой системы, ее нельзя измерить в каких-либо единицах, более того, нельзя даже с однозначным результатом сравнивать безопасность двух систем — одна будет обладать лучшей защитой в одном случае, другая — в другом. Кроме того, у каждой группы специалистов, занимающихся проблемами безопасности информационных технологий, имеется свой взгляд на безопасность и средства ее достижения, а, следовательно, и свое представление о том, что должна представлять собой защищенная система. Разумеется, любая точка зрения имеет право на существование и развитие, но для того, чтобы объединить усилия всех специалистов в направлении конструктивной работы над созданием защищенных систем все таки необходимо определить, что является целью исследований, что мы хотим получить в результате и чего в состоянии достичь ?

Для ответа на эти вопросы и согласования всех точек зрения на проблему создания защищенных систем разработаны и продолжают разрабатываться стандарты информационной безопасности. Это документы,

регламентирующие основные понятия и концепции информационной безопасности на государственном или межгосударственном уровне, определяющие понятие "защищенная система" посредством стандартизации требований и критериев безопасности, образующих шкалу оценки степени защищенности ВС.

В соответствии с этими документами ответ на поставленный вопрос в общем виде звучит так: защищенная система обработки информации — это система отвечающая тому или иному стандарту информационной безопасности. Конечно, это условная характеристика, она зависит от критериев, по которым оценивается безопасность, но у нее есть одно неоспоримое преимущество — объективность. Именно это позволяет осуществлять сопоставление степени защищенности различных систем относительно установленного стандарта.

Итак, рассмотрим, что представляет собой защищенная система с точки зрения существующих стандартов безопасности.

2.2. Основные понятия и определения

Для того, чтобы приступить к дальнейшему изложению, необходимо установить некоторые термины и определения, составляющие базовые концепции безопасности компьютерных систем. Несмотря на то, что практически каждый из рассматриваемых далее стандартов представляет оригинальный подход к определению понятия безопасной системы обработки информации, существует ряд понятий и концепций используемых всеми стандартами. Приведем те из них, которые являются наиболее важными для понимания и трактуются во все стандартах(за редким исключением) практически одинаково.

Политика безопасности (Security Policy). Совокупность норм и правил, обеспечивающих эффективную защиту системы обработки информации от заданного множества угроз безопасности.

Модель безопасности (Security Model). Формальное представление политики безопасности.

Дискреционное, или произвольное, управление доступом (Discretionary Access Control). Управление доступом, осуществляемое на основании заданного администратором множества разрешенных отношений доступа (например в виде троек *<объект, субъект, тип доступа>*).

Мандатное, или нормативное, управление доступом (Mandatory Access Control). Управление доступом, основанное на совокупности правил предоставления доступа, определенных на множестве атрибутов безопасности субъектов и объектов, например, в зависимости от грифа секретности информации и уровня допуска пользователя.

Ядро безопасности — Trusted Computing Base (TCB). Совокупность аппаратных, программных и специальных компонент ВС, реализующих функции защиты и обеспечения безопасности. Далее в тексте будет использоваться аббревиатура ТСВ ввиду ее распространенности и неточности предложенного перевода.

Идентификация (Identification). Процесс распознавания сущностей путем присвоения им уникальных меток (идентификаторов).

Аутентификация (Authentication). Проверка подлинности идентификаторов сущностей с помощью различных (преимущественно криптографических) методов.

Адекватность (Assurance). Показатель реально обеспечиваемого уровня безопасности, отражающий степень эффективности и надежности реализованных средств защиты и их соответствия поставленным задачам (в большинстве случаев это задача реализации политики безопасности).

Квалификационный анализ, квалификация уровня безопасности (Evaluation). Анализ ВС с целью определения уровня ее защищенности и соответствия требованиям безопасности на основе критериев стандарта безопасности. *Квалификация уровня безопасности* является конечным этапом технологического цикла создания защищенных систем, непосредственно предшествует процедуре сертификации, и завершается присвоением ВС того или иного класса или уровня безопасности. Авторы долго колебались, прежде чем остановили свой выбор на этом термине. В русском языке слова, в точности соответствующего английскому *evaluation* нет, а вводить очередной "англоизм"("эвалюация") не хотелось. В терминологии Гостехкомиссии РФ есть близкий по значению термин "сертификационные испытания", но не совсем точен — *квалификационный анализ* не ограничивается простыми испытаниями, а включает в себя подробное исследование архитектуры ВС и технологии ее разработки, также анализ слабых сторон ее защиты. Соответственно специалистов, занимающихся *квалификационным анализом* будем называть *экспертами по квалификации.*

Таксономия (Taxonomy). Наука о систематизации и классификации сложноорганизованных объектов и явлений, имеющих иерархическое строение (от греческого *taxis* — строй, порядок и *nomos* — закон). В отличие от классификации, устанавливающий связи и отношения между объектами (иерархия строится снизу-вверх), таксономия основана на декомпозиции явлений и поэтапном уточнении свойств объектов (иерархия строится сверху-вниз).

Прямое взаимодействие (Trusted Path). Принцип организации информационного взаимодействия (как правило, между пользователем и

системой), гарантирующий, что передаваемая информация не подвергается перехвату или искажению.

Отметим, что часть приведенных определений не совпадает с официальной трактовкой руководящими документами Гостехкомиссии России[1]. С точки зрения авторов предложенные определения совпадают по смыслу с общепринятыми и распространенными английскими терминами.

2.3. Угрозы безопасности компьютерных систем

Под угрозой безопасности вычислительной системе понимаются воздействия на систему, которые прямо или косвенно могут нанести ущерб ее безопасности. Разработчики требований безопасности и средств защиты выделяют три вида угроз: угрозы нарушения конфиденциальности обрабатываемой информации, угрозы нарушения целостности обрабатываемой информации и угрозы нарушения работоспособности системы (отказа в обслуживании).

Угрозы конфиденциальности направлены на разглашение секретной информации, т. е. информация становится известной лицу, которое не должно иметь к ней доступ. Иногда для обозначения этого явления используется понятие "несанкционированный доступ" (НСД), особенно популярное у отечественных специалистов. Традиционно противостоянию угрозам этого типа уделялось максимальное внимание и, фактически, подавляющее большинство исследований и разработок в области компьютерной безопасности было сосредоточено именно в этой области, т. к. она непосредственно относится к задаче охраны государственных и военных секретов.

Угрозы целостности представляют собой любое искажение или изменение неавторизованным на это действие лицом хранящейся в вычислительной системе или передаваемой информации. Целостность информации может быть нарушена как злоумышленником, так и результате объективных воздействий со стороны среды эксплуатации системы. Наиболее актуальна эта угроза для систем передачи информации — компьютерных сетей и систем телекоммуникаций,

Угрозы нарушения работоспособности (отказ в обслуживании) направлены на создание ситуаций, когда в результате преднамеренных действий ресурсы вычислительной системы становятся недоступными, или снижается ее работоспособность.

Цель защиты систем обработки информации — противодействие угрозам безопасности. Следовательно, безопасная или защищенная систе-

ма — это система, обладающая средствами защиты которые успешно и эффективно противостоят угрозам безопасности.

2.4. Роль стандартов информационной безопасности

Главная задача стандартов информационной безопасности — создать основу для взаимодействия между производителями, потребителями и экспертами по квалификации продуктов информационных технологий. Каждая из этих групп имеет свои интересы и свои взгляды на проблему информационной безопасности.

Потребители, во-первых, заинтересованы в методике, позволяющей обоснованно выбрать продукт, отвечающий их нуждам и решающий их проблемы, для чего им необходима шкала оценки безопасности и, во-вторых, нуждаются в инструменте, с помощью которого они могли бы формулировать свои требования производителям. При этом потребителей (что вполне естественно) интересуют исключительно характеристики и свойства конечного продукта, а не методы и средства их достижения. С этой точки зрения идеальная шкала оценки безопасности должна была бы выглядеть примерно следующим образом:

Уровень 1. Система для обработки информации с грифом не выше "*для служебного пользования*";

Уровень 2. Система для обработки информации с грифом не выше "*секретно*";

и т. д.

Соответственно и требования потребители хотели бы формулировать примерно в такой форме: "Мы хотим, чтобы у нас все было защищено для обработки совершенно секретной информации". Этот, хотя и не очень конструктивный, подход сам по себе не так страшен, гораздо хуже другое — многие потребители не понимают, что за все надо платить (и не только деньгами) и что требования безопасности обязательно противоречат функциональным требованиям (удобству работы, быстродействию и т. д.), накладывают ограничения на совместимость и, как правило, вынуждают отказаться от очень широко распространенных и поэтому незащищенных прикладных программных средств.

Производители также нуждаются в стандартах, как средстве сравнения возможностей своих продуктов, и в применении процедуры сертификации как механизме объективной оценки их свойств, а также в стандартизации определенного набора требований безопасности, который мог бы ограничить фантазию заказчика конкретного продукта и заставить его выбирать требования из этого набора. С точки зрения производителя требования должны быть максимально конкретными и регламентировать не-

обходимость применения тех или иных средств, механизмов, алгоритмов и т. д. Кроме того, требования не должны противоречить существующим парадигмам обработки информации, архитектуре вычислительных систем и технологиям создания информационных продуктов. Этот подход также не может быть признан в качестве доминирующего, т. к. не учитывает нужд пользователей (а ведь это главная задача разработчика) и пытается подогнать требования защиты под существующие системы и технологии, а это далеко не всегда возможно осуществить без ущерба для безопасности.

Эксперты по квалификации и специалисты по сертификации рассматривают стандарты как инструмент, позволяющий им оценить уровень безопасности, обеспечиваемый продуктами информационных технологий, и предоставить потребителям возможность сделать обоснованный выбор. Производители в результате квалификации уровня безопасности получают от объективную оценку возможностей своего продукта. Эксперты по квалификации находятся в двойственном положении: с одной стороны они как и производители заинтересованы в четких и простых критериях, над которыми не надо ломать голову как их применить к конкретному продукту (проще всего в виде анкеты с ответами типа да/нет), а с другой стороны, они должны дать обоснованный ответ пользователям — удовлетворяет продукт их нужды, или нет, ведь к конечном счете именно они принимают на себя ответственность за безопасность продукта, получившего квалификацию уровня безопасности и прошедшего сертификацию.

Таким образом, перед стандартами информационной безопасности стоит непростая задача — примирить эти три точки зрения и создать эффективный механизм взаимодействия всех сторон. Причем "ущемление" потребностей хотя бы одной из них приведет к невозможности взаимопонимания и взаимодействия и, следовательно, не позволит решить общую задачу — создание защищенной системы обработки информации. Необходимость в подобных стандартах была осознана уже достаточно давно (по меркам развития информационных технологий), и в этом направлении достигнут существенный прогресс, закрепленный в новом поколении документов разработки 90-годов. Наиболее значимыми стандартами информационной безопасности являются (в хронологическом порядке): "Критерии безопасности компьютерных систем министерства обороны США"[7], Руководящие документы Гостехкомиссии России[1,2,3,4,5] (только для нашей страны), "Европейские критерии безопасности информационных технологий"[16], "Федеральные критерии безопасности информационных технологий США"[17], "Канадские критерии безопасности компьютерных систем"[18] и "Единые критерии безопасности информационных технологий"[19].

Представляется целесообразным проанализировать эти документы, сопоставить их структуру, содержащиеся в них требования и критерии, а также оценить эффективность их практического применения. Следующий далее обзор стандартов строится по следующей схеме: цель разработки, основные положения, таксономия и ранжирование требований и критериев.

2.5. Критерии безопасности компьютерных систем министерства обороны США ("Оранжевая книга")

2.5.1. Цель разработки

"Критерии безопасности компьютерных систем" (Trusted Computer System Evaluation Criteria)[16], получившие неформальное, но прочно закрепившееся название "Оранжевая книга", были разработаны Министерством обороны США в 1983 году с целью определения требований безопасности, предъявляемых к аппаратному, программному и специальному обеспечению компьютерных систем и выработки соответствующей методологии и технологии анализа степени поддержки политики безопасности в компьютерных системах военного назначения.

В данном документе были впервые нормативно определены такие понятия, как "политика безопасности", ТСВ и т. д. Согласно "Оранжевой книге" безопасная компьютерная система — это система, поддерживающая управление доступом к обрабатываемой в ней информации, таким образом, что только соответствующим образом авторизованные пользователи или процессы, действующие от их имени, получают возможность читать, писать, создавать и удалять информацию. Предложенные в этом документе концепции защиты и набор функциональных требований послужили основой для формирования всех появившихся впоследствии стандартов безопасности.

2.5.2. Таксономия требований и критериев "Оранжевой книги"

В "Оранжевой книге" предложены три категории требований безопасности — политика безопасности, аудит и корректность, в рамках которых сформулированы шесть базовых требований безопасности. Первые четыре требования направлены непосредственно на обеспечение безопасности информации, а два последних — на качество самих средств защиты. Рассмотрим эти требования подробнее.

2.5.2.1 Политика безопасности

Требование 1. *Политика безопасности.* Система должна поддерживать точно определенную политику безопасности. Возможность осуществления субъектами доступа к объектам должна определяться на основании их идентификации и набора правил управления доступом. Там, где необходимо, должна использоваться политика нормативного управления доступом, позволяющая эффективно реализовать разграничение доступа к категорированной информации (информации, отмеченной грифом секретности — типа "секретно", "сов. секретно" и т.д.).

Требование 2. *Метки.* С объектами должны быть ассоциированы метки безопасности, используемые в качестве атрибутов контроля доступа. Для реализации нормативного управления доступом система должна обеспечивать возможность присваивать каждому объекту метку или набор атрибутов, определяющих степень конфиденциальности (гриф секретности) объекта и/или режимы доступа к этому объекту.

2.5.2.2 Аудит

Требование 3. *Идентификация и аутентификация.* Все субъекты должен иметь уникальные идентификаторы. Контроль доступа должен осуществляться на основании результатов идентификации субъекта и объекта доступа, подтверждения подлинности их идентификаторов (аутентификации) и правил разграничения доступа. Данные, используемые для идентификации и аутентификации, должны быть защищены от несанкционированного доступа, модификации и уничтожения и должны быть ассоциированы со всеми активными компонентами компьютерной системы, функционирование которых критично с точки зрения безопасности.

Требование 4. *Регистрация и учет.* Для определения степени ответственности пользователей за действия в системе, все происходящие в ней события, имеющие значение с точки зрения безопасности, должны отслеживаться и регистрироваться в защищенном протоколе. Система регистрации должна осуществлять анализ общего потока событий и выделять из него только те события, которые оказывают влияние на безопасность для сокращения объема протокола и повышения эффективность его анализа. Протокол событий должен быть надежно защищен от несанкционированного доступа, модификации и уничтожения.

2.5.2.3 Корректность

Требование 5. *Контроль корректности функционирования средств защиты.* Средства защиты должны содержать независимые аппаратные и/или программные компоненты, обеспечивающие работоспособность

функций защиты. Это означает, что все средства защиты, обеспечивающие политику безопасности, управление атрибутами и метками безопасности, идентификацию и аутентификацию, регистрацию и учет, должны находиться под контролем средств, проверяющих корректность их функционирования. Основной принцип контроля корректности состоит в том, что средства контроля должны быть полностью независимы от средств защиты.

Требование 6. *Непрерывность защиты.* Все средства защиты (в т.ч. и реализующие данное требование) должны быть защищены от несанкционированного вмешательства и/или отключения, причем эта защита должна быть постоянной и непрерывной в любом режиме функционирования системы защиты и компьютерной системы в целом. Данное требование распространяется на весь жизненный цикл компьютерной системы. Кроме того, его выполнение является одним из ключевых аспектов формального доказательства безопасности системы.

Рассматриваемые далее критерии безопасности компьютерных систем представляют собой конкретизацию данных обобщенных требований.

2.5.2.4. Таксономия критериев безопасности

Приведенные выше базовые требования к безопасности служат основой для критериев, образующих единую шкалу оценки безопасности компьютерных систем, определяющую семь классов безопасности.

Ввиду широкой доступности самой "Оранжевой книги" и ее многочисленных обзоров и интерпретаций приведем только схему, отражающую таксономию предложенных в ней функциональных требований безопасности (рис. 2.1).

2.5.3. Классы безопасности компьютерных систем

Поскольку "Оранжевая книга" достаточно подробно освещалась в отечественных исследованиях[6], ограничимся только кратким обзором классов безопасности.

"Оранжевая книга" предусматривает четыре группы критериев, которые соответствуют различной степени защищенности: от минимальной (группа **D**) до формально доказанной (группа **A**). Каждая группа включает один или несколько классов. Группы **D** и **A** содержат по одному классу (классы **D** и **A** соответственно), группа **C** — классы **C1**, **C2**, а группа **B** — **B1**, **B2**, **B3**, характеризующиеся различными наборами требований безопасности. Уровень безопасности возрастает при движении от группы **D** к группе **A**, а внутри группы — с возрастанием номера класса.

Группа **D**. Минимальная защита.

Рис. 2.1 Таксономия требований "Оранжевой книги"

Класс **D**. Минимальная защита. К этому классу относятся все системы, которые не удовлетворяют требованиям других классов.

Группа **C**. Дискреционная защита.

Группа **C** характеризуется наличием произвольного управления доступом и регистрацией действий субъектов.

Класс **C1**. Дискреционная защита. Системы этого класса удовлетворяют требованиям обеспечения разделения пользователей и информации и включают средства контроля и управления доступом, позволяющие задавать ограничения для индивидуальных пользователей, что дает им возможность защищать свою приватную информацию от других пользователей. Класс **C1** рассчитан на многопользовательские системы, в которых осуществляется совместная обработка данных одного уровня секретности.

Класс **C2**. Управление доступом. Системы этого класса осуществляют более избирательное управление доступом, чем системы класса **C1**, с помощью применения средств индивидуального контроля за действиями пользователей, регистрацией, учетом событий и выделением ресурсов.

Группа **B**. Мандатная защита.

Основные требования этой группы — нормативное управление доступом с использованием меток безопасности, поддержка модели и политики безопасности, а также наличие спецификаций на функции ТСВ. Для систем этой группы монитор взаимодействий должен контролировать все события в системе.

Класс **B1**. Защита с применением меток безопасности. Системы класса **B1** должны соответствовать всем требованиям, предъявляемым к системам класса **C2**, и, кроме того, должны поддерживать определенную неформально модель безопасности, маркировку данных и нормативное управление доступом. При экспорте из системы информация должна подвергаться маркировке. Обнаруженные в процессе тестирования недостатки, должны быть устранены.

Класс **B2**. Структурированная защита. Для соответствия классу **B2** ТСВ системы должна поддерживать формально определенную и четко документированную модель безопасности, предусматривающую произвольное и нормативное управление доступом, которое распространяется по сравнению с системами класса **B1** на все субъекты. Кроме того, должен осуществляться контроль скрытых каналов утечки информации. В структуре ТСВ должны быть выделены элементы, критичные с точки зрения безопасности. Интерфейс ТСВ должен быть четко определен, а ее архитектура и реализация должны быть выполнены с учетом возможности проведения тестовых испытаний. По сравнению с классом **B1** должны быть усилены средства аутентификации. Управление безопасностью осу-

ществляется администраторами системы. Должны быть предусмотрены средства управления конфигурацией.

Класс **B3**. Домены безопасности. Для соответствия этому классу ТСВ системы должна поддерживать монитор взаимодействий, который контролирует все типы доступа субъектов к объектам, который невозможно обойти. Кроме того, ТСВ должна быть структурирована с целью исключения из нее подсистем, не отвечающих за реализацию функций защиты, и быть достаточно компактной для эффективного тестирования и анализа. В ходе разработки и реализации ТСВ должны применяться методы и средства, направленные на минимизацию ее сложности. Средства аудита должны включать механизмы оповещения администратора при возникновении событий, имеющих значение для безопасности системы. Требуется наличие средств восстановления работоспособности системы.

Группа **A**. Верифицированная защита.

Данная группа характеризуется применением формальных методов верификации корректности работы механизмов управления доступом (произвольного и нормативного). Требуется дополнительная документация, демонстрирующая, что архитектура и реализация ТСВ отвечают требованиям безопасности.

Класс **A1**. Формальная верификация. Системы класса **A1** функционально эквивалентны системам класса **B3**, и к ним не предъявляется никаких дополнительных функциональных требований. В отличие от систем класса **B3** в ходе разработки должны применяться формальные методы верификации, что позволяет с высокой уверенностью получить корректную реализацию функций защиты. Процесс доказательства адекватности реализации начинается на ранней стадии разработки с построения формальной модели политики безопасности и спецификаций высокого уровня. Для обеспечения методов верификации системы класса **A1** должны содержать более мощные средства управления конфигурацией и защищенную процедуру дистрибуции.

Приведенные классы безопасности надолго определили основные концепции безопасности и ход развития средств защиты.

2.5.4. Интерпретация и развитие "Оранжевой книги"

Опубликование "Оранжевой книги" стало важным этапом и сыграло значительную роль в развитии технологий обеспечения безопасности компьютерных систем. Тем не менее, в ходе применения ее положений выяснилось, что часть практически важных вопросов осталась за рамками данного стандарта, и, кроме того, с течением времени (с момента опубли-

кования прошло пятнадцать лет) ряд положений устарел и потребовал пересмотра.

Круг специфических вопросов по обеспечению безопасности компьютерных сетей и систем управления базами данных нашел отражение в отдельных документах, изданных Национальным центром компьютерной безопасности США в виде дополнений к "Оранжевой книге" — "Интерпретация <"Оранжевой книги"> для компьютерных сетей" (Trusted Network Interpretation [8]) è "Интерпретация <"Оранжевой книги"> для систем управления базами данных" (Trusted Database Management System Interpretation [9]). Эти документы содержат трактовку основных положений "Оранжевой книги" применительно к соответствующим классам систем обработки информации.

Устаревание ряда положений "Оранжевой книги" обусловлено прежде всего интенсивным развитием компьютерных технологий и переходом с мэйнфреймов (типа вычислительных комплексов IBM-360, 370, советский аналог — машины серии ЕС) к рабочим станциям, высокопроизводительным персональным компьютерам и сетевой модели вычислений. Именно для того, чтобы исключить возникшую в связи с изменением аппаратной платформы некорректность некоторых положений "Оранжевой книги", адаптировать их к современным условиям и сделать адекватными нуждам разработчиков и пользователей программного обеспечения, и была проделана огромная работа по интерпретации и развитию положений этого стандарта. В результате возник целый ряд сопутствующих "Оранжевой книге" документов, многие их которых стали ее неотъемлемой частью. К наиболее часто упоминаемым относятся:

- Руководство по произвольному управлению доступом в безопасных системах (A guide to understanding discretionary access control in trusted systems) [10].
- Руководство по управлению паролями (Password management guideline)[11].
- Руководство по применению Критериев безопасности компьютерных систем в специфических средах (Guidance for applying the Department of Defence Trusted Computer System Evaluation Criteria in specific environment)[12].
- Руководство по аудиту в безопасных системах (A Guide to Understanding Audit in Trusted Systems)[13].
- Руководство по управлению конфигурацией в безопасных системах (Guide to understanding configuration management in trusted systems)[14].

Количество подобных вспомогательных документов, комментариев и интерпретаций значительно превысило объем первоначального документа, и в 1995 году Национальным центром компьютерной безопасности

США был опубликован документ под названием "Интерпретация критериев безопасности компьютерных систем"[15], объединяющий все дополнения и разъяснения. При его подготовке состав подлежащих рассмотрению и толкованию вопросов обсуждался на специальных конференциях разработчиков и пользователей защищенных систем обработки информации. В результате открытого обсуждения была создана база данных, включающая все спорные вопросы, которые затем в полном объеме были проработаны специально созданной рабочей группой. В итоге появился документ, проинтегрировавший все изменения и дополнения к "Оранжевой книге", сделанные с момента ее опубликования, что привело к обновлению стандарта и позволило применять его в современных условиях.

2.5.5. Выводы

"Критерии безопасности компьютерных систем" министерства обороны США представляют собой первую попытку создать единый стандарт безопасности, рассчитанный на разработчиков, потребителей и специалистов по сертификации компьютерных систем. В свое время этот документ явился настоящим прорывом в области безопасности информационных технологий и послужил отправной точной для многочисленных исследований и разработок. Основной отличительной чертой этого документа является его ориентация на системы военного применения, причем в основном на операционные системы. Это предопределило доминирование требований, направленных на обеспечение секретности обрабатываемой информации и исключение возможностей ее разглашения. Большое внимание уделено меткам (грифам секретности) и правилам экспорта секретной информации.

Критерии адекватности реализации средств защиты и политики безопасности отражены слабо, соответствующий раздел по существу ограничивается требованиями контроля целостности средств защиты и поддержания их работоспособности, чего явно недостаточно.

Высший класс безопасности, требующий осуществления верификации средств защиты, построен на доказательстве соответствия программного обеспечения его спецификациям с помощью специальных методик, однако это доказательство (очень дорогостоящее, трудоемкое и практически неосуществимое для реальных операционных систем) не подтверждает корректность и адекватность реализации политики безопасности.

"Оранжевая книга" послужила основой для разработчиков всех остальных стандартов информационной безопасности и до сих пор используется в США в качестве руководящего документа при сертификации компьютерных систем обработки информации.

2.6. Европейские критерии безопасности информационных технологий

Проблемы информационной безопасности актуальны не только для Соединенных штатов — вслед за выходом "Оранжевой книги" страны Европы разработали согласованные "Критерии безопасности информационных технологий" (Information Technology Security Evaluation Criteria, далее "Европейские критерии")[16]. Данный обзор основывается на версии 1.2 этих Критериев, опубликованной в июне 1991 года от имени соответствующих органов четырех стран: Франции, Германии, Нидерландов и Великобритании.

2.6.1. Основные понятия

"Европейские Критерии" рассматривают следующие задачи средств информационной безопасности:

- защита информации от несанкционированного доступа с целью обеспечение конфиденциальности;
- обеспечение целостности информации посредством защиты от ее несанкционированной модификации или уничтожения;
- обеспечение работоспособности систем с помощью противодействия угрозам отказа в обслуживании.

Для того, чтобы удовлетворить требованиям конфиденциальности, целостности и работоспособности, необходимо реализовать соответствующий набор функций безопасности, таких как идентификация и аутентификация, управление доступом, восстановление после сбоев и т. д.. Чтобы средства защиты можно было признать эффективными, требуется определенная степень уверенности в правильности их выбора и надежности функционирования. Для решения этой проблемы в "Европейских критериях" впервые вводится понятие адекватности (assurance) средств защиты.

Адекватность включает в себя два аспекта: эффективность, отражающую соответствие средств безопасности решаемым задачам, и корректность, характеризующую процесс их разработки и функционирования. Эффективность определяется соответствием между задачами, поставленными перед средствами безопасности, и реализованным набором функций защиты — их функциональной полнотой и согласованностью, простотой использования, а также возможными последствиями использования злоумышленниками слабых мест защиты. Под корректностью понимается правильность и надежность реализации функций безопасности.

Общая оценка уровня безопасности системы складывается из функциональной мощности средств защиты и уровня адекватности их реализации.

2.6.2. Функциональные критерии

В "Европейских критериях" средства, имеющие отношение к информационной безопасности, рассматриваются на трех уровнях детализации. На первом уровне рассматриваются цели, которые преследует обеспечение безопасности, второй уровень содержит спецификации функций защиты, а третий — реализующие их механизмы. Спецификации функций защиты предлагается рассматривать с точки зрения следующих требований:

- идентификация и аутентификация;
- управление доступом;
- подотчетность;
- аудит;
- повторное использование объектов;
- целостность информации;
- надежность обслуживания;
- безопасность обмен данными.

Большинство из перечисленных требований совпадает с аналогичными требованиями "Оранжевой книги". Остановимся лишь на специфичных для "Европейских критериев" моментах.

Требования безопасности обмена данными регламентируют работу средств, обеспечивающих безопасность данных, передаваемых по каналам связи, и включают следующие разделы:

- аутентификация;
- управление доступом;
- конфиденциальность данных;
- целостность данных;
- невозможность отказаться от совершенных действий.

Набор функций безопасности может специфицироваться с использованием ссылок на заранее определенные классы-шаблоны. В "Европейских критериях" таких классов десять. Пять из них (**F-C1**, **F-C2**, **F-B1**, **F-B2**, **F-B3**) соответствуют классам безопасности "Оранжевой книги" с аналогичными обозначениями. Рассмотрим подробнее другие пять классов, т. к. их требования отражают точку зрения разработчиков стандарта на проблему безопасности.

Класс **F-IN** предназначен для систем с высокими потребностями в обеспечении целостности, что типично для систем управления базами данных. Его описании основано на концепции "ролей", соответствующих видам деятельности пользователей, и предоставлении доступа к опреде-

ленным объектам только посредством доверенных процессов. Должны различаться следующие виды доступа: чтение, запись, добавление, удаление, создание, переименование и выполнение объектов.

Класс **F-AV** характеризуется повышенными требованиями к обеспечению работоспособности. Это существенно, например, для систем управления технологическими процессами. В требованиях этого класса указывается, что система должна восстанавливаться после отказа отдельного аппаратного компонента таким образом, чтобы все критически важные функции постоянно оставались доступными. В таком же режиме должна происходить и замена компонентов системы. Независимо от уровня загрузки должно гарантироваться определенное время реакции системы на внешние события.

Класс **F-DI** ориентирован на распределенные системы обработки информации. Перед началом обмена и при получении данных стороны должны иметь возможность провести идентификацию участников взаимодействия и проверить ее подлинность. Должны использоваться средства контроля и исправления ошибок. В частности, при пересылке данных должны обнаруживаться все случайные или намеренные искажения адресной и пользовательской информации. Знание алгоритма обнаружения искажений не должно позволять злоумышленнику производить нелегальную модификацию передаваемых данных. Должны обнаруживаться попытки повторной передачи ранее переданных сообщений.

Класс **F-DC** уделяет особое внимание требованиями к конфиденциальности передаваемой информации. Информация по каналам связи должна передаваться в зашифрованном виде. Ключи шифрования должны быть защищены от несанкционированного доступа.

Класс **F-DX** предъявляет повышенные требования и к целостности и к конфиденциальности информации. Его можно рассматривать как объединение классов **F-DI** и **F-DC** с дополнительными возможностями шифрования и защиты от анализа трафика. Должен быть ограничен доступ к ранее переданной информации, которая в принципе может способствовать проведению криптоанализа.

2.6.3. Критерии адекватности

"Европейские критерии" уделяют адекватности средств защиты значительно больше внимания чем функциональным требованиям. Как уже говорилось адекватность складывается из двух компонентов — эффективности и корректности работы средств защиты. Для оценки степени адекватности используются следующие критерии — рис. 2.2.

Рис. 2.2. Таксономия критериев адекватности "Европейских критериев".

"Европейские критерии" определяют семь уровней адекватности — от **E0** до **E6** (в порядке возрастания). Уровень **E0** обозначает минимальную адекватность (аналог уровня **D** "Оранжевой книги"). При проверке адекватности анализируется весь жизненный цикл системы — от начальной фазы проектирования до эксплуатации и сопровождения. Уровни адекватности от **E1** до **E6** выстроены по нарастанию требований тщательности контроля. Так, на уровне **E1** анализируется лишь общая архитектура системы, а адекватность средств защиты подтверждается функциональным тестированием. На уровне **E3** к анализу привлекаются исходные тексты программ и схемы аппаратного обеспечения. На уровне **E6** требуется формальное описание функций безопасности, общей архитектуры, а также политики безопасности.

Степень безопасности системы определяется самым слабым из критически важных механизмов защиты. В "Европейских критериях" определены три уровня безопасности — базовый, средний и высокий.

Безопасность считается базовой, если средства защиты способны противостоять отдельным случайным атакам.

Безопасность считается средней, если средства защиты способны противостоять злоумышленникам, обладающим ограниченными ресурсами и возможностями.

Наконец, безопасность можно считать высокой, если есть уверенность, что средства защиты могут быть преодолены только злоумышленником с высокой квалификацией, набор возможностей и ресурсов которого выходит за рамки возможного.

2.6.4. Выводы

"Европейские критерии безопасности информационных технологий" появившиеся, вслед за "Оранжевой книгой" оказали существенное влияние на стандарты безопасности и методику сертификации.

Главное достижение этого документа — введение понятия адекватности средств защиты и определение отдельной шкалы для критериев адекватности. Как уже упоминалось, "Европейские критерии" придают адекватность средств защиты даже большее значение чем их функциональности. Этот подход используется во многих появившихся позднее стандартах информационной безопасности.

Необходимо отметить, что "Европейские критерии" тесно связаны с "Оранжевой книгой", что делает их не вполне самостоятельным документом.

На первый взгляд довольно странным выглядит тот факт, что "Европейские критерии" признают возможность наличия недостатков в сертифицированных системах (критерий возможности использования не-

достатков защиты), однако на самом деле это свидетельствует о реалистичном взгляде существующее положение и признании того очевидного факта, что существующие системы еще весьма несовершенны и далеки от идеала (см. гл. 3).

"Европейские критерии" безопасности информационных технологий наряду с "Оранжевой книгой" легли в основу многих стандартов безопасности компьютерных систем.

2.7. Руководящие документы Гостехкомиссии России

2.7.1. Основные положения

В 1992 г. Гостехкомиссия (ГТК) при Президенте Российской федерации (РФ) опубликовала пять Руководящих документов, посвященных вопросам защиты от несанкционированного доступа к информации [1,2,3,4,5]. Мы рассмотрим важнейшие их них — "Концепция защиты средств вычислительной техники от несанкционированного доступа к информации", "Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации", "Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации".

Идейной основой этих документов является "Концепция защиты средств вычислительной техники от несанкционированного доступа к информации(НСД)"[1], содержащая систему взглядов ГТК на проблему информационной безопасности и основные принципы защиты компьютерных систем. С точки зрения разработчиков данных документов основная и едва ли не единственная задача средств безопасности — это обеспечение защиты от несанкционированного доступа к информации. Если средствам контроля и обеспечения целостности еще уделяется некоторое внимание, то поддержка работоспособности систем обработки информации (как мера защиты от угроз работоспособности) вообще не упоминается. Определенный уклон в сторону поддержания секретности объясняется тем, что эти документы были разработаны в расчете на применение в информационных системах министерства обороны и спецслужб РФ, а также недостаточно высоким уровнем информационных технологий этих систем по сравнению с современным.

2.6.2. Таксономия критериев и требований безопасности

Руководящие документы ГТК предлагают две группы критериев безопасности — показатели защищенности средств вычислительной техники (СВТ) от НСД и критерии защищенности автоматизированных систем (АС) обработки данных. Первая группа позволяет оценить степень защищенности (правда только относительно угроз одного типа — НСД) отдельно поставляемых потребителю компонентов ВС, а вторая рассчитана на полнофункциональные системы обработки данных.

Поскольку данные документы легко доступны и часто служили объектами комментариев и критики[6], ограничимся только кратким обзором их основных положений.

2.7.2.1. Показатели защищенности СВТ от НСД

Данный руководящий документ устанавливает классификацию СВТ по уровню защищенности от НСД к информации на базе перечня показателей защищенности и совокупности описывающих их требований. Под СВТ понимается совокупность программных и технических элементов систем обработки данных, способных функционировать самостоятельно или в составе других систем.

Данные показатели содержат требования защищенности СВТ от НСД к информации и применяются к общесистемным программным средствам и операционным системам (с учетом архитектуры ЭВМ). Конкретные перечни показателей определяют классы защищенности СВТ и описываются совокупностью требований. Совокупность всех средств защиты составляет комплекс средств защиты (КСЗ).

Установлено семь классов защищенности СВТ от НСД к информации. Самый низкий класс — седьмой, самый высокий — первый.

Показатели защищенности и установленные требования к классам приведены в табл. 2.1.

2.7.2.2. Требования к защищенности автоматизированных систем

Данные требования являются составной частью критериев защищенности автоматизированных систем обработки информации от НСД. Требования сгруппированы вокруг реализующих их подсистем защиты. В отличие от остальных стандартов отсутствует раздел, содержащий требования по обеспечению работоспособности системы, зато присутствует раздел, посвященный криптографическим средствам. Другие стандарты информационной безопасности, не содержат даже упоминания о криптографии, т. к. рассматривают ее исключительно в качестве механизма за-

щиты, реализующего требования аутентификации, контроля целостности и т. д. Исключением являются только "Единые критерии" (раздел 2.10), однако и в них требования раздела криптографии касаются распределения ключей, все остальное регламентируется отдельными стандартами. Таксономия требований к средствам защиты АС от НСД приведена на рис. 2.3.

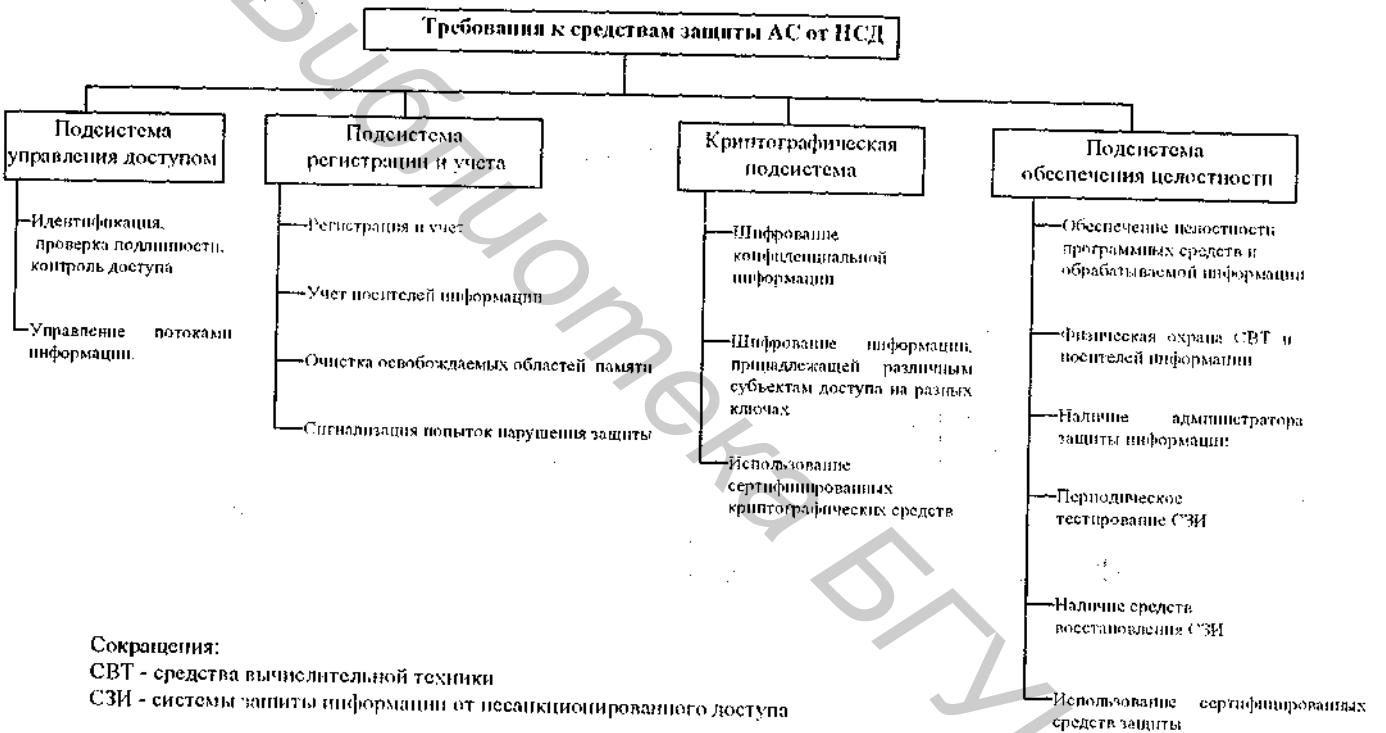
Таблица 2.1. Распределение показателей защищенности по классам СВТ.

Наименование показателя	Класс защищенности					
	6	5	4	3	2	1
Дискреционный принцип контроля доступа	+	+	+	=	+	=
Мандатный принцип контроля доступа	-	-	+	=	=	=
Очистка памяти	-	+	+	+	=	=
Изоляция модулей	-	-	+	=	+	=
Маркировка документов	-	-	+	=	=	=
Защита ввода и вывода на отчужденный физический носитель информации	-	-	+	=	=	=
Сопоставление пользователя с устройством	-	-	+	=	=	=
Идентификация и аутентификация	+	=	+	=	=	=
Гарантии проектирования	-	+	+	+	+	+
Регистрация	-	+	+	+	=	=
Взаимодействие пользователя с КСЗ	-	-	-	+	=	=
Надежное восстановление	-	-	-	+	=	=
Целостность КСЗ	-	+	+	+	=	=
Контроль модификации	-	-	-	-	+	=
Контроль дистрибуции	-	-	-	-	+	=
Гарантии архитектуры	-	-	-	-	-	+
Тестирование	+	+	+	+	+	=
Руководство пользователя	+	=	=	=	=	=
Руководство по КСЗ	+	+	=	+	+	=
Текстовая документация	+	+	+	+	+	=
Конструкторская (проектная) документация	+	+	+	+	+	+

Обозначения:
"-" - нет требований к данному классу;
"+" - новые или дополнительные требования;
"=" - требования совпадают с требованиями к СВТ предыдущего класса;
"КСЗ" - комплекс средств защиты.

2.7.3. Классы защищенности автоматизированных систем

Документы ГТК устанавливают девять классов защищенности АС от НСД, каждый из которых характеризуется определенной совокупностью требований к средствам защиты. Классы подразделяются на три группы, отличающиеся спецификой обработки информации в АС. Группа АС определяется на основании следующих признаков:



Сокращения:
СВТ - средства вычислительной техники
СЗИ - системы защиты информации от несанкционированного доступа

Рис 2.3. Таксономия требований к средствам защиты АС от НСД

- наличие в АС информации различного уровня конфиденциальности;
- уровень полномочий пользователей АС на доступ к конфиденциальной информации;
- режим обработки данных в АС (коллективный или индивидуальный).

В пределах каждой группы соблюдается иерархия классов защищенности АС. Класс, соответствующий высшей степени защищенности для данной группы, обозначается индексом *N***А**, где *N* — номер группы(от 1 до 3). Следующий класс обозначается *N***Б** и т.д.

Третья группа включает АС, в которых работает один пользователь, допущенный ко всей информации АС, размещенной на носителях одного уровня конфиденциальности. Группа содержит два класса — **3Б** и **3А**.

Вторая группа включает АС, в которых пользователи имеют одинаковые полномочия доступа ко всей информации, обрабатываемой и/или хранимой в АС на носителях различного уровня конфиденциальности. Группа содержит два класса — **2Б** и **2А**.

Первая группа включает многопользовательские АС, в которых одновременно обрабатывается и/или хранится информация разных уровней конфиденциальности. Не все пользователи имеют равные права доступа. Группа содержит пять классов — **1Д**, **1Г**, **1В**, **1Б** и **1А**.

В таб. 2.2 приведены требования к подсистемам защиты для каждого класса.

2.7.4. Выводы

Разработка руководящих документов ГТК явилась следствием бурно развивающегося в России процесса внедрения информационных технологий. До начала 90-х годов необходимости в подобных документах не было, т. к. в большинстве случаев обработка и хранение конфиденциальной информации осуществлялись без применения вычислительной техники. Поэтому разработка стандартов подобного рода представляет собой абсолютно новую и незнакомую область деятельности для соответствующих институтов и учреждений, что позволяет трактовать данные документы как первую стадию формирования отечественных стандартов в области информационной безопасности.

На разработку этих документов наибольшее влияние оказала "Оранжевая книга", однако это влияние в основном отражается в ориентированности обоих документов на системы военного применения и в использовании единой универсальной шкалы оценки степени защищенности.

К недостаткам данного стандарта относятся уже упоминавшиеся отсутствие требований к защите от угроз работоспособности, ориентация

на противодействие НСД и отсутствие требований к адекватности реализации политики безопасности. Понятие "политика безопасности" трактуется исключительно как поддержание режима секретности и отсутствие НСД[1], что принципиально неверно. Из-за этого средства защиты ориентируются исключительно на противодействие внешним угрозам, а к структуре самой системе и ее функционированию не предъявляется никаких требований. Ранжирование требований по классам защищенности по сравнению с остальными стандартами информационной безопасности максимально упрощено и сведено до определения наличия/отсутствия заданного набора механизмов защиты, что существенно снижает гибкость требований и возможность их практического применения.

Несмотря на указанные недостатки документы ГТК заполнили правовой вакуум в области стандартов информационной безопасности в нашей стране и на определенном этапе оперативно решили актуальную проблему.

Таблица 2.2. Требования к классам защищенности АС.

Подсистемы и требования	Классы								
	3Б	3А	2Б	2А	1Д	1Г	1В	1Б	1А
1. Подсистема управления доступом 1.1. Идентификация. Проверка подлинности и контроль доступа субъектов в систему,	+	+	+	+	+	+	+	+	+
к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ,				+		+	+	+	+
к программам,				+		+	+	+	+
к томам, каталогам, файлам, записям, полям записей,				+		+	+	+	+
1.2. Управление потоками информации.				+			+	+	+
2. Подсистема регистрации и учета 2.1.Регистрация и учет: входа/выхода субъектов доступа в/из системы (узла сети),	+	+	+	+	+	+	+	+	+
выдачи печатных (графических) выходных документов,		+		+		+	+	+	+
запуска/завершения программ и процессов (заданий, задач),				+		+	+	+	+
доступа программ к защищаемым файлам, включая их создание и удаление, передачу по линиям и каналам связи,				+		+	+	+	+

доступа программ к терминалам ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ, программам, каталогам, файлам, записям, полям записей,				+		+	+	+	+
изменения полномочий субъектов доступа,							+	+	+
создаваемых защищаемых объектов доступа.				+			+	+	+
2.2. Учет носителей информации	+	+	+	+	+	+	+	+	+
2.3. Очистка (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей		+		+		+	+	+	+
2.4. Сигнализация попыток нарушения защиты							+	+	+
3. Криптографическая подсистема 3.1. Шифрование конфиденциальной информации				+				+	+
3.2. Шифрование информации, принадлежащей различным субъектам доступа (группам субъектов) на разных ключах									+
3.3. Использование аттестованных (сертифицированных) криптографических средств				+				+	+
4. Подсистема обеспечения целостности 4.1. Обеспечение целостности программных средств и обрабатываемой информации	+	+	+	+	+	+	+	+	+
4.2. Физическая охрана средств вычислительной техники и носителей информации	+	+	+	+	+	+	+	+	+
4.3. Наличие администратора (службы) защиты информации в АС				+			+	+	+
4.4. Периодическое тестирование СЗИ НСД	+	+	+	+	+	+	+	+	+
4.5. Наличие средств восстановления СЗИ НСД	+	+	+	+	+	+	+	+	+
4.6. Использование сертифицированных средств защиты		+		+			+	+	+

Обозначения: "+" - требование к данному классу присутствует;
"СЗИ НСД" - система защиты информации от несанкционированного доступа.

2.8. Федеральные критерии безопасности информационных технологи

2.8.1. Цель разработки

"Федеральные критерии безопасности информационных технологий" (Federal Criteria for Information Technology Security)[17] разрабатывались как одна из составляющих "Американского федерального стандарта по обработке информации" (Federal Information Processing Standard), призванного заменить "Оранжевую книгу". Разработчиками стандарта выступили Национальный институт стандартов и технологий США (National Institute of Standards and Technology) и Агентство национальной безопасности США (National Security Agency). Данный обзор основан на версии 1.0 этого документа, опубликованной в декабре 1992 года.

Этот документ разработан на основе результатов многочисленных исследований в области обеспечения безопасности информационных технологий 80-х — начала 90-х годов, а также на основе анализа опыта использования "Оранжевой книги". Документ представляет собой основу для разработки и сертификации компонентов информационных технологий с точки зрения обеспечения безопасности. Создание "Федеральных критериев безопасности информационных технологий" преследовало следующие цели:

1. Определение универсального и открытого для дальнейшего развития базового набора требований безопасности, предъявляемых к современным информационным технологиям. Требования к безопасности и критерии оценки уровня защищенности должны соответствовать современному уровню развития информационных технологий и учитывать его прогресс в будущем. Стандарт предлагает обоснованный и структурированный подход к разработке требований безопасности к продуктам информационных технологий с учетом областей их применения.

2. Совершенствование существующих требований и критериев безопасности. В связи с развитием информационных технологий назрела необходимость пересмотра фундаментальных принципов безопасности с учетом появления новых областей их применения как в государственном так и в частном секторе.

3. Приведение в соответствие принятых в разных странах требований и критериев безопасности информационных технологий.

4. Нормативное закрепление основополагающих принципов информационной безопасности. Стандарт является обобщением основных принципов обеспечения безопасности информационных технологий, раз-

работанных в 80-е годы, и обеспечивает преемственность по отношению к ним с целью сохранения достижений в области защиты информации.

2.8.2. Основные положения

"Федеральные критерии безопасности информационных технологий" (далее, просто "Федеральные критерии") охватывают практически полный спектр проблем, связанных с защитой и обеспечением безопасности, т.к. включают все аспекты обеспечения конфиденциальности, целостности и работоспособности.

Основными объектами применения требований безопасности "Федеральных критериев", являются продукты информационных технологий (Information Technology Products) и системы обработки информации (Information Technology Systems). Под продуктом информационных технологий (далее просто "ИТ-продукт") понимается совокупность аппаратных и/или программных средств, которая представляет собой поставляемое конечному потребителю готовое к использованию средство обработки информации. Как правило, ИТ-продукт эксплуатируется не автономно, а интегрируется в систему обработки информации, представляющую собой совокупность ИТ-продуктов, объединенных в функционально полный комплекс, предназначенный для решения прикладных задач. В ряде случаев система обработки информации может состоять только из одного ИТ-продукта, обеспечивающего решение всех стоящих перед системой задач и удовлетворяющего требованиям безопасности. С точки зрения безопасности принципиальное различие между ИТ-продуктом и системой обработки информации определяется средой их эксплуатации. Продукт информационных технологий обычно разрабатывается в расчете на то, что он будет использован во многих системах обработки информации, и, следовательно, разработчик должен ориентироваться только на самые общие предположения о среде эксплуатации своего продукта, включающие условия применения и общие угрозы. Напротив, система обработки информации разрабатывается для решения прикладных задач в расчете на требования конечных потребителей, что позволяет в полной мере учитывать специфику воздействий со стороны конкретной среды эксплуатации.

"Федеральные критерии" содержат положения, относящиеся только к отдельным продуктам информационных технологий. Вопросы построения систем обработки информации из набора ИТ-продуктов не являются предметом рассмотрения этого документа.

Положения "Федеральных критериев" касаются только собственных средств обеспечения безопасности ИТ-продуктов, т.е. механизмов защиты, встроенных непосредственно в эти продукты в виде соответст-

вующих программных, аппаратных или специальных средств. Для повышения их эффективности могут дополнительно применяться внешние системы защиты и средства обеспечения безопасности, к которым относятся как технические средства, так и организационные меры, правовые и юридические нормы. В конечном счете, безопасность ИТ-продукта определяется совокупностью собственных средств обеспечения безопасности и внешних средств, являющихся частью среды эксплуатации.

Ключевым понятием концепции информационной безопасности "Федеральных критериев" является понятие "Профиля защиты" (Protection Profile). Профиль защиты — это нормативный документ, который регламентирует все аспекты безопасности ИТ-продукта в виде требований к его проектированию, технологии разработки и квалификационному анализу. Как правило, один Профиль защиты описывает несколько близких по структуре и назначению ИТ-продуктов. Основное внимание в Профиле защиты уделяется требованиям к составу средств защиты и качеству их реализации, а также их адекватности предполагаемым угрозам безопасности.

"Федеральные критерии" представляют процесс разработки систем обработки информации, начинающийся с формулирования требований потребителями и заканчивающийся введением в эксплуатацию, в виде следующих основных этапов:

1. Разработка и анализ Профиля защиты. Требования, изложенные в Профиле защиты, определяют функциональные возможности ИТ-продуктов по обеспечению безопасности и условия эксплуатации, при соблюдении которых гарантируется соответствие предъявляемым требованиям. Кроме требований безопасности Профиль содержит требования по соблюдению технологической дисциплины в процессе разработки, тестирования и квалификационного анализа ИТ-продукта. Профиль безопасности анализируется на полноту, непротиворечивость и техническую корректность.

2. Разработка и квалификационный анализ ИТ-продуктов. Разработанные ИТ-продукты подвергаются независимому анализу, целью которого является определение степени соответствия характеристик продукта сформулированным в Профиле защиты требованиям и спецификациям.

3. Компоновка и сертификация системы обработки информации в целом. Успешно прошедшие квалификацию уровня безопасности ИТ-продукты интегрируются в систему обработки информации. Полученная в результате система должна удовлетворять заявленным в Профиле защиты требованиям при соблюдении указанных в нем условий эксплуатации.

"Федеральные критерии" регламентируют только первый этап этой схемы — разработку и анализ Профиля защиты, процесс создания ИТ-

продуктов и компоновка систем обработки информации остаются вне рамок этого стандарта.

2.8.3. Профиль защиты

Как уже говорилось, Профиль защиты является центральным понятием "Федеральных критериев", большая часть содержания которых представляет собой описание разделов Профиля защиты, включающее таксономию требований безопасности и их ранжирование. Рассмотрим назначение, структуру и этапы разработки Профиля защиты.

2.8.3.1 Назначение и структура Профиля защиты

Профиль защиты предназначен для определения и обоснования состава и содержания средств защиты, спецификации технологии разработки и регламентации процесса квалификационного анализа ИТ-продукта. Профиль защиты состоит из следующих пяти разделов:

- описание;
- обоснование;
- функциональные требования к ИТ-продукту;
- требования к технологии разработки ИТ-продукта;
- требования к процессу квалификационного анализа ИТ-продукта.

Описание Профиля содержит классификационную информацию, необходимую для его идентификации в специальной картотеке. "Федеральные критерии" предлагают поддерживать такую картотеку на общегосударственном уровне. Это позволит любой организации воспользоваться созданными ранее Профилями защиты непосредственно, или использовать их в качестве прототипов для разработки новых. В описании Профиля защиты должна быть охарактеризована основная проблема или группа проблем обеспечения безопасности, решаемых с помощью применения данного Профиля.

Обоснование содержит описание среды эксплуатации, предполагаемых угроз безопасности и методов использования ИТ-продукта. Кроме того, этот раздел содержит подробный перечень задач по обеспечению безопасности, решаемых с помощью данного Профиля. Эта информация дает возможность определить, в какой мере данный Профиль защиты пригоден для применения в той или иной ситуации. Предполагается, что данный раздел ориентирован на службы безопасности организаций, которые изучают возможность использования ИТ-продукта, соответствующего данному Профилю защиты.

Раздел *функциональных требований к ИТ-продукту* содержит описание функциональных возможностей средств защиты ИТ-продукта и оп-

ределяет условия, в которых обеспечивается безопасность в виде перечня угроз, которым успешно противостоят предложенные средства защиты. Угрозы, лежащие вне этого диапазона, должны быть устранены с помощью дополнительных, не входящих в состав продукта, средств обеспечения безопасности. Очевидно, что чем сильнее и опаснее угрозы, тем более мощными и стойкими должны быть средства, реализующие функции защиты.

Раздел *требований к технологии разработки ИТ-продукта* охватывает все этапы его создания, начиная от разработки проекта и заканчивая вводом готовой системы в эксплуатацию. Раздел содержит требования как к самому процессу разработки, так и к условиям, в которых она проводится, к используемым технологическим средствам, а также к документированию этого процесса. Выполнение требований этого раздела является непременным условием для проведения квалификационного анализа и сертификации ИТ-продукта.

Раздел *требований к процессу квалификационного анализа ИТ-продукта* регламентирует порядок проведения квалификационного анализа в виде методики исследований и тестирования ИТ-продукта. Объем и глубина требуемых исследований зависят от наиболее вероятных типов угроз, среды применения и планируемой технологии эксплуатации.

"Федеральные критерии" содержат подробное описание всех трех разделов Профиля защиты, посвященных требованиям, включающее их таксономию и ранжирование для каждого раздела. В данном обзоре основное внимание уделено функциональным требованиям, т. к. именно в этой области "Федеральные критерии" проделали значительный шаг вперед по сравнению с предшествующими стандартами.

2.8.3.2. Этапы разработки Профиля защиты

Разработка Профиля защиты осуществляется в три этапа: анализ среды применения ИТ-продукта с точки зрения безопасности, выбор Профиля-прототипа и синтез требований. На рис. 2.4 приведена общая схема разработки Профиля защиты.

На первой стадии проводится анализ информации о среде предполагаемого применения ИТ-продукта, действующих в этой среде угрозах безопасности и используемых этими угрозами недостатках защиты. Анализ проводится с учетом технологии использования продукта, а также существующих стандартов и нормативов, регламентирующих его эксплуатацию.

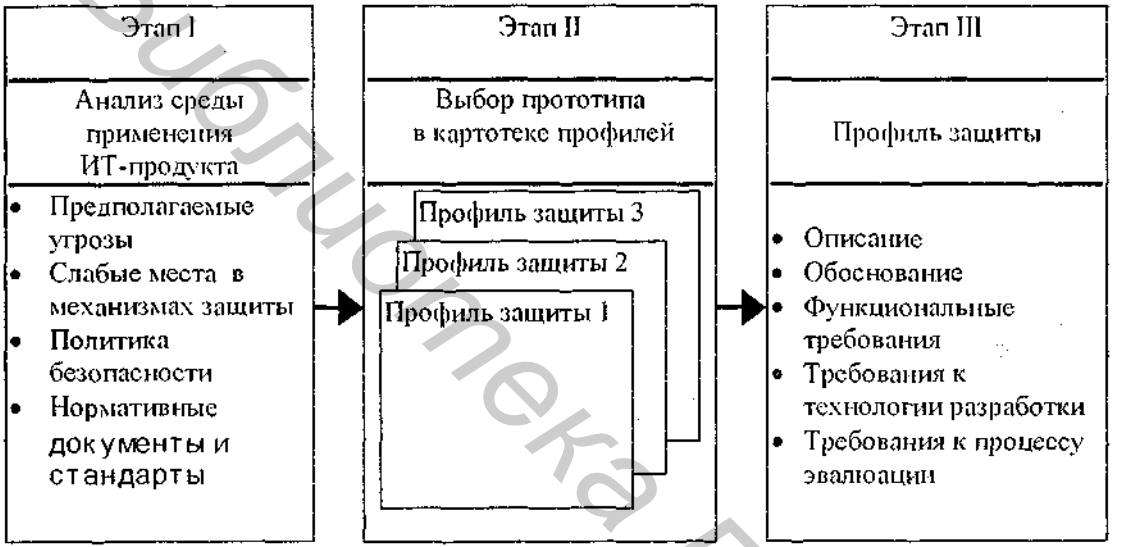


Рис. 2.4. Схема разработки Профиля защиты согласно "Федеральным критериям

Второй этап состоит в поиске Профиля, который может быть использован в качестве прототипа. Как уже говорилось, "Федеральные критерии" предусматривают создание специальной, доступной для разработчиков ИТ-продуктов, картотеки, в которую должны быть помещены все разработанные когда-либо Профили защиты. Это позволит минимизировать затраты на создание Профилей и учесть опыт предыдущих разработок.

Этап синтеза требований включает выбор наиболее существенных для условий функционирования продукта функций защиты и их ранжирование по степени важности с точки зрения обеспечения качества защиты. Выбор специфичных для среды требований безопасности должен быть основан на их эффективности для решения задачи противодействия угрозам. Разработчик Профиля должен показать, что выполнение выдвинутых требований ведет к обеспечению требуемого уровня безопасности посредством успешного противостояния заданному множеству угроз и устранения недостатков защиты.

При разработке Профиля защиты необходимо анализировать связи и взаимозависимости, существующие между функциональными требованиями и требованиями к процессу разработки, а также между отдельными требования внутри этих разделов. По завершению разработки Профиль защиты подвергается проверке, целью которой является подтверждение его полноты, корректности, непротиворечивости и реализуемости.

2.8.4. Функциональные требования к ИТ-продукту

"Федеральные критерии" предлагают набор функциональных требований, реализация которых позволяет противостоять наиболее распространенным угрозам безопасности, относящихся к широкому спектру ИТ-продуктов и областей их применения. Данные требования разработаны с учетом возможности расширения и адаптации к конкретным условиям эксплуатации ИТ-продукта, и допускают возможность совершенствования параллельно процессу развития информационных технологий. Требования, изложенные в "Федеральных критериях", разработаны на основе обобщения существовавших на момент их создания стандартов информационной безопасности — "Оранжевой книги" и "Европейских критериев" (п. 2.5 и 2.6).

Функциональные требования, приведенные в "Федеральных критериях", определяют состав и функциональные возможности ТСВ. Как говорилось в п.2.2, ТСВ объединяет все компоненты ИТ-продукта (аппаратные, программные и специальные средства), реализующие функции защиты. Таким образом, функциональные требования, направленные на

обеспечение безопасности, относятся либо к внутренним элементам ТСВ, либо к ее внешним функциям, доступным через специальные интерфейсы.

Для того, чтобы расширить спектр потенциального применения Профиля защиты, в "Федеральных критериях" при описании функциональных требований предполагается, что ТСВ является единственной частью ИТ-продукта, которая нуждается в защите и обладает такой характеристикой как уровень защищенности. По этой причине предполагается достаточным установить множество требований, касающихся только безопасности ТСВ.

Функциональные требования Профиля защиты задаются в виде общих положений и косвенным образом определяют множество угроз, которым может успешно противостоять удовлетворяющий им ИТ-продукт.

2.8.4.1. Таксономия функциональных требований

Функциональные требования "Федеральных критериев" разделены на восемь классов и определяют все аспекты функционирования ТСВ. Таксономия классов функциональных требований приведена на рис. 2.5. Все классы имеют непосредственное отношение к обеспечению безопасности функционирования ТСВ. Реализация политики безопасности должна быть поддержана средствами, обеспечивающими надежность функционирования как самой ТСВ так и механизмов осуществления политики безопасности. Эти средства также входят в состав ТСВ, хотя, с точки зрения противодействия угрозам, вносят только косвенный вклад в общую защиту ИТ-продукта.

Поскольку "Федеральные критерии" по сравнению с "Оранжевой книгой" являются стандартом нового поколения, и, кроме того, никогда не рассматривались в отечественных публикациях, остановимся на функциональных требованиях более подробно.

Требования к реализации политики безопасности описывают функции ТСВ, реализующие политику безопасности, и состоят из четырех групп: требования к политике аудита, политике управления доступом, политике обеспечения работоспособности и управлению безопасностью. Эти требования носят весьма общий характер, что позволяет рассматривать их в качестве прототипа, обеспечивающего поддержку широкого спектра политик и моделей безопасности (см. гл. 4).

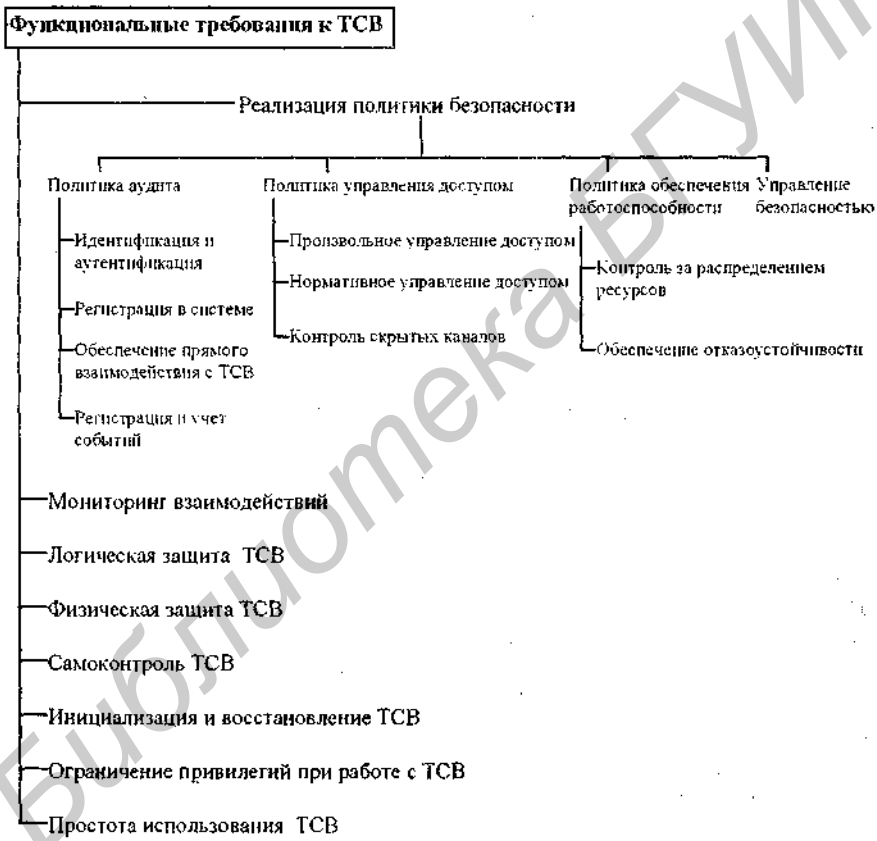


Рис 2.5. Таксономия функциональных требований "Федеральных критериев".

Политика аудита включают разделы, относящиеся к идентификации и аутентификации, процедуре регистрации пользователя в системе, обеспечению прямого взаимодействия с ТСВ, а также регистрацию и учет событий. Основная задача политики управления аудитом — обеспечить возможность однозначной идентификации субъекта, ответственного за те или иные действия в системе.

Идентификация и аутентификация позволяют установить однозначное соответствие между пользователями и представляющими их в ВС субъектами разграничения доступа, а также подтвердить подлинность этого соответствия.

Регистрация пользователя в системе означает создание субъекта взаимодействия, с идентификатором которого будут ассоциироваться все последующие действия пользователя. К процедуре регистрации также относится учет места, времени и других параметров подключения к системе и ее блокирование во время отсутствия пользователя.

Обеспечение прямого взаимодействия с ТСВ гарантирует, что пользователь взаимодействует с компонентами ТСВ напрямую, т.е. информация, которая передается в ТСВ и обратно, не подвергается перехвату или искажению. Поддержка прямого взаимодействия с ТСВ особенно важна для управления безопасностью (например, при администрировании прав доступа и полномочий пользователей).

Регистрация и учет событий в системе позволяет распознавать потенциально опасные ситуации и сигнализировать о случаях нарушения безопасности. Регистрация событий включает распознавание, учет и анализ действий пользователя, представляющих интерес с точки зрения безопасности.

Политика управления доступом содержит следующие разделы: произвольное управление доступом, нормативное управление доступом и контроль скрытых каналов утечки информации. Политика управления доступом является основным механизмом защиты т. к. непосредственно обеспечивает конфиденциальность и целостность обрабатываемой информации.

Произвольное управление доступом позволяет осуществлять назначение прав доступа с точностью до идентифицируемых субъектов и объектов, а также поддерживаемых типов доступа и, кроме того, обеспечивает контроль за распространением прав доступа среди субъектов.

Нормативное управление доступом, в отличие от произвольного, основано на контроле информационных потоков между субъектами и объектами и их атрибутах безопасности, что позволяет регламен-

тировать порядок использования информации в системе и противостоять атакам типа "троянского коня".

Контроль скрытых каналов утечки информации включает технические и административные меры, направленные на ликвидацию таких каналов посредством минимизации объема совместно используемых ресурсов и введения активных "шумовых помех".

Политика обеспечения работоспособности системы включают контроль за распределением ресурсов и обеспечение отказоустойчивости. Обеспечение работоспособности позволяет гарантировать доступность ресурсов и сервиса системы, а также корректное восстановление системы после сбоев.

Контроль за распределением ресурсов осуществляется посредством введения ограничений (квот) на их потребление или приоритетной системы распределения ресурсов.

Обеспечение отказоустойчивости входит в сферу безопасности наравне с другими требованиями, т. к. противостоит угрозам работоспособности.

Управление безопасностью регламентируют следующие аспекты функционирования системы:

- компоновка, установка, конфигурация и поддержка ТСВ;
- администрирование атрибутов безопасности пользователей (идентификаторов, полномочий, доступных ресурсов и т.д.);
- администрирование политики управления доступом;
- управление потреблением ресурсов системы;
- аудит действий пользователей.

Мониторинг взаимодействий. Требования этого раздела регламентируют порядок взаимодействия между компонентами системы и прохождения информационных потоков через ТСВ. Реализация политики безопасности будет эффективна только в том случае, если все без исключения взаимодействия в системе, т. е. доступ к объектам, ресурсам и сервису осуществляются при обязательном посредничестве ТСВ.

Логическая защита ТСВ. Требования данной группы устанавливают порядок доступа к внутренним компонентам ТСВ (данным и программам). ТСВ должна быть защищена от внешних воздействий со стороны непривилегированных пользователей, в противном случае искажение программ и данных, находящихся в ТСВ может привести к полному подавлению функций защиты.

Необходимо подчеркнуть, что политика безопасности, мониторинг взаимодействвий и логическая защита ТСВ являются обязательными компонентами всех Профилей защиты вне зависимости от назначения и среды применения ИТ-продукта.

Физическая защита ТСВ. Требования этой группы задают ограничения на физический доступ к компонентам ТСВ, а также допустимые физические параметры среды функционирования ВС.

Самоконтроль ТСВ. Требования, касающиеся самоконтроля ТСВ, определяют возможности обеспечения контроля корректности выполнения функций ТСВ и целостности программ и данных, входящих в ТСВ. Выполнение этих требований позволяет вовремя обнаруживать нарушения целостности компонентов ТСВ, произошедшие либо в результате целенаправленного воздействия, либо вследствии сбоя в работе аппаратных или программных средств, и осуществлять восстановление целостности ТСВ.

Инициализация и восстановление ТСВ. Требования данной группы устанавливают возможности ТСВ по контролю за процессом собственной инициализации и способности к самовосстановлению после сбоев. Процесс восстановления после сбоя должен происходить без нарушений функционирования, даже временного, средств защиты. Восстановленное состояние ТСВ должно соответствовать требованиям политики безопасности, мониторинга взаимодействий и самоконтроля целостности.

Ограничение привилегий при работе с ТСВ. Требования этой группы устанавливают порядок назначения полномочий для работы с ТСВ. Основным принципом назначения таких полномочий является принцип минимальной достаточности. Это обеспечивается посредством постоянного контроля и, при необходимости, автоматического понижения привилегий пользователей при обращении к компонентам или сервису ТСВ. Соблюдение этого принципа позволяет минимизировать нарушения целостности в случае возникновения сбоев или нарушений безопасности.

Простота использования ТСВ. Эти требования обеспечивают удобство пользования возможностями ТСВ как для высококвалифицированных администраторов, ответственных за функционирование и безопасность системы, так и для рядовых пользователей, а также для разработчиков прикладных программ, взаимодействующих с ТСВ. К этому классу требований относятся: порядок реагирования ТСВ на ошибки в действиях пользователей и попытки нарушения безопасности, устанавливаемые по умолчанию полномочия, интерфейс пользователей и администратора.

Объем и глубина реализации функциональных требований зависит от того, какую степень защищенности должна обеспечивать ТСВ конкретного ИТ-продукта, а также от того, какие угрозы безопасности возможны в среде его эксплуатации. Степень обеспечения требуемого уровня защищенности зависит от реализованной политики безопасности, от квалификации ответственного за безопасность персонала, от правильности адми-

нистрирования ТСВ и соблюдения рядовыми пользователями правил политики безопасности.

2.8.4.2. Ранжирование функциональных требований

Состав и содержание включенных в Профиль защиты функциональных требований определяются средой эксплуатации ИТ-продукта. Чтобы обосновать выбор тех или иных требований и не вступать в противоречие с существующими стандартами в области безопасности ИТ-продуктов, функциональные требования, приведенные в "Федеральных критериях", ранжируются по уровням с помощью следующих четырех критериев: широта сферы применения, степень детализации, функциональный состав средств защиты, обеспечиваемый уровень безопасности.

Широта сферы применения определяется множеством сущностей к которому могут быть применены данные требования, а именно:

- пользователи системы, субъекты и объекты доступа;
- функции ТСВ и интерфейс взаимодействия с ТСВ;
- аппаратные, программные и специальные компоненты ТСВ;
- множество параметров конфигурации ТСВ.

Например, требования из разделов управления доступом, аудита, обеспечения работоспособности, мониторинга взаимодействий и простоты использования ТСВ могут относиться только к определенному подмножеству объектов доступа и параметров конфигурации ТСВ. Обеспечение прямого взаимодействия с ТСВ требуется только для некоторого подмножества функций ТСВ.

Степень детализации требований определяется множеством атрибутов сущностей, к которым применяются данные требования — либо ко всем атрибутам пользователей, субъектов или объектов, либо только к некоторому подмножеству этих атрибутов. Например, требования из разделов управления доступом, аудита и мониторинга взаимодействий могут относится только к некоторому подмножеству атрибутов субъектов и объектов — к правам доступа, групповым идентификаторам пользователей, но не к атрибутам состояния субъектов и объектов и не к индивидуальным идентификаторам.

Функциональный состав средств защиты определяется множеством функций включенных в ТСВ для реализации той или иной группы функциональных требований. Например, политика управления доступом может включать либо произвольное, либо нормативное управление доступом, или и то, и другое одновременно.

Обеспечиваемый уровень безопасности определяется условиями, в которых функциональные компоненты ТСВ способны противостоять за-

данному множеству угроз, отказам и сбоям. Например, нормативное управление доступом обеспечивает более высокий уровень безопасности чем произвольное в силу его способности противостоять атакам типа "троянского коня".

Ранжирование всегда предполагает установление некоторого отношения порядка. Однако, независимое ранжирование функциональных требований по каждому из описанных критериев, хотя и дает некоторое представление о различиях между функциональными возможностями средств защиты, не позволяет установить четкую, линейную шкалу оценки уровня безопасности. Строгого отношения порядка, определенного на множестве функциональных требований не существует, т.к. значение требований и уровень обеспечиваемой ими защиты зависят не только от их содержания, но и от назначения ИТ-продукта и среды его эксплуатации. Для одних систем наиболее важными будут идентификация и аутентификация пользователей, а для других — реализация политики управления доступом или обеспечение работоспособности.

Поэтому в "Федеральных критериях" отсутствуют рекомендации как по выбору и применению тех или иных функциональных требований, так и по определению их роли в системе обеспечения безопасности. Вместо жестких указаний этот документ содержит согласованный с предшествующими ему стандартами ("Оранжевая книга", "Европейские критерии") ранжированный перечень функциональных требований и предоставляет разработчикам Профиля защиты возможность самостоятельно сделать выбор необходимых методов и средств обеспечения безопасности, основанный на назначении и специфике среды эксплуатации ИТ-продукта.

Приводимое ранжирование не противоречит предшествующим стандартам и вводится для исключения ошибок в определении степени защищенности системы из-за неправильной оценки значимости отдельных групп требований. Кроме того ранжирование предоставляет разработчикам и пользователям возможность для обоснованной оценки реально обеспечиваемого уровня безопасности.

Применение критериев ранжирования к различным группам функциональных требований представлено в таб. 2.3.

В приложении I приведен полный ранжированный перечень функциональных требований "Федеральных критериев".

Таблица 2.3. Ранжирование функциональных требований "Федеральных критериев"

Функциональные требования	Широта сферы применения	Степень детализации	Функциональный состав средств защиты	Обеспечиваемый уровень безопасности
Реализация политики безопасности				
Политика аудита				
Идентификация и аутентификация			*	*
Регистрация в системе			*	
Обеспечение прямого взаимодействия с ТСВ	*		*	
Регистрация и учет событий			*	*
Политика управления доступом	*	*	*	
Контроль скрытых каналов	*		*	
Политика обеспечения работоспособности				
Контроль за распределением ресурсов	*		*	
Отказоустойчивость	-	-	-	-
Управление безопасностью			*	*
Мониторинг взаимодействий	*	*	*	
Логическая защита ТСВ			*	
Физическая защита ТСВ			*	*
Самоконтроль ТСВ	*		*	
Инициализация и восстановление ТСВ			*	
Ограничение привилегий при работе с ТСВ		*		
Простота использования ТСВ	*		*	

2.8.5. Требования к технологии разработки ИТ-продукта

Основное назначение требований к технологии разработки ИТ-продукта — обеспечить адекватность условий разработки функциональным требованиям, выдвинутым в соответствующем разделе Профиля защиты, и установить ответственность разработчика за корректность реализации этих требований. Данный раздел регламентирует процесс создания, тестирования, документирования и сопровождения ИТ-продукта. Таксономия требований к технологии разработки ИТ-продукта приведена на рис. 2.6.

Требования к технологии разработки ИТ-продуктов включают четыре раздела: требования к процессу разработки, к среде разработки, документированию и сопровождению.

Требования к процессу разработки содержат подразделы, относящиеся к проектированию, реализации, тестированию и анализу ИТ-продукта. Особую роль играют требования адекватности реализации функций ТСВ, обеспечивающие корректность

выполнения функциональных требований Профиля защиты.

Требования к среде разработки позволяют обеспечить качество процесса создания ИТ-продукта с помощью применения современных технологий проектирования, программирования и тестирования, а также регламентируют управление процессом разработки и дистрибуцию конечного продукта.

Требования к документированию определяют состав и содержание технологической документации, позволяющей производителю ИТ-продукта доказать соответствие самого продукта и технологии его изготовления выдвинутым требованиям.

Требования к сопровождению ИТ-продукта содержат обязательства производителя перед пользователями, выполнение которых позволяет обеспечить эффективную и надежную эксплуатацию ИТ-продукта. Данные требования регламентируют состав пользовательской и административной документации, процедуру обновления версий и исправления ошибок, а также инсталляцию продукта.

"Федеральные критерии" содержат ранжированный перечень типовых требований к технологии разработки ИТ-продуктов. Выполнение требований к технологии разработки является необходимым условием для проведения процедуры квалификационного анализа.

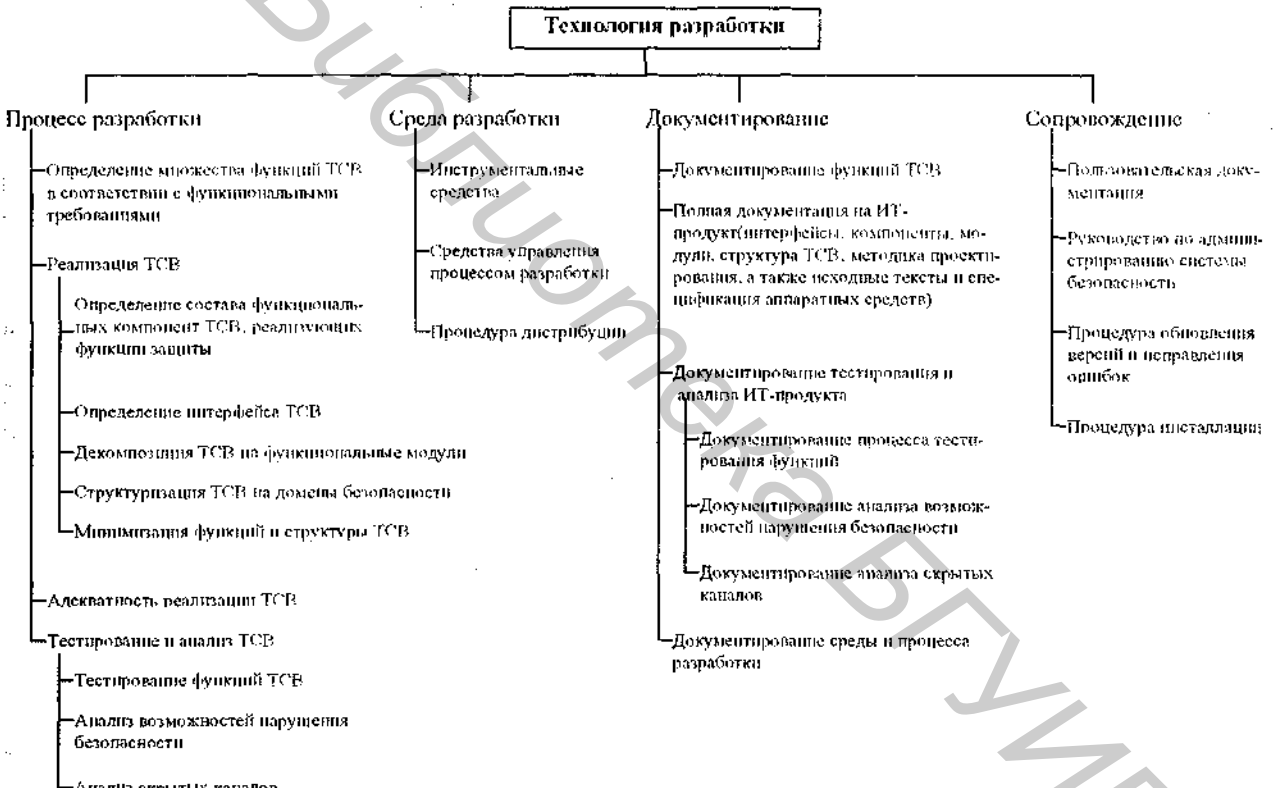


Рис.2.6. Таксономия требований "Федеральных критериев" к технологии разработки ИТ-продукта.

2.8.6. Требования к процессу квалификационного анализа ИТ-продукта

Требования к процессу квалификационного анализа ИТ-продукта призваны обеспечить надежность и корректность этого процесса. Раздел содержит три группы требований, регламентирующих анализ, контроль и тестирование ИТ-продукта. Таксономия требований этого раздела приведена на рис. 2.7.

Раздел требований к анализу ИТ-продукта содержит требования к проведению независимого анализа предложенного решения (архитектуры), и его реализации как конкретного средства.

Раздел требований к контролю регламентирует проверку соответствия среды разработки ИТ-продукта и обеспечиваемого производителем сопровождения требованиям к технологии разработки.

Требования к тестированию описывают процедуру проведения тестирования функций ТСВ как самим разработчиком ИТ-продукта, так и независимыми экспертами.

Нетрудно заметить, что эти требования регламентируют процесс квалификационного анализа только в общих чертах и, по замыслу разработчиков стандарта, должны послужить основой для разработки специализированных методик квалификации уровня безопасности, ориентированных на различные области применения и классы ИТ-продуктов.

2.8.7. Выводы

"Федеральные критерии безопасности информационных технологий" первый стандарт информационной безопасности, в котором определяются три независимые группы требований: функциональные требования к средствам защиты, требования к технологии разработки и к процессу квалификационного анализа. Авторами этого стандарта впервые предложена концепция Профиля защиты — документа, содержащего описание всех требований безопасности как к самому ИТ-продукту, так и к процессу его проектирования, разработки, тестирования и квалификационного анализа.

Функциональные требования безопасности хорошо структурированы и описывают все аспекты функционирования ТСВ. Требования к технологии разработки, впервые появившиеся в этом документе, побуждают производителей использовать современные технологии программирования как основу для подтверждения безопасности своего продукта.

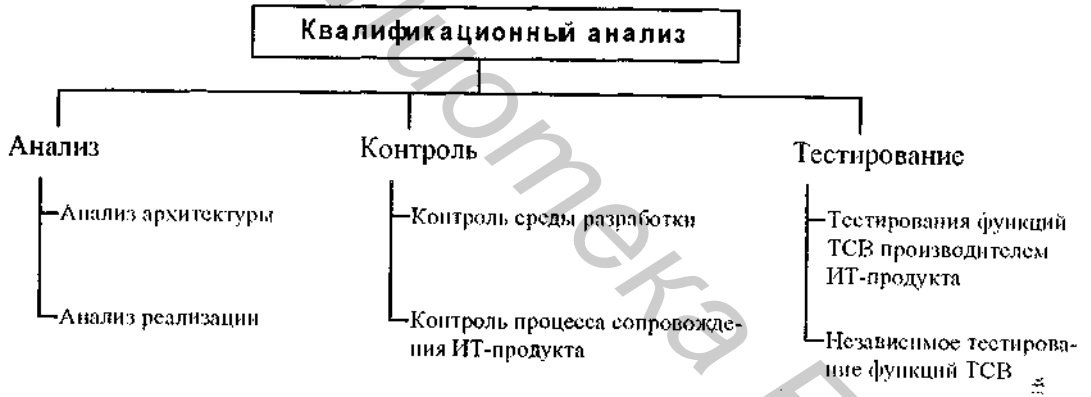


Рис. 2.7. Таксономия требований "Федеральных критериев" к процессу квалификационного анализа ИТ-продукта.

Требования к процессу квалификационного анализа носят довольно общий характер и не содержат конкретных методик тестирования и исследования безопасности ИТ-продуктов.

Разработчики "Федеральных критериев" отказались от используемого в "Оранжевой книге" подхода к оценки уровня безопасности ИТ-продукта на основании обобщенной универсальной шкалы классов безопасности. Вместо этого предлагается независимое ранжирование требований каждой группы, т. е. вместо единой шкалы используется множество частных шкал-критериев, характеризующих обеспечиваемый уровень безопасности. Данный подход позволяет разработчикам и пользователям ИТ-продукта выбрать наиболее приемлемое решение и точно определить необходимый и достаточный набор требований для каждого конкретного ИТ-продукта и среды его эксплуатации.

Особо отметим, что этот стандарт рассматривает устранение недостатков существующих средств безопасности как одну из задач защиты наряду с противодействием угрозам безопасности и реализацией модели безопасности.

Данный стандарт ознаменовал появление нового поколения руководящих документов в области информационной безопасности, а его основные положения послужили базой для разработки "Канадских критериев безопасности компьютерных систем" (п. 2.9) и "Единых критериев безопасности информационных технологий"(п. 2.10).

2.9. Канадские критерии безопасности компьютерных систем

2.9.1. Цель разработки

"Канадские критерии безопасности компьютерных систем" (Canadian Trusted Computer Product Evaluation Criteria, далее просто "Канадские критерии")[18] были разработаны в Центре безопасности ведомства безопасности связи Канады (Canadian System Security Centre Communication Security Establishment) для использования в качестве национального стандарта безопасности компьютерных систем. Этот обзор основан на третьей версии стандарта, опубликованной в январе 1993 года.

"Канадских критерии" разрабатывались как основа для оценки эффективности средств обеспечения безопасности компьютерных систем, при этом преследовались следующие цели:

1. Предложить единую шкалу критериев оценки безопасности компьютерных систем, позволяющую сравнивать системы обработки конфиденциальной информации по степени обеспечения безопасности.

2. Создать основу для разработки спецификаций безопасных компьютерных систем, которая могла бы использоваться разработчиками при проектировании подобных систем в качестве руководства для определения состава функций средств защиты.

3. Предложить унифицированный подход и стандартные средства для описания характеристик безопасных компьютерных систем.

"Канадские критерии" разрабатывались на основе "Оранжевой книги" (п. 2.5), и под влиянием "Федеральных критериев безопасности информационных технологий" (п. 2.8). В отличие от "Оранжевой книги", ориентированной в основном на разработку и сертификацию многопользовательских операционных систем, и требующей определенной интерпретации для применения в других областях (например, для баз данных и сетей), "Канадские критерии" были изначально нацелены на широкий диапазон компьютерных систем. Этот стандарт может быть использован для разработки требований безопасности, спецификаций средств защиты и сертификации программного обеспечения как рабочих станций, так и многопроцессорных вычислительных систем, персональных и многопользовательских операционных систем, систем управления базами данных, распределенных, сетевых, встроенных, объектно-ориентированных и других систем.

2.9.2. Базовые концепции "Канадских критериев"

В "Канадских критериях" используется отличное от общепринятого толкование ряда терминов, что определяет оригинальность предложенного ими подхода к описанию процесса взаимодействия пользователей с компьютерной системой и его инвариантность по отношению к политике безопасности.

2.9.2.1 Объекты и субъекты

В "Канадских критериях" все компоненты системы, находящиеся по управлением ТСВ называются *объектами*. Объекты могут находится в одном из следующих трех состояний: *объект-пользователь*, *объект-процесс*, *пассивный объект*, и, в зависимости от состояния, обозначают *пользователей*, *процессы* и *объекты* соответственно.

Пользователь представляет собой физическое лицо, взаимодействующее с компьютерной системой. Каждый пользователь имеет собственный уникальный идентификатор, права доступа, уровень привилегий и т.д.

С пользователями ассоциируются все *процессы*, существующие в системе. *Процесс*, или *активный объект*, представляющий *пользователя* в компьютерной системе, — это программа, выполнение которой инициировано пользователем. *Объект* представляет собой пассивный элемент, над которым выполняют действия *пользователи* и *процессы*. Все взаимодействия *объектов* контролируются в соответствии с реализованной в компьютерной системе политикой безопасности.

Таким образом в "Канадских критериях" отсутствует понятие *субъект*, широко используемое в других стандартах информационной безопасности для описания процесса взаимодействия[7, 17]. Обычно под *субъектом* принято понимать активного участника взаимодействия, а под *объектом* — пассивного. Напротив, в "Канадских критериях" все сущности компьютерной системы рассматриваются как *объекты*, а их взаимодействие описывается тройкой *пользователь-процесс-объект*. Однако противоречия здесь нет. Обычно (например, в "Оранжевой книге") понятие *субъект* представляет собой комбинацию понятий *пользователь* и *процесс*, действующий от его имени. Соответственно, для описания ситуации, когда один пользователь запускает много процессов, с позиций "Оранжевой книги" описывается с помощью множества субъектов, каждый из которых с точки зрения политики безопасности будет рассматриваться как независимый участник взаимодействия. "Канадские критерии" позволяют использовать для описания этой ситуации один *объект-пользователь* и множество ассоциированных с ним *объектов-процессов*. При этом политика безопасности будет применяться по отношению к одному пользователю, осуществляющему доступ к объектам посредством нескольких процессов. "Канадские критерии" просто конкретизируют отношение доступа пользователя к информации, находящейся в компьютерной системе, с помощью терминов, описывающих реальный механизм осуществления доступа. Соответствие между базовыми понятиями "Канадских критериев" и "Оранжевой" книги показано на рис. 2.8.

2.9.2.2 Теги

При описания критериев конфиденциальности и целостности (произвольного и нормативного управления доступом и целостностью) в "Канадских критериях" для обеспечения максимальной степени абстракции и инвариантности по отношению к политике безопасности и методам ее реализации используется понятие *тег*, обозначающее совокупность атрибутов безопасности, ассоциированных с пользователем, процессом или объектом.

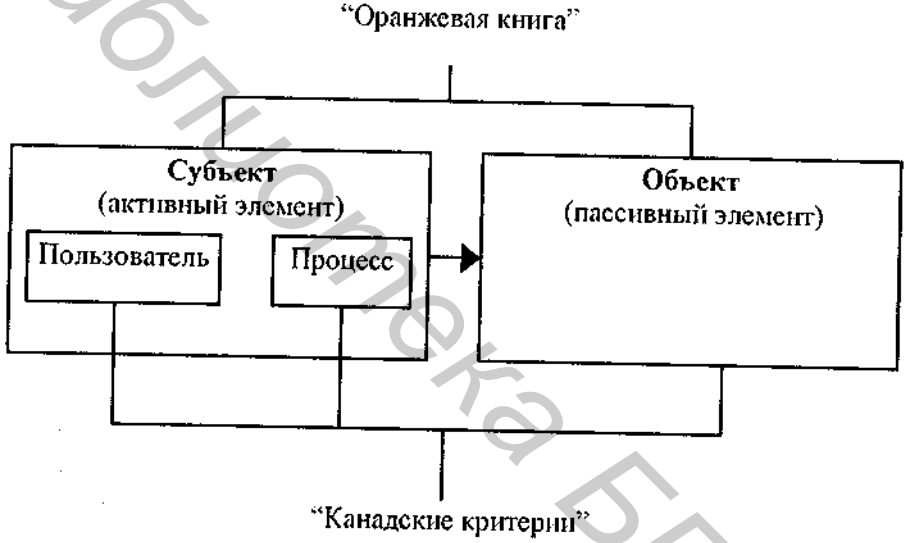


Рис. 2.8. Соответствие понятий "Канадских критериев" и "Оранжевой книги"

В качестве *тега* пользователя, процесса или объекта могут выступать соответствующий уникальный идентификатор, метка безопасности или целостности, криптографический ключ, таблица прав доступа или другие атрибуты в соответствии с реализованной в компьютерной системе политикой безопасности.

2.9.3. Основные положения и структура "Канадских критериев"

Возможность применения "Канадских критериев" к такому широкому кругу различных по назначению систем определяется используемым в них принципом дуального представления требований безопасности в виде функциональных требований к средствам защиты и требований к адекватности их реализации.

Функциональные критерии представляют собой частные метрики, предназначенные для определения показателей эффективности средств защиты в виде уровня их возможностей по отражению угроз соответствующего типа. Функциональные критерии разделяются на четыре группы: критерии конфиденциальности, целостности, работоспособности и аудита.

Каждая из этих групп (кроме аудита) отражает функциональные возможности системы по отражению соответствующего класса угроз. Угрозы несанкционированного доступа к информации предотвращаются с помощью средств, требования к которым содержатся в разделе критериев конфиденциальности. Угрозам несанкционированного изменения информации или ее искажения противостоят средства защиты, функциональные требования к которым задаются критериями целостности. Требования к средствам, обеспечивающим защиту от угроз работоспособности, описаны в разделе критериев работоспособности. Угрозы, направленные на фальсификацию протоколов и манипуляции с внутрисистемной информацией, предотвращаются средствами аудита, требования к которым содержатся в одноименном разделе функциональных критериев.

Такая специализация критериев и требований и, соответственно, реализующих эти требования средств защиты, позволяет четко определить стоящие перед ними задачи и разграничить их функции.

Внутри каждой группы критериев определены уровни безопасности, отражающие возможности средств защиты по решению задач данного раздела. Ранжирование по уровням производится на основании мощности используемых методов защиты и класса отражаемых угроз соответствующего типа. Уровни с большим номером обеспечивают более полную функциональность и, соответственно, более высокую степень безопасности.

Таксономия функциональных критериев показана на рис. 2.9, а в табл. 2.4 приведены идентификаторы уровней.

Адекватность реализации определяется тем, насколько точно и последовательно средства, обеспечивающие защиту, реализуют принятую в компьютерной системе политику безопасности. Согласно "Канадским критериям" политика безопасности представляет собой множество правил, регламентирующих обработку, хранение и использование информации. Критерии адекватности рассматриваются без разделения на подгруппы и определяют требования к процессу проектирования и разработки компьютерной системы.

Таблица 2.4. Идентификаторы уровней "Канадских критериев"

Идентификатор	Наименование	Уровни
	Критерии конфиденциальности	
CC	Контроль скрытых каналов	CC-0—CC-3
CD	Произвольное управление доступом	CD-0—CD-4
CM	Нормативное управление доступом	CM-0—CM-4
CR	Повторное использование объектов	CR-0—CR-1
	Критерии целостности	
IB	Домены целостности	IB-0—IB-2
ID	Произвольное управление целостностью	ID-0—ID-4
IM	Нормативное управление целостностью	IM-0—IM-4
IP	Физическая целостность	IP-0—IP-4
IR	Возможность осуществления отката	IR-0—IR-2
IS	Разделение ролей	IS-0—IS-3
IT	Самотестирование	IT-0—IT-3
	Критерии работоспособности	
AC	Контроль за распределением ресурсов	AC-0—AC-3
AF	Устойчивость к отказам и сбоям	AF-0—AF-3
AR	Живучесть	AR-0—AR-3
AY	Восстановление	AY-0—AY-3
	Критерии аудита	
WA	Регистрация и учет событий в системе	WA-0—WA-3
WI	Идентификация и аутентификация	WI-0—WI-3
WT	Прямое взаимодействие с TCB	WT-0—WT-3
T	**Адекватность**	T-0—T-7

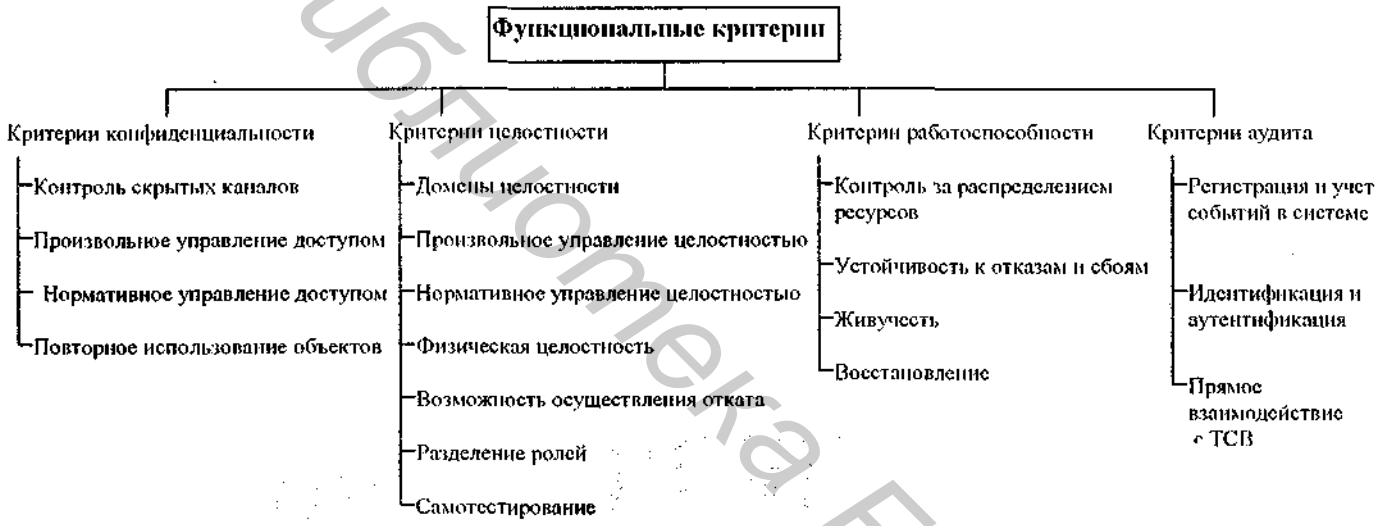


Рис 2.9. Таксономия функциональных критериев "Канадских критериев".

Уровень адекватность присваивается всей системе в целом, причем более высокий уровень означает более полную и корректную реализацию политики безопасности. Таксономия критериев адекватности показана на рис. 2.10. Критерии адекватности отражают уровень корректности реализации политики безопасности и охватывают все стадии проектирования, разработки и эксплуатации компьютерной системы. За некоторым исключением (контроль скрытых каналов) взаимосвязь между функциональными требованиями средствам защиты и требованиями адекватности реализации политики безопасности отсутствует.

Таким образом "Канадские критерии" определяют степень безопасность компьютерной системы как совокупность функциональных возможностей используемых средств защиты, характеризующуюся частными показателями обеспечиваемого уровня безопасности, и одного обобщенного параметра — уровня адекватности реализации политики безопасности.

В состав приложений к "Канадским критериям" входит подробное описание преложенной в них концепции обеспечения безопасности информации, а также руководства по применению функциональных критериев и критериев адекватности реализации. Присутствует приложение, включающее набор стандартных профилей защиты, содержащих типовые наборы требований к компьютерным системам, применяющимся в государственных учреждениях. Этот подход имеет много общего с концепцией профилей защиты, предлагаемой в "Федеральных критериях"(п. 2.8). Приложение II содержит ранжированный перечень функциональных критериев и критериев адекватности "Канадских критериев".

2.9.4. Выводы

"Канадские критерии оценки безопасности компьютерных систем" явились первым стандартом информационной безопасности, в котором на уровне структуры документа функциональные требования к средствам защиты отделены от требований адекватности и качества реализации политики безопасности. Функциональные требования к средствам защиты четко структурированы и описывают все аспекты функционирования ТСВ. Требования к адекватности реализации политики безопасности, впервые появившиеся в виде отдельного раздела, позволяют определить степень доверия к средствам обеспечения безопасности.

Впервые столько внимания уделено взаимному соответствию и взаимодействию всех систем средств обеспечения безопасности. Наиболее прогрессивными являются требования доказательств корректности реализации функциональных требований и их формального соответствия политике и модели безопасности.

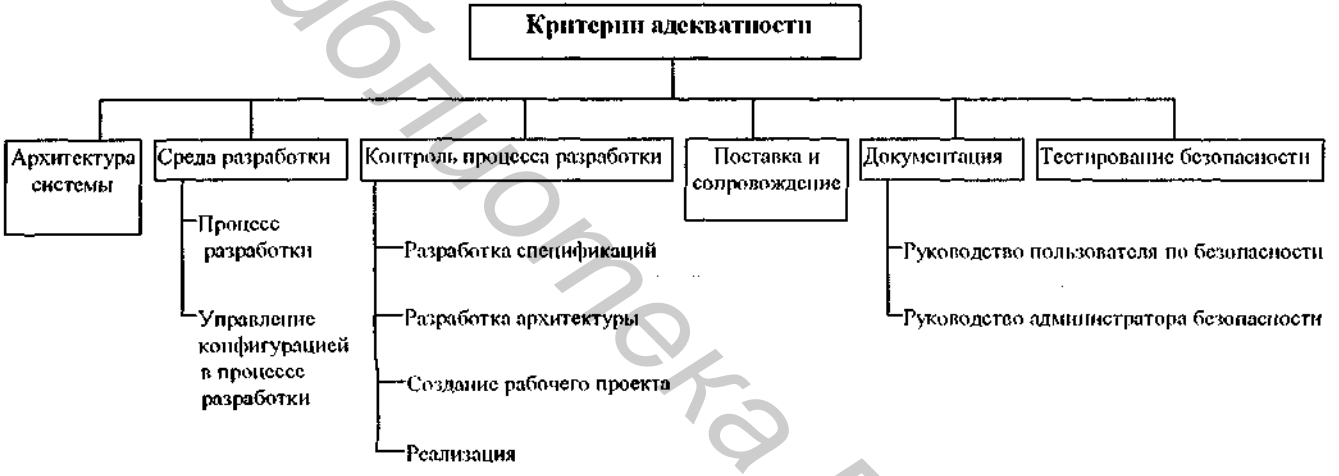


Рис. 2.10. Таксономия критериев адекватности реализации политики безопасности "Канадских критериев".

В "Канадских критериях", как и в "Федеральных критериях", отвергается подход к оценке уровня безопасности с помощью универсальной шкалы и используется независимое ранжирование требований по каждому разделу, образующее множество частных критериев, характеризующих работу подсистем обеспечения безопасности. Кроме того, уровень адекватности реализации политики безопасности характеризует качество всей системы в целом. Однако, по сравнению с "Федеральными критериями", требования к технологии разработки выражены слабо и недостаточно конкретизированы в части используемых методов и средств.

"Канадские критерии оценки безопасности компьютерных систем" представляют собой хорошо сбалансированный конгломерат "Оранжевой книги" и "Федеральных критериев", усиленный требованиями адекватности реализации политики безопасности, и наравне с другими стандартами послужили основой для разработки "Единых критериев безопасности информационных технологий" (п. 2.10).

2.10 Единые критерии безопасности информационных технологий

2.10.1. Цель разработки

"Единые критерии безопасности информационных технологий" (Common Criteria for Information Technology Security Evaluation, далее просто "Единые критерии")[19] являются результатом совместных усилий авторов "Европейских критериев безопасности информационных технологий", "Федеральных критериев безопасности информационных технологий" и "Канадских критериев безопасности компьютерных систем", направленных на объединение основных положений этих документов и создание единого международного стандарта безопасности информационных технологий. Работа над этим самым масштабным в истории стандартов информационной безопасности проектом началась в июне 1993 года с целью преодоления концептуальных и технических различий между указанными документами, их согласования и создания единого международного стандарта. Версия 2.1 этого стандарта утверждена Международной организацией по стандартизации (ISO) в 1999 в качестве международного стандарта информационной безопасности ISO/IEC 15408.

Первая версия "Единых критериев" была опубликована 31 января 1996 г. Разработчиками документа выступили Национальный институт стандартов и технологий и Агентство национальной безопасности США, а также соответствующие организации Великобритании, Канады, Франции

и Нидерландов. Вторая версия вышла в мае 1998, причем она отличается от первоначальной довольно существенными исправлениями и дополнениями. Данный обзор базируется на версии 2.1, отличающейся незначительными исправлениями, сделанными в ходе утверждения стандарта в комитетах ISO.

"Единые критерии" сохраняют совместимость с существующими стандартами и развивают их путем введения новых концепций, соответствующих современному уровню развития информационных технологий и интеграции национальных информационных систем в единое мировое информационное пространство. Этот документ разработан на основе достижений многочисленных исследований в области безопасности информационных технологий 90-х годов и на результатах анализа опыта применения положенных в его основу стандартов. "Единые критерии" оперируют уже знакомым нам по Федеральным критериям понятием "продукт информационных технологий", или ИТ-продукт, и используют предложенную в них концепцию Профиля защиты.

"Единые критерии" разрабатывались в расчете на то, чтобы удовлетворить запросы трех группы специалистов, в равной степени являющихся пользователями этого документа: производителей и потребителей продуктов информационных технологий, а также экспертов по квалификации уровня их безопасности (см. п. 2.2).

Потребители рассматривают квалификацию уровня безопасности ИТ-продукта как метод определения соответствия ИТ-продукта их запросам. Обычно эти запросы составляются на основании результатов проведенного анализа рисков и выбранной политики безопасности. "Единые критерии" играют существенную роль в процессе формирования запросов потребителей, так как содержат механизмы, позволяющие сформулировать эти запросы в виде стандартизованных требований. Это позволяет потребителям принять обоснованное решение о возможности использования тех или иных продуктов. Наконец, "Единые критерии" предоставляют потребителям механизм Профилей защиты, с помощью которого они могут выразить специфичные для них требования, не заботясь о механизмах их реализации.

Производители должны использовать "Единые критерии" в ходе проектирования и разработки ИТ-продуктов, а также для подготовки к квалификационному анализу и сертификации. Этот документ дает возможность производителям на основании анализа запросов потребителей определить набор требований, которым должен удовлетворять разрабатываемый ими продукт. Производители используют предлагаемую "Едиными критериями" технологию для обоснования своих претензий на то, что поставляемый ими ИТ-продукт успешно противостоит угрозам

безопасности, на основании того, что он удовлетворяет выдвинутым функциональным требованиям и их реализация осуществлена с достаточным уровнем адекватности. Для осуществления этой технологии "Единые критерии" предлагают производителям специальный механизм, названный Проект защиты, дополняющий Профиль защиты и позволяющий соединить описания требований, на которые ориентировался разработчик, и спецификации механизмов реализации этих требований.

Кроме того, производители могут использовать "Единые критерии" для определения границ своей ответственности, а также условий, которые необходимо выполнить для успешного прохождения квалификационного анализа и сертификации созданного ими продукта.

Эксперты по квалификации используют этот документ в качестве основных критериев определения соответствия средств защиты ИТ-продукта требованиям, предъявляемым к нему потребителями и угрозам, действующим в среде его эксплуатации. "Единые критерии" описывают только общую схему проведения квалификационного анализа и сертификации, но не регламентируют процедуру их осуществления. Вопросам методологии квалификационного анализа и сертификации посвящен отдельный документ — "Общая методология оценки безопасности информационных технологий" [20].

Таким образом, "Единые критерии" обеспечивают нормативную поддержку процесса выбора ИТ-продукта, к которому предъявляются требования функционирования в условиях действия определенных угроз, служат руководящим материалом для разработчиков таких систем, а также регламентируют технологию их создания и процедуру оценки обеспечиваемого уровня безопасности.

"Единые критерии" рассматривают информационную безопасность, во-первых, как совокупность конфиденциальности и целостности информации, обрабатываемой ИТ-продуктом, а также доступности ресурсов ВС, и, во-вторых, ставят перед средствами защиты задачу противодействия угрозам, актуальным для среды эксплуатации этого продукта и реализации политики безопасности, принятой в этой среде эксплуатации.

Поэтому в концепцию "Единых критериев" входят все аспекты процесса проектирования, производства и эксплуатации ИТ-продуктов, предназначенных для работы в условиях действия определенных угроз безопасности. Причинно-следственные связи, установленные между базовыми понятиями "Единых критериев", показаны на рис. 2.11. Потребители ИТ-продуктов озабочены наличием угроз безопасности, приводящих к определенным рискам для обрабатываемой информации.

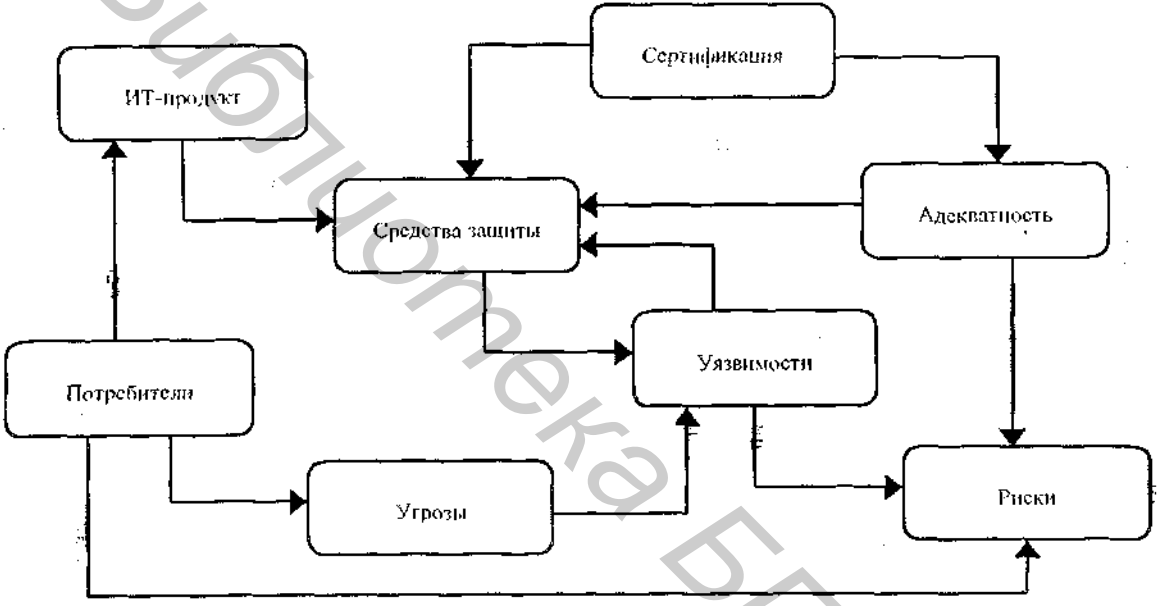


Рис. 2.11 Основные понятия "Единых критериев".

Для противодействия этим угрозам ИТ-продукты должны включать в свой состав средства защиты, противодействующие этим угрозам, и направленные на устранение уязвимостей, однако ошибки в средствах защиты в свою очередь могут приводить к появлению новых уязвимостей. Сертификация средств защиты позволяет подтвердить их адекватность угрозам и рискам.

2.10.2. Основные положения

"Единые критерии" регламентируют все стадии разработки, квалификационного анализа и эксплуатации ИТ-продуктов используя схему уже знакомую нам по "Федеральным критериям" (п. 2.8.2). "Единые критерии" предлагают достаточно сложную и бюрократичную концепцию процесса разработки и квалификационного анализа ИТ-продуктов, требующую от потребителей и производителей огромного количества работы по составлению и оформлению весьма объемных и подробных нормативных документов. Коротко рассмотрим основные положения и разделы этих документов, но сначала введем определения для некоторых базовых понятий "Единых критериев":

Задачи защиты — базовое понятие "Единых критериев", выражающее потребность потребителей ИТ-продукта в противостоянии заданному множеству угроз безопасности или в необходимости реализации политики безопасности.

Профиль защиты — специальный нормативный документ, представляющий собой совокупность Задач защиты, функциональных требований, требований адекватности и их обоснования. Служит руководством для разработчика ИТ-продукта при создании Проекта защиты.

Проект защиты — специальный нормативный документ, представляющий собой совокупность Задач защиты, функциональных требований, требований адекватности, общих спецификаций средств защиты и их обоснования. В ходе квалификационного анализа служит в качестве описания ИТ-продукта.

Согласно "Единым критериям", безопасность информационных технологий может быть достигнута посредством применения предложенной в их авторами технологии разработки, сертификации и эксплуатации ИТ-продуктов. На рис. 2.12. представлена схема технологического цикла применения "Единых критериев".

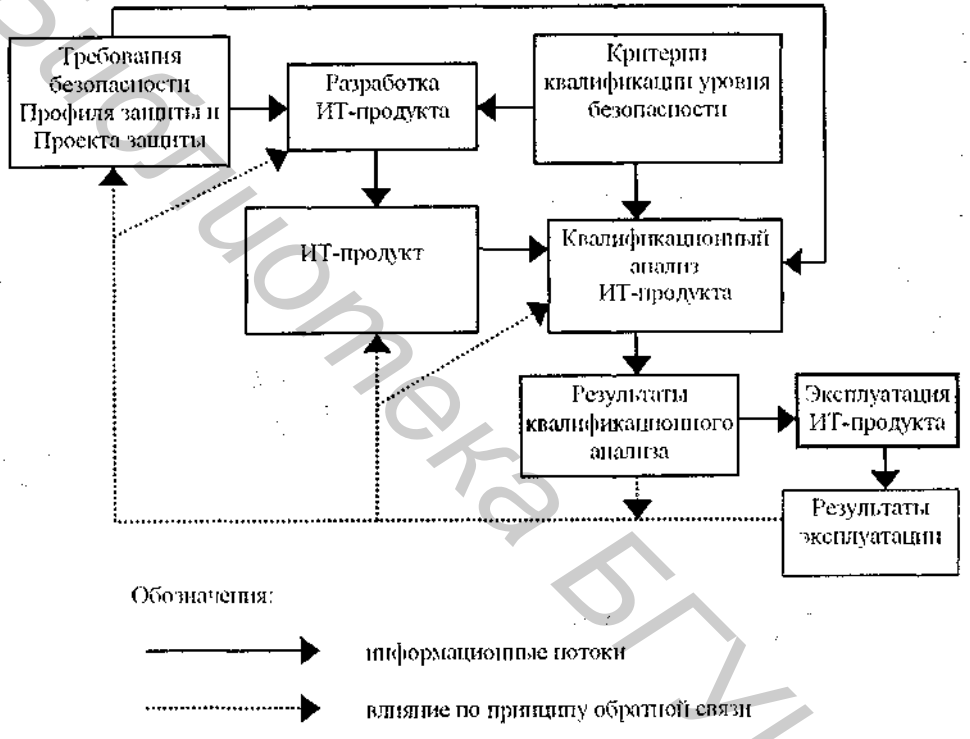


Рис 2.12. Схема процесса разработки и квалификационного анализа ИТ-продукта с точки зрения "Единых критериев".

С точки зрения авторов "Единых критериев" наиболее существенным аспектом требований безопасности, на которые ориентируются разработчики при создании ИТ-продукта, является их соответствие нуждам его потребителей. Только при соблюдении этого условия будет достигнута поставленная цель — обеспечение безопасности информационных технологий в условиях действия угроз безопасности.

"Единые критерии" определяют множество типовых требований, которые в совокупности с механизмом Профилей защиты позволяют потребителям создавать частные наборы требований, отвечающие их нуждам. Разработчики могут использовать Профиль защиты как основу для создания спецификаций своих продуктов. Профиль защиты и спецификации средств защиты составляют Проект защиты, который и представляет ИТ-продукт в ходе квалификационного анализа.

Квалификационный анализ может осуществляться как параллельно с разработкой ИТ-продукта, так и после ее завершения. Для проведения квалификационного анализа разработчик продукта должен представить следующие материалы:

- Профиль защиты, описывающий назначение ИТ-продукта и характеризующий среду его эксплуатации, а также устанавливающий Задачи защиты и требования, которым должен отвечать продукт;
- Проект защиты, включающий спецификации средств защиты, а также обоснование соответствия ИТ-продукта задачам защиты из Профиля защиты и указанным в нем требованиям "Единых критериев";
- различные обоснования и подтверждения свойств и возможностей ИТ-продукта, полученные разработчиком;
- сам ИТ-продукт;
- дополнительные сведения, полученные путем проведения различных независимых экспертиз.

Процесс квалификационного анализа включает три стадии:

1. Анализ Профиля защиты на предмет его полноты, непротиворечивости, реализуемости и возможности использования в качестве набора требований для анализируемого продукта.

2. Анализ Проекта защиты на предмет его соответствия требованиям Профиля защиты, а также полноты, непротиворечивости, реализуемости и возможности использования в качестве эталона при анализе ИТ-продукта.

3. Анализ ИТ-продукта на предмет соответствия Проекту защиты.

Результатом квалификационного анализа является заключение о том, что проанализированный ИТ-продукт соответствует представленному Проекту защиты. Заключение состоит из нескольких отчетов, отличающихся уровнем детализации, и содержащих мнение экспертов по квали-

фикации об ИТ-продукте на основании критериев квалификации — "Единых критериев". Эти отчеты могут использоваться как производителями, так и потребителями ИТ-продукта.

Применение квалификационного анализа и сертификации приводит к повышению качества работы производителей в процессе проектирования и разработки ИТ-продуктов. В продуктах прошедших квалификацию уровня безопасности вероятность появления ошибок и изъянов защиты и уязвимостей существенно меньше, чем в обычных продуктах. Все это позволяет говорить о том, что применение "Единых критериев" оказывают положительное и конструктивное влияние на процесс формирование требований, разработку ИТ-продукта, сам продукт и его эксплуатацию.

Основными документами, описывающими все аспекты безопасности ИТ-продукта, с точки зрения пользователей и разработчиков являются соответственно Профиль защиты и Проект защиты. Рассмотрим структуру и содержание этих документов.

2.10.2.1. Профиль защиты

Профиль защиты определяет требования безопасности к определенной категории ИТ-продуктов, не уточняя методы и средства их реализации. С помощью Профилей защиты потребители формулируют свои требования к производителям.

Структура Профиля защиты "Единых критериев" существенно отличается от структуры документа с тем же названием, обсуждавшегося в разделе, посвященном "Федеральным критериям" (п. 2.8.3). Эта структура представлена на рис. 2.13.

Рассмотрим назначение и содержание разделов Профиля защиты.

Введение содержит информацию, необходимую для поиска Профиля защиты в библиотеке профилей.

Идентификатор Профиля защиты представляет собой уникальное имя, пригодное для его поиска среди подобных ему профилей и обозначения ссылок на него.

Обзор содержания содержит краткую аннотацию Профиля защиты, на основании которой потребитель может сделать вывод о соответствии данного профиля его запросам.

Описание ИТ-продукта содержит его краткую характеристику, функциональное назначение, принципы работы, методы использования и т.д. Эта информация не подлежит анализу и сертификации, но предоставляется экспертам для пояснения требований безопасности и определения их соответствия задачам, решаемым с помощью ИТ-продукта, а также для общего понимания его структуры и принципов работы.

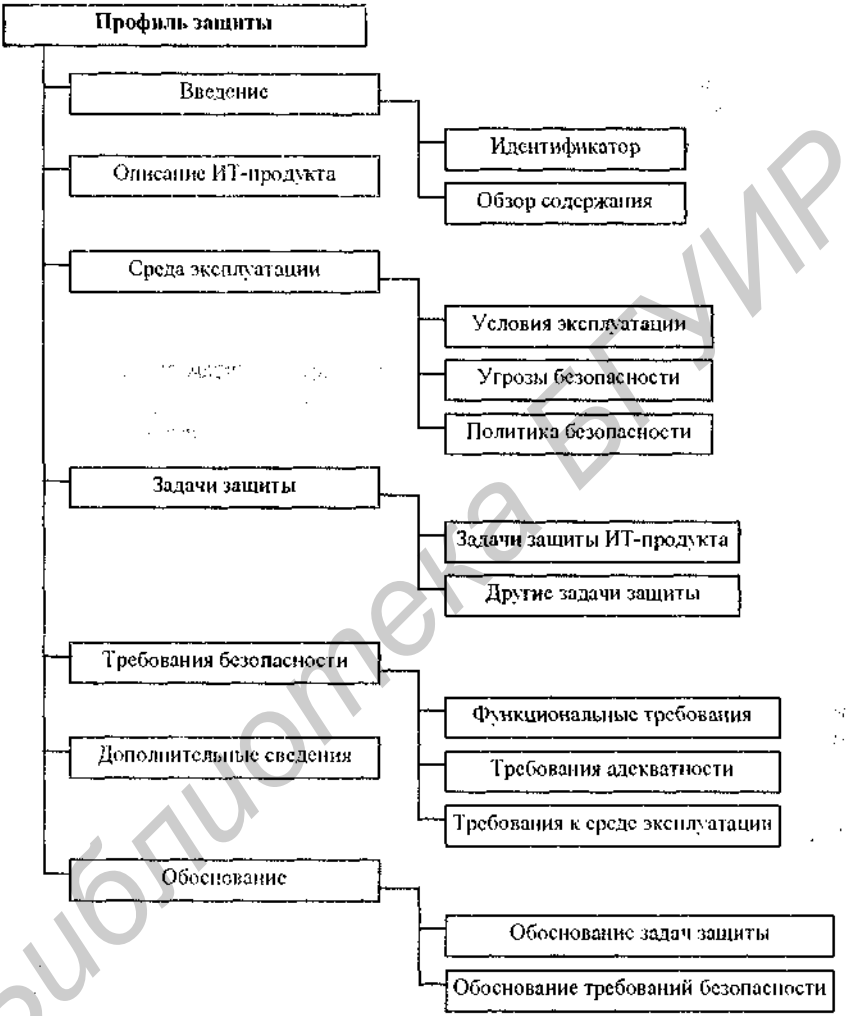


Рис. 2.13 Структура Профиля защиты согласно "Единым критериям"

Среда эксплуатации. Этот раздел содержит описание среды функционирования ИТ-продукта с точки зрения безопасности.

Условия эксплуатации. Описание условий эксплуатации ИТ-продукта должно содержать исчерпывающую характеристику среды его эксплуатации с точки зрения безопасности, в том числе ограничения на условия его применения.

Угрозы безопасности. Описание угроз безопасности, действующих в среде эксплуатации, которым должна противостоять защита ИТ-продукта. Для каждой угрозы должен быть указан ее источник, метод и объект воздействия.

Политика безопасности. Описание политики безопасности должно определять и, при необходимости, объяснять правила политики безопасности, которая должна быть реализована в ИТ-продукте.

Задачи защиты отражают потребности пользователей в противодействии указанным угрозам безопасности и/или реализации политики безопасности.

Задачи защиты ИТ-продукта отражают потребности пользователей в противодействии угрозам безопасности и/или реализации политики безопасности.

Другие задачи защиты отражают необходимость участия средств защиты ИТ-продукта в противодействии угрозам безопасности и/или реализации политики безопасности совместно с другими компонентами информационных технологий.

Требования безопасности. В этом разделе Профиля защиты содержатся требования безопасности, которым должен удовлетворять ИТ-продукт для решения задач защиты.

Раздел *функциональных требований* должен содержать только типовые требования, предусмотренные соответствующими разделами "Единых критериев". Необходимо обеспечить такой уровень детализации требований, который позволяет продемонстрировать их соответствие задачам защиты. Функциональные требования могут предписывать или запрещать использование конкретных методов и средств защиты.

Раздел *требований адекватности* содержит ссылки на типовые требования уровней адекватности "Единых критериев", но допускает и определение дополнительные требований адекватности.

Раздел *требований к среде эксплуатации* является необязательным и может содержать функциональные требования и требования адекватности, которым должны удовлетворять компоненты

информационных технологий, составляющие среду эксплуатации ИТ-продукта. В отличие от предыдущих разделов использование типовых требований "Единых критериев" является желательным, но не обязательным.

Дополнительные сведения — необязательный раздел, содержащий любую дополнительную информацию, которая может быть полезна для проектирования, разработки, квалификационного анализа и сертификации ИТ-продукта.

Обоснование должно демонстрировать, что Профиль защиты содержит полное и связное множество требований, и что удовлетворяющий им ИТ-продукт будет эффективно противостоять угрозам безопасности среды эксплуатации.

Обоснование задач защиты должно демонстрировать, что задачи защиты, предложенные в профиле, соответствуют параметрам среды эксплуатации, и их решение позволит эффективно противостоять угрозам безопасности и реализовать политику безопасности.

Обоснование требований безопасности показывает, что требования безопасности позволяют эффективно решить задачи защиты, поскольку:

- совокупность целей, преследуемых отдельными функциональными требованиями, соответствует установленным задачам защиты;
- требования безопасности являются согласованными, т. е. не противоречат друг другу, а, наоборот, взаимно усиливают;
- выбор требований является оправданным (особенно это относится к дополнительным требованиям, не содержащимся в "Единых критериях");
- выбранный набор функциональных требований и уровень требований адекватности соответствуют Задачам защиты.

Профиль защиты служит отправной точкой для производителя в процессе создания Проекта защиты, который является техническим проектом для разработки ИТ-продукта и представляет его в ходе квалификационного анализа.

2.10.2.1 Проект защиты

Проект защиты содержит требования и задачи защиты ИТ-продукта, а также описывает уровень функциональных возможностей реализованных в нем средств защиты, их обоснование и подтверждение степени их адекватности. Проект защиты, с одной стороны является отправной точкой для разработчика системы, а с другой представляет собой эталон системы в ходе квалификационного анализа. Структура Проекта защиты представлена на рис. 2.14.

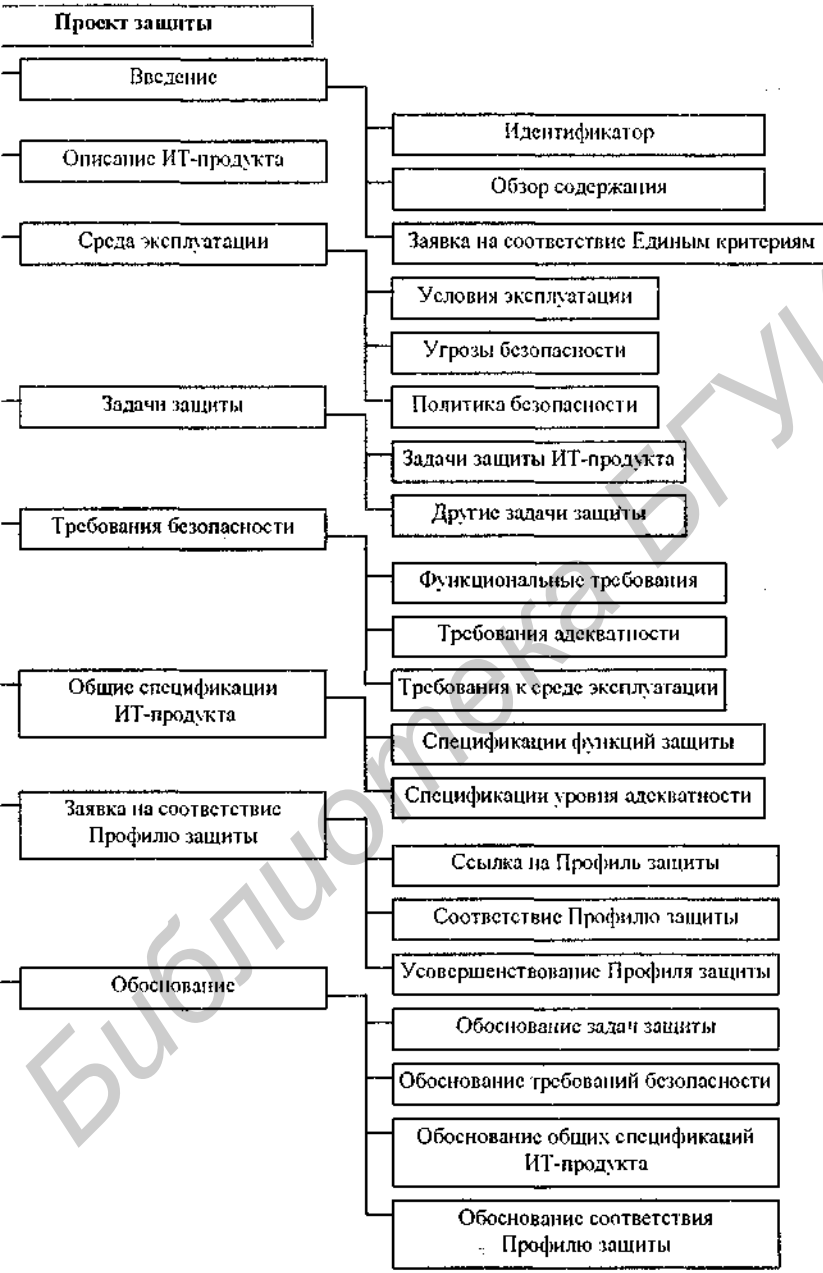


Рис. 2.14. Структура Проекта защиты согласно "Единым критериям".

Многие разделы Проекта защиты совпадают с одноименными разделами Профиля защиты, поэтому рассмотрим только те разделы, которые специфичны для Проекта защиты, а также те, которые претерпели изменения.

Введение содержит информацию, необходимую для идентификации Проекта защиты, определения назначения, а также обзор его содержания.

Идентификатор представляет собой уникальное имя Проекта защиты, необходимое для поиска и идентификации Проекта защиты и соответствующего ему ИТ-продукта.

Обзор содержания представляют собой достаточно подробную аннотацию Проекта защиты, позволяющую потенциальным потребителям определить пригодность ИТ-продукта для решения их задач.

Заявка на соответствие "Единым критериям" содержит описание всех свойств ИТ-продукта, подлежащих квалификационному анализу на основе "Единых критериев".

Раздел *Требований безопасности* Проекта защиты содержит требования безопасности к ИТ-продукту, которыми руководствовался производитель в ходе его разработки. Этот раздел несколько отличается от аналогичного раздела Профиля защиты.

Раздел *функциональных требований* к ИТ-продукту в отличие от соответствующего раздела Профиля защиты допускает использование кроме типовых требований "Единых критериев" других, специфичных для данного продукта и среды его эксплуатации. При описании таких специальных требований необходимо сохранять стиль "Единых критериев" и обеспечивать присущую им степень подробности.

Раздел *требований адекватности* может включать уровни адекватности, не предусмотренные в "Единых критериях". В этом случае описание уровня адекватности должно быть четким, непротиворечивым и обладать степенью подробности, допускающей его использование в ходе квалификационного анализа. При этом желательно использовать стиль и подробность описания уровней адекватности, принятые в "Единых критериях".

Общие спецификации ИТ-продукта описывают механизмы осуществления Задач защиты с помощью определения высокоуровневых спецификаций средств защиты в соответствии с предъявляемыми функциональными требованиями и требованиями адекватности.

Спецификации функций защиты описывают функциональные возможности средств защиты ИТ-продукта, заявленные его производителем как реализующие требования безопасности. Форма пред-

ставления спецификаций должна позволять определять соответствия между функциями защиты и требованиями безопасности

Спецификации уровня адекватности определяют заявленный уровень адекватности защиты ИТ-продукта и его соответствие требованиям адекватности в виде представления параметров технологии проектирования и создания ИТ-продукта. Эти параметры должны быть представлены в форме, позволяющей определить их соответствие требованиям адекватности.

Заявка на соответствие Профилю защиты. Проект защиты претендует на удовлетворение требований одного или нескольких Профилей защиты. Этот необязательный раздел содержит материалы, необходимые для подтверждения заявки. Для каждого Профиля защиты, на реализацию которого претендует Проект защиты, этот раздел должен содержать следующую информацию:

Ссылка на Профиль защиты однозначно идентифицирует Профиль защиты, на реализацию которого претендует Проект защиты, с указанием случаев, в которых обеспечиваемый уровень защиты превосходит требования Профиля. Корректная реализация Профиля защиты подразумевает корректную реализацию всех его требований без исключения.

Соответствие Профилю защиты определяет возможности ИТ-продукта, которые реализуют задачи защиты и требования, содержащиеся в Профиле защиты.

Усовершенствования Профиля защиты отражают возможности ИТ-продукта, которые выходят за рамки задач защиты и требований, установленных в Профиле защиты.

Обоснование должно показывать, что Проект защиты содержит полное и связное множество требований, что реализующий его ИТ-продукт будет эффективно противостоять угрозам безопасности, действующим в среде эксплуатации, и что общие спецификации функций защиты соответствуют требованиям безопасности. Кроме того, обоснование содержит подтверждение соответствия Профилю защиты. Обоснование Проекта защиты включает следующие разделы:

Обоснование задач защиты должно демонстрировать, что задачи защиты, предложенные в Проекте защиты, соответствуют свойствам среды эксплуатации, и что их решение позволит эффективно противодействовать угрозам безопасности и реализовать требуемую политику безопасности.

Обоснование требований безопасности показывает, что выполнение этих требований позволяет решить задачи защиты, т. к.:

- совокупность функциональных требований и требований адекватности, а также условий эксплуатации ИТ-продукта соответствуют задачам защиты;
- все требования безопасности являются непротиворечивыми и взаимно усиливают друг друга;
- выбор требований является оправданным;
- уровень функциональных возможностей средств защиты соответствует задачам защиты.

Обоснование общих спецификаций ИТ-продукта должно демонстрировать, что средства защиты и методы обеспечения их адекватности соответствуют предъявляемым требованиям поскольку:
- совокупность средств защиты удовлетворяет функциональным требованиям;
- требуемый уровень безопасности и надежности защиты обеспечивается предложенными средствами;
- меры, направленные на обеспечение адекватности реализации функциональных требований, соответствуют предъявленным требованиям адекватности.

Обоснование соответствия Профилю защиты показывает, что требования Проекта защиты поддерживают все требования Профиля защиты. Для этого должно быть показано, что:
- все усовершенствования задач защиты по сравнению с Профилем защиты осуществлены корректно и в направлении их развития и конкретизации;
- все усовершенствования требований безопасности по сравнению с Профилем защиты осуществлены корректно и в направлении их развития и конкретизации;
- все задачи защиты Профиля защиты успешно решены и все требования Профиля защиты удовлетворены;
- никакие дополнительно введенные в Проект защиты специальные задачи защиты и требования безопасности не противоречат Профилю защиты.

Как видно из приведенной структуры Профиля и Проекта защиты и краткого обзора их содержания, эти документы практически исчерпывающим образом регламентируют взаимодействие потребителей, производителей и экспертов по квалификации в процессе создания ИТ-продукта. Фактически, положения этих документов определяют технологию разработки защищенных систем.

Самым важным элементом этой технологии являются требования безопасности. Поскольку "Единые критерии" обобщают все предшествующие решения в этой области, рассмотрим их более подробно.

2.10.4. Требования безопасност.

"Единые критерии" разделяют требования безопасности на две категории: функциональные требования и требования адекватности.

Функциональные требования регламентируют функционирование обеспечивающих безопасность компонентов ИТ-продукта и определяют возможности средств защиты.

Адекватность представляет собой характеристику ИТ-продукта, которая показывает, насколько эффективно обеспечивается заявленный уровень безопасности, а также степень корректности реализации средств защиты. Адекватность определяется технологиями, используемыми в процессе проектирования, создания и эксплуатации ИТ-продукта. Поэтому требования адекватности регламентируют технологию и процесс создания ИТ-продукта, а также необходимость проведения анализа слабых мест защиты.

2.10.4.1. Функциональные требования

Как и все остальные положения "Единых критериев" функциональные требования представлены в виде хорошо проработанной формальной структуры. Набор функциональных требований обобщает все существующие стандарты информационной безопасности и впечатляет своей всеобъемлющей полнотой и подробнейшей детализацией.

Функциональные требования разбиты на классы, которые, в свою очередь состоят из разделов. Структура функциональных требований приведена на рис. 2.15.

Рассмотрим назначение некоторых элементов структуры описания функциональных требований, необходимых для проведения их анализа:

Название и обозначение раздела. Каждый раздел имеет свое уникальное название и семисимвольный идентификатор, состоящий из префикса — трехбуквенного идентификатора класса, знака подчеркивания и трехбуквенного обозначения раздела. Используется для ссылок на раздел.

Ранжирование требований. "Единые критерии" развивают впервые предложенный в "Федеральных критериях" подход, связанный с отказом от единой шкалы ранжирования функциональных требований и введением множества частных шкал. Специфика "Единых критериев" состоит в том, что шкалы, по которым ранжируются требования, являются лишь частично упорядоченными.

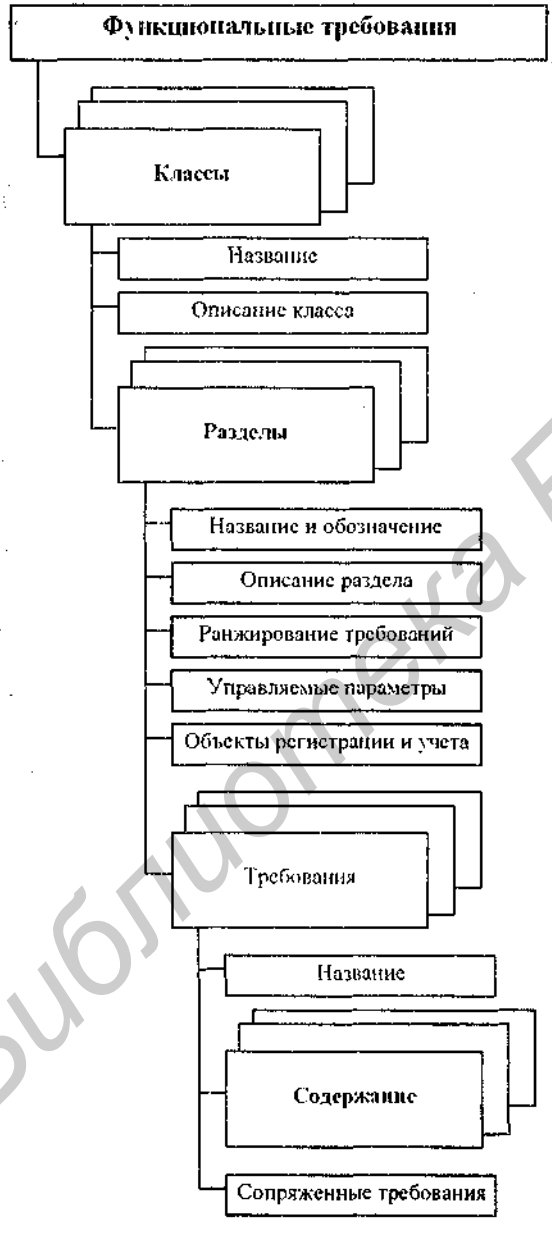


Рис. 2.15 Структура функциональных требований "Единых критериев"

Это нововведение требует подробного рассмотрения, т. к. открывает новые возможности для пользователей и разработчиков при составлении набора требований Профиля защиты и Проекта защиты.

Согласно "Единым критериям" набор ранжированных требований представляет собой иерархическую структуру, в которой усиление требований происходит монотонно, но при этом не является линейным упорядоченным списком. Требования, стоящие в иерархии выше других включают в себя нижестоящие требования. Это означает, что в Профиле защиты имеет смысл использовать только одно их таких требований. Требования, не связанные отношениями иерархичности являются независимыми и могут быть использованы одновременно. Таким образом, структура, отображающая иерархию требований, имеет вид направленного графа. Усиление требований происходит при движении по его ребрам от корневой вершины. Вершины графа, между которыми нельзя проложить путь, следуя по направлению ребер, обозначают несравнимые между собой требования. Строго говоря, такой граф канонически индуцирует отношение частичного порядка на множестве требований. В качестве примера рассмотрим ранжирование требований защиты информации при передаче по внутренним каналам и к использованию псевдонимов (рис. 2.16).

Реализация требований защиты информации при передаче по внутренним каналам осуществляется по двум направлениям — обеспечение безопасности при передаче и контроль целостности информации. Для каждого направления существует две степени реализации требований, в зависимости от того учитываются атрибуты безопасности передаваемой информации, или нет. Требования, находящиеся на разных ветках являются независимыми и дополняют друг друга.

Иерархия требований к использованию псевдонимов представляет собой более сложную структуру. Минимальный уровень соответствия этому требованию обеспечивается посредством использования псевдонимов, скрывающих личности пользователей, работающих с системой. Существуют два независимых направления усиления этого требования — наличие механизма, позволяющего в случае необходимости определить пользователя, скрывающегося за псевдонимом, и использование при назначении псевдонимов некоторых правил, позволяющих установить личность пользователя. Эти два требования находятся на разных ветвях иерархии и несопоставимы между собой.

Управляемые параметры. В этом пункте могут быть перечислены параметры, путем настройки которых должно осуществляться управление средствами защиты, реализующими требования данного раздела.

Рис. 2.16. Пример иерархического ранжирования функциональных требований "Единых критериев".

Объекты регистрации и учета. В этом пункте требований перечислены операции и события, которые должны являться объектами регистрации и учета.

Само описание функционального требования строится по следующей схеме:

1. Уникальное название требования, которое используется для ссылок на него в Профиле и Проекте защиты.

2. Содержание требования. "Единые критерии" разрешают использовать требования только без изменений, что обеспечивает их стандартизацию.

3. Сопряженные требования. Необязательный пункт, содержащий список требований различных разделов и классов, выполнение которых рассматривается как необходимое предварительное условие для реализации данного требования.

Таксономия классов функциональных требований "Единых критериев" показана на рис. 2.17.

Набор функциональных требований "Единых критериев" отличается от других стандартов, во-первых, своей всеобъемлющей полнотой (135 требований), и, во-вторых, многоуровневым подходом к обеспечению безопасности. Впервые отдельные классы требований направлены на обеспечение надежности работы самих средств защиты, контролю за эксплуатацией системы, обеспечению конфиденциальности сеансов доступа к системе и организации обмена информацией. Особо стоит отметить появление в последней версии критериев требований к криптографическим средствам и управлению безопасностью. Таксономия функциональных требований для всех классов "Единых критериев" показана на рис. 2.18, 2.19 и 2.20. Приложение III содержит перечень требований "Единых критериев" и их ранжирование.

Требования конфиденциальности, целостности и управления доступом объединены в один класс "защита информации", что выглядит вполне логичным и полностью соответствует их назначению. Следует отметить разделение требований к политике управления доступом (произвольные модели управления доступом) от требований к управлению информационными потоками (нормативные модели управления доступом), а также отделение требований к политике безопасности от требований к средствам ее реализации.

Класс требований к надежности работы средств защиты является самым объемным, что определяется высокой степенью детализации включенных в него требований к методам и средствам обеспечения нормального функционирования средств защиты.

Рис. 2.17. Таксономия классов функциональных требований "Единых критериев"

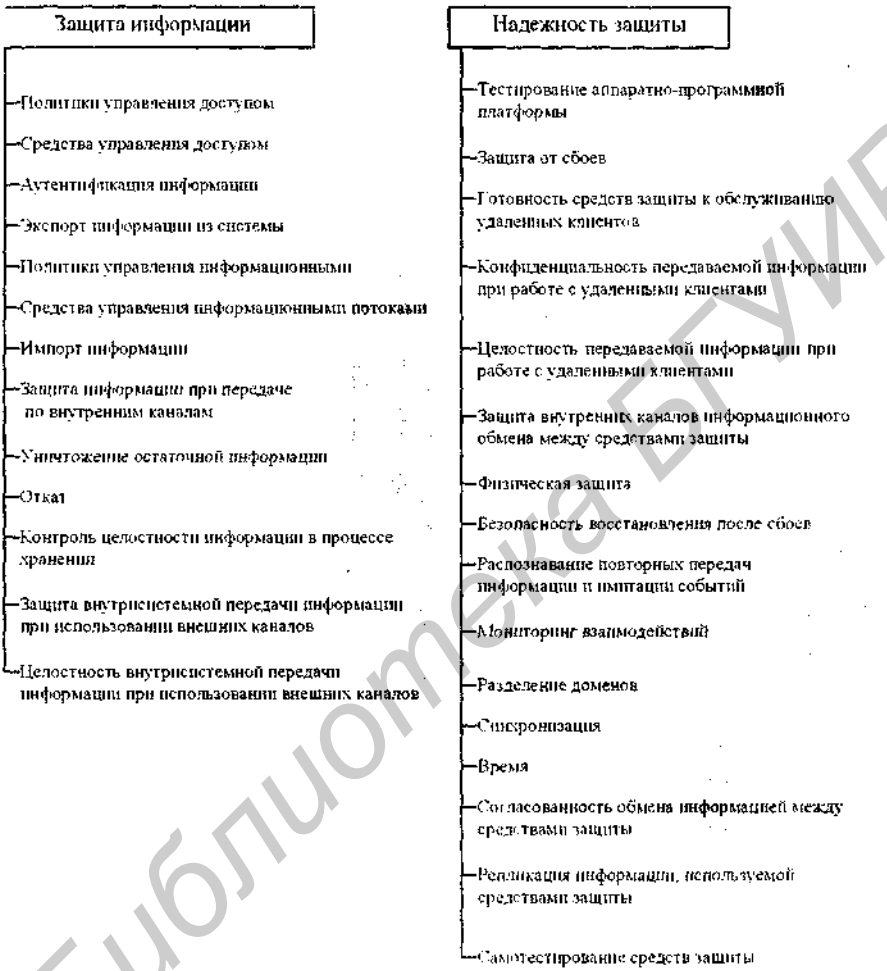


Рис. 2.18. Таксономия функциональных требований классов *Защита информации* и *Надежность защиты* "Единых критериев".

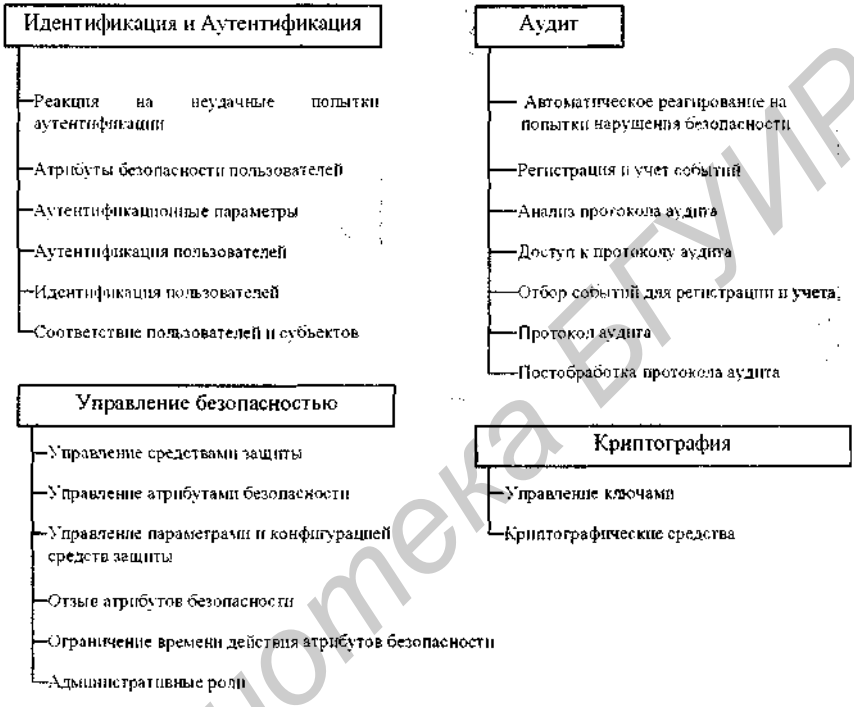


Рис. 2.19. Таксономия функциональных требований **классов** *Идентификация и Аутентификация*, *Аудит*, *Управление безопасностью* **и** *Криптография* "Единых критериев"

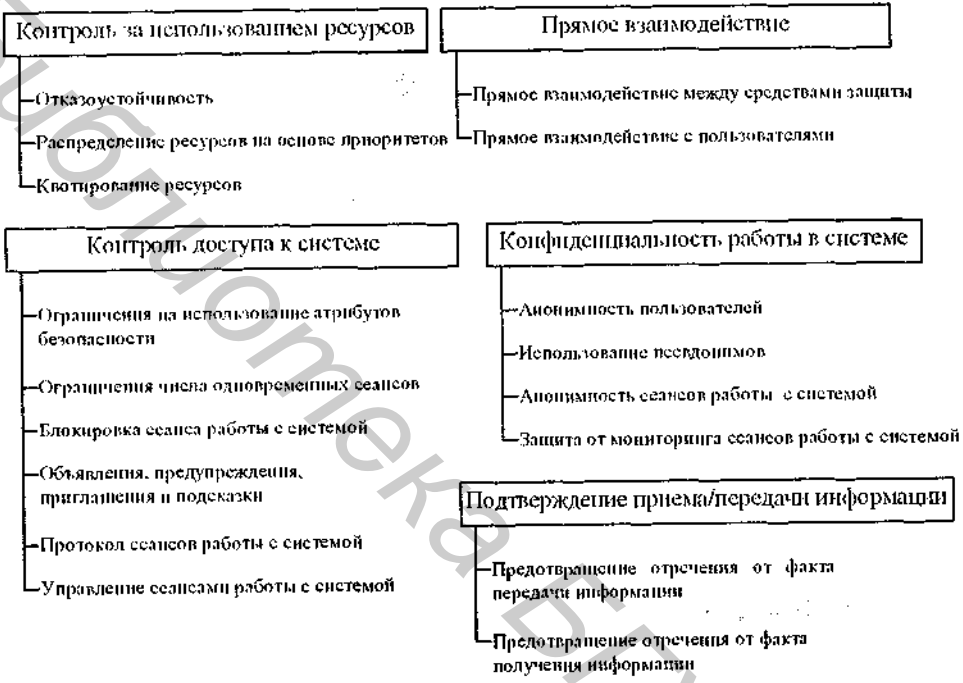


Рис. 2.20. Таксономия функциональных требований классов *Контроль за использованием ресурсов, Прямое взаимодействие, Контроль доступа к системе, Конфиденциальность работы в системе* и *Подтверждение приема/передачи информации* "Единых критериев".

2.10.4.2. Требования адекватности

Требования адекватности "Единых критериев" жестко структурированы и регламентируют все этапы проектирования, создания и эксплуатации ИТ-продукта с точки зрения обеспечения надежности работы средств защиты и их адекватности функциональным требованиям, задачам защиты и угрозам безопасности, действующим в среде эксплуатации ИТ-продукта.

Таксономия требований адекватности "Единых критериев" представлена на рис. 2.21.

Ранжирование требований адекватности представлено в виде упорядоченных списков. Критерии адекватности используются в ходе квалификационного анализа ИТ-продукта для определения степени корректности реализации функциональных требований и назначения ИТ-продукту соответствующего уровня адекватности. Для этого "Единые критерии" предлагают семь стандартных уровней адекватности, жесткость требований адекватности возрастает от первого уровня к седьмому. Каждый уровень характеризуется набором требований адекватности, регламентирующих применение различных методов и технологий разработки, тестирования, контроля и верификации, ИТ-продукта:

Уровень 1. Функциональное тестирование.

Уровень 2. Структурное тестирование.

Уровень 3. Методическое тестирование и проверка.

Уровень 4. Методическая разработка, тестирование и анализ.

Уровень 5. Полуформальные методы разработки и тестирование.

Уровень 6. Полуформальные методы верификации разработки и тестирование.

Уровень 7. Формальные методы верификации разработки и тестирование.

Кратко охарактеризуем каждый уровень адекватности с точки зрения области его применения, условий разработки, уровня проводимого анализа, состава материалов, подтверждающих результаты анализа, и проследим усиление требований от уровня к уровню.

Уровень 1. Функциональное тестирование предназначено для тех случаев, когда угрозам безопасности не придается большого значения. Предлагается использовать его в тех ситуациях, когда все что требуется — это независимая гарантия того, что в состав продукта входят средства защиты персональной или подобной информации, реализованные в соответствии с указанными требованиями.

Рис. 2.21. Таксономия требований адекватности "Единых критериев".

Анализ ИТ-продукта на соответствие данному уровню адекватности ограничивается исследованием функций защиты и проверкой функциональных спецификаций, интерфейсов и документации.

Результаты анализа подтверждаются независимым тестированием средств защиты ИТ-продукта на соответствие спецификациям и документации.

Сертификация ИТ-продукта на данный уровень адекватности является подтверждением соответствия свойств ИТ-продукта его документации и спецификациям, а также наличия работоспособной защиты против угроз безопасности.

Уровень 2. Структурное тестирование используется тогда, когда пользователи или разработчики согласны удовлетвориться низкой или умеренной степенью независимого подтверждения адекватности обеспечиваемого уровня безопасности. Особо рекомендуется применять требования данного уровня для унаследованных систем, уже находящихся в эксплуатации.

Разработка продукта в соответствии с требованиями данного уровня не требует от производителя никаких дополнительных затрат, по сравнению с разработкой обычных коммерческих или промышленных продуктов, кроме предоставления результатов тестирования.

Для этого уровня анализ должен проводится не в отношении функциональных спецификаций, интерфейсов и документации, но и для архитектуры защиты ИТ-продукта.

Кроме независимого тестирования средств защиты ИТ-продукта результаты анализа подтверждаются протоколами тестирования функциональных спецификаций, представленными разработчиком, а также независимым выборочным контролем результатов этих испытаний и глубины проведенного тестирования, и независимым подтверждением поиска разработчиком явных уязвимостей. Кроме того, для этого уровня требуется наличие документированного состава конфигурации продукта и доказательство безопасности процедуры поставки.

Данный уровень расширяет требования предыдущего за счет включения в материалы, подтверждающие анализ, результатов тестов, проведенных разработчиком ИТ-продукта, необходимостью осуществления анализа уязвимостей и независимого тестирования с использованием более детальных спецификаций

Уровень 3. Методическое тестирование применяется тогда, когда потребителям или пользователям требуются умеренная степень независимого подтверждения свойств ИТ-продукта, а также полное и последовательное исследование свойств продукта и контроль в процессе создания,

но без проведения дорогостоящего обратного проектирования (reengineering).

Этот уровень позволяет получить максимальной степени адекватности, не требующий никаких изменений в обычную процедуру разработки, поскольку все регламентированные им меры, направленные на обеспечение адекватности, применяются на этапе проектирования.

Для этого уровня проводятся те же виды анализа, что и для уровня 2, но в дополнение к материалам тестирования спецификаций функций защиты от разработчика требуется предоставление результатов тестирования архитектуры защиты ИТ-продукта. Требования к процессу создания продукта дополняются использованием средств управления конфигурацией проекта.

Уровень расширяет требования предыдущего за счет более полного тестирования функций защиты и средств их реализации, а также применением мер, дающих определенную уверенность в том, что ИТ-продукта не был подменен в процессе разработки.

Уровень 4. Уровень методической разработки, тестирования и анализа применим в тех обстоятельствах, когда разработчики или пользователи требуют умеренную или высокую степень независимого подтверждения адекватности защиты ИТ-продукта и готовы нести определенные дополнительные технические затраты.

Этот уровень, несмотря на достаточно сильные требования, не требует от разработчика специальных знаний в области разработки защищенных систем и применения специальных методов и технологий отличающихся от общепринятых. Это наивысший уровень адекватности, на который можно рассчитывать без дополнительных экономических затрат.

В отличие от предыдущих уровней для сертификации продукта на четвертый уровень адекватности анализу подвергаются все интерфейсы без исключения, все частные спецификации, а также детали реализации средств защиты. Кроме того должна быть предоставлена неформальная модель политики безопасности.

В дополнение к предыдущему уровню результаты анализа подтверждаются независимым исследованием уязвимостей средств защиты, демонстрирующим стойкость системы против слабых атак. Требования к процессу создания продукта расширяются дополнительными требованиями применения автоматизированных средств управления конфигурацией.

Данный уровень отличается от предыдущего ужесточением требований к процессу проектирования и разработки, а также усилением мер, гарантирующих, что ИТ-продукт не был подменен в процессе создания.

Уровень 5. Полуформальные методы разработки и тестирование рекомендуется применять в том случае, когда разработчики или пользова-

тели требуют высокой степени независимого подтверждения адекватности средств защиты, а также строгого применения определенных технологий разработки ИТ-продукта, но без чрезмерных затрат.

Этот уровень требует от разработчика применения определенных технологий и методов разработки, однако, их использование может ограничиваться проектированием и реализацией средств защиты.

В отличие от предыдущих уровней анализу подвергаются все средства защиты без исключения. Кроме того, требуется наличие формальной модели политики безопасности и полуформальное представление функциональных спецификаций и архитектуры защиты, а также полуформальная демонстрация их взаимного соответствия. Архитектура ИТ-продукта должна отвечать требованиям модульности.

В дополнение к предыдущим уровням для подтверждения результатов анализа требуется тестирование разработчиком частных спецификаций, а анализ уязвимостей должен демонстрировать стойкость против атак умеренной силы. Кроме того, требуется независимая проверка проведенного разработчиком анализа скрытых каналов.

Требования к процессу разработки дополняются необходимостью расширения состава конфигурации продукта, управляемой с помощью автоматических средств.

Таким образом, этот уровень усиливает требования предыдущего в части полуформального описания процесса проектирования и реализации, более структурированной архитектуры защиты, более тщательного анализа скрытых каналов, более полного контроля в процессе разработки.

Уровень 6. Полуформальные методы верификации, разработки и тестирование применяются при разработке защищенных продуктов, предназначенных для использования в ситуациях с высокой степенью риска, где ценность защищаемой информации оправдывает дополнительные затраты.

Данный уровень требует строгого и последовательного применения определенных методов проектирования и разработки, позволяющих обеспечивать адекватность защиты при эксплуатации в условиях повышенного риска.

Требования этот уровня дополняют предыдущие необходимостью структурного описания реализации продукта и полуформального представления частных спецификаций, а также требованием многоуровневой архитектуры.

Для подтверждения результатов анализа кроме мер, предусмотренных уровнем 5, анализ уязвимостей должен демонстрировать стойкость системы против сильных атак, и должны быть получены независимые

подтверждения проведения разработчиком систематического поиска скрытых каналов.

Требования к процессу разработки расширяются необходимостью структурирования этого процесса и полной автоматизации управления конфигурацией проекта.

Уровень расширяет требования предыдущего применением более глубокого анализа, структуризацией представления ИТ-продукта, многоуровневой архитектурой, усилением анализа уязвимостей, применением систематического поиска скрытых каналов, а также усовершенствованным управлением конфигурацией и среды разработки.

Уровень 7. Формальные методы верификации, разработки и тестирование могут быть использованы для разработки защищенных продуктов, предназначенных для использования в ситуациях с исключительно высокой степенью риска, и/или там, где ценность защищаемых объектов оправдывает высокие дополнительные затраты. Практическое применение этого уровня в настоящее время ограничено компактными продуктами, в которых сконцентрированы средства защиты, и которые легко поддаются формальному анализу.

В отличие от предыдущих уровней требуется формальное представление функциональных спецификаций и архитектуры защиты, а также формальная и полуформальная демонстрация соответствия между ними. Архитектура системы должна быть не только модульной, но и простой и ясной.

В дополнение к предыдущим уровням результаты анализа подтверждаются тестированием формы реализации, а также обоснованным независимым подтверждением всех результатов проведенных разработчиком испытаний.

Таким образом, этот уровень усиливает требования предыдущего за счет более последовательного анализа с использованием формального описания системы на различных уровнях представления и формального доказательства взаимного соответствия этих описаний, а также всеобъемлющего тестирования.

Приложение III содержит таблицу, показывающую распределение требований адекватности по рассмотренным уровням.

2.10.7. Выводы

"Единые критерии безопасности информационных технологий" представляют собой результат обобщения всех достижений последних лет в области информационной безопасности. Впервые документ такого уров-

ня содержит разделы, адресованные потребителям, производителям и экспертам по квалификации ИТ-продуктов.

Предложенные "Едиными критериями" механизмы Профиля защиты и Проекта защиты позволяют потребителям и производителям в полной мере выразить свой взгляд на требования безопасности и задачи защиты, а с другой стороны дают возможность экспертам по квалификации проанализировать взаимное соответствие между требованиями, нуждами потребителей, задачами защиты и средствами защиты ИТ-подукта. В отличие от Профиля защиты "Федеральных критериев", который ориентирован исключительно на среду применения ИТ-продукта, Профиль защиты "Единых критериев" предназначен непосредственно для удовлетворения запросов потребителей.

Разработчики "Единых критериев" продолжили подход "Федеральных критериев", направленный на отказ от единой шкалы безопасности, и усилили гибкость предложенных в них решений путем введения частично упорядоченных шкал, благодаря чему потребители и производители получили дополнительные возможности по выбору требований и их адаптации к свои прикладным задачам.

Особое внимание этот стандарт уделяет адекватности реализации функциональных требований, которая обеспечивается как независимым тестированием и анализом ИТ-продукта, так и применением соответствующих технологий на всех этапах его проектирования и реализации.

Таким образом, требования "Единых критериев" охватывают практически все аспекты безопасности ИТ-продуктов и технологии их создания, а также содержат все исходные материалы, необходимые потребителям и разработчикам для формирования Профилей и Проектов защиты.

Кроме того, требования "Единых критериев" являются практически всеобъемлющей энциклопедией информационной безопасности, поэтому их можно использовать в качестве справочника по безопасности информационных технологий.

Данный стандарт ознаменовал собой новый уровень стандартизации информационных технологий, подняв его на межгосударственный уровень. За этим проглядывается реальная перспектива создания единого безопасного информационного пространства, в котором сертификация безопасности систем обработки информации будет осуществляться на глобальном уровне, что предоставит возможности для интеграции национальных информационных систем, что в свою очередь откроет совершенно новые сферы применения информационных технологий. Остается только посетовать, что наша страна, как всегда, осталась в стороне этого процесса.

2.11. Анализ стандартов информационной безопасности

Главная задача стандартов информационной безопасности — согласовать позиции и цели производителей, потребителей и аналитиков-классификаторов в процессе создания и эксплуатации продуктов информационных технологий. Каждая из перечисленных категорий специалистов оценивает стандарты и содержащихся в них требования и критерии по своим собственным параметрам.

Для большинства потребителей наибольшее значение имеют простота критериев безопасности и однозначность параметров выбора защищенной системы, а для наиболее квалифицированной части пользователей первую роль играет гибкость требований и возможность их применения к специфическим ИТ-продуктам и средам эксплуатации. Производители ИТ-продуктов требуют от стандартов максимальной конкретности и совместимости требований и критериев с современными архитектурами ВС, распространенными ОС и технологиями обработки информации.

Эксперты по квалификации хотят, чтобы стандарты информационной безопасности детально регламентировали процедуру квалификационного анализа и нуждаются в четких, простых, однозначных и легко применяемых критериях. Очевидно, что подобный идеал недостижим, и реальность требует от каждой стороны определенных компромиссов. Поэтому не будем проводить субъективный анализ стандартов с точки зрения каждого из участников процесса создания защищенных систем, а попытаемся ввести общие для всех "объективные" критерии сопоставления.

В качестве обобщенных показателей, характеризующих стандарты информационной безопасности и имеющих значение для всех трех сторон, предлагается использовать универсальность, гибкость, гарантированность, реализуемость и актуальность.

Универсальность стандарта определяется множеством типов ВС и областью информационных технологий, к которым могут быть корректно применены его положения. Это очень важная характеристика стандарта, ибо информационные технологии переживают период бурного развития, архитектура компьютерных систем совершенствуется, а сфера их применения постоянно расширяется. Стандарты информационной безопасности в своем развитии не должны отставать от современных информационных технологий или обходить ту или иную сферу их применения.

Под *гибкостью стандарта* понимается возможность и удобство его применения к постоянно развивающимся информационным технологиям и время его "устаревания". Гибкость может быть достигнута исклю-

чительно через фундаментальность требований и критериев и их инвариантность по отношению к механизмам реализации и технологиям создания ИТ-продуктов. Однако, очевидно, что чрезмерная абстрактность требований и оторванность их от практики снижает их реализуемость.

Гарантированность определяется мощностью предусмотренных стандартом методов и средств подтверждения надежности результатов квалификационного анализа. Вначале этому вопросу не уделялось много внимания, но анализ опыта применения первых стандартов информационной безопасности показал, что для достижения поставленных целей аналитики-классификаторы должны иметь возможность обосновывать свои заключения, а разработчики нуждаются в механизмах, с помощью которых они могли бы подтвердить корректность своих притязаний и предоставить потребителям определенные гарантии.

Реализуемость — это возможность адекватной реализации требований и критериев стандарта на практике, с учетом затрат на этот процесс. Реализуемость во многом связана с универсальностью и гибкостью, но отражает чисто практические и технологические аспекты реализации положений и требований стандарта.

Актуальность отражает соответствие требований и критериев стандарта постоянно развивающемуся множеству угроз безопасности и новейшим методам и средствам, используемым злоумышленниками. Эта характеристика, наряду с универсальностью является одной из важнейших, т. к. способность противостоять угрозам и прогнозировать их развитие фактически определяет пригодность стандарта и является решающим фактором при определении его пригодности.

Классификация рассмотренных стандартов информационной безопасности по предложенным показателям приведена в таб. 2.5.

Степень соответствия стандартов предложенным показателям определяется по следующей качественной шкале:

- ограниченное соответствие — недостаточное соответствие, при применении стандарта возникают существенные трудности;
- умеренное соответствие — минимальное соответствие, при применении стандарта в большинстве случаев существенных трудностей не возникает;
- достаточное соответствие — удовлетворительное соответствие, при применении стандарта в большинстве случаев не возникает никаких трудностей, однако эффективность предлагаемых решений не гарантируется;
- высокое соответствие — стандарт предлагает специальные механизмы и процедуры, направленные на улучшение данного показателя, применение которых позволяет получать достаточно эффективные решения;
- превосходное соответствие — улучшение данного показателя рассматривалось авторами стандарта в качестве одной из основных целей его разработки, что обеспечивает эффективность применения предложенных решений.

Таблица 2.5. Сопоставление стандартов информационной безопасности.

Стандарты безопасности	Показатели сопоставления стандартов информационной безопасности				
	Универсальность	Гибкость	Гарантированность	Реализуемость	Актуальность
Оранжевая книга (1983 г.)	ограниченная	ограниченная	ограниченная	высокая (за исключением класса А)	умеренная
Европейские критерии (1986 г.)	умеренная	умеренная	умеренная	высокая	умеренная
Документы ГТК (1992 г.)	ограниченная	ограниченная	отсутствует	высокая	ограниченная
Федеральные критерии (1992 г.)	высокая	отличная	достаточная	высокая	высокая
Канадские критерии (1993 г.)	умеренная	достаточная	достаточная	достаточная	достаточная
Единые критерии (1999 г.)	превосходная	превосходная	превосходная	превосходная	превосходная

Рассмотрим, в какой степени стандарты информационной безопасности отвечают предложенным показателям, и проследим направления, по которым шло их развитие.

2.11.1. Универсальность

На этапе начального становления и развития стандартов информационной безопасности универсальности уделялось мало внимания, т. к., во-первых, разработчикам стандартов казалось, что в обеспечении безопасности нуждается только ограниченный круг потребителей, находящихся в правительственных и военных сферах, и, во-вторых, темпы информатизации тогда были относительно низкими.

Поэтому первый стандарт безопасности — "Оранжевая книга" предназначался для систем военного применения, основанных в те годы исключительно на мэйнфреймах, и его адаптация для распределенных систем и баз данных потребовала разработки дополнительных документов. С учетом этого опыта сфера применения вышедших несколькими годами позже "Европейских критериев" значительно расширена, — уже на уровне базового документа в этот стандарт вошли распределенные системы, сети, системы телекоммуникаций и СУБД.

Однако, в этом документе по-прежнему явным образом оговаривается архитектура и назначение систем, к которым он может быть применен, и никак не регламентируется среда их эксплуатации. Документы Гостехкомиссии России имеют довольно ограниченную сферу применения — это персональные (видимо это объясняется тотальным распространением PC в нашей стране) и многопользовательские системы, причем ориентация системы на обслуживание конечных пользователей является обязательным условием. "Федеральные критерии" подняли область применения стандартов на новый уровень, начав рассматривать в качестве объекта их применения любые продукты информационных технологий, независимо от их назначения, проводя различие только между характеристиками среды их эксплуатации. "Канадские критерии" рассматривают в качестве области своего применения все типы компьютерных систем.

Наконец, "Единые критерии" завершили процесс расширения сферы применения стандартов информационной безопасности предложив такую технологию создания ИТ-продуктов, при которой использование данного стандарта является неотъемлемым компонентом.

2.11.2. Гибкость

Гибкость положений стандарта определяет удобство его использования потребителями и производителями систем обработки информации. Возможно именно потому, что требования первого стандарта ("Оранжевой книги") были в большинстве своем инвариантными к механизмам реализации, они оказались слишком абстрактными для непосредственного применения во многих случаях, что потребовало их комментирования, дополнения и расширения. "Европейские критерии" унаследовали такой стиль изложения требований, но пошли по пути экстенсивного развития, предусмотрев специальные уровни и требования, рассчитанные на типовые системы (СУБД, телекоммуникации и т. д.).

Руководящие документы ГТК по конкретности своих требований превзошли даже уровень "Оранжевой книги", т. к. подробно регламентируют реализацию функций защиты (например, это единственный стандарт, который в ультимативной форме требует применения криптографии), что значительно снижает удобство их использования — в конкретных ситуациях многие требования часто оказываются избыточными и ненужными.

Более того, ограничение области применения данного стандарта системами, ориентированными на конечного пользователя, ставит отечественных разработчиков и экспертов по сертификации в затруднительное положение при применении эти документов, скажем, к программному обеспечению маршрутизатора или межсетевого экрана (у этих систем в принципе нет пользователей как физических лиц). "Федеральные критерии" обеспечивают гибкость на качественно новом по сравнению с предшествующими стандартами уровне, впервые предложив механизм Профилей защиты, с помощью которых можно создавать специальные наборы требований, соответствующие запросам потребителей конкретного продукта и угрозам среды его эксплуатации. "Канадские критерии" не рассматривают Профиль защиты в качестве обязательного элемента безопасности информационных технологий, и также обладают определенной спецификой в своем подходе к основным понятиям безопасности, поэтому их гибкость можно оценить только как достаточную.

Наконец, "Единые критерии" обладают практически совершенной гибкостью, т. к. позволяют потребителям выразить свои требования с помощью механизма Профилей защиты, в форме инвариантной к механизмам реализации, а производителям — продемонстрировать с помощью Проекта защиты как эти требования преобразуются в задачи защиты и реализуются на практике.

В этом случае процесс квалификации уровня безопасности ИТ-продукта представляет собой проверку взаимного соответствия Профиля

защиты, Проекта защиты и реализованного ИТ-продукта, а также их соответствия "Единым критериям".

2.11.3. Гарантированность

Гарантированность обеспечиваемого уровня защиты сначала рассматривалась разработчиками стандартов только для высших уровней безопасности. Поэтому "Оранжевая книга" предусматривала обязательное применение формальных методов верификации при только создании систем высшего класса защищенности (класс А).

Однако необходимость контроля корректности реализации требований и подтверждения эффективности средств защиты для систем всех уровней была осознана достаточно быстро. Уже в "Европейских критериях" появляется специальный раздел требований — требования адекватности, которые регламентируют технологию и инструментарий разработки, а также контроль за процессами проектирования и разработки.

К сожалению, документы ГТК практически полностью проигнорировали этот, на наш взгляд ключевой аспект безопасности информационных технологий, и обходят данный вопрос молчанием. "Федеральные критерии" содержат два специальных раздела требований, посвященных этому аспекту безопасности, содержащие требования к технологии разработки и к процессу квалификационного анализа.

"Канадские критерии" включают раздел требований адекватности, количественно и качественно ни в чем не уступающий разделу функциональных требований. "Единые критерии" рассматривают гарантированность реализации защиты как самый важный компонент информационной безопасности и предусматривают многоэтапный контроль на каждой стадии разработки ИТ-продукта, позволяющий подтвердить соответствие полученных результатов поставленным целям путем доказательства адекватности задач защиты требованиям потребителей, адекватности Проекта защиты "Единым критериям" и адекватности ИТ-продукта Проекту защиты.

2.11.4. Реализуемость

Плохие показатели реализуемости говорят о практической бесполезности стандарта, поэтому все рассмотренные документы отвечают этому показателю в достаточной или высокой степени. Реализация требований "Оранжевой книги", за исключением высшего класса (класса А), большой сложности не представляет, что подтверждается большим числом систем, сертифицированных на соответствие классам В и С(около 30 систем).

Авторам известна только две системы, сертифицированная на соответствие классу А, причем политика безопасности в одной из них реализована на аппаратном уровне с помощью контроля операндов каждой инструкции, т. е. разработчикам пришлось спроектировать и реализовать специальный процессор. Остальные стандарты решают эту проблему за счет гибкости предлагаемых требований и критериев.

О "Канадских критериях" стоит упомянуть особо, поскольку своим нетрадиционным толкованием понятий "объект" и "субъект" они осложняют разработчикам процесс реализации предложенных требований, что и определило уровень их соответствия данному показателю только как достаточный. "Единые критерии" и здесь оказались на практически недосягаемой для остальных стандартов высоте за счет потрясающей степени подробности функциональных требований (135 требований), фактически служащих исчерпывающем руководством по разработке защищенных систем. Отметим, что это единственный показатель, по которому документы ГТК не отстают от остальных стандартов информационной безопасности.

2.11.5. Актуальность

Актуальность стандартов информационной безопасности возрастает с расширением сферы их применения и появлением опыта их использования. "Оранжевая книга", хотя и содержит предпосылки для противодействия всем основным видам угроз, содержит требования, в основном направленные на противодействие угрозам конфиденциальности, что объясняется ее ориентированностью на системы военного назначения. "Европейские критерии" находятся примерно на том же уровне, хотя и уделяют угрозам целостности гораздо больше внимания.

Документы ГТК с точки зрения этого показателя выглядят наиболее отсталыми, — уже в самом их названии определена единственная рассматриваемая в них угроза — несанкционированный доступ. "Федеральные критерии" рассматривают все виды угроз достаточно подробно и предлагают механизм Профилей защиты для описания угроз безопасности, присущих среде эксплуатации конкретного ИТ-продукта, что позволяет учитывать специфичные виды угроз.

"Канадские критерии" ограничиваются типовым набором угроз безопасности, достаточным для большинства применений. "Единые критерии" ставят во главу угла удовлетворение нужд пользователей и предлагают для этого соответствующие механизмы (Профиль и Проект защиты), что дает возможность выстроить на их основе динамичную и постоянно адаптирующуюся к новым задачам технологию создания безопасных информационных систем.

Представленный анализ стандартов информационной безопасности и основных тенденций их развития позволяет сделать следующие выводы:

1. Развитие стандартов привело к отказу от единой шкалы ранжирования требований и критериев, замене их множеством независимых частных показателей и введению частично упорядоченных шкал.

2. Неуклонное возрастание роли требований адекватности реализации защиты и политики безопасности свидетельствует о тенденции преобладания "качества" обеспечения защиты над ее "количеством".

3. Определение ролей производителей, потребителей и экспертов по квалификации ИТ-продуктов и разделение их функций в процессе создания защищенных систем обработки информации свидетельствует о полноправной интеграции стандартов обеспечения безопасности в сферу информационных технологий.

4. Сложившееся на основе современных стандартов разделение ролей участников процесса создания и эксплуатации защищенных систем, применение соответствующих механизмов и технологий привело к сбалансированному распределению ответственности между всеми участниками процесса.

5. Современные тенденции интеграции информационных технологий и стремление к созданию безопасного всемирного информационного пространства привели к необходимости интернационализации стандартов информационной безопасности.

2.12. Заключение

Итак, мы рассмотрели стандарты информационной безопасности, начиная с самых первых и заканчивая последним, обобщившим опыт применения предшествующих документов. Что это дает для решения поставленной задачи — построения защищенной системы, кроме изучения накопленного за четырнадцать лет опыта?

Во-первых, мы определили задачи, которые должны быть решены в ходе создания защищенной системы (задачи защиты): эффективное противостояние угрозам безопасности, действующим в среде ее эксплуатации, и корректная реализация политики безопасности.

Во-вторых, таксономия функциональных требований или критериев, приведенная для каждого стандарта, позволяет определить набор функциональных возможностей средств защиты, которые должны быть реализованы в защищенной системе.

Наконец, впервые в отечественной литературе столь подробно представлены основные стандарты информационной безопасности, содержащиеся в них требования, технологии их применения.

Смысл включения в книгу данной главы состоит в том, что изложенный материал позволяет определить, какими свойствами должна обладать защищенная система, каким условиям должна отвечать ее архитектура, какие функциональные возможности должны обеспечивать средства защиты, какие требования могут быть предъявлены в ходе сертификации, каким образом должны строиться процессы проектирования, разработки и сертификации. Таким образом представленный обзор служит руководством по всем существенным элементам современных технологий создания защищенных систем.

Представленный материал позволяет сформулировать задачи каждого из участников процесса создания защищенных систем (потребители, производители, эксперты по сертификации), которые должны быть решены для достижения поставленной цели. Наряду с исследованием причин нарушений безопасности (гл. 3), теоретическими основами защиты — моделями безопасности и криптографией (гл. 4), а также архитектурой построения защищенных систем (гл. 5), этот материал служит основой для технологии создания защищенных систем.

Литература к главе 2

1. Гостехкомиссия России. Руководящий документ. Концепция защиты средств вычислительной техники от несанкционированного доступа к информации. Москва, 1992 г.

2. Гостехкомиссия России. Руководящий документ. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации. Москва, 1992 г.

3. Гостехкомиссия России. Руководящий документ. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. Москва, 1992 г.

4. Гостехкомиссия России. Руководящий документ. Временное положение по организации разработки, изготовления и эксплуатации программных и технических средств защиты информации от несанкционированного доступа в автоматизированных системах и средствах вычислительной техники. Москва, 1992 г.

5. Гостехкомиссия России. Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения. Москва, 1992 г.

6. В. А. Галатенко. Информационная безопасность. "Открытые системы", NN4-6 1995 г.

7. Trusted Computer System Evaluation Criteria. US Department of Defense 5200.28-STD, 1993.

8. Trusted Network Interpretation. National Computer Security Center. NCSC-TG-005 Version 1, July 1987.

9. Trusted Database Management System Interpretation. National Computer Security Center. NCSC-TG-021 Version 1, April 1991.

10. A guide to understanding discretionary access control in trusted systems. National Computer Security Center. NCSC-TG-003 Version 1, September 1987.

11. Password management guideline. US Department of Defense. CSC-STD-002-85, April 1985.

12. Guidance for applying the Department of Defense Trusted Computer System Evaluation Criteria in specific environment. US Department of Defense. CSC-STD-003-85, June 1985.

13. A Guide to Understanding Audit in Trusted Systems. National Computer Security Center. NCSC-TG-001, July 1987.

14. Guide to understanding configuration management in trusted systems. National Computer Security Center. NCSC-TG-006-88, March 1988.

15. The Interpreted Trusted Computer System Evaluation Criteria Requirements. National Computer Security Center. NCSC-TG-001-95, January 1995.

16. Information Technology Security Evaluation Criteria. Harmonized Criteria of France-Germany-Netherlands-United Kingdom. - Department of Trade and Industry, London, 1991.

17. Federal Criteria for Information Technology Security. National Institute of Standards and Technology & National Security Agency. Version 1.0, December 1992.

18. Canadian Trusted Computer Product Evaluation Criteria. Canadian System Security Centre Communication Security Establishment, Government of Canada. Version 3.0e. January 1993.

19. Common Criteria for Information Technology Security Evaluation. National Institute of Standards and Technology & National Security Agency (USA), Communication Security Establishment (Canada), Communications -Electronics Security Group (United Kingdom), Bundesamt fur Sichereit in der Informationstechnik (Germany), Service Central de la Securite des Systemes d'Information (France), National Communications Security Agency (Netherlands). Version 2.1, August 1999.

20. Common Evaluation Methodology for Information Technology Security. National Institute of Standards and Technology & National Security Agency (USA), Communication Security Establishment (Canada), UK IT Security and Certification Scheme (United Kingdom), Bundesamt fur Sichereit in der Informationstechnik (Germany), Service Central de la Securite des Systemes (France), National Communications Security Agency (Netherlands). Version 0.6. 11.01.97.

Sublata causa, tollitur morbus.

Устраните причину,
тогда исчезнет и болезнь.
Гиппократ

Глава 3. Исследование причин нарушений безопасности

В данной главе продолжается исследование понятия защищенной системы обработки информации, но в отличие от предыдущей главы, рассматривающей проблему с точки зрения норм и стандартов, безопасность ВС анализируется с точки зрения практики. Таким образом, если стандарты информационной безопасности (наравне с моделями безопасности — глава 4) представляет собой попытку решить проблему сверху, то исследования нарушений безопасности и обуславливающих их причин являют собой прагматический подход к проблеме (как бы взгляд снизу), основанный на очевидной предпосылке, — чтобы предотвратить появление нарушений безопасности, надо устранить их причину. Задача данной главы — выявить обстоятельства, которые приводят к успешной реализации угроз безопасности, систематизировать их и определить первопричины их появления.

3.1. Нарушения безопасности ВС и изъяны защиты

Знание истории нарушений безопасности вычислительных систем и понимание причин, по которым были успешно осуществлены, является одним из необходимых условий создания защищенных систем. Перспективным направлением исследований в этой области является анализ успешно реализовавшихся угроз безопасности (атак) с целью их обобщения, классификации и выявления причин и закономерностей их появления и существования. Проведение такого анализа позволяет при разработке и создании защищенных систем сконцентрировать основные усилия именно на устранении этих причин путем исправления в механизмах защиты выявленных недостатков, что позволяет эффективно противостоять угрозам безопасности. Очевидно, что основой данного подхода является глубокое исследование частных случаев нарушения безопасности и слабых сторон систем защиты, сделавших возможным их осуществление. Для этого необходимо провести анализ существующих данных о нарушении безопас-

ности ВС и разработать их классификацию. Эта задача стояла в центре ряда зарубежных исследований [1, 4, 5, 6, 11, 12, 13, 16, 17, 18].

Наибольший интерес представляет одно из новейших направлений исследований в данной области, предложенное в работе "Таксономия нарушений безопасности программного обеспечения, с примерами" [1]. Данная глава опирается на результаты этого исследования и представляет собой его обобщение и развитие с точки зрения объяснения причин нарушений информационной безопасности и построения защищенных систем.

Любая атака на вычислительную систему (подразумеваются успешные атаки, в результате которых происходит нарушение информационной безопасности) опирается на определенные особенности построения и функционирования последней, иными словами — использует имеющиеся недостатки средств обеспечения безопасности. В контексте изучения истории атак на вычислительные системы и построения таксономии причин нарушений информационной безопасности наиболее точно отражения сущности этого явления предлагается ввести термин "изъян защиты" (ИЗ).

Под ИЗ понимается совокупность причин, условий и обстоятельств, наличие которых в конечном итоге может привести к нарушению нормального функционирования ВС и нарушению безопасности (несанкционированный доступ, ознакомление, уничтожение или искажение данных). Иными словами (в большинстве случаев) под этим термином понимаются особенности построения входящих в состав ВС программных средств защиты, которые не в состоянии противостоять угрозам безопасности и обеспечивать безусловное выполнение возложенных на них задач.

Данная глава не преследует своей целью исчерпывающего описания всех известных случаев нарушения безопасности и детального изучения механизмов осуществления атак. Для того подкрепить реальными фактами предлагаемую таксономию причин нарушения безопасности в Приложении IV содержится список имевших место случаев нарушения безопасности, заимствованный из [1].

3.2. Исследования нарушений безопасности компьютерных систем

Наибольший интерес среди работ в области выявления, исследования и систематизации ÈÇ представляют труды [2,12] и одна из последних публикаций в данной области [1]. Представляется целесообразным проанализировать результаты, полученные в работах [2,12,17] для того, чтобы представить фактографическую основу, на которую опираются результаты, полученные в последних научных исследованиях. Наиболее серьезные усилия по выявлению ИЗ в системах защиты с помощью экспериментов

по их преодолению нашли свое отражение в разработанной в начале 70-х годов т.н. методологии гипотетического выявления ИЗ (Flaw Hypothesis Methodology) [12]. Данная методология предусматривает проведение исследований в три этапа. Первый этап состоит в изучении системы, причем особое внимание уделяется принципам функционирования механизмов защиты. На втором этапе происходит выдвижение предположений (гипотез) о потенциально уязвимых компонентах, которые затем тщательно проверяются на основании анализа документации, реальных особенностей ВС и деталей ее функционирования, и с помощью проведения специальных тестов, призванных подтвердить или опровергнуть присутствие предполагаемого изъяна в системе защиты. Наконец на третьем, заключительном этапе проводится обобщение полученных результатов и формируются списки (перечни) выявленных ИЗ и успешных атак. Хотя в [12] и присутствует перечень ИЗ исследованных систем, разработанных и успешно осуществленных атак, систематизация полученных данных проведена не была.

В середине 70-х годов подобные исследования проводились по проектам *RISOS* (Research in Secured Operating System — Исследования безопасности защищенных операционных систем) и *PA* (Protection Analysis — Анализ защиты). В обоих проектах были предприняты попытки формального описания и систематизации информации о ИЗ.

В заключительном отчете по проекту *RISOS* [2] приведено описание следующих категорий ИЗ в защищенных операционных системах:

1. Неполная проверка допустимости значений критичных параметров.
2. Противоречия в проверке допустимости значений критичных параметров.
3. Неявное совместное использование несколькими пользователями конфиденциальной информации.
4. Асинхронный контроль параметров или неадекватная последовательность действий.
5. Недостаточно надежная идентификация / аутентификация / авторизация.
6. Возможность нарушения запретов и ограничений.
7. Использование ошибок в логике функционирования механизмов защиты.

В итоговом отчете по этому проекту описываются разработанные примеры атак для каждого типа ИЗ и содержится детальная информация по семнадцати реально выявленным ИЗ в трех операционных системах: IBM MVT, Univac 1100 и TENEX. Каждый из ИЗ четко соответствует одной из семи приведенных категорий.

Целью проекта *PA* был сбор информации и построение абстрактных моделей, которые в дальнейшем могли бы быть применены для автоматизированного поиска ИЗ. В результате исследования шести операционных систем (GCOS, Multics, Unix, IBM MVT, Univac 1100 и TENEX) было обнаружено более ста ИЗ, способных привести к несанкционированному проникновению в систему [5]. Для анализа обнаруженных ИЗ была разработана классификационная таблица, содержащая десять категорий ИЗ, которые можно обобщить следующими классами:

1. Ошибки выделения областей (доменов), включающие незащищенное (в открытом виде) хранение и представление данных, неполное уничтожение объектов или их окружения.

2. Ошибки контроля и проверки, состоящие в некорректной проверке значений параметров и границ объектов.

3. Ошибки назначения имен, допускающие возможность использования нескольких имен для одного объекта и неполное запрещение доступа к уничтоженным объектам.

4. Ошибки в последовательности действий, влекущие за собой некорректное использование множественных ссылок (указателей) на объект и ошибки прерывания атомарных операций.

Предлагаемые в работе [5] методики поиска ошибок безопасности в операционных системах достаточно ограничены в практическом применении. Это можно объяснить предпринятой попыткой обеспечить универсальность методик, что отрицательно сказалось на возможности их развития и адаптации для новых ОС. С другой стороны, усилия исследователей слишком рано были перенаправлены от изучения ИЗ в сторону разработки универсальной технологии создания защищенных операционных систем, свободных от подобных ошибок.

Данная работа преследует более определенную и реальную цель: на основании опубликованной информации о ИЗ разработать их детальную классификацию и выявить причины их возникновения.

3.3. Таксономия ИЗ

Таксономия, в отличие от классификации, представляющей собой абстрактную структуру разнесенных по категориям экземпляров, включает в себя комплексное исследование предметной области и создание модели полного множества изучаемых объектов, что позволяет определить признаки, которые могут быть положены в основу той или иной классификации. На основании предложенной модели таксономия дает возможность построить полное множество категорий исследуемых объектов для любой выбранной классификации.

С точки зрения технологии создания защищенных систем наибольшее значение имеют следующие вопросы, на которые должна дать ответ таксономия ИЗ:

1. Каким образом ошибки, приводящие к появлению ИЗ, вносятся в систему защиты? (классификация ИЗ по источнику появления).

2. Когда, на каком этапе, они вносятся? (классификация ИЗ по этапу возникновения.)

3. Где, в каких компонентах, системы защиты (или ВС в целом) они возникают и проявляются? (классификация ИЗ по размещению в системе.)

Каждый вопрос предопределяет наличие соответствующей классификации.

3.3.1. Классификация ИЗ по источнику появления

Как следует из определения ИЗ, источником их появления является ошибка или недоработка в системе безопасности. Исследованию ошибок, тем или иным образом связанных с безопасностью, всегда уделялось много внимания. В качестве примера можно привейти работы [2,3,4,11,12,13,16,17]. Классификация ИЗ по источнику появления, приведенная в работе [1] представлена в табл. 3.1.

Данная таблица нуждается в комментарии. Под источником появления (эквивалентный перевод емкого английского термина genesis), в [1] понимается основа существования ИЗ, т. е. либо характеристики ВС, которые обуславливают его существование, либо принцип функционирования средств, использующих ИЗ для осуществления атаки. С точки зрения авторов данный подход можно подвергнуть серьезной критике, т. к. он порождает некоторое противоречие между заявленным принципом классификации и полученными результатами. Действительно, (см. табл. 3.1) классификация преднамеренных ИЗ фактически представляет собой классификацию разрушающих программных средств (подробнее об РПС и способов борьбы с ними см. гл. 2 в [20]) по принципам функционирования, а непреднамеренных — классификацию ошибок, возникающих в ходе программной реализации системы. Для решения практических задач наибольший интерес представляет классификация ИЗ по причинам возникновения, поэтому авторы провели собственное исследование, результаты которого будут представлены далее.

Ошибки, служащие источником появления ИЗ, могут быть внесены в систему защиты специально (преднамеренно), либо возникнуть неумышленно (непреднамеренно). Для выявления таких непреднамеренных, в определенном смысле случайных, ошибок применяются различные стратегии и методики.

Таблица 3.1. Классификация ИЗ по источнику появления.

					Кол-во примеров	Индексы в Приложении IV1
Ошибки в системах защиты, служащие источником появления ИЗ	Преднамеренные	С наличием деструктивных функций (активные)	Разрушающие программные средства (РПС)	Несамовоспроизводящиеся РПС("троянские кони")	3	PC1,PC3, I8
				Самовоспроизводящиеся РПС (вирусы)	7	U1,PC2,PC4, MA1,MA2,CA1,AT1
			Черные ходы, люки, Проникновения в	скрытые возможности систему	2	U1, U10
		Без деструктивных Функций (пассивные)	Скрытые каналы утечки информации	С использованием памяти	1	DT1
				С использованием времени	2	I9,D2
			Другие		5	I7,B1,U3,U6,U10
	Непреднамеренные (случайные)	Ошибки контроля допустимых значений параметров.			10	I4,I5,MT1,MU2,MU4, MU8,U7, U11,U12,U13
		Ошибки определения областей (доменов)			7	I3,I6,MT2,MT3,MU3, UN1,D1
		Ошибки последовательности действий и использования нескольких имен для одного объекта (в том числе TOCTTOU)			2	I1,I2
		Ошибки идентификации/аутентификации.			5	MU1,U2,U4,U5,U14
		Ошибки проверки границ объектов.			4	MT4,MU5,MU6,U9
		Другие ошибки в логике функционирования			4	MU7,MU9,U8,IN1

1 Приложение IV содержит ряд типовых примеров нарушения безопасности ВС. Все примеры в этом приложении отмечены уникальными идентификаторами, которые используется в тексте этой главы для ссылок на примеры из приложения.

Например, большинство случайных ошибок может быть выявлено и устранено при увеличении времени и глубины анализа и тестирования кода систем защиты. Но в отношении наиболее серьезных преднамеренно внесенных (и, вероятнее всего, замаскированных) ошибок этот способ окажется малоэффективным. Для того, чтобы сделать процесс поиска таких ошибок более продуктивным, необходимо проведение специальных испытаний, заключающихся в осуществлении попыток проникновения в ВС и проведении атак на систему защиты. Обобщение информации о ИЗ, позволяющее осуществить аргументированный выбор эффективной стратегии выявления ошибок в системах обеспечения безопасности путем проведения экспериментальных атак является одной из задач данной работы.

Характеристика преднамеренности является в определенной мере относительной. В некоторых случаях определенные функции, намеренно добавленные в программное обеспечение, предопределяли внесение в систему защиты непреднамеренных ошибок. Например, возможность использования удаленной отладки или настройки системы может привести к появлению ИЗ. Такая ошибка может быть классифицирована как преднамеренная, но не имеющая деструктивной функции.

Хотя определенные преднамеренные ошибки могут рассматриваться как случайные, трудно представить, скажем, случайно возникшую "троянскую" программу. В спорных случаях при классификации ошибок по их происхождению в работе [1] рекомендуется обратиться к другим классификационным признакам, например, к этапу внесения ошибки. Как правило, случайные ошибки вносятся в систему при разработке требований к безопасности и спецификаций системы защиты (откуда они автоматически переносятся в реализующие их средства), а также в процессе сопровождения системы (обновления версии, поставки новых утилит и т.п.). Напротив, преднамеренные ошибки внедряются в систему именно на этапе ее применения(если, конечно, среди разработчиков системы не было лица, заинтересованного в ее компрометации).

Преднамеренные ошибки являются достаточно трудно обнаруживаемыми, потому что они специально скрыты, замаскированы именно с целью не быть обнаруженными. Они же являются самыми опасными: специально внесенные ошибки с деструктивными функциями могут серьезно нарушить функционирование системы. Случайные ошибки, безобидные сами по себе, также могут быть использованы для этих же целей специально написанными программами. Рассмотрим подробнее основные классы преднамеренно внесенных ИЗ.

3.3.1.1. Преднамеренно внедренные ИЗ с наличием деструктивных функций.

Преднамеренно внесенные в систему защиты изъяны и связанные с ними каналы утечки информации по сути дела являются результатом функционирования предварительно внедренных специальных средств - РПС, о которых говорилось выше. Общей характерной чертой РПС является активный характер их функционирования — противодействие системе обеспечения безопасности и обеспечение утечки информации. Как правило, они имеют жаргонные названия, подчеркивающие их суть, например, "троянский конь", "логическая бомба", "часовая мина". Более подробно авторы рассмотрели проблему РПС в [20]. "Логические бомбы" и "часовые мины" - это РПС, которые не выполняют никаких функций до наступления определенного события в системе, после чего "срабатывают", что, как правило, приводит к серьезным нарушениям работы системы, уничтожению информации и другим деструктивным последствиям. Особенностью проявления данного вида преднамеренно внедряемого кода является отсутствие каких-либо мер маскировки выполнения деструктивных функций. Последствия "срабатывания" чаще всего носят катастрофический характер и могут привести к полной потере работоспособности ВС и уничтожению хранящихся в ней данных. РПС являются одним из самых опасных и самым разрушительным видом ИЗ.

"Черным ходом" называется скрытая (замаскированная) возможность получения доступа к ресурсам в обход стандартных механизмов контроля. Например, разработчик программы проверки идентификатора может предусмотреть при совпадении контролируемого параметра с известной только ему константой осуществление некоторых непредусмотренных действий, скажем, отменить контроль доступа для субъекта с этим идентификатором. В дальнейшем разработчик может использовать этот "черный ход" для бесконтрольного доступа к инфорiации.

3.3.1.2. Преднамеренно внедренные ИЗ без деструктивных функций

РПС могут осуществлять взаимодействие с атакующим систему злоумышленником через т.н. "скрытые каналы" утечки информации. Под скрытым каналом будем понимать любую возможность обмена информацией, не предусмотренную разработчиками средств защиты и, как следствие, ничем не контролируемую [10]. Отнести скрытые каналы к непреднамеренно возникшим ИЗ нельзя, так как их использование носит отнюдь не случайный характер. В силу отсутствия контроля за скрытыми каналами со стороны системы защиты они широко используются злоумышленника-

ми. Для использования скрытых каналов требуется наличие двух процессов: один (как правило, РПС) осуществляет сбор информации, интересующей злоумышленника, и помещает ее в скрытый канал, который не контролируется системой защиты. Другой процесс осуществляет прослушивание канала в ожидании начала передачи, и при ее появлении выполняет необходимую обработку и сохранение собранной информации.

Скрытые каналы утечки в зависимости от способа кодирования информации, передаваемой между этими процессами, подразделяются на два типа: с использованием памяти и с использованием времени.

В первом случае для кодирования передаваемой информации используется либо область памяти, не имеющая важного значения (например, установление характерных признаков в имени и атрибутах файла), либо вообще неиспользуемая область (например, зарезервированные поля в заголовке сетевого пакета).

Во втором случае, более сложном для реализации, информация кодируется определенной последовательностью и длительностью событий, происходящих в системе (например, с помощью модуляции интервалов обращения к устройствам, введения задержек между приемом и посылкой сетевых пакетов и т.д.).

Кроме скрытых каналов, в системе могут присутствовать и другие преднамеренно внесенные ошибки, не влекущие за собой разрушительных последствий. К их появлению обычно приводит расхождение между требованиями безопасности и требованиями к функциональным возможностям ВС. В силу присущей скрытым каналам трудности выделения общих особенностей и уникальности более подробная их классификация не проводилась, т. к. выходит за рамки данной работы.

3.3.1.3. Непреднамеренные ошибки и ИЗ

Непреднамеренные ИЗ могут возникнуть из-за ошибок, допущенных на этапе разработки требований, при создании спецификаций и на этапе реализации. Ошибки этого типа были подробно исследованы в работах [2,4]. Хотя большинство из них обнаруживается и устраняется в процессе тестирования, ряд ошибок может остаться незамеченным и вызвать определенные проблемы на этапе эксплуатации системы. Наиболее трудно выявляются такие ошибки в сложных системах, состоящих из многочисленных компонентов и разработанных при участии большого коллектива специалистов. Одна из неотъемлемых проблем таких систем — невозможность составления исчерпывающих спецификаций, то есть невозможность адекватного документирования. Недостатки проектной документации в процессе сопровождения и эксплуатации системы приводят к тому, что

при попытке устранения одних ошибок, в нее вносятся другие. И хотя наличие неумышленных ошибок не приводит к немедленному их использованию и нарушению безопасности системы, это является лишь вопросом времени.

Неумышленные ИЗ могут быть классифицированы в соответствии со следующими группами ошибок, предопределяющими их существование:

1. Ошибки контроля допустимых значений параметров.
2. Ошибки определения доменов.
3. Ошибки последовательности действий и использования нескольких имен для одного объекта.
4. Ошибки идентификации/аутентификации.
5. Ошибки проверки границ объектов.
6. Ошибки в логике функционирования.

Ошибки контроля допустимых значений параметров заключаются в принятии соответствующим механизмом неправильного заключения о соответствии проверяемого параметра допустимым значениям. Это касается числа, состава, типа, размера, статуса (передаваемые или принимаемые) параметров или ряда других их характеристик. Ошибки контроля можно рассматривать как неадекватную реакцию механизмов защиты на возникающие в системе события.

Ошибки определения доменов выражаются в наличии неконтролируемого способа доступа в защищенную область. Например, возможность получения доступа к объекту файловой системы, непосредственный доступ к которому запрещен, посредством доступа к его физическому представлению на магнитных носителях. Другим примером является возможность доступа к остаточной информации в занимаемой объектом области памяти после его уничтожения.

Наличие ошибок последовательности действий предопределяется асинхронным функционированием компонент системы, которое может быть использовано для нарушения безопасности. Выявить такие ошибки в системе достаточно трудно. Их суть заключается в том, что в силу невозможности представления каждой из функций системы единственной атомарной операцией, многие из функций выполняются как некоторая асинхронная последовательность действий (операций), одновременно осуществляемых несколькими компонентами системы. Например, одной из одновременно выполняющихся операций может быть проверка идентификатора процесса, а второй — установка для него соответствующих полномочий, или, проверка допустимости параметра, а затем — его использование. Асинхронность может быть использована злоумышленником для обмана механизмов контроля путем подмены параметра на другой, запрещенный,

после проверки его допустимости, но перед использованием. Данная ошибка носит название TOCTTOU (time-of-check to time-of-use) [16] — возможность подмены информации между моментом ее проверки и моментом использования (I1 в Приложении IV).

Ошибки идентификации/аутентификации приводят к тому, что неуполномоченный на соответствующие действия пользователь получает доступ к защищенным объектам в объеме полномочий другого пользователя. Эти ошибки в принципе можно рассматривать как ошибки контроля в том смысле, что происходит неправильная проверка параметров идентификации и подлинности пользователя и объекта. Однако по причине значительного числа случаев нарушений безопасности из-за неправильной идентификации/аутентификации целесообразно рассматривать данную категорию ошибок как отдельную.

Ошибки проверки границ объектов и связанные с ними каналы утечки обычно возникают из-за игнорирования контроля за пересечением объектом границ области памяти, отведенной для его хранения (контроль длины строки, размера массива, размера и положения файла и т.д.). Самый известный пример такого рода — программа finger в ОС Unix (U12 в Приложении IV).

Кроме того существуют другие ошибки, не попадающие непосредственно ни в одну из перечисленных категорий. В целом их можно назвать ошибками логики функционирования системы и механизмов защиты, которые потенциально могут быть использованы злоумышленниками для проникновения в систему и нарушения безопасности.

3.3.2. Классификация ИЗ по этапам внедрения

В отличие от исследования вопроса об источниках возникновения ИЗ, анализу этапа появления ошибок уделяется недостаточное внимание. Результаты исследования данного вопроса подробно изложены в нескольких работах, наибольший интерес из которых представляет [2], в которой проблема выявления этапа внедрения ошибок рассматривается по отношению к жизненному циклу программного обеспечения.

Существует большое число моделей жизненного цикла программных систем и подходов к процессу разработки программного обеспечения. Для выделения категорий ИЗ относительно времени их внедрения можно построить простую абстрактную схему, описывающую широкий спектр технологий разработки ПО.

На самом верхнем уровне представления жизненного цикла систем можно выделить три фазы:

- фазу разработки, которая охватывает весь период создания первой рабочей версии системы;
- фазу сопровождения, в ходе которой происходит модификация, совершенствование, развитие системы и появление ее очередных версий;
- фазу эксплуатации, то есть непосредственного функционального применения конкретной версии системы.

Хотя на практике фазы сопровождения и эксплуатации перекрываются во времени, им присущи различные особенности, которые позволяют выделить соответствующие категории ИЗ. Классификация ИЗ по этапу внедрения, основанная на этих положениях, приведена в табл. 3.2.

Таблица. 3.2 Классификация ИЗ по этапу внедрения

			Количество примеров	Индексы в Приложении IV
Этап внедрения ошибки и возникновения ИЗ	На стадии разработки	Ошибки в требованиях, и спецификациях	22	I1,I2,I3,I4,I5,I6
		Ошибки в исходных Текстах программ	15	MT1,MT4,MU1,MU2, MU5,MU7,MU8, DT1.U2,U3,U4
		Ошибки в исполняемом коде	1	U1
	В ходе сопровождения		3	D1,MU3,MU9
	В ходе эксплуатации		9	I8,PC1,PC2,PC3, PC4,MA1,MA2, CA1.AT1

Рассмотрим подробнее перечисленные этапы с точки зрения возможности появления ошибок, приводящих к возникновению ИЗ.

3.3.2.1. Возникновение ИЗ в процессе разработки системы

Процесс разработки любой программной системы включает в себя несколько этапов. Прежде всего формулируются требования к системе, затем, исходя из этих требований, разрабатываются спецификации. На основании спецификаций создаются исходные тексты программ, которые затем компилируются и превращаются в исполняемый код. И на каждом из этих этапов в создаваемую систему могут быть внесены ошибки, которые приводят к возникновению ИЗ.

3.3.2.1.1. Составление требований и спецификаций

Требования к программному обеспечению описывают, *что* должна делать программа. Спецификации определяют то, *каким образом* эти действия должны выполняться. Требования и спецификации взаимосвязаны

настолько тесно, что относить обусловленные ими ИЗ к разным категориям представляется нецелесообразным.

Маловероятно, что требования или спецификации могут содержать положения, явно обуславливающие преднамеренные ошибки и каналы утечки информации. И требования, и спецификации достаточно открыты, поддаются пониманию и анализу, что позволяют относительно легко выявить и устранить ошибки типа наличия "черного хода" и им подобные.

Более реально появление ошибок, обусловленных необходимостью одновременного выполнения как требований безопасности, так и общих требований к функциональным возможностям системы. Конкуренция этих требований и неизбежно возникающие противоречия требуют от разработчиков принятия компромиссных решений, в которых предпочтение может быть отдано функциональности системы в ущерб ее безопасности. В качестве примера можно привести выполнение пользователем программ в режиме супервизора с целью увеличения быстродействия или разрешение на модификацию кода прямо во время выполнения программ для осуществления отладки.

3.3.2.1.2. Создание исходных текстов программ

Создание исходных текстов программ обеспечивает воплощение требований и спецификаций и является логическим продолжением соответствующих этапов. Большинство ошибок (но не все) в исходных текстах, как случайных, так и внесенных преднамеренно, может быть обнаружено при тщательном их изучении.

Наиболее распространены случайные ошибки в исходных текстах программ. Чаще всего они возникают в результате неадекватной реализации определенных в требованиях интерфейсов, либо просто из-за ошибок программистов.

Преднамеренные ошибки могут быть внесены в систему по целому ряду причин. Программист может внедрить в систему код, не предусмотренный ее спецификациями, но необходимый ему для отладки и тестирования разрабатываемой программы. Однако, если по завершению разработки этот код не будет удален из системы, он превратится в реальный канал утечки информации и может быть использован злоумышленником. Самый известный пример такого рода — программа sendmail, позволившая широко распространиться сетевому вирусу Морриса (U10 в Приложении IV).

3.3.2.1.3. Генерация исполняемого кода

Исполняемый код генерируется компиляторами на основе исходных текстов программ и представляет собой инструкции процессора. Поскольку компиляторы предназначены только для формального преобразования исходных текстов в исполняемый код, они автоматически переносят ошибки из первых во второй. Однако, если ошибки содержатся в самом компиляторе, они могут использоваться злоумышленниками для получения в компилируемых программах нужных им фрагментов кода (U1 в Приложении IV).

3.3.2.2. Возникновение ИЗ в процессе сопровождения и развития системы

Случайные ошибки, внесенные в систему во время ее сопровождения, чаще всего обусловлены неправильным представлением программистами всех аспектов функционирования системы в целом. Любые изменения, вносимые ими в систему, потенциально могут превратиться в каналы утечки информации. Поэтому каждое вносимое в ПО изменение должно сопровождаться тщательной проверкой всей системы в целом, так как если бы это было испытание абсолютно новой системы.

3.3.2.3. Возникновение ИЗ в процессе функционирования системы

Возникновение ошибок и сбоев, утечка информации и другие подобные явления в процессе функционирования системы в большинстве случаев происходят по причине воздействия на нее специально написанных программ (РПС).

Хорошо известные случаи вирусных атак являются наиболее ярким обоснованием необходимости постоянного контроля и анализа состояния системы и исполняемых файлов [19]. Основной задачей такого анализа в процессе функционирования системы является выявление несанкционированной модификации каких-либо фрагментов исполняемого кода. Очевидно, что целью РПС является не просто модификация кода, а нарушение функционирования системы и, возможно, доступ к конфиденциальной информации, перехват паролей, создание скрытых каналов утечки и т.д.

3.3.3. Классификация ИЗ по размещению в системе

ИЗ можно классифицировать по их размещению в ВС, в зависимости от того, в каких компонентах системы они могут находиться. Большинство ошибок, приводящих к возникновению ИЗ и нарушению требо-

ваний защиты, присутствует в программном обеспечении, хотя они могут встречаться и в аппаратуре. В данной работе основное внимание уделено исследованию таксономии ИЗ в программном обеспечении вообще и в операционных системах в частности, однако функционирование программ всецело зависит от аппаратной платформы. Это, а также то, что ИЗ может использовать ошибки аппаратуры, определяет необходимость внесения в классификацию соответствующих категорий: ошибки в программном обеспечении и ошибки аппаратных платформ (табл. 3.3).

Компоненты программного обеспечения, вне зависимости от их конкретного назначения, чрезвычайно сильно связаны и взаимозависимы. Поэтому предлагаемое разделение программ по категориям носит достаточно условный характер.

Среди всего комплекса программного обеспечения в отдельную категорию в первую очередь выделим операционную систему. В ней определена и реализована архитектура всей вычислительной системы, и наличие в ней ошибок, связанных с обеспечением безопасности, автоматически повлечет за собой серьезные последствия для всей ВС в целом.

Непосредственно с ОС связано сервисное программное обеспечение, обеспечивающее поддержку различных аспектов функционирования системы. Кроме того, существует прикладное программное обеспечение, с которым непосредственно работают пользователи.

3.3.3.1. Системное программное обеспечение

Операционные системы обычно включают в себя функции управления процессами, устройствами, распределением памяти, файловой системой и т.д. Кроме того, они обеспечивают инициализацию вычислительной системы при загрузке и содержат основные механизмы обеспечения безопасности. Однако, поскольку состав этих механизмов не стандартизован и сильно различается от системы к системе, в данной классификации выделен только механизм идентификации/аутентификации — как один из важнейших и присущий всем системам без исключения.

Поэтому, ИЗ в операционной системе будем разделять по следующим категориям:

- инициализация (загрузка) системы;
- управление распределением памяти;
- управление устройствами;
- управление процессами;
- управление файловой системой;
- идентификация и аутентификация.

Таблица 3.3. Классификация ИЗ по размещению в вычислительной системе

				Кол-во примеров	Индексы в Приложении IV
Размещение в ВС	Программное обеспечение	Операционная система	Инициализация (загрузка)	8	U5,U13,PC2,PC4, MA1,MA2,AT1,CA1
			Управление выделением памяти	2	MT3,MU5
			Управление процессами	10	I6,I9,MT1,MT2,MU2,MU3, MU4,MU6,U7,UN1
			Управление устройствами	3	I2,I3,I4
			Управление файловой системой	6	I1,I5,MU8,U2,U3,U9
			Средства идентификации и аутентификации	5	MU1,DT1,U6,U11,D1
			Другие	1	MT4
		Сервисные программы и утилиты	Привилегированные утилиты	10	I7,B1,U4,U7,U8,U10, U12,U14,PC1,PC3
			Непривилегированные утилиты	1	U1
		Прикладные программы		1	I8
	Аппаратное обеспечение			3	MU9,D2,IN1

Для ошибок, не попадающих ни в одну из данных категорий, введем дополнительную категорию "другие".

Процесс инициализации системы достаточно четко определен по своим функциям, но все же достаточно сложен и значительно различается от системы к системе. Ошибки на этапе инициализации могут возникнуть в результате неправильного взаимодействия с аппаратурой (например, если произошли изменения в составе аппаратных средств), или при задании неверных параметров конфигурации. Ошибки такого рода приводят, как правило, к неправильному назначению полномочий доступа к ресурсам системы.

Управление процессами и управление распределением памяти — основные задачи операционной системы, и ошибки в данных механизмах приводят к получению злоумышленником контроля над всей системой и свободному доступу к любой информации.

Управление устройствами подразумевает наличие комплекса программ ввода/вывода, обеспечивающих функционирование этих устройств параллельно и независимо от центрального процессора. Ошибки в таких программах, например, ошибки приема/передачи управляющих параметров и команд устройствам (возможно являющиеся результатом преднамеренной подмены после проверки допустимости значений) приводят либо к отказам и сбоям в работе устройств, либо позволяют злоумышленнику получить информацию, доступ к которой ему запрещен.

Файловая система использует значительное число функций операционной системы — управление процессами, устройствами, распределением памяти и т.д. Соответственно, ошибки в этих компонентах автоматически распространяются и на файловую систему.

Кроме того, файловая система может содержать и собственные ошибки, касающиеся хранения данных и ограничения доступа к ним. Неверное представление данных влечет за собой неправильное функционирование механизмов контроля.

Таким образом, средства обеспечения безопасности, принадлежащие операционной системе, оказываются непосредственно связанными с механизмами управления файловой системой. Наличие ошибок в механизмах управления файловой системой способно привести к нарушению функционирования и безопасности всей ВС в целом.

Идентификация и аутентификация являются отправными точками в функционировании любой системы защиты. Как правило, операционная система содержит специальные файлы, в которых хранятся имена и пароли, на основании которых и выполняются указанные процедуры. Чрезвычайно важно обеспечить не только адекватную реализацию процедур идентификации и аутентификации, но и всестороннюю защиту этих фай-

лов от несанкционированного доступа и изменения, иначе злоумышленник легко сможет выдать себя за легального пользователя и получить соответствующие полномочия.

3.3.3.2. Сервисное программное обеспечение

Термин "сервисное программное обеспечение" включает в себя компиляторы, отладчики, редакторы, библиотеки функций, системы управления базами данных и т.п. Операционная система при запуске таких программ обычно предоставляет им специальные привилегии, превышающие привилегии работающего с ними пользователя. Поэтому о сервисном программном обеспечении можно говорить как о множестве привилегированных утилит.

Привилегированные утилиты, как правило, являются сложными программами, и часто обеспечивают выполнение функций, не предусмотренных операционной системой. Кроме того, они разрабатываются отдельно от последней и могут не поддерживать принятые в ней ограничения и требования безопасности, даже при наличии собственной защиты. Это означает, что привилегированные утилиты являются потенциально опасными с точки зрения защиты ВС в целом.

Наличие ошибок в реализации защиты привилегированных утилит или каналов утечки информации в них может быть использовано злоумышленником, который в случае успеха получит возможности, соответствующие специальным привилегиям утилиты, с которой он работает. Наиболее известным примером подобного рода является наличие специального бита доступа SUID у многих программ ОС Unix (U5 в Приложении IV).

3.3.3.3. Прикладное программное обеспечение

Нарушения в функционировании вычислительных систем, вызванные неумышленными ошибками в прикладном программном обеспечении, обычно ограничиваются только содержащим эту ошибку процессом, который либо некорректно функционирует либо останавливается или зацикливается. Однако это не означает, что все ошибки в прикладном программном обеспечении столь же безобидны. Преднамеренно внесенные программные закладки, вирусы, троянские кони и логические бомбы находятся именно на уровне прикладного программного обеспечения. Объектами их атак потенциально могут стать любые компоненты вычислительной системы, и последствия такого воздействия могут быть очень серьезными — вплоть до выведения операционной системы из строя. В этом случае успех атаки зависит от того, насколько защищена конкретная

ОС от разрушительных действий прикладных программ. Многопользовательские многозадачные ОС (такие как Unix) сравнительно легко справляются с подобной проблемой, а широко распространенные DOS и Windows в этой ситуации оказываются практически бессильными.

3.3.4 Результаты исследования таксономии ИЗ и их практическое применение

Рассмотренная таксономия ИЗ, основанная на результатах исследования [1], дает достаточно полное представление о классификации ИЗ с точки зрения источника их появления, этапа возникновения и размещения в ВС. Эти результаты, несомненно, представляют собой самостоятельный интерес и имеют ценность для пассивного исследования и классификации ИЗ. С точки зрения поиска путей и способов противостояния угрозам безопасности этого недостаточно, т. к. отсутствует классификация причин нарушений безопасности. Поэтому представляется необходимым развить эти исследования и провести анализ случаев нарушения безопасности с точки зрения таксономии причин появления ИЗ, чтобы получить представление о первоисточниках данного явления. Такая таксономия будет служить основой для разработки принципиально новых технологий и средств защиты, устраняющих причины существования ИЗ, и, следовательно, обеспечивающих максимальную безопасность.

3.4. Исследование причин возникновения ИЗ

По мнению авторов сложившаяся практика исследования случаев нарушения безопасности, уделяющая основное внимание методам и средствам преодоления защиты, обладает существенным недостатком — отталкиваясь от действий злоумышленника, она фактически представляет собой анализ технологии преодоления средств защиты и не позволяет выявить недостатки средств обеспечения безопасности.

Кроме того, подобный подход сразу разделяет все случаи нарушения безопасности на умышленные, классифицируемые по способам преодоления защиты, и неумышленные, обусловленные ошибками программирования. Авторы придерживаются прагматической точки зрения на этот вопрос — важен сам факт нарушения безопасности и те меры, которые необходимо предпринять для предотвращения подобных нарушений, а их преднамеренность не имеет значения. С этой точки зрения залог успешных действий злоумышленника как и предпосылки случайных нарушений предопределены свойствами самой ВС — ее архитектурой, реализацией и администрированием.

Это означает, что в основе каждого факта нарушения безопасности ВС лежит соответствующий изъян средств защиты, обусловивший успешное осуществление атаки. Анализ случаев нарушения безопасности должен основываться не столько на исследовании методов, используемых нарушителем, сколько на выявлении свойств системы, позволивших ему осуществить свои действия.

Поэтому для решения задачи данной главы — определения обстоятельств, приводящих к успешной реализации угроз безопасности, их систематизации и выявления первопричин их появления, предлагается разработанная авторами таксономия причин нарушений безопасности ВС, или причин возникновения ИЗ, которая связывает случаи нарушения безопасности с принципами организации защиты ВС, обусловившими их осуществимость. Таксономия причин возникновения ИЗ должна дать ответ на имеющий ключевое значение с практической точки зрения вопрос: что явилось причиной успешного осуществления нарушения безопасности в том или ином случае ?

Для ответа на него необходимо выявить те свойства и особенности архитектуры ВС, которые привели к возможности успешного осуществления соответствующих атак. Только знание природы этих причин позволит оценить способность системы противостоять атакам на ее безопасность, а также понять природу недостатков, присущих существующим средствам обеспечения безопасности, которые привели к соответствующим нарушениям, и построить защищенную систему, лишенную этих недостатков.

3.4.1. Таксономия причин возникновения ИЗ

Рассмотрим с этой точки зрения известные случаи нарушения безопасности ВС, используя в качестве примеров статистику из Приложения IV. Анализ показывает, что все случаи произошли по одной из следующих причин (рис. 3.1):

1. Выбор модели безопасности, не соответствующей назначению или архитектуре ВС. Модель безопасности (модели безопасности, их свойства и методы реализации рассмотрены в гл. 4) должна соответствовать как требованиям безопасности, предъявляемым к ВС, так и принятой в ней парадигме обработки информации, в основном определяемой архитектурой ОС. В настоящий момент наблюдается определенное несоответствие между моделями безопасности и архитектурой ОС. Фактически формальные модели безопасности существуют только в виде теории, а разработчики ОС вынуждены подвергать их интерпретации, чтобы приспособить к конкретной ОС.

Рис. 3.1. Таксономия причин нарушения безопасности ВС.

При этом приходится идти на определенные компромиссы, и может оказаться, что модель безопасности в ходе реализации подверглась существенным искажениям. Это означает, что при выборе модели безопасности нельзя не учитывать специфику архитектуры ВС, в противном случае, несмотря на все достоинства модели, гарантированного ею уровня безопасности достичь не удастся.

2. Неправильное внедрение модели безопасности. Несмотря на вполне адекватный выбор модели безопасности ее реализация и применение к архитектуре конкретной ОС в силу свойств самой модели или ОС было проведено неудачно. Это означает, что в ходе реализации были потеряны все теоретические достижения, полученные при формальном доказательстве безопасности модели. Это наиболее распространенная причина нарушения безопасности ВС. Обычно неправильное внедрение модели безопасности в систему выражается в недостаточном ограничении доступа к наиболее важным для безопасности системы службам и объектам, а также во введении различных исключений из предусмотренных моделью правил разграничения доступа типа привилегированных процессов, утилит и т. д. В качестве примеров можно привести случаи I1, I4, I6, I7, MU3, B1, UN1, U4, U5, U14 из Приложения IV.

3. Отсутствие идентификации и/или аутентификации субъектов и объектов. Во многих современных ОС (различные версии Unix, Novell Netware, Wndows) идентификация и аутентификация субъектов и объектов взаимодействия находятся на весьма примитивном уровне — субъект может сравнительно легко выдать себя за другого субъекта и воспользоваться его полномочиями доступа к информации. Кроме того, можно внедрить в систему "ложный" объект, который будет при взаимодействии выдавать себя за другой объект. Часто идентификация и аутентификация носят непоследовательный характер и не распространяются на все уровни взаимодействия — так, например, в ОС Novell Netware предусмотрена аутентификация пользователя, но отсутствует аутентификация рабочей станции и сервера. В стандартной версии ОС Unix аутентификация пользователей находится на очень примитивном уровне — программы подбора пароля легко справляются со своей задачей при наличии у злоумышленника идентификатора пользователя и зашифрованного пароля. Ряд служб ОС Unix (в первую очередь сетевые службы, использующие стек протоколов TCP/IP) и всемирная информационная сеть Internet в силу сложившихся обстоятельств и необходимости соблюсти требования по совместимости в глобальном масштабе вообще не предусматривает аутентификации при сетевых взаимодействиях [21](пример MU1 из Приложения IV).

4. Отсутствие контроля целостности средств обеспечения безопасности. Во многих ОС контролю целостности самих механизмов, реализующих функции защиты, уделяется слабое внимание, но, как уже говорилось, в условиях распространения РПС контроль целостности приобретает решающее значение. Многие системы допускают прозрачную для служб безопасности подмену компонентов. Например, ОС Unix традиционно построена таким образом, что для обеспечения ее функционирования многие процессы должны выполняться с уровнем полномочий, превышающим обычный пользовательский уровень (с помощью механизма замены прав пользователя на права владельца программы). Такие приложения являются потенциальной брешью в системе защиты, т.к. нуждаются в проверке на безопасность при установке в систему и постоянном контроле целостности в ходе эксплуатации. С точки зрения безопасности такая ситуация является недопустимой, т. к. не соблюдается принцип минимальной достаточности при распределении полномочий пользователей и процессов. Список критичных приложений и пользователей, обладающих высоким уровнем привилегий должен быть максимальной ограничен. Этого можно достичь путем последовательного применения принципа локализации функций обеспечения безопасности и целостности в рамках ядра ОС (пример U1 Приложения IV).

5. Ошибки, допущенные в ходе программной реализации средств обеспечения безопасности. Эта группа причин нарушения безопасности будет существовать до тех пор, пока не появятся технологии программирования, гарантирующие производство безошибочных программ. По-видимому, такие технологии не появятся, и ошибки такого рода будут возникать всегда. Исчерпывающее тестирование и верификация программных продуктов (особенно реализующих функции защиты) позволяет сократить вероятность появления подобных ошибок (примеры MT1, MU2, DT1, D1, U2, U3, U7, U11, U12 Приложения IV).

6. Наличие средств отладки и тестирования в конечных продуктах. Многие разработчики оставляют в коммерческих продуктах т. н. "люки", "дыры", отладочные возможности и т.д. Наиболее известные примеры — отладочная опция в программе sendmail и встроенный отладчик ОС Novell NetWare. Причины, по которым это происходит, вполне понятны — программные продукты становятся все сложнее, и отладить их в лабораторных условиях становится просто невозможно. Следовательно, для определения причин сбоев и ошибок уже в процессе эксплуатации, разработчикам приходится оставлять в своих продуктах возможности для отладки и

диагностики в ходе эксплуатации. Очевидно, что для тех ситуаций, где безопасность имеет решающее значение, применение подобной практики недопустимо (пример U10 Приложения IV).

7. Ошибки администрирования. Наличие самых современных и совершенных средств защиты не гарантирует от возможных нарушений безопасности, т. к. в безопасности любой системы присутствует человеческий фактор — администратор, управляющий средствами обеспечения безопасности, может совершит ошибку, и все усилия разработчиков будут сведены на нет. Ошибки администрирования является достаточно распространенной причиной нарушений безопасности, но часто списываются на ошибки разработчиков средств защиты.

Предложенный подход к классификации причин нарушения безопасности в отличие от существующих подходов позволяет определить полное множество независимых первопричин нарушений безопасности, несводимых одна к другой, и образующих ортогональное пространство факторов, определяющих реальную степень безопасности системы.

3.4.2. Взаимосвязь таксономии причин нарушения безопасности и классификацией ИЗ

Рассмотрим взаимосвязь между рассмотренной таксономией причин возникновения ИЗ и предложенной в работе [1] классификацией источников появления и этапов внедрения ИЗ.

Сопоставление предложенной таксономии причин нарушений безопасности и классификации источников появления ИЗ [1] показано на рис. 3.2, на котором демонстрируется, что источником появления наибольшего количества категорий ИЗ является неправильное внедрение модели безопасности и ошибки в ходе программной реализации. Это означает, что эти причины являются более значимыми, и должны быть устранены в первую очередь.

На рис. 3.3 показано соответствие между причинами нарушений безопасности и классификацией ИЗ по этапу внесения[1]. Из рис 3.3 видно, что появление основных причин нарушения безопасности закладывается на этапе разработки, причем в основном на стадии задания спецификаций. Это вполне ожидаемый результат т. к. именно этап составления спецификаций является одним из самых трудоемких, а последствия ошибок в спецификациях сказываются на всех последующих этапах разработки и распространяются на все взаимосвязанные компоненты системы.

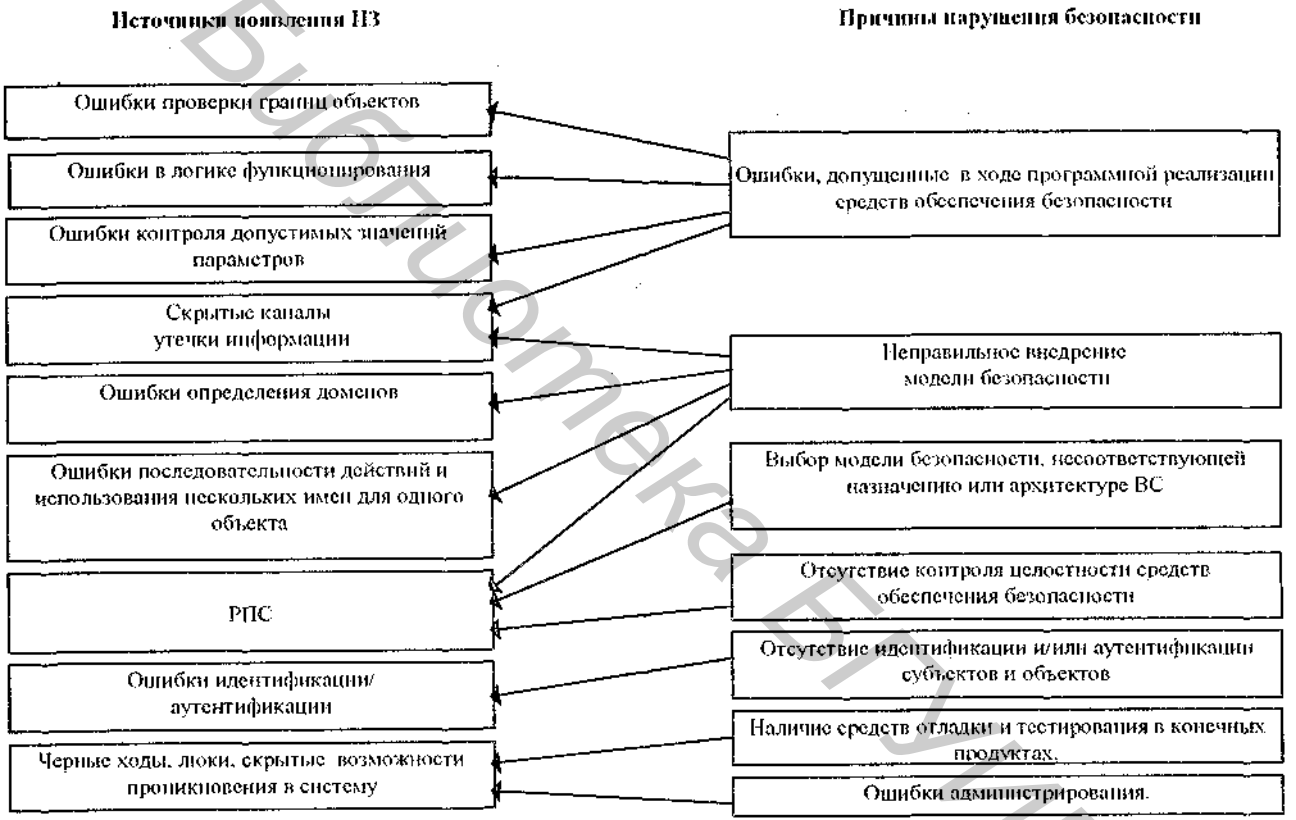


Рис.3.2. Сопоставление таксономии причин нарушения безопасности и классификации источников появления НЗ.

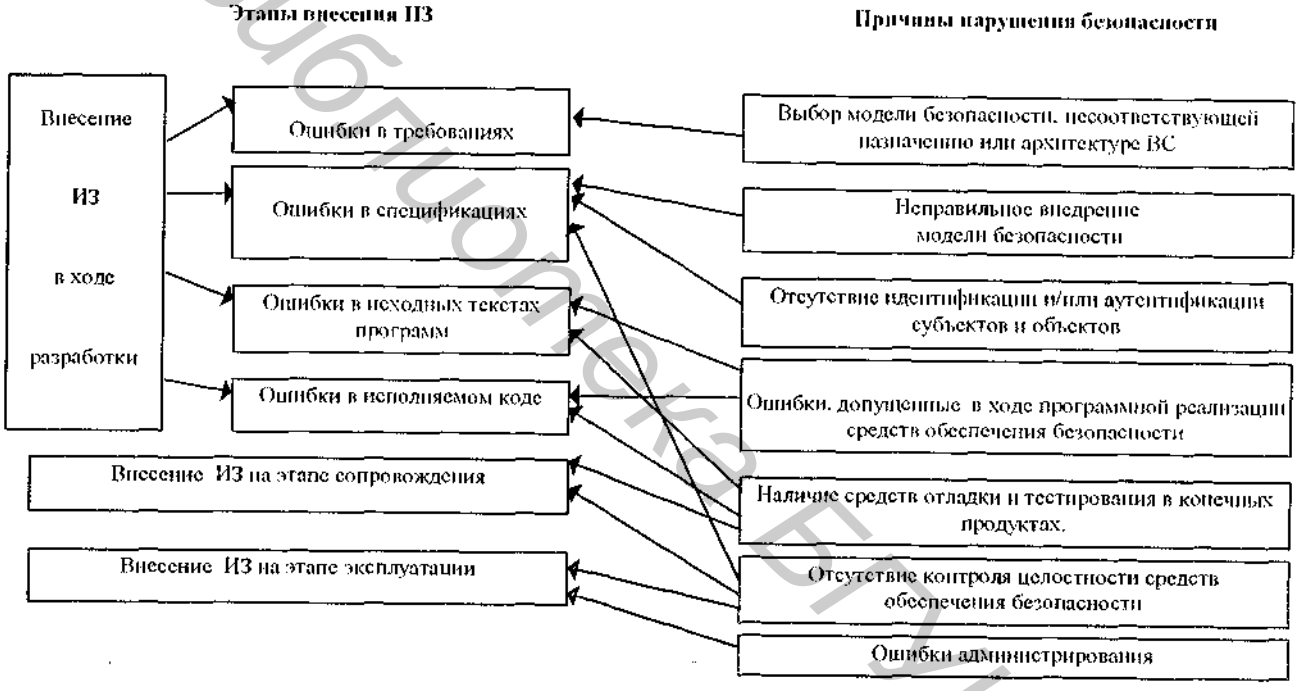


Рис. 3.3. Сопоставление таксономии причин нарушения безопасности и классификации НЗ по этапу внесения.

Поскольку большинство причин нарушения безопасности определяет появление ИЗ практически в любом компоненте ВС, сопоставление причин нарушения безопасности с классификацией ИЗ по размещению в системе проводить нецелесообразно.

Перекрестный анализ таксономии причин нарушений безопасности и классификаций ИЗ по источнику появления и этапу внесения показывает, что наиболее значимые причины (неправильное внедрение модели безопасности и ошибки программной реализации) действуют в ходе процесса разработки и реализации. Следовательно, именно на этих этапах должны быть сосредоточены основные усилия при создании защищенных систем.

Приведенное сопоставление показывает, что предложенные причины нарушения безопасности полностью охватывают как все аспекты появления, так и этапы внесения ИЗ. Таким образом, предложенная таксономия причин нарушения безопасности полностью подтверждается существующими исследованиями ИЗ, однако носит более принципиальный характер и, следовательно, обладает более высокой степенью общности.

3.5. Заключение к главе 3

Представленная таксономия причин нарушения безопасности позволяет определить значимость каждой из причин и выявить этапы разработки ВС, на которых они могут быть устранены. Как следует из проведенных исследований наибольшее значение имеют принципы организации защиты и управление доступом, т. е. те составляющие безопасности ВС, которые закладываются на ранних стадиях определения требований, выбора модели безопасности и архитектуры средств защиты. Все эти действия можно объединить в один этап — выбор технологии защиты ВС.

Кроме того, данная таксономия имеет и самостоятельное значение – ее можно использовать как основу для анализа и экспериментального тестирования систем защиты, т. к. она позволяет выбрать направления и способы проведения тестовых атак на наиболее уязвимые компоненты систем защиты. На основании данного подхода в Центре защиты информации Санкт-Петербургского технического университета были проведены исследования безопасности распространенных ОС и компьютерных сетей. Эти результаты на практике подтверждают правильность предложенной таксономии и доказывают адекватность рассмотренных причин нарушения безопасности механизмам действия реальных средств нарушения безопасности.

Разработанная авторами таксономия причин нарушения безопасности является первым и необходимым этапом для решения задач построения защищенных систем обработки информации, и может служить отправной точкой для адекватной реализации требований безопасности, и корректной реализации функций защиты.

Литература к главе 3

1. Carl E. Landwehr, Alan R. Bull, John P. McDermott, and William S.Choi. Information Technology Division, Code 5542, Naval Research Laboratory, Washington, D.C. 20375-5337// *A Taxonomy of Computer Security Flaws, with Examples*.

2. Abbott R. P., Chin J. S., Donnelley J. E., Konigsford W. L., Tokubo S and Webb D. A.1976 *Security analysis and enhancements of computer operating system*. NBSIR 76-1041, National Bureau of Standards, ICST, April 1976.

3. Anderson J. P. 1972. *Computer security technology planning study*. ESD-TR-73-51, Vols I and II, NTIS AD 758206, Hanscom Field, Bedford, MA (October 1972).

4. Bisbey II, R. and Hollingworth, D. 1978. *Protection analysis project report*. ISI/RR-78-13, DTIC AD A056816, USC/Information Sciences Institute (May 1978).

5. Brehmer C. L. and Carl J. R. Hollingworth, 1993 *Incorporating IEEE Standard 1044 into your anomaly tracking process*. CrossTalk, J. Defense Software Engineering, 6 (Jan.1993), 9-16.

6. Chillarege R., Bhandari I. S., Chaar J. K., Halliday M. J., Moebus D. S., Ray B. K., and Wong, M-Y. 1992. *Orthogonal defect classification-a concept for in-process measurements*. IEEE Trans. on Software Engineering, 18,11, (Nov. 1992), 943-956.

7. Cohen, F. 1984. *Computer viruses: theory and experiments*, 7th DoD/NBS Computer Security Conference, Gaithersburg, MD (Sept. 1984), 240-263.

8. Federal Criteria for Information Technology Security. USA National Institute of Standards and Technology & USA National Security Agency.

9. Florac, W. A. 1992. *Software Quality Measurement: A Framework for Counting Problems and Defects*. CMU/SEI-92-TR-22, Software Engineering Institute, Pittsburgh, PA, (Sept.).

10. Lampson, B. W. 1973. *A note on the confinement problem*, Comm. ACM 16,10 (October), 613-615.

11. Leveson N. and Turner C. S. 1992. *An investigation of the Therac-25 accidents*. UCI TR92-108, Inf. and Comp. Sci. Dept, of Cal.-Irvine, Irvine, CA.

12. Linde R. R. 1975. Operating system penetration. AFIPS national computer Conference, 361-368.

13. McDermott J.P. 1988. *A technique for removing an important class of Trojan horses from high order languages*, Proc.11th National Computer Security Conference NBS/NCSC, Gaithersburg, MD (October.), 114-117.

14. Pfleeger C. P. 1989. *Security in Computing*. Prentice Hall, Englewood Cliffs, NJ.

15. Schoch J. F. and Hupp J. A. 1982. *The "worm" programs-early experience with a distributed computation*. Comm. ACM.25,3 (March) 172-180.

16. Sullivan M.R. and Chillarege R. 1992. *A comparison of software defects in database management systems and operating systems*. Proc.22nd Int. Sump. on Fault-Tolerant Computer Systems (FTCS-22), Boston, MA IEEE CS Press.

17. Weiss D. M. and Basili V. R. 1985. *Evaluating software development by analysis of changes: some data from the Software Engineering Laboratory*. IEEE Trans. Software Engineering SE-11,2 (February), 157-168.

18. Use of A Taxonomy of Security Faults. COAST Laboratory, Purdue University, Technical Report TR-96-051.

19. Д. Зегжда, А. Мешков, П. Семьянов, Д. Шведов. Как противостоять вирусной атаке. BHV — Санкт-Петербург, 1995 г. 318 стр.

20. Под ред. П. Д. Зегжда. Теория и практика обеспечения информационной безопасности. М.: Издательство Агентства "Яхтсмен". 1996 г.

Не путайте метод решения с постановкой задачи,
особенно если это ваш собственный метод.
Д. Гаусс и Дж. Уейнберг

Глава 4. Теоретические основы методов защиты информационных систем

Технология защиты информационных систем начала развиваться относительно недавно, но сегодня уже существует значительное число теоретических моделей, позволяющих описывать практически все аспекты безопасности и обеспечивать средства защиты формально подтвержденной алгоритмической базой. Однако на практике далеко не всегда удается воспользоваться результатами этих исследований, потому что слишком часто теория защиты не согласуется с реальной жизнью. Дело в том, что теоретические исследования в области защиты информационных систем носят разрозненный характер и не составляют комплексной теории безопасности. Все существующие теоретические разработки основаны на различных подходах к проблеме, вследствие чего предлагаемые ими постановки задачи обеспечения безопасности и методы ее решения существенно различаются. Наибольшее развитие получили два подхода, каждый из которых основан на своем видении проблемы безопасности и нацелен на решение определенных задач — это формальное моделирование политики безопасности и криптография. Причем эти различные по происхождению и решаемым задачам подходы дополняют друг друга: криптография может предложить конкретные методы защиты информации в виде алгоритмов идентификации, аутентификации, шифрования и контроля целостности, а формальные модели безопасности предоставляют разработчикам защищенных систем основополагающие принципы, которые лежат в основе архитектуры защищенной системы и определяют концепцию ее построения.

Данная глава содержит краткий обзор наиболее распространенных моделей безопасности и криптографических методов идентификации, аутентификации, шифрования и контроля целостности. При создании данной главы авторы не ставили своей задачей полное изложение математической теории со всеми теоремами и формальными доказательствами, но хотели представить читателю базовые принципы, на которых строятся методы защиты информационных систем, и показать каким образом эта теория может помочь при решении практических задач.

4.1. Формальные модели безопасности

Напомним, что под политикой безопасности понимается совокупность норм и правил, регламентирующих процесс обработки информации, выполнение которых обеспечивает защиту от определенного множества угроз и составляет необходимое (а иногда и достаточное) условие безопасности системы. Формальное выражение политики безопасности называют моделью политики безопасности.

Для чего же нужны модели безопасности? Неужели нельзя обойтись неформальным описанием политики безопасности, ведь составление формальных моделей требует существенных затрат и привлечения высококвалифицированных специалистов, они трудны для понимания и требуют определенной интерпретации для применения в реальных системах. Тем не менее формальные модели необходимы и используются достаточно широко, потому что только с их помощью можно доказать безопасность системы опираясь при этом на объективные и неопровержимые постулаты математической теории. По своему назначению модели безопасности аналогичны аэродинамическим моделям самолетов и моделям плавучести кораблей — и те, и другие позволяют обосновать жизнеспособность системы и определяют базовые принципы ее архитектуры и используемые при ее построении технологические решения. Основная цель создания политики безопасности информационной системы и описания ее в виде формальной модели — это определение условий, которым должно подчиняться поведение системы, выработка критерия безопасности и проведение формального доказательства соответствия системы этому критерию при соблюдении установленных правил и ограничений. На практике это означает, что только соответствующим образом уполномоченные пользователи получат доступ к информации, и смогут осуществлять с ней только санкционированные действия.

Кроме того, формальные модели безопасности позволяют решить еще целый ряд задач, возникающих в ходе проектирования, разработки и сертификации защищенных систем, поэтому их используют не только теоретики информационной безопасности, но и другие категории специалистов, участвующих в процессе создания и эксплуатации защищенных информационных систем (производители, потребители, эксперты-квалификаторы).

Производители защищенных информационных систем используют модели безопасности в следующих случаях:

- при составлении формальной спецификации политики безопасности разрабатываемой системы;
- при выборе и обосновании базовых принципов архитектуры защищенной системы, определяющих механизмы реализации средств защиты;

- в процессе анализа безопасности системы в качестве эталонной модели;
- при подтверждении свойств разрабатываемой системы путем формального доказательства соблюдения политики безопасности.

Потребители путем составления формальных моделей безопасности получают возможности довести до сведения производителей свои требования в четко определенной и непротиворечивой форме, а также оценить соответствие защищенных систем своим потребностям.

Эксперты по квалификации в ходе анализа адекватности реализации политики безопасности в защищенных системах используют модели безопасности в качестве эталонов.

В данном разделе изложены основные положения наиболее распространенных политик безопасности, основанных на контроле доступа субъектов к объектам, и моделирующих поведение системы с помощью пространства состояний, одни из которых являются безопасными, а другие — нет. Все рассматриваемые модели безопасности основаны на следующих базовых представлениях:

1. Система является совокупностью взаимодействующих сущностей — *субъектов* и *объектов*. *Объекты* можно интуитивно представлять в виде контейнеров, содержащих информацию, а *субъектами* считать выполняющиеся программы, которые воздействуют на *объекты* различными способами. При таком представлении системы безопасность обработки информации обеспечивается путем решения задачи управления доступом *субъектов* к *объектам* в соответствии с заданным набором правил и ограничений, которые образуют политику безопасности. Считается, что система безопасна, если субъекты не имеют возможности нарушить правила политики безопасности. Необходимо отметить, что общим подходом для всех моделей является именно разделение множества сущностей, составляющих систему, на множества субъектов и объектов, хотя сами определения понятий *объект* и *субъект* в разных моделях могут существенно различаться.

2. Все взаимодействия в системе моделируются установлением отношений определенного типа между субъектами и объектами. Множество типов отношений определяется в виде набора операций, которые субъекты могут производить над объектами.

3. Все операции контролируются монитором взаимодействий и либо запрещаются, либо разрешаются в соответствии с правилами политики безопасности.

4. Политика безопасности задается в виде правил, в соответствии с которыми должны осуществляться все взаимодействия между субъектами и объектами. Взаимодействия, приводящие к нарушению этих правил, пресекаются средствами контроля доступа и не могут быть осуществлены.

5. Совокупность множеств субъектов, объектов и отношений между ними (установившихся взаимодействий) определяет *состояние* системы. Каждое состояние системы является либо *безопасным*, либо *небезопасным* в соответствии с предложенным в модели критерием безопасности.

6. Основной элемент модели безопасности — это доказательство утверждения (теоремы) о том, что система, находящаяся в безопасном состоянии, не может перейти в небезопасное состояние при соблюдении всех установленных правил и ограничений.

Среди моделей политик безопасности можно выделить два основных класса: дискреционные (произвольные) и мандатные (нормативные). Мы рассмотрим наиболее распространенные политики произвольного управления доступом, в основе которых лежат модель Харрисона-Руззо-Ульмана и модель типизованной матрицы доступа, фундаментальную нормативную модель безопасности Белла-ЛаПадулы, а также модели двух прикладных политик — ролевой политики и политики доменов и типов.

4.1.1. Дискреционная модель Харрисона-Руззо-Ульмана

Модель безопасности Харрисона-Руззо-Ульмана[1], являющаяся классической дискреционной моделью, реализует произвольное управление доступом субъектов к объектам и контроль за распространением прав доступа.

В рамках этой модели система обработки информации представляется в виде совокупности активных сущностей — субъектов (множество S), которые осуществляют доступ к информации, пассивных сущностей — объектов (множество O), содержащих защищаемую информацию, и конечного множества прав доступа $R = \{r_1, \ldots, r_n\}$, означающих полномочия на выполнение соответствующих действий (например, *чтение*, *запись*, *выполнение*).

Причем для того, чтобы включить в область действия модели и отношения между субъектами, принято считать, что все субъекты одновременно являются и объектами — $S \subset O$. Поведение системы моделируется с помощью понятия состояния. Пространство состояний системы образуется декартовым произведением множеств составляющих ее объектов, субъектов и прав — $O \times S \times R$. Текущее состояние системы Q в этом пространстве определяется тройкой, состоящей из множества субъектов, множества объектов и матрицы прав доступа M, описывающей текущие права доступа субъектов к объектам, — $Q=(S,O,M)$. Строки матрицы соответствуют субъектам, а столбцы — объектам, поскольку множество объектов включает в себя множество субъектов, матрица имеет вид прямоугольника. Любая ячейка матрицы M[s,o] содержит набор прав субъекта s к объекту o, принадлежащих множеству прав доступа R. Поведение

системы во времени моделируется переходами между различными состояниями. Переход осуществляется путем внесения изменений в матрицу M с помощью команд следующего вида:

command $\alpha(x_1,.., x_k)$
  if r_1 **in** $M[x_{s_1}, x_{o_1}]$ **and** (условия выполнения команды)
    r_2 **in** $M[x_{s_2}, x_{o_2}]$ **and**
    .
    .
    r_m **in** $M[x_{s_m}, x_{o_m}]$

  then
    $op_1, op_2 \ldots op_n$ (операции, составляющие команду)

Здесь α — имя команды; x_i — параметры команды, являющиеся идентификаторами субъектов и объектов, s_i и o_i — индексы субъектов и объектов в диапазоне от 1 до k; op_i - элементарные операции. Элементарные операции, составляющие команду, выполняются только в том случае, если все условия, означающие присутствие указанных прав доступа в ячейках матрицы M, являются истинными. В классической модели допустимы только следующие элементарные операции:

enter r **into** M[s,o] (добавление субъекту s права r для объекта o)
delete r **from** M[s,o] (удаление у субъекта s права r для объекта o)
create subject s (создание нового субъекта s)
create object o (создание нового объекта o)
destroy subject s (удаление существующего субъекта s)
destroy object o (удаление существующего объекта o)

Применение любой элементарной операции op в системе, находящейся в состоянии Q=(S,O,M) влечет за собой переход в другое состояние Q'=(S',O',M'), которое отличается от предыдущего состояния Q по крайней мере одним компонентом. Выполнение базовых операций приводит к следующим изменениям в состоянии системы:

enter r **into** M[s, o] (где $s \in S$, $o \in O$)
  $O' = O$
  $S' = S$
  $M'[x_s, x_o] = M[x_s, x_o]$, если $(x_s, x_o) \neq (s,o)$
  $M'[s, o] = M[s, o] \cup \{r\}$

delete r **from** M[s, o] (где $s \in S$, $o \in O$)

$O' = O$
$S' = S$
$M'[x_s, x_o] = M[x_s, x_o]$, если $(x_s, x_o) \neq (s, o)$
$M'[s, o] = M[s, o] \setminus \{r\}$

create subject s (где $s \notin S$)
$O' = O \cup \{s\}$
$S' = S \cup \{s\}$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S \times O$
$M'[s, x_o] = \varnothing$ для всех $x_o \in O'$
$M'[s, x_s] = \varnothing$ для всех $x_s \in S'$

destroy subject s (где $s \in S$)
$O' = O \setminus \{s\}$
$S' = S \setminus \{s\}$
$M[x_s, x_o]' = M[x_s, x_o]$ для всех $(x_s, x_o) \in S' \times O'$

create object o (где $o \notin O$)
$O' = O \cup \{o\}$
$S' = S$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S \times O$
$M'[x_s, o] = \varnothing$ для всех $x_s \in S'$

destroy object o (где $o \in O \setminus S$)
$O' = O \setminus \{o\}$
$S' = S$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S' \times O'$

Операция **enter** вводит право r в существующую ячейку матрицы доступа. Содержимое каждой ячейки рассматривается как множество, т.е. если это право уже имеется, то ячейка не изменяется. Операция называется **enter** *монотонной*, поскольку она только добавляет права в матрицу доступа и ничего не удаляет. Действие операции **delete** противоположно действию операции **enter**. Она удаляет право из ячейки матрицы доступа, если оно там присутствует. Поскольку содержимое каждой ячейки рассматривается как множество, **delete** не делает ничего, если удаляемое право отсутствует в указанной ячейке. Поскольку **delete** удаляет информацию из матрицы доступа, она называется *немонотонной* операцией. Операции **create subject** и **destroy subject** представляют собой аналогичную пару монотонной и немонотонной операции.

Заметим, что для каждой операции существует еще и предусловие ее выполнения: для того чтобы изменить ячейку матрицы доступа с по-

мощью операций **enter** или **delete** необходимо, чтобы эта ячейка существовала, т. е. чтобы существовали соответствующие субъект и объект. Предусловиями операций создания **create subject/object**, является отсутствие создаваемого субъекта/объекта, операций удаления **destroy subject/object** — наличие субъекта/объекта. Если предусловие любой операции не выполнено, то ее выполнение безрезультатно.

Формальное описание системы $\Sigma(Q,R,C)$ состоит из следующих элементов:

1. конечный набор прав доступа $R = \{r_1, \ldots, r_n\}$;
2. конечные наборы исходных субъектов $S_0 = \{s_1, \ldots, s_l\}$ и объектов $O_0 = \{o_1, \ldots, o_m\}$, где $S_0 \subseteq O_0$;
3. исходная матрица доступа, содержащая права доступа субъектов к объектам — M_0;
4. конечный набор команд $C=\{\alpha_i(x_1,.., x_k)\}$, каждая из которых состоит из условий выполнения и интерпретации в терминах перечисленных элементарных операций.

Поведение системы во времени моделируется с помощью последовательности состояний $\{Q_i\}$, в которой каждое последующее состояние является результатом применения некоторой команды из множества C к предыдущему $Q_{n+1} = C_n(Q_n)$. Таким образом для заданного начального состояния только от условий команд из C и составляющих их операций зависит, сможет ли система попасть в то или иное состояние, или нет. Каждое состояние определяет отношения доступа, которые существуют между сущностями системы в виде множества субъектов, объектов и матрицы прав. Поскольку для обеспечения безопасности необходимо наложить запрет на некоторые отношения доступа, для заданного начального состояния системы должна существовать возможность определить множество состояний, в которые она сможет из него попасть. Это позволит задавать такие начальные условия (интерпретацию команд C, множества объектов O_0, субъектов S_0 и матрицу доступа M_0), при которых система никогда не сможет попасть в состояния, нежелательные с точки зрения безопасности. Следовательно, для построения системы с предсказуемым поведением необходимо для заданных начальных условий получить ответ на вопрос: сможет ли некоторый субъект s когда-либо приобрести право доступа r для некоторого объекта o?

Поэтому критерий безопасности модели Харрисона-Руззо-Ульмана формулируется следующим образом:

Для заданной системы начальное состояние $Q_0=(S_0,O_0,M_0)$ *является безопасным относительно права* r, *если не существует применимой к* Q_0 *последовательности команд, в результате которой право* r *будет занесено в ячейку матрицы* M, *в которой оно отсутствовало в состоянии* Q_0.

Смысл данного критерия состоит в том, что для безопасной конфигурации системы субъект никогда получит право r доступа к объекту, если он не имел его изначально. На первый взгляд такая формулировка кажется довольно странной, поскольку невозможность получения права r вроде бы влечет за собой отказ от использования команд, в которых присутствует операция **enter** r **into** M[s,o], однако это не так. Дело в том, что удаление субъекта или объекта приводит к уничтожению всех прав в соответствующей строке или в столбце матрицы, но не влечет за собой уничтожение самого столбца или строки и сокращение размеров матрицы. Следовательно, если в какой-то ячейке в начальном состоянии существовало право r, и после удаления субъекта или объекта, к которым относилось это право, ячейка будет очищена, но впоследствии в результате создания субъекта или объекта появится вновь, и в эту ячейку с помощью соответствующей команды **enter** снова будет занесено право r, то это не будет означать нарушения безопасности.

Из критерия безопасности следует, что для данной модели ключевую роль играет выбор значений прав доступа и их использование в условиях команд. Хотя модель не налагает никаких ограничений на смысл прав и считает их равнозначными, те из них, которые участвуют в условиях выполнения команд, фактически представляют собой не права доступа к объектам (как, например, чтение и запись), а права управления доступом, или права на осуществление модификации ячеек матрицы доступа. Таким образом, по сути дела данная модель описывает не только доступ субъектов к объектам, а распространение прав доступа от субъекта к субъекту, поскольку именно изменение содержания ячеек матрицы доступа определяет возможность выполнения команд, в том числе команд, модифицирующих саму матрицу доступа, которые потенциально могут привести к нарушению критерия безопасности.

Необходимо отметить, что с точки зрения практики построения защищенных систем модель Харрисона-Руззо-Ульмана является наиболее простой в реализации и эффективной в управлении, поскольку не требует никаких сложных алгоритмов, и позволяет управлять полномочиями пользователей с точностью до операции над объектом, чем и объясняется ее распространенность среди современных систем. Кроме того, предложенный в данной модели критерий безопасности является весьма сильным в практическом плане, поскольку позволяет гарантированность недоступность определенной информации для пользователей, которым изначально не выданы соответствующие полномочия.

Однако, Харрисон, Руззо и Ульман доказали, что в общем случае не существует алгоритма, который может для произвольной системы, ее начального состояния $Q_0=(S_0,O_0,M_0)$ и общего права r решить, является ли данная конфигурация безопасной. Доказательство опирается на свой-

ства машины Тьюринга, с помощью которой моделируется последовательность переходов системы из состояния в состояние.

Для того, чтобы можно было доказать указанный критерий, модель должна быть дополнена рядом ограничений[2]. Мы не будем утомлять читателя многостраничными математическими выкладками, а только отметим, что указанная задача является разрешимой в любом из следующих случаев:

- команды $\alpha_i(x_1,.., x_k)$ являются монооперационными, т. е. состоят не более чем из одной элементарной операции;
- команды $\alpha_i(x_1,.., x_k)$ являются одноусловными и монотонными, т. е. содержат не более одного условия и не содержат операций **destroy** и **delete**;
- команды $\alpha_i(x_1,.., x_k)$ не содержат операций **create**.

Эти условия существенно ограничивает сферу применения модели, поскольку трудно представить себе реальную систему, в которой не будет происходить создания или удаления сущностей.

Кроме того, все дискреционные модели уязвимы по отношению к атаке с помощью "троянского коня", поскольку в них контролируются только операции доступа субъектов к объектам, а не потоки информации между ними. Поэтому, когда "троянская" программа, которую нарушитель подсунул некоторому пользователю, переносит информацию из доступного этому пользователю объекта в объект, доступный нарушителю, то формально никакое правило дискреционной политики безопасности не нарушается, но утечка информации происходит.

Таким образом дискреционная модель Харрисона-Руззо-Ульмана в своей общей постановке не дает гарантий безопасности системы, однако именно она послужила основой для целого класса моделей политик безопасности, которые используются для управления доступом и контроля за распространением прав во всех современных системах.

4.1.2. Типизованная матрица доступа

Другая дискреционная модель, получившая название "Типизованная матрица доступа" (Type Access Matrix — далее TAM)[3], представляет собой развитие модели Харрисона-Руззо-Ульмана, дополненной концепцией типов, что позволяет несколько смягчить те условия, для которых возможно доказательство безопасности системы.

Формальное описание модели TAM включает следующие элементы:

1. конечный набор прав доступа $R = \{r_1, \ldots, r_l\}$;
2. конечный набор типов $T = \{t_1, \ldots, t_g\}$;
3. конечные наборы исходных субъектов $S_0 = \{s_1, \ldots, s_n\}$ и объектов $O_0 = \{o_1, \ldots, o_m\}$, где $S_0 \subset O_0$;

4. матрица M, содержащая права доступа субъектов к объектам, и ее начальное состояние M_0;
5. конечный набор команд $C = \{\alpha_i(x_1,.., x_k)\}$, включающий условия выполнения команд и их интерпретацию в терминах элементарных операций

Тогда состояние системы описывается четверкой Q=(S,O,t,M), где S, O, и M обозначают соответственно множество субъектов, объектов и матрицу доступа, а t: $O \rightarrow T$ - функция, ставящая в соответствие каждому объекту некоторый тип.

Состояние системы изменяется с помощью команд из множества C. Команды TAM имеют тот же формат, что и в модели Харрисона-Руззо-Ульмана, но всем параметрам приписывается определенный тип:

command $\alpha(x_1:t_1,.., x_k:t_k)$
  if r_1 **in** $M[x_{S_1}, x_{O_1}]$ **and** (условия выполнения команды)
    r_2 **in** $M[x_{S_2}, x_{O_2}]$ **and**
    .
    .
    r_m **in** $M[x_{S_m}, x_{O_m}]$
  then
    $op_1, op_2 \ldots op_n$ (операции, составляющие команду)

Перед выполнением команды происходит проверка типов фактических параметров, и, если они не совпадают с указанными в определении, команда не выполняется. Фактически, введение контроля типов для параметров команд приводит к неявному введению дополнительных условий, т. к. команды могут быть выполнены только при совпадении типов параметров. В модели используются следующие шесть элементарных операций, отличающихся от аналогичных операций модели Харрисона-Руззо-Ульмана только использованием типизованных параметров:

Монотонные операции	Немонотонные операции
enter r **into** M[s,o]	**delete** r **from** M[s,o]
create subject s **of type** t	**destroy subject** s
create object o **of type** t	**destroy object** o

Смысл элементарных операций совпадает со смыслом аналогичных операций из классической модели Харрисона-Руззо-Ульмана с точностью до использования типов:

enter r **into** M[s, o] (где $s \in S$, $o \in O$)

$O' = O$
$S' = S$
$t'(o) = t(o)$ для всех $o \in O$
$M'[x_s, x_o] = M[x_s, x_o]$, если $(x_s, x_o) \neq (s, o)$
$M'[s, o] = M[s, o] \cup \{r\}$

delete r **from** M[s, o] (где $s \in S$, $o \in O$)

$O' = O$
$S' = S$
$t'(o) = t(o)$ для всех $o \in O$
$M'[x_s, x_o] = M[x_s, x_o]$, если $(x_s, x_o) \neq (s, o)$
$M'[s, o] = M[s, o] \setminus \{r\}$

create subject s **of type** t_s (где $s \notin S$)

$O' = O \cup \{s\}$
$S' = S \cup \{s\}$
$t'(o) = t(o)$ для всех $o \in O$
$t'(s) = t_s$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S \times O$
$M'[s, x_o] = \varnothing$ для всех $x_o \in O'$
$M'[s, x_s] = \varnothing$ для всех $x_s \in S'$

destroy subject s (где $s \in S$)

$O' = O \setminus \{s\}$
$S' = S \setminus \{s\}$
$t'(o) = t(o)$ для всех $O \in O'$
$t'(s) =$ не определено
$M[x_s, x_o]' = M[x_s, x_o]$ для всех $(x_s, x_o) \in S' \times O'$

create object o **of type** t_o (где $o \notin O$)

$O' = O \cup \{o\}$
$S' = S$
$t'(O) = t(O)$ для всех $O \in O$
$t'(o) = t_o$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S \times O$
$M'[x_s, o] = \varnothing$ для всех $x_s \in S'$

destroy object o (где $o \in O \setminus S$)

$O' = O \setminus \{o\}$

$S' = S$

$t'(x_o) = t(x_o)$ для всех $x_o \in O'$

$t'(o)$ = не определено

$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S' \times O'$

Таким образом ТАМ является обобщением модели Харрисона-Руззо-Ульмана, которую можно рассматривать как частный случай ТАМ с одним единственным типом, к которому относятся все объекты и субъекты. Появление в каждой команде дополнительных неявных условий, ограничивающих область применения команды только сущностями соответствующих типов, позволяет несколько смягчить жесткие условия классической модели, при которых критерий безопасности является разрешимым.

В работе[2] Харрисон, Руззо и Ульман показали, что критерий безопасности дискреционной модели может быть доказан для систем, в которых все команды $\alpha_i(x_1, .., x_k)$ являются одноусловными и монотонными. Строгий контроль соответствия типов позволяет смягчить требование одноусловности, заменив его ограничением на типы параметров команд, при выполнении которых происходит создание новых сущностей.

Для того чтобы сформулировать это ограничение определим отношения между типами. Пусть $\alpha(x_1{:}t_1, x_2{:}t_2, \ldots, x_k{:}t_k)$ – некоторая команда ТАМ. Будем говорить, что t_i является *дочерним типом* в α, если в теле α имеет место одна из следующих элементарных операций: **create subject** x_i **of type** t_i или **create object** x_i **of type** t_i. В противном случае будем говорить, что t_i является *родительским типом* в α.

Заметим, что в одной команде тип может быть одновременно и родительским, и дочерним, например:

command foo(s_1 : u, s_2 : u, s_3 : w, o : b)
create subject s_2 **of type** u;
create subject s_3 **of type** v;
end

Здесь u является родительским типом относительно s_1 и дочерним типом относительно s_2. Кроме того, w и b являются родительскими типами, а v - дочерним типом. Тогда можно описать взаимосвязи связи между различными типами с помощью графа, определяющего отношение "наследственности" между типами, устанавливаемые через операции порождения сущностей (объектов и субъектов). Такой граф называется *графом*

создания и представляет собой направленный граф с множеством вершин T, в котором ребро от u к v существует тогда и только тогда, когда в системе имеется создающая команда, в которой u является родительским типом, а v -дочерним типом. Этот граф для каждого типа позволяет определить:

1) сущности каких типов должны существовать в системе, чтобы в ней мог появиться объект или субъект заданного типа;

2) сущности каких типов могут быть порождены при участии сущностей заданного типа.

Модель монотонной типизированной матрицы доступа (MTAM) идентична TAM за исключением того, что в ней отсутствуют немонотонные элементарные операции **delete**, **destroy subject** и **destroy object**.

Реализация MTAM, состоящая из множеств объектов, субъектов, типов, матрицы прав доступа и множества команд, называется *ациклической* тогда и только тогда, когда ее граф создания не содержит циклов, в противном случае говорят, что реализация *циклическая*. Например, граф создания для приведенной выше команды foo, содержит следующие ребра: {(u, u), (u, v), (w, u), (w, v), (b, u), (b, v)}. Реализация MTAM, содержащая эту команду, будет циклической, поскольку тип u является для нее одновременно и родительским и дочерним, что приводит к появлению на графе цикла (u, u).

Доказано, что критерий безопасности, предложенный Харрисоном, Руззо и Ульманом, разрешим для ациклических реализаций MTAM, и что требование одноусловности команд можно заменить требованием ацикличности графа создания[3]. Смысл этой замены состоит в том, что последовательность состояний системы должна следовать некоторому маршруту на графе создания, поскольку невозможно появление сущностей дочерних типов, если в системе отсутствуют сущности родительских типов, которые должны участвовать в их создании. Отсутствие циклов на графе создания позволяет избежать зацикливания при доказательстве критерия безопасности, т. к. количество путей на графе без циклов является ограниченным.

Это означает, что поведение системы становится предсказуемым, поскольку в любом состоянии можно определить сущности каких типов могут появиться в системе, а каких — нет. Но, к сожалению, доказано, что в общем случае сложность проверки критерия безопасности для MTAM является NP-трудной задачей, т. е. с ростом размерности задачи (количества объектов и субъектов) время на ее решение растет в степенной зависимости от ее размерности. Этот недостаток может быть преодолен с помощью *тернарной* TAM, в которой команды могут иметь не более трех параметров. Тернарная MTAM является монотонной версией тернарной TAM. Для тернарной MTAM доказательство безопасности радикально упрощается, поскольку при проверке условной части команды всегда ис-

пользуется только небольшой фрагмент матрицы доступа. Тернарная МТАМ по своим выразительным способностям эквивалентна МТАМ, но несмотря на это, доказано, что безопасность ее ациклической реализации разрешима за время, зависящее от размера начальной матрицы доступа полиномиально.

Следовательно введение строгого контроля типов в дискреционную модель Харрисона-Руззо-Ульмана позволило доказать критерий безопасности систем для более приемлемых ограничений, что существенно расширило область ее применения.

4.1.3. Мандатная модель Белла-ЛаПадулы

Мандатная модель управления доступом основана на правилах секретного документооборота, принятых в государственных и правительственных учреждениях многих стран. Основным положением политики Белла-ЛаПадулы, взятым ими из реальной жизни, является назначение всем участникам процесса обработки защищаемой информации, и документам, в которых она содержится, специальной метки, например, *секретно, сов. секретно* и т. д, получившей название уровня безопасности. Все уровни безопасности упорядочиваются с помощью установленного отношения доминирования, например, уровень *сов. секретно* считается более высоким чем уровень *секретно,* или доминирует над ним. Контроль доступа осуществляется в зависимости от уровней безопасности взаимодействующих сторон на основании двух простых правил:

1. Уполномоченное лицо (субъект) имеет право читать только те документы, уровень безопасности которых не превышает его собственный уровень безопасности.

2. Уполномоченное лицо (субъект)имеет право заносить информацию только в те документы, уровень безопасности которых не ниже его собственного уровня безопасности.

Первое правило обеспечивает защиту информации, обрабатываемой более доверенными (высокоуровневыми) лицами, от доступа со стороны менее доверенных (низкоуровневых). Второе правило (далее мы увидим, что оно более важное) предотвращает утечку информации (сознательную или несознательную) со стороны высокоуровневых участников процесса обработки информации к низкоуровневым.

Таким образом, если в дискреционных моделях управление доступом происходит путем наделения пользователей полномочиями осуществлять определенные операции над определенными объектами, то мандатные модели управляют доступом неявным образом — с помощью назначения всем сущностям системы уровней безопасности, которые определяют все допустимые взаимодействия между ними. Следовательно, мандатное управление доступом не различает сущностей, которым присвоен

одинаковый уровень безопасности, и на их взаимодействия ограничения отсутствуют. Поэтому в тех ситуациях, когда управление доступом требует более гибкого подхода, мандатная модель применяется совместно с какой-либо дискреционной, которая используется для контроля за взаимодействиями между сущностями одного уровня и для установки дополнительных ограничений, усиливающих мандатную модель.

Система в модели безопасности Белла-ЛаПадулы[4], как и в модели Харрисона-Руззо-Ульмана, представляется в виде множеств субъектов S, объектов O(множество объектов включает множество субъектов, $S \subset O$) и прав доступа read(чтение) и write(запись). В мандатной модели рассматриваются только эти два вида доступа, и, хотя она может быть расширена введением дополнительных прав (например, правом на добавление информации, выполнение программ и т.д.), все они будут отображаться в базовые (чтение и запись). Использование столь жесткого подхода, не позволяющего осуществлять гибкое управление доступом, объясняется тем, что в мандатной модели контролируются не операции, осуществляемые субъектом над объектом, а потоки информации, которые могут быть только двух видов: либо от субъекта к объекту (запись), либо от объекта к субъекту (чтение).

Уровни безопасности субъектов и объектов задаются с помощью функции уровня безопасности $F: S \cup O \rightarrow L$, которая ставит в соответствие каждому объекту и субъекту уровень безопасности, принадлежащий множеству уровней безопасности L, на котором определена решетка Λ[5].

4.1.3.1. Решетка уровней безопасности

Решетка уровней безопасности Λ — это формальная алгебра $(L, \leq, \bullet, \otimes)$, где L — базовое множество уровней безопасности, а оператор $\leq$ определяет частичное нестрогое отношение порядка для элементов этого множества, т.е. оператор $\leq$ — антисимметричен, транзитивен и рефлексивен. Отношение $\leq$ на L:

1) рефлексивно, если $\forall a \in L: a \leq a$;

2) антисимметрично, если $\forall a_1, a_2 \in L: (a_1 \leq a_2 \wedge a_2 \leq a_1) \Rightarrow a_1 = a_2$;

3) транзитивно, если $\forall a_1, a_2, a_3 \in L: (a_1 \leq a_2 \wedge a_2 \leq a_3) \Rightarrow a_1 \leq a_3$.

Другое свойство решетки состоит в том, что для каждой пары a_1 и a_2 элементов множества L можно указать единственный элемент *наименьшей верхней границы* и единственный элемент *наибольшей нижней границы*. Эти элементы также принадлежат L и обозначаются с помощью операторов $\bullet$ и $\otimes$ соответственно:

$$a_1 \bullet a_2 = a \Leftrightarrow a_1, a_2 \leq a \wedge \forall a' \in L: (a' \leq a) \Rightarrow (a' \leq a_1 \vee a' \leq a_2)$$

$$a_1 \otimes a_2 = a \Leftrightarrow a \leq a_1, a_2 \wedge \forall a' \in L: (a' \leq a_1 \wedge a' \leq a_2) \Rightarrow (a' \leq a)$$

Смысл этих определений заключается в том, что для каждой пары элементов (или множества элементов, поскольку операторы • и ⊗ транзитивны) всегда можно указать единственный элемент, ограничивающий ее сверху или снизу таким образом, что между ними и этим элементом не будет других элементов.

Функция уровня безопасности F назначает каждому субъекту и объекту некоторый уровень безопасности из L, разбивая множество сущностей системы на классы, в пределах которых их свойства с точки зрения модели безопасности являются эквивалентными. Тогда оператор ≤ определяет направление потоков информации, то есть, если F(A) ≤ F(B), то информация может передаваться от элементов класса A элементам класса B.

Покажем, почему в модели Белла-ЛаПадулы для описания отношения доминирования на множестве уровней безопасности используется решетка.

Если информация может передаваться от сущностей класса A к сущностям класса B, а также от сущностей класса B к сущностям класса A, то классы A и B содержат одноуровневую информацию и с точки зрения безопасности эквивалентны одному классу (AB). Поэтому для удаления избыточных классов необходимо, чтобы отношение ≤ было антисимметричным.

Если информация может передаваться от сущностей класса A сущностям класса B, а также от сущностей класса B к сущностям класса C, то очевидно, что она будет также передаваться от сущностей класса A к сущностям класса C. Таким образом, отношение ≤ должно быть транзитивным.

Так как класс сущности определяет уровень безопасности содержащейся в ней информации, то все сущности одного и того же класса содержат с точки зрения безопасности одинаковую информацию. Следовательно, нет смысла запрещать потоки информации между сущностями одного и того же класса. Более того, из чисто практических соображений нужно предусмотреть возможность для сущности передавать информацию самой себе. Следовательно, отношение ≤ должно быть рефлексивным.

Покажем, что для любого множества сущностей должны существовать единственная наименьшая верхняя и наибольшая нижняя границы множества соответствующих им уровней безопасности. Для пары сущностей x и y, обладающих уровнями безопасности a и b соответственно, обозначим наибольший уровень безопасности их комбинации как (a•b), при этом a ≤ (a•b) и b ≤ (a•b). Тогда, если существует некоторый уровень c такой, что a ≤ c и b ≤c, то должно иметь место отношение (a•b) ≤ c, поскольку (a•b) — это минимальный уровень субъекта, для которого дос-

тупна информация как из x, так и из y. Следовательно, (a•b) должен быть наименьшей верхней границей a и b. Аналогично обозначим наименьший уровень безопасности комбинации сущностей x и y как (a⊗b), при этом (a⊗b)≤a и (a⊗b)≤b. Тогда, если существует некоторый уровень c такой, что c≤a и c≤b, то должно иметь место отношение c≤(a⊗b), поскольку (a⊗b) — это максимальный уровень субъекта, для которого разрешена передача информации как в x, так и в y. Следовательно, (a⊗b) должен быть наибольшей нижней границей a и b.

Использование решетки для описания отношений между уровнями безопасности позволяет использовать в качестве атрибутов безопасности (элементов множества L) не только целые числа, для которых определено отношение "меньше или равно", но и более сложные составные элементы. Например, в государственных организациях достаточно часто в качестве атрибутов безопасности используется комбинация, состоящая из уровня безопасности, представляющего собой целое число, и набора категорий из некоторого множества. Такие атрибуты невозможно сравнивать с помощью арифметических операций, поэтому отношение доминирования ≤ определяется как композиция отношения "меньше или равно" для уровней безопасности и отношения включения множеств ⊆ для наборов категорий. Причем это никак не сказывается на свойствах модели, поскольку отношения "меньше или равно" и "включение множеств" обладают свойствами антисимметричности, транзитивности и рефлексивности, и, следовательно, их композиция также будет обладать этими свойствами, образуя над множеством атрибутов безопасности решетку. Точно также можно использовать любые виды атрибутов и любое отношение частичного порядка, лишь бы их совокупность представляла собой решетку.

4.1.3.2. Классическая мандатная модель Белла-ЛаПадулы

В мандатных моделях функция уровня безопасности F вместе с решеткой уровней определяют все допустимые отношения доступа между сущностями системы, поэтому множество состояний системы V представляется в виде набора упорядоченных пар (F,M), где M – это матрица доступа, отражающая текущую ситуацию с правами доступа субъектов к объектам, содержание которой аналогично матрице прав доступа в модели Харрисона-Руззо-Ульмана, но набор прав ограничен правами read и write. Модель системы $\Sigma(v_0,R,T)$ состоит из начального состояния v_0, множества запросов R и функции перехода $T:(V\times R)\to V$, которая в ходе выполнения запроса переводит систему из одного состояния в другое. Система, находящаяся в состоянии $v\in V$, при получении запроса $r\in R$, переходит в следующее состояние $v^* = T(v,r)$. Состояние v достижимо в системе $\Sigma(v_0,R,T)$ тогда и только тогда, когда существует последователь-

ность $<(r_0,v_0),\ldots,(r_{n-1},v_{n-1}),(r_n,v)>$ такая, что $T(r_i, v_i) = v_{i+1}$ для $0 \le i < n$. Заметим, что для любой системы v_0 тривиально достижимо.

Как и для дискреционной модели состояния системы делятся на безопасные, в которых отношения доступа не противоречат установленным в модели правилам, и небезопасные, в которых эти правила нарушаются и происходит утечка информации.

Белл и ЛаПадула предложили следующее определение безопасного состояния:

1. Состояние (F,M) называется безопасным по чтению(или *просто безопасным*) тогда и только тогда, когда для каждого субъекта, осуществляющего в этом состоянии доступ чтения к объекту, уровень безопасности этого субъекта доминирует над уровнем безопасности этого объекта: $\forall s \in S, \forall o \in O$, $read \in M[s,o] \rightarrow F(s) \ge F(o)$.

2. Состояние (F,M) называется безопасным по записи(или **-безопасным*) тогда и только тогда, когда для каждого субъекта, осуществляющего в этом состоянии доступ записи к объекту, уровень безопасности этого объекта доминирует над уровнем безопасности этого субъекта: $\forall s \in S, \forall o \in O$, $write \in M[s,o] \rightarrow F(o) \ge F(s)$.

3. Состояние безопасно тогда и только тогда, когда оно безопасно и по чтению, и по записи.

В соответствии с предложенным определением безопасного состояния критерий безопасности системы выглядит следующим образом:

Система $\Sigma(v_0,R,T)$ *безопасна тогда и только тогда, когда ее начальное состояние* v_0 *безопасно и все состояния, достижимые из* v_0 *путем применения конечной последовательности запросов из* R *безопасны.*

Белл и ЛаПадула доказали теорему, формально доказывающую безопасность системы при соблюдении определенных условий, получившую название основной теоремы безопасности.

Основная теорема безопасности Белла-ЛаПадулы.

Система $\Sigma(v_0,R,T)$ безопасна тогда и только тогда, когда:

а) начальное состояние v_0 безопасно и

б) для любого состояния v, достижимого из v_0 путем применения конечной последовательности запросов из R таких, что $T(v,r)= v^*$, $v=(F,M)$ и $v^* =(F^*,M^*)$ для каждого $s \in S$ и $o \in O$ выполняются следующие условия:

1) если $read \in M^*[s,o]$ и $read \notin M[s,o]$, то $F^*(s) \ge F^*(o)$;

2) если $read \in M[s,o]$ и $F^*(s) < F^*(o)$, то $read \notin M^*[s,o]$;

3) если $write \in M^*[s,o]$ и $write \notin M[s,o]$, то $F^*(o) \ge F^*(s)$;

4) если $write \in M[s,o]$ и $F^*(o) < F^*(s)$, то $write \notin M^*[s,o]$.

Доказательство:

1. Необходимость. Если система безопасна, то состояние v_0 безопасно по определению. Допустим, существует некоторое состояние v^*, достижимое из v_0 путем применения конечного числа запросов из R и полученное путем перехода из безопасного состояния v: $T(v,r)= v^*$. Тогда, если при этом переходе нарушено хотя бы одно из первых двух ограничений, накладываемых теоремой на функцию T, то состояние v^* не будет безопасным по чтению, а если функция T нарушает хотя бы одно из последних двух условий теоремы, то состояние v^* не будет безопасным по записи. В любом случае при нарушении условий теоремы система небезопасна.

2. Достаточность. Проведем доказательство от противного. Предположим, что система небезопасна. В этом случае, либо v_0 небезопасно, что явно противоречит условиям теоремы, либо должно существовать небезопасное состояние v^*, достижимое из безопасного v_0 путем применения конечного числа запросов из R. В этом случае обязательно будет иметь место переход $T(v,r)= v^*$, при котором состояние v — безопасно, а состояние v^* — нет, однако четыре условия теоремы делают такой переход невозможным.

Таким образом, теорема утверждает, что система с безопасным начальным состоянием является безопасной тогда и только тогда, когда при любом переходе системы из одного состояния в другое не возникает никаких новых и не сохраняется никаких старых отношений доступа, которые будут небезопасны по отношению к функции уровня безопасности нового состояния. Формально эта теорема определяет все необходимые и достаточные условия, которые должны быть выполнены для того, чтобы система, начав свою работу в безопасном состоянии, никогда не достигла небезопасного состояния.

4.1.3.3. Безопасная функция перехода

Недостаток основной теоремы безопасности Белла-ЛаПадулы состоит в том, что ограничения, накладываемые теоремой на функцию перехода, совпадают с критериями безопасности состояния, поэтому данная теорема является избыточной по отношению к определению безопасного состояния. Кроме того, из теоремы следует только то, что все состояния, достижимые из безопасного состояния при определенных ограничения, будут в некотором смысле безопасны, но при этом не гарантируется, что они будут достигнуты без потери свойства безопасности в процессе осуществления перехода. Поскольку у нас нет никаких определенных ограничений на вид функции перехода, кроме указанных в условиях теоремы, и допускается, что уровни безопасности субъектов и объектов могут изменяться, то можно представить такую гипотетическую систему (она получила название Z-системы[7]), в которой при попытке низкоуровневого

субъекта прочитать информацию из высокоуровневого объекта будет происходить понижение уровня объекта до уровня субъекта и осуществляться чтение. Функция перехода Z-системы удовлетворяет ограничениям основной теоремы безопасности, и все состояния такой системы также являются безопасными в смысле критерия Белла-ЛаПадулы, но вместе с тем в этой системе любой пользователь сможет прочитать любой файл, что, очевидно, несовместимо с безопасностью в обычном понимании.

Следовательно, необходимо сформулировать теорему, которая бы не только констатировала, безопасность всех достижимых состояний для системы, соответствующей определенным условиям, но и гарантировала бы безопасность в процессе осуществления переходов между состояниями. Для этого необходимо регламентировать изменения уровней безопасности при переходе от состояния к состоянию с помощью дополнительных правил.

Такую интерпретацию мандатной модели осуществил Мак-Лин[6], предложивший свою формулировку основной теоремы безопасности, основанную не на понятии *безопасного состояния*, а на понятии *безопасного перехода*. При таком подходе функция уровня безопасности представляется с помощью двух функций, определенных на множестве субъектов и объектов: F_s:S→L и F_o: O→L.

Функция перехода T считается безопасной по чтению, если для любого перехода T(r,v)=v*, выполняются следующие три условия:

если read ∈ M*[s,o] и

read ∉ M[s,o] то:

$F_s(s) \geq F_o(o)$ и F=F*;

если $F_s \neq F^*_s$ то:

M=M*,

$F_o=F^*_o$,

для ∀ s и o, для которых $F^*_s(s)<F^*_o(o)$, read ∉ M[s,o];

если $F_o \neq F^*_o$ то:

M=M*,

$F_s=F^*_s$,

для ∀ s и o, для которых $F^*_s(s)<F^*_o(o)$, read ∉ M[s,o].

Функция перехода T считается безопасной по записи, если для любого перехода T(r,v) =v* выполняются следующие три условия:

если write ∈ M*[s,o] и

write ∉ M[s,o] то:

$F_o(o) \geq F_s(s)$ и $F=F^*$;

если $F_s \neq F^*_s$ то:

$M=M^*$,
$F_o=F_o^*$,
для $\forall$ s и o, для которых $F^*_s(s)>F^*_o(o)$, write $\notin$ $M[s,o]$;

если $F_o \neq F^*_o$ то:

$M=M^*$,
$F_s=F^*_s$,
для $\forall$ s и o, для которых $F^*_s(s)>F^*_o(o)$, write $\notin$ $M[s,o]$.

Функция перехода является безопасной тогда и только тогда, когда она одновременно безопасна и по чтению и по записи.

Смысл введения перечисленных ограничений и их принципиальное отличие от условий теоремы Белла-ЛаПадулы состоит в следующем: нельзя изменять одновременно более одного компонента состояния системы — в процессе перехода либо возникает новое отношение доступа, либо изменяется уровень объекта, либо изменяется уровень субъекта.

Следовательно, *функция перехода является безопасной тогда и только тогда, когда она изменяет только один из компонентов состояния и изменения не приводят к нарушению безопасности системы.*

Поскольку безопасный переход из состояния v в состояние v^* позволяет изменяться только одному элементу из v и так как этот элемент может быть изменен только способами, сохраняющими безопасность состояния, была доказана следующая теорема о свойствах безопасной системы[7]:

Теорема безопасности Мак-Лина. Система безопасна в любом состоянии и в процессе переходов между ними, если ее начальное состояние является безопасным, а ее функция перехода удовлетворяет критерию Мак-Лина.

Обратное утверждение неверно. Система может быть безопасной по определению Белла-ЛаПадулы, но не иметь безопасной функции перехода, о чем свидетельствует рассмотренный пример Z-системы.

Такая формулировка основной теоремы безопасности предоставляет в распоряжение разработчиков защищенных систем базовый принцип их построения, в соответствии с которым для того, чтобы обеспечить безопасность системы как в любом состоянии, так и в процессе перехода

между ними, необходимо реализовать для нее такую функцию перехода, которая соответствует указанным условиям.

4.1.3.4. Уполномоченные субъекты

Формулировка основной теоремы безопасности в интерпретации Мак-Лина позволяет расширить область ее применения по сравнению с классической теоремой Белла-ЛаПадулы, однако, используемый критерий безопасности перехода не всегда соответствует требованиям контроля доступа, возникающим на практике. Поскольку в процессе осуществления переходов могут изменяться уровни безопасности сущностей системы, желательно контролировать этот процесс, явным образом разрешая или запрещая субъектам осуществлять подобные переходы. Для решения этой задачи Мак-Лин расширил базовую модель путем выделения подмножества уполномоченным субъектов, которым разрешается инициировать переходы, в результате которых у сущностей системы изменяются уровни безопасности[7]. Система с уполномоченными субъектами также описывается множествами S, O и L, смысл которых совпадает с аналогичными понятиями модели Белла-ЛаПадулы, а ее состояние также описывается набором упорядоченных пар (F,M), причем функция перехода F и матрица отношений M доступа играют ту же роль. Новым элементом модели является функция управления уровнями $C:S\cup O\rightarrow \mathcal{P}(S)$ (здесь и далее $\mathcal{P}(S)$ обозначает множество всех подмножеств S). Эта функция определяет подмножество субъектов, которым позволено изменять уровень безопасности, для заданного объекта или субъекта. Модель системы $\Sigma(v_0,R,T^a)$ состоит из начального состояния v_0, множества запросов R и функции перехода T^a, которая переводит систему из состояния в состояние по мере выполнения запросов. Но теперь у функции перехода, которая определяет следующее состояние системы после выполнения определенным субъектом некоторого запроса, появился еще один аргумент — субъект, от которого исходит этот запрос, поскольку результат перехода зависит от того, какой субъект его инициировал: $T^a:(S\times V\times R)\rightarrow V$. Как и прежде система, находящаяся в состоянии $v\in V$, при получении запроса $r\in R$, от субъекта $s\in S$ переходит из состояния v в состояние $v^*=T^a(s,v,r)$.

Функция перехода в модели с уполномоченными субъектами T^a называется *авторизованной функцией перехода* тогда и только тогда, когда для каждого перехода $T^a(s,v,r)=v^*$, при котором $v=(F,M)$ и $v^*=(F^*,M^*)$ выполняется следующее условие:

для $\forall\, x\in S\cup O$: если $F^*(x)\neq F(x)$, то $s\in C(x)$.

Другими словами, в ходе авторизованного перехода уровень безопасности субъекта или объекта может изменяться только тогда, когда

субъект, выполняющий переход, принадлежит множеству субъектов, уполномоченных изменять уровень этого субъекта или объекта.

С точки зрения модели уполномоченных субъектов система $\Sigma(v_0, R, T^a)$ считается безопасной в том случае, если:

1) начальное состояние v_0 и все состояния, достижимые из него путем применения конечного числа запросов из R являются безопасными по критерию Белла-ЛаПадулы;

2) функция перехода T^a является *авторизованной функцией перехода* согласно предложенному определению.

Отметим, что из этого определения следует только необходимое условие безопасности системы. В качестве достаточного условия может использоваться совокупность критерия авторизации функции перехода и критериев безопасного состояния Белла-ЛаПадулы, либо критериев безопасности функции перехода Мак-Лина.

4.1.3.5. Модель совместного доступа

Практическое применение всех представленных формулировок мандатной модели безопасности ограничено еще одним фактором — они не учитывают широко распространенные в государственных учреждениях правила, согласно которым доступ к определенной информации или модификация ее уровня безопасности могут осуществляться только в результате совместных действий нескольких пользователей (т. н. групповой доступ). Например, может потребоваться, чтобы гриф секретности документа мог изменяться только с обоюдной санкции его владельца и администратора безопасности. В системе обработки информации это может быть реализовано либо как параллельное выполнение несколькими пользователями специальной программы, изменяющей уровень безопасности, либо последовательной обработкой запроса несколькими пользователями, каждый из которых должен санкционировать его выполнение. Однако на уровне политики безопасности механизм реализации не имеет значения, поскольку он никак не отражается на формальной модели системы.

Для того, чтобы мандатная модель предусматривала совместный доступ необходимо модифицировать ее следующим образом. Вместо множества субъектов системы S будем рассматривать множество непустых подмножеств S, которое обозначим как $\mathbf{S}=\wp(S)\setminus\{\varnothing\}$. Расширим матрицу прав доступа, отражающую текущее состояние доступа в системе, путем добавления в нее строк, содержащих права групповых субъектов, и обозначим ее как $\mathbf{M}$. Кроме функции уровня безопасности $F: S \cup O \to L$ для групповых субъектов вводятся дополнительные функции: $\mathbf{F}^L: \mathbf{S} \to L$, такая, что $\mathbf{F}^L(\mathbf{s})$ есть наибольшая нижняя граница множества $\{F(s) \mid s \in \mathbf{s}\}$ и $\mathbf{F}^H: \mathbf{S} \to L$, такая, что $\mathbf{F}^H(\mathbf{s})$ есть наименьшая верхняя граница множества

$\{F(s) \mid s \in \mathbf{s}\}$. Например, если $x \in S$ и $y \in S$ с уровнями безопасности *секретно* и *совершенно секретно* соответственно, то для субъекта $\{x\} \in \mathbf{S}$ $F^H(\{x\}) = F^L(\{x\}) =$ *секретно*, для субъекта $\{y\} \in \mathbf{S}$ $F^H(\{y\}) = F^L(\{y\}) =$ *совершенно секретно*, но для группового субъекта $\{x,y\} \in \mathbf{S}$ значением $F^L(\{x,y\})$ является *секретно*, а значением $F^H(\{x,y\})$ — *совершенно секретно*. Соответственно, если право $write \in \mathbf{M}[x,o_1]$ и право $write \in \mathbf{M}[\{x,y\},o_2]$, то x имеет индивидуальный доступ записи в o_1, а x и y имеют групповой доступ записи в o_2, т. е. x и y могут изменять o_2, но только в том случае, если они будут делать это совместно.

Критерии безопасности состояния для такой системы формулируются следующим образом:

- состояние системы является безопасным по чтению тогда и только тогда, когда для каждого индивидуального или группового субъекта, имеющего в этом состоянии доступ *чтения* к объекту, наибольшая нижняя граница множества уровней безопасности этого субъекта доминирует над уровнем безопасности этого объекта: $\forall \mathbf{s} \in \mathbf{S}$, $\forall o \in O$, $read \in \mathbf{M}[\mathbf{s},o] \rightarrow F^L(\mathbf{s}) \geq F(o)$.
- состояние системы является безопасным по записи тогда и только тогда, когда для каждого индивидуального или группового субъекта, имеющего в этом состоянии доступ *записи* к объекту, уровень безопасности этого объекта доминирует над наименьшей верхней границей множества уровней безопасности этого субъекта: $\forall \mathbf{s} \in \mathbf{S}$, $\forall o \in O$, $write \in \mathbf{M}[\mathbf{s},o] \rightarrow F(o) \geq F^H(\mathbf{s})$.

Как и прежде состояние безопасно тогда и только тогда, когда оно одновременно безопасно и по чтению и по записи.

Благодаря тому, что множество уровней безопасности и отношение доминирования образуют решетку, удалось задать функции, определяющие границы множества уровней безопасности групповых субъектов, таким образом, что одни условия безопасности одновременно учитывают как индивидуальные, так и совместные доступы.

Переопределим функцию перехода, которая определяет следующее состояние системы после выполнения определенным субъектом некоторого запроса, как $T:(V \times R) \rightarrow V$, где $T(v,r) = v^*$, причем в описании состояния $v = ((F,F^H,F^L),\mathbf{M})$ и $v^* = ((F^*,F^{*H},F^{*L})^*,\mathbf{M}^*)$ участвуют три функции уровня безопасности: F — для объектов, F^H и F^L — наименьшая верхняя и наибольшая нижняя границы для групповых субъектов.

Тогда теорема Белла-ЛаПадулы для совместного доступа формулируется следующим образом:

Система $\Sigma(v_0,R,\mathbf{T})$ безопасна тогда и только тогда, когда:

а) начальное состояние v_0 безопасно и

б) функция перехода **T** такова, что для любого состояния v, достижимого из v_0 путем применения конечной последовательности запросов из R, таких, что T(v,r)=v*, v=((F,F^H,F^L),**M**) и v*=((F*,F^{*H},F^{*L})*,**M***) для каждого **s**∈**S**, и o∈O выполняются следующие условия:

1) если read ∈ **M***[**s**,o] и read ∉ M[**s**,o], то F^{*L}(**s**)≥ F*(o);

2) если read ∈ **M**[**s**,o] и F^{*L}(**s**)<F*(o), то read ∉ **M***[**s**,o];

3) если write ∈ **M***[**s**,o] и write ∉ **M**[**s**,o], то F*(o)≥ F^{*H}(**s**);

4) если write ∈ **M**[**s**,o] и F*(o)< F^{*H}(**s**), то write ∉ **M***[**s**,o].

Заметим, что аналогичные рассуждения можно провести и в отношении множественного доступа к нескольким объектам сразу (групповым объектам), когда в ходе одной операции читаются или записываются сразу несколько объектов. В этом случае множество объектов и матрица прав доступа расширяются аналогичным образом за счет групповых объектов, для которых вводятся аналогичные функции верней и нижней границы уровня безопасности.

4.1.3.6. Безопасная функция перехода для модели совместного доступа

Общность методов применяемых при построении мандатных моделей безопасности и использование математического аппарата решеток позволяют комбинировать модели практически произвольным образом. Тот же подход, на основании которого для модели совместного доступа были сформулированы критерии безопасности состояния Белла-ЛаПадулы и основная теорема безопасности, позволяет сформулировать для нее критерии безопасного перехода и теорему Мак-Лина.

Для краткости будем использовать обозначение F_s для функции уровня безопасности индивидуальных субъектов. Поскольку значения функций F^H и F^L границ множества уровней безопасности групповых субъектов однозначно определяются уровнями безопасности составляющих их индивидуальных субъектов, то, если в результате перехода из одного состояния в другое F_s осталась без изменений, из этого следует, что F^H и F^L также не изменились. С учетом этого определения сформулируем критерии безопасного перехода для модели с множественным доступом.

Функция перехода **T** для модели совместного доступа считается безопасной по чтению, если для любого перехода T(v,r)=v*, при котором v=((F,F^H,F^L),**M**) и v*=((F*,F^{*H},F^{*L})*,**M***) выполняются следующие три условия:

если read ∈ **M***[**s**,o] и
read ∉ **M**[**s**,o] то:
F^{*L}(**s**)≥ F^*_o(o) и F_s=F^*_s, F_o=F^*_o;
если F_s ≠ F^*_s то:
M=**M***

$F_o=F_o^*$,
для $\forall$ **s**,o для которых $F^{*L}(s)<F^*_o(o)$, read $\notin$ **M**[**s**,o];

если $F_o \neq F^*_o$ то:

M=**M***,
$F_s=F^*_s$,
для $\forall$ **s**,o для которых $F^{*L}(s)<F^*_o(o)$, read $\notin$ **M**[**s**,o].

Функция перехода τ считается безопасной по записи, если для любого перехода $\tau(v,r)=v^*$, выполняются следующие три условия:

если write $\in$ **M***[**s**,o] и
write $\notin$ **M**[**s**,o] то:

$F^*(o) \geq F^{*H}(s)$ и $F_s=F^*_s$, $F_o=F^*_o$;

если $F_s \neq F^*_s$ то:

M=**M***,
$F_o=F_o^*$,
для $\forall$ **s**,o для которых $F^{*H}(s)>F^*_o(o)$, write $\notin$ **M**[**s**,o];

если $F_o \neq F^*_o$ то:

M=**M***,
$F_s=F^*_s$,
для $\forall$ **s**,o для которых $F^{*H}(s)>F^*_o(o)$, write $\notin$ **M**[**s**,o].

Функция перехода является безопасной тогда и только тогда, когда она одновременно безопасна и по чтению и по записи.

Смысл введенных ограничений ничем не отличается от критериев безопасности перехода для классической постановки мандатной модели, но в них учитываются совместные доступы групповых субъектов. Следовательно, теорема Мак-Лина о свойствах безопасной системы будет верна и для модели совместного доступа, с учетом сформулированного для нее критерия безопасного перехода.

4.1.3.7. Модель совместного доступа с уполномоченными субъектами

Чтобы сформулировать условие авторизации функции перехода для системы с уполномоченными субъектами, поддерживающей совместный доступ, переопределим функцию перехода τ добавив к ее аргументам субъект (групповой или индивидуальный), который инициирует переход

— $T^a:(S_\times V_\times R)_\to V$. Поскольку уровень безопасности группового субъекта {x,y} определяется уровнями безопасности составляющих его субъектов {x} и {y}, достаточно контролировать только изменения уровней безопасности индивидуальных субъектов. Поэтому область определения функции управления уровнями C можно ограничить множествами объектов и простых субъектов, а ее область значений необходимо расширить за счет групповых субъектов: $C:S_\cup O_\to \tilde{P}(S)$. Как и прежде, система, находящаяся в состоянии $v_\in V$, при получении запроса $r_\in R$ от субъекта $s_\in S$ переходит из v в состояние $v^* = T^a(s,v,r)$.

Функция перехода T^a является *авторизованной функцией перехода* тогда и только тогда, когда для каждого перехода $T^a(s,v,r)=v^*$, при котором $v=((F,F^H,F^L),M)$ и $v^*=((F^*,F^{*H},F^{*L})^*,M^*)$ выполняются следующие условия:

(1) для всех $x_\in S$ если $F^{*H}(x)\neq F^H(x)$ или $F^{*L}(x)\neq F^L(x)$, то $s_\in C(x)$;

(2) для всех $o_\in O$ если $F^*(o)\neq F(o)$, то $s_\in C(o)$.

Система $\Sigma(v_0,R,T^a)$ с совместным доступом и уполномоченными субъектами считается безопасной в том случае, если:

1) начальное состояние v_0 и все состояния, достижимые из него путем применения конечного числа запросов из R являются безопасными по критерию Белла-ЛаПадулы;

2) функция перехода T^a является *авторизованной функцией перехода*.

Как и для модели индивидуального доступа из этого определения следует только необходимое условие безопасности системы.

4.1.3.8 Решетка мандатных моделей.

Итак, мы рассмотрели шесть вариантов мандатной модели, которые различаются представлением взаимодействующих сущностей (индивидуальные или групповые субъекты), правилами контроля за изменением уровней безопасности (наличие/отсутствие уполномоченных субъектов), а также критериями безопасности (безопасные состояния или безопасные переходы). Соответственно, различны и доказанные для каждой из рассмотренных моделей теоремы о свойствах безопасности: необходимые и достаточные условия безопасности состояний для классической модели и модели совместного доступа, условие авторизации функции перехода, изменяющей уровни безопасности, для моделей с уполномоченными субъектами, достаточное условие безопасности состояний и переходов для моделей безопасного перехода. Вместе с тем, несмотря на некоторые различия, все рассмотренные модели построены на одних и тех же принципах, поскольку все они применяют единый механизм представления атрибутов безопасности в виде решетки и используют одни и те же мето-

ды доказательства свойств безопасности. Благодаря этому рассмотренные модели можно связать между собой отношениями обобщения/конкретизации, которые показаны на рис. 4.1.

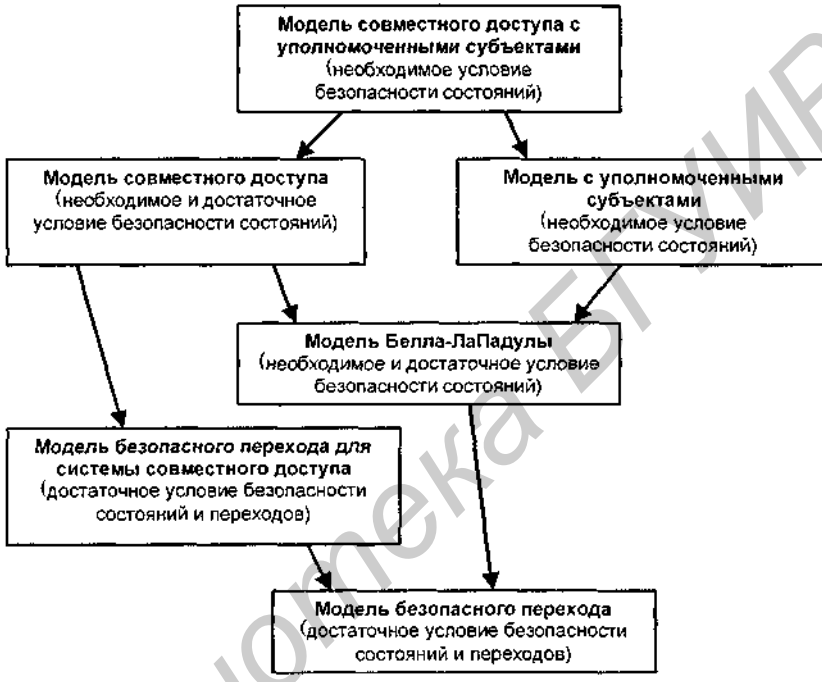


Рис. 4.1. Мандатные модели безопасности.

Каждая стрелка на этом рисунке означает, что модель, из которой она исходит, является обобщением модели, на которую она указывает. Или, другими словами, модель, на которую указывает стрелка, представляет собой частный случай модели, из которой она исходит. Транзитивные отношения не показаны. Отношения между мандатными моделями, показанные на рис. 4.1 позволяют сделать следующие заключения:

1. Модель совместного доступа представляет собой строгое обобщение классической модели Белла-ЛаПадулы, которая является ее частным случаем. Поскольку уровни безопасности всех групповых субъектов определяются уровнями составляющих их индивидуальных субъектов, то ограничения доступа для индивидуальных субъектов $\{s\} \in S$ идентичны ограничениям классической модели, однако все ограничения модели Белла-ЛаПадулы для субъектов $s \in S$, действуют и в отношении любого груп-

пового субъекта $s \in S$, содержащего s. Поэтому модель совместного доступа является более строгой, чем классическая модель Белла-ЛаПадулы. Например, в соответствии с моделью группового доступа субъекту будет отказано в совместном доступе *записи* к объекту, несмотря на то, что он имеет к нему одиночный доступ *записи*, только из-за того, что субъект, с которым он желает разделять доступ, не имеет такого права. Проще говоря, для каждой операции права групповых субъектов определяются правами входящего в нее субъекта, который наименее уполномочен в отношении этой операции и этого объекта.

2. Модели с уполномоченными субъектами обобщают модели, не предусматривающие контроль за изменением уровней, которые можно рассматривать как их частный случай, когда все субъекты уполномочены изменять уровни безопасности любых субъектов и объектов. Поэтому условия на функцию перехода, налагаемые моделями с уполномоченными субъектами, являются дополнительными по отношению к критериям безопасности других моделей.

3. Предложенные Мак-Лином критерий безопасности функции перехода и теорема о достаточных условиях безопасности системы во всех достижимых состояниях и при переходах между ними дополняют условия классической модели Белла-ЛаПадулы и позволяют доказать безопасность системы в процессе осуществления переходов. Поэтому модели безопасного перехода являются конкретизацией моделей безопасного состояния. Следовательно, множество систем с безопасной функцией перехода представляет собой подмножество систем, безопасных по критерию Белла-ЛаПадулы.

Таким образом, единый подход к представлению системы в виде последовательности состояний и описанию взаимодействий, на котором основаны все мандатные модели, связывает все изложенные теоремы о свойствах безопасных систем в единую шкалу критериев безопасности, различающиеся достаточными и необходимыми условиями. Мак-Лин показал, что на множестве мандатных моделей безопасности может быть построена формальная алгебра[7]. Отношение обобщения/конкретизации, показанное на рис. 4.1 является отношением порядка, позволяет сравнивать модели по степени строгости ограничений, и образует на множестве мандатных моделей формальную решетку. Модель M_1 является более строгой, чем модель M_2 ($M_1 \le M_2$), когда она является ее подмножеством ($M_1 \cap M_2 = M_1$). Отношение $M_1 \le M_2$ означает, что если система является безопасной в соответствии моделью M_1, то она тем более безопасна с точки зрения модели M_2, или что модель M_2 является обобщением модели M_1. Решетка моделей упрощает использование мандатных моделей при решении различных задач, поскольку она упорядочивает критерии безопасности — необходимые условия распространяются по иерархии сверху

вниз (от общих моделей к конкретным), а достаточные — снизу наверх(от конкретных к обобщенным). Если некоторое необходимое условие безопасности доказано для модели M_1, то оно будет справедливо и для любой модели $M_2 \leq M_1$. Напротив, любое достаточное условие безопасности, доказанное для модели M_2 будет справедливо и для любой модели $M_1 \leq M_2$.

Отсюда следует, что наиболее эффективным решением задачи создания защищенной системы является построение ее функции перехода в соответствии с достаточными условиями модели безопасного перехода. При этом система будет безопасна как в любом состоянии, так и процессе переходов между ними.

С точки зрения анализа безопасности существующих систем предварительная проверка необходимых условий, сформулированных для моделей уполномоченных субъектов, модели Белла-ЛаПадулы и модели совместного доступа, в случае их невыполнения позволяет избежать трудоемкой проверки достаточных условий.

Кроме того, решетка мандатных моделей позволяет выбирать для каждого конкретного применения наиболее подходящую по уровню ограничений модель, без необходимости проведения формального доказательства безопасности, и предоставляет возможность с однозначным результатом сравнивать системы, построенные в соответствии с различными моделями. Построенная Мак-Лином решетка мандатных моделей безопасности обобщает результаты всех теоретических исследований в этой области, поскольку дает полную картину всех их возможностей и свойств.

4.1.3.9. Применение мандатных моделей

В завершении обзора мандатных моделей необходимо отметить трудности, которые связаны с их применением на практике. Все мандатные модели, как и модель Белла-ЛаПадулы, используют только два права доступа — *чтение* и *запись*. На практике информационные системы поддерживают значительно более широкий спектр операций над информацией, например, *создание*, *удаление*, *передача* и т. д. Следовательно, для того чтобы применить мандатную модель к реальной системе, необходимо установить подходящее соответствие между *чтением* и *записью* и операциями, реализованными в конкретной системе. Идеальным соответствием считается такое, при котором объект не может воздействовать на поведение субъекта до тех пор, пока субъект не осуществит к нему доступ *чтения*, и когда субъект не может воздействовать на объект, пока не осуществит к нему доступ *записи*. Определение такого соответствия представляет собой нетривиальную задачу, поскольку в реальной жизни невозможно ограничиться однонаправленными потоками информации, идущими строго от субъекта к объекту, или наоборот. Ведь для того, чтобы, например, осуществить операцию *чтения* субъект должен сначала послать запрос

службе, реализующей доступ к интересующему его объекту, т. е. осуществить передачу информации, или операцию *записи*. Если рассматривать функционирование системы с такой точки зрения, то применение мандатной политики становится невозможным, потому что попытки распространить мандатную модель на низкоуровневые механизмы, реализующие контролируемые взаимодействия, автоматически приводят к нарушению этой политики. Самым простым примером непрактичности мандатной модели является невозможность ее применения для сетевых взаимодействий — нельзя построить распределенную систему, в которой информация передавалась бы только в одном направлении, потому что всегда будет существовать обратный поток информации, содержащий ответы на запросы, подтверждения получения и т.д.

Поэтому, когда в системе используется мандатная политика, все взаимодействия рассматриваются только на достаточно высоком уровне абстракции, на котором не учитываются детали реализации операций доступа. Такой подход позволяет отобразить любое множество разнообразных операций доступа в обобщенные операции *чтения* и *записи*. Для оценки возможности нарушений безопасности с использованием методов, основанных на несоответствии этих абстрактных операций и реальных механизмов доступа, применяется анализ т.н. скрытых каналов утечки информации. Целью этих исследований является выявление тех способов, с помощью которых информация может передаваться в обход правил мандатной модели. Например для кодирования информации может использоваться число открытых файлов или пустые полях в заголовках сетевых пакетов, или даже размер и количество этих пакетов. Передача информации такими способами не контролируется политикой безопасности, поэтому нарушители могут использовать их как косвенные каналы получения информации и для организации обмена данными. Очевидно, что проконтролировать эти каналы не сможет ни одна модель, а ограничения, с помощью которых их можно было бы ликвидировать, являются слишком жесткими и, как правило, неприемлемы для большинства применений, поэтому на практике ограничиваются применением специальных мер, чтобы, по возможности, сократить их пропускную способность.

Чем больше потоков информации мы поставим под контроль мандатной модели, тем менее гибкой будет наша система, но и тем меньше потоков информации придется исследовать в процессе анализа скрытых каналов.

В заключение обзора мандатной модели управления доступом необходимо отметить, что хотя она является базовой моделью безопасности, составляющей основу теории защиты информации, однако ее применение на практике связано с серьезными трудностями. Поэтому в реальной жизни она используется только в системах, обрабатывающих классифициро-

ванную информацию, и применяется только в отношении ограниченного подмножества субъектов и объектов.

4.1.4. Ролевая политика безопасности

Ролевая политика безопасности представляет собой существенно усовершенствованную модель Харрисона-Руззо-Ульмана, однако ее нельзя отнести ни к дискреционным, ни к мандатным, потому что управление доступом в ней осуществляется как на основе матрицы прав доступа для ролей, так и с помощью правил, регламентирующих назначение ролей пользователям и их активацию во время сеансов [8]. Поэтому ролевая модель представляет собой совершенно особый тип политики, основанной на компромиссе между гибкостью управления доступом, характерной для дискреционных моделей, и жесткостью правил контроля доступа, присущей мандатным моделям.

В ролевой модели классическое понятие *субъект* и замещается понятиями *пользователь* и *роль*. Пользователь — это человек, работающий с системой и выполняющий определенные служебные обязанности. Роль — это активно действующая в системе абстрактная сущность, с которой связан ограниченный, логически связанный набор полномочий, необходимых для осуществления определенной деятельности. Самым распространенным примером роли является присутствующий почти в каждой системе административный бюджет(например root для UNIX и Administrator для Windows NT), который обладает специальными полномочиями и может использоваться несколькими пользователями.

Ролевая политика распространена очень широко, потому что она, в отличие от других более строгих и формальных политик, очень близка к реальной жизни. Ведь на самом деле работающие в системе пользователи действуют не от своего личного имени — они всегда осуществляют определенные служебные обязанности, т. е. выполняют некоторые роли, которые никак не связаны с их личностью.

Поэтому вполне логично осуществлять управление доступом и назначать полномочия не реальным пользователям, а абстрактным (не персонифицированным) ролям, представляющим участников определенного процесса обработки информации. Такой подход к политике безопасности позволяет учесть разделение обязанностей и полномочий между участниками прикладного информационного процесса, т. к. с точки зрения ролевой политики имеет значение не личность пользователя, осуществляющего доступ к информации, а то, какие полномочия ему необходимы для выполнения его служебных обязанностей. Например, в реальной системе обработки информации могут работать системный администратор, менеджер баз данных и простые пользователи.

В такой ситуации ролевая политика позволяет распределить полномочия между этими ролями в соответствии с их служебными обязанностями: роли администратора назначаются специальные полномочия, позволяющие ему контролировать работу системы и управлять ее конфигурацией, роль менеджера баз данных позволяет осуществлять управление сервером БД, а права простых пользователей ограничиваются минимумом, необходимым для запуска прикладных программ. Кроме того, количество ролей в системе может не соответствовать количеству реальных пользователей — один пользователь, если на нем лежат различные обязанности, требующие различных полномочий, может выполнять (одновременно или последовательно) несколько ролей, а несколько пользователей могут пользоваться одной и той же ролью, если они выполняют одинаковую работу.

При использовании ролевой политики управление доступом осуществляется в две стадии: во-первых, для каждой роли указывается набор полномочий, представляющий набор прав доступа к объектам, и, во-вторых, каждому пользователю назначается список доступных ему ролей. Полномочия назначаются ролям в соответствии с принципом наименьших привилегий, из которого следует, что каждый пользователь должен обладать только минимально необходимым для выполнения своей работы набором полномочий.

Ролевая модель описывает систему в виде следующих множеств:

- U - множество пользователей;
- R - множество ролей;
- P - множество полномочий на доступ к объектам, представленное, например, в виде матрицы прав доступа;
- S - множество сеансов работы пользователей с системой.

Для перечисленных множеств определяются следующие отношения (рис. 4.2):

$PA \subseteq P \times R$ - отображает множество полномочий на множество ролей, устанавливая для каждой роли набор присвоенных ей полномочий;

$UA \subseteq U \times R$ - отображает множество пользователей на множество ролей, определяя для каждого пользователя набор доступных ему ролей.

Правила управления доступом ролевой политики безопасности определяются следующими функциями:

$user : S \rightarrow U$ - для каждого сеанса s эта функция определяет пользователя, который осуществляет этот сеанс работы с системой : $user(s)=u$;

$roles : S \rightarrow 2^{(R)}$ - *для каждого сеанса* s эта функция определяет набор ролей из множества R, которые могут быть одновременно доступны пользователю в этом сеансе: $roles(s)=\{r_i \mid (user(s), r_i) \in UA\}$;

permissions : S→P - для каждого сеанса s эта функция задает набор доступных в нем полномочий, который определяется как совокупность полномочий всех ролей, задействованных в этом сеансе: permissions(s) = $U_{r \in roles(s)}\{p_i \mid (p_i, r) \in PA\}$.

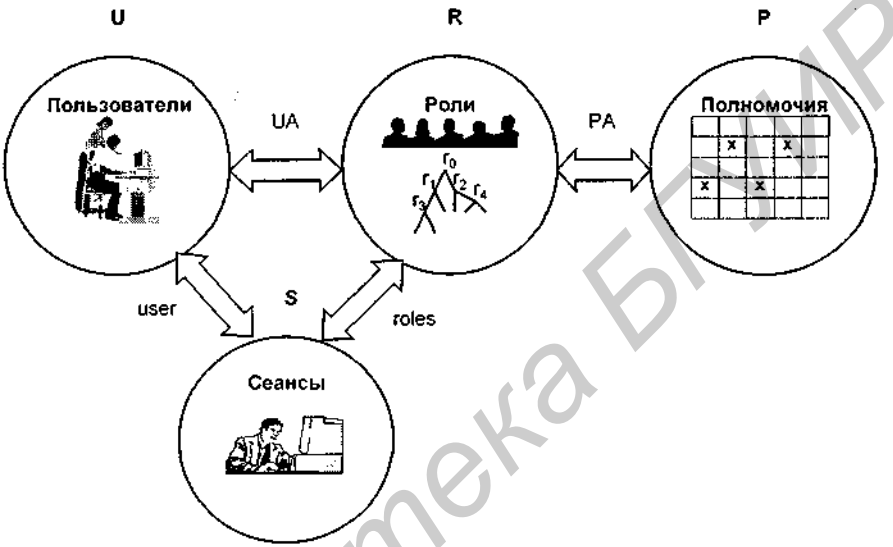


Рис. 4.2 Ролевая модель управления доступом.

В качестве критерия безопасности ролевой модели используется следующее правило: *система считается безопасной, если любой пользователь системы, работающий в сеансе* s, *может осуществлять действия, требующие полномочия* p *только в том случае, если* p ∈ permissions(s).

Из формулировки критерия безопасности ролевой модели следует, что управление доступом осуществляется главным образом не с помощью назначения полномочий ролям, а путем задания отношения UA, назначающего роли пользователям, и функции roles, определяющей доступный в сеансе набор ролей. Поэтому многочисленные интерпретации ролевой модели различаются видом функций user, roles и permission, а также ограничениями, накладываемыми на отношения PA и UA. В качестве примеров рассмотрим ролевую политику управления доступом с иерархической организацией ролей[9], а также несколько наиболее часто встречающихся типовых ограничений на отношения PA и UA и функции user и roles[10].

Иерархическая организация ролей представляет собой наиболее распространенный тип ролевой модели поскольку она очень точно отражает установившееся в реальном мире отношение подчиненности между участниками процессов обработки информации и разделение между ними сфер ответственности. Роли в иерархии упорядочиваются по уровню предоставляемых полномочий. Чем выше роль находится в иерархии, тем больше с ней связано полномочий, поскольку считается, что если пользователю присвоена некоторая роль, то ему автоматически назначаются и все подчиненные ей по иерархии роли. Иерархия ролей допускает множественное наследование. Иерархическая ролевая модель отличается от классической следующими отношениями[9]:

$RH \subseteq R_X R$ - частичное отношение порядка на множестве R, которое определяет иерархию ролей и задает на множестве ролей оператор доминирования $\geq$, такой что, если $r_1 \geq r_2$, то r_1 находится в иерархии выше чем r_2;

$UA^h \subseteq UxR$ - назначает каждому пользователю набор ролей, причем вместе с каждой ролью в него включаются и все роли, подчиненные ей по иерархии, т. е. для $\forall r,r' \in R, u \in U$: $r \geq r' \wedge (u,r) \in UA^h \Rightarrow (u,r') \in UA^h$.

$roles^h : S \rightarrow^{\oplus} (R)$ - назначает каждому сеансу s набор ролей из иерархии ролей пользователя, работающего в этом сеансе: $roles^h(s) \subseteq \{r_i \mid (\exists r' \geq r_i\ (user(s),r') \in UA^h)\}$;

$permissions^h : S \rightarrow P$ - определяет полномочия сеанса как совокупность полномочий всех задействованных в нем ролей и полномочий всех ролей, подчиненных им: $permissions^h(s) = U_{r \in roles^h(s)}\{p_i \mid (\exists\ r'' \leq r\ (p_i,r'') \in PA)\}$.

Таким образом каждому пользователю назначается некоторое подмножество иерархии ролей, а в каждом сеансе доступна совокупность полномочий ролей, составляющих фрагмент этой иерархии. Такой подход позволяет существенно упростить управление доступом за счет неявного назначения полномочий, поскольку в реальной жизни, как правило, пользователи жестко упорядочены по степени ответственности, соответствующей уровню полномочий, которыми они обладают. Причем более доверенные пользователи, стоящие на служебной лестнице выше, всегда обладают всеми полномочиями менее доверенных, подчиненным им. Иерархия ролей в точности отражает эту ситуацию.

Другие реализации ролевой политики безопасности также связаны с введением различных ограничений на отношения PA, UA, и функции user, roles и permissions. Главным для этих ограничений является то, что все они отражают специфику распределения полномочий и сфер ответственности между участниками различных процессов обработки информации.

Рассмотрим несколько примеров, демонстрирующих богатые возможности применения ролевой модели управления доступом[10]:

1. Взаимоисключающие роли. Множество ролей разбивается на подмножества, объединяющие роли, которые не могут быть назначены пользователю одновременно и считают несовместимыми. Таким образом пользователю может быть назначено только по одной роли из каждого подмножества несовместимых ролей. Для определения отношения несовместимости на множестве ролей R задается функция $\text{exclusive}: R \to 2^{R}$, которая для каждой роли определяет множество несовместимых с ней ролей. Тогда на отношение UA, отображающее множество ролей на множество пользователей, накладывается следующее ограничение:

если $(u,r) \in UA \wedge r' \in \text{exclusive}(r) \Rightarrow (u,r') \notin UA$.

Взаимоисключающие роли реализуют т. н. статическое разделение обязанностей, когда конфликт несовместимости полномочий разрешается на стадии назначения ролей. Такая политика хорошо подходит для системы обработки информации, в которой пользователям запрещается совмещать определенные обязанности. Например, в банковской системе одному и тому же пользователю не могут быть одновременно назначены роли оператора, отвечающего за выполнение определенных операций, и аудитора, осуществляющего контроль за их выполнением.

2. Ограничения на одновременное использование ролей в рамках одной сессии. В этом случае множество ролей также разбивается на подмножества несовместимых ролей, но отношение UA может назначить пользователю любую комбинацию ролей. Однако в ходе сеанса работы с системой пользователь может одновременно активировать не более одной роли из каждого подмножества несовместимых ролей.

В этом случае на функцию $\text{roles}: S \to 2^{R}$, которая назначает каждому сеансу s некоторый набор ролей, накладывается следующее ограничение:

$\forall r_1, r_2 \in R$: $r_1 \in \text{roles}(s) \wedge r_2 \in \text{exclusive}(r_1) \Rightarrow r_2 \notin \text{roles}(s)$.

Поскольку в процессе сеанса пользователь может переключаться между различными ролями, но должен при этом избегать конфликтов несовместимости между ними, эта политика получила название динамического разделения обязанностей. Такая политика является более гибкой по сравнению со статическим разделением обязанностей, поскольку позволяет реализовать более сложные схемы контроля доступа. В частности она позволяет запретить пользователю, обладающему значительным набором ролей и полномочий, пользоваться ими всеми одновременно. В определенных ситуациях это позволяет защититься от атаки "троянского коня" — например, пользователю можно запретить одновременно осуществлять доступ к ценной информации и запускать "недоверенные" программы, внесенные в систему другими пользователями. Правильно подобранные

ограничения на одновременное использование ролей позволяют реализовать контроль за информационными потоками, что вообще то характерно для мандатных моделей безопасности.

3. Количественные ограничения при назначении ролей и полномочий. Эта модель предназначена для тех случаев, когда роль может быть назначена только ограниченному числу пользователей, и/или предоставление некоторых полномочий допускается только для ограниченного числа ролей. Функция $\text{cardinality}^r: R \rightarrow N$ показывает какому числу пользователей может быть назначена каждая роль, а функция $\text{cardinality}^p: P \rightarrow N$ определяет для каждого полномочия скольким ролям оно может быть присвоено. На отношения UA и PA накладываются следующие ограничения:

для $\forall r_i \ |\{u_j \mid (u_j, r_i) \in UA\}| \leq \text{cardinality}^r(r_i)$ и

для $\forall p_i \ |\{r_j \mid (p_i, r_j) \in PA\}| \leq \text{cardinality}^p(p_i)$.

Смысл данных условий состоит в том, что благодаря ограничению количества пользователей, осуществляющих те или иные операции, сужается круг лиц, на которых лежит ответственность за совершение соответствующих действий. Например, в системе не должно быть более одного администратора, или, скажем, право уничтожать документы может быть назначено только одной роли.

4. Группирование ролей и полномочий. Роли и полномочия, которые дополняют друг друга, и назначение которых по отдельности не имеет смысла, объединяются в группы, которые могут быть назначены только целиком. Для этого вводятся дополнительные правила, в соответствии с которыми любая роль может быть назначена пользователю только в том случае, если ему уже присвоен определенный набор ролей, а роль может быть наделена полномочием только тогда, когда с ней уже связан определенный набор полномочий.

На множестве ролей R задается функция $\text{prerequisite}^r: R \rightarrow 2^{(R)}$, определяющая для каждой роли подмножество ролей, которые должны быть назначены пользователю прежде чем он получит эту роль. Соответственно на множестве P задается функция $\text{prerequisite}^p: P \rightarrow 2^{(P)}$, определяющая для каждого полномочия подмножество полномочий, которые должны быть присвоены этой роли перед назначением этого полномочия. Для отношения UA, назначающего пользователям роли, вводится ограничение, требующее наличия у пользователя всех ролей, принадлежащих одной группе:

$\forall (p,r) \in UA \wedge p' \in \text{prerequisite}^r(r) \Rightarrow (p',r) \in UA.$

На отношение PA, которое определяет полномочия ролей, накладывается ограничение, вынуждающее присваивать роли сразу все полномочия из группы:

$\forall (p,r) \in PA \wedge p' \in \text{prerequisite}^p(p) \Rightarrow (p',r) \in PA.$

Введение подобных ограничений упрощает администрирование системы в тех случаях, когда полномочия должны предоставляться определенным набором, или когда назначение ролей должно производится в определенной последовательности. Например, предоставлять доступ к некоторым объектам (скажем личным каталогам) имеет смысл только сразу и по чтению и по записи. Типичным примером группирования ролей является ситуация, когда некоторый пользователь, осуществляющий руководство работой других пользователей, должен обладать полномочиями, равными совокупности полномочий всех своих подчиненных, т.е. роль руководителя образует одну группу с ролями исполнителей. Следует отметить, что иерархия ролей является частным случаем группирования ролей и полномочий.

Поскольку все перечисленные варианты ограничений, а также любые другие могут использоваться в различных комбинациях, ролевая модель очень легко адаптируется для каждого конкретного случая, что является ее основным преимуществом перед другими моделями, рассмотренными в этой главе. Ролевая политика предоставляет широкий простор для разработчиков систем управления доступом, — с одной стороны, использование матрицы прав доступа может превратить ее в разновидность дискреционной модели, но, с другой стороны, применение жестких правил распределения ролей между сеансами и пользователями, а также полномочий между ролями, позволяет построить на ее основе полноценную нормативную политику. Следовательно, свойства системы, построенной в соответствии с ролевой моделью, определяются исключительно характером используемых ограничений и могут находиться в очень широком диапазоне, что не позволяет провести формальное доказательство безопасности модели для общего случая.

Подводя итоги свойств ролевой политики управления доступом следует констатировать, что в отличие от других политик она практически не гарантирует безопасность с помощью формального доказательства, а только определяет характер ограничений, соблюдение которых и служит критерием безопасности системы. Такой подход позволяет получать простые и понятные правила контроля доступа, которые легко могут быть применены на практике, но лишает систему теоретической доказательной базы. В некоторых ситуациях это обстоятельство затрудняет использование ролевой политики, однако, в любом случае, оперировать ролями гораздо удобнее чем субъектами, поскольку это более соответствует распространенным технологиям обработки информации, предусматривающим разделение обязанностей и сфер ответственности между пользователями. Кроме того, ролевая политика может использоваться одновременно с другими политиками безопасности, когда полномочия ролей, назначаемых пользователям, контролируются дискреционной или мандатной политикой, что позволяет строить многоуровневые схемы контроля доступа.

4.1.5. Политика доменов и типов для ОС UNIX

Политика доменов и типов (Domain and Type Enforcement — далее DTE) представляет собой практическую реализацию модели типизированной матрицы доступа в системах, построенных на базе ОС UNIX[11]. В классическом варианте управление доступом в UNIX осуществляется с помощью т. н. битов защиты (см. п. 5.1.2.1), что является явно недостаточным с точки зрения потребностей сегодняшнего дня. Поэтому главной задачей усовершенствования защиты UNIX-систем является внедрение новых механизмов управления доступом. Однако, поскольку ОС UNIX является давно признанной и очень широко распространенной системой, существенное возрастание затрат на администрирование, утрата совместимости с традиционными приложениями и необходимость дополнительного обучения пользователей обойдутся очень дорого и являются абсолютно недопустимыми. Политика DTE разрабатывалась именно для того, чтобы решить проблему управления доступом в UNIX не создавая перечисленных трудностей.

С формальной точки зрения DTE — это дискреционная модель, являющаяся расширением типизированной матицы доступа, в которой типы приписаны не только объектам, но и субъектам. С точки зрения реализации DTE представляет собой надстройку над ядром ОС UNIX, которая осуществляет контроль доступа со стороны процессов (субъектов, представляющих интересы пользователей), к традиционным для UNIX объектам системы (файлам, сообщениям, сегментам общей памяти, семафорам и т.д.). Каждому объекту присвоен некоторый *тип*, а с каждым субъектом связан определенный *домен*. Домен представляет собой совокупность субъектов, имеющих одинаковые полномочия. Контроль доступа субъектов к объектам осуществляется на основании правил, задающих отношения доступа между доменами и типами.

Система описывается с помощью множества субъектов S, образующих множество доменов D, и множества объектов O, которые принадлежат типам T. На множестве субъектов S задается функция domain:S→D которая определяет для каждого субъекта домен, к которому он принадлежит, а на множестве объектов O задается функция type:O→T, присваивающая каждому объекту некоторый тип. Отношения доступа между доменами и типами описываются в таблице полномочий доменов DDT=D×T, строки которой соответствуют доменам, столбцы — типам, а в ячейках содержатся права доступа субъектов соответствующего домена к объектам соответствующего типа (например, чтение, запись, выполнение). В таблице отношений между доменами DIT=D×D, задаются полномочия на осуществление процессами одних доменов операций над процессами из других доменов (например, создание процесса, уничтожение процесса, посылка ему определенных сигналов и т. д.). Контроль доступа

осуществляется на уровне ядра ОС UNIX с помощью следующих правил, соблюдение которых и составляет критерий безопасности политики DTE:

- выпрлонение субъектом $s \in S$ операции op над объектом $o \in O$ разрешается только в том случае, если $op \in DDT[domain(s), type(o)]$;
- выполнение субъектом $s_1 \in S$ операции op над субъектом $s_2 \in S$ разрешается только тогда, когда $op \in DIT[domain(s_1), domain(s_2)]$.

В процессе работы подсистема контроля доступа перехватывает системные вызовы, анализирует их и блокирует запросы процессов к ядру ОС, если необходимые для их осуществления полномочия отсутствуют в таблицах DTE.

Такой механизм контроля доступа является достаточно сильным и одновременно гибким, однако его практическое применение в ОС UNIX требует разрешения следующих проблем.

Во-первых, применять политику DTE имеет смысл только тогда, когда в системе существует значительное количество доменов, для которых используются различные ограничения доступа. Однако, в этом случае таблицы прав доступа могут стать слишком громоздкими, вследствие чего администрирование системы превратится в сложный и неуправляемый процесс. Поэтому, для того чтобы управление доступом было эффективным, необходимо предусмотреть возможности как индивидуального, так и группового назначения доменов и типов, а также параметры доступа по умолчанию.

Во-вторых, не существует естественного соответствия между таблицами DTE и стандартными системными структурами ОС UNIX. Таблицы прав доступа не учитывают положение субъектов и объектов в системных иерархиях ОС UNIX и не опираются на дерево порождения процессов и структуру каталогов файловых систем, несмотря на то, что именно положение сущностей в этих структурах как раз и определяет их атрибуты безопасности (процесс наследует домен своего родителя, а в каталоге группируются файлы, принадлежащие одному типу).

Кроме того, необходимость хранения типа для каждого файла требует пересмотра форматов служебных структур файловой системы, что приведет к потере совместимости с существующими системами.

Разработанная специалистами фирмы Trusted Information Systems реализация политики DTE позволяет преодолеть эти препятствия[12]. В предложенной ими реализации атрибуты безопасности субъектов и объектов не хранятся в таблицах прав доступа DIT и DDT, а содержатся в неявном виде в спецификации политики, которая описываются на специально разработанном для этой цели языке DTEL (DTE language). Такое решение позволяет создавать многократно используемые типовые конфигурации управления доступом, рассчитанные на решение определенных прикладных задач.

В процессе загрузки ОС UNIX обрабатывает специальный файл, содержащий спецификацию политики безопасности, описанную на DTEL, и устанавливает соответствующие правила управления доступом. Язык DTEL представляет собой язык высокого уровня, специально предназначенный для описания многократно используемых конфигураций политики DTE в компактной и легко читаемой форме. Этот язык позволяет администратору с помощью высокоуровневых средств манипулировать информацией, традиционно содержащейся в низкоуровневых системных таблицах.

Другой особенностью предложенной архитектуры, позволяющей решить указанные проблемы, является то, что типов объектов не хранятся в служебных структурах файловой системы, а неявно определяются их положением в иерархии каталогов. Тип объекта определяется на основании принципа наследования, в соответствии с которым назначение типа каталогу автоматически влечет за собой его присвоение и всем объектам, находящимся ниже его в дереве каталогов, однако каждое явное назначение типа подавляет тип, унаследованный объектом по умолчанию от родительского каталога.

Такая реализация политики DTE существенно упрощает администрирование безопасности и устраняет необходимость какой-либо модификации файловой системы. Благодаря этим решениям удалось реализовать политику DTE в существующих UNIX системах при соблюдении полной обратной совместимости с существующими форматами файловых систем и незначительными накладными расходами.

Мы не будем здесь приводить полную документацию по DTEL, а просто перечислим наиболее важные операторы языка и покажем на небольшом примере, что сложные правила политики DTE можно выразить в достаточно простой и краткой форме, а администрировать ее не труднее, чем обычную UNIX-систему.

Наиболее важными элементами языка DTEL являются четыре основных оператора описания конфигурации:

`type`, **`domain`**, **`initial_domain`**, **`assign`**.

Оператор **`type`** (тип) объявляет один или несколько типов, которые после могут использоваться для спецификации правил политики безопасности.

Наиболее часто используемый оператор **`domain`** (домен) решает сразу несколько задач: определяет имя домена, задает точки входа в него, описывает полномочия входящих в него субъектов на доступ к объектам различных типов и на осуществление операций над субъектами из других доменов. Таким образом оператор **`domain`** как бы "сжимает" таблицы DDT и DIT в одно целое.

Определение домена состоит из объявления его имени и перечня программ, которые могут использоваться для входа в него. Запуск любой из них приводит к созданию процесса в этом домене, который будет наследоваться всеми его потомками. Например, точкой входа в домены пользователей является программа `/bin/login`.

Полномочия на доступ к объектам задаются путем перечисления режимов доступа и типов объектов. В качестве режимов доступа используются как традиционные для UNIX режимы "`rwx`"(см. п. 5.2.1), так и дополняющие их специальные режимы DTE. Например, в UNIX право выполнения "`x`" для каталогов трактуется специальным образом как право сделать его текущим. Его же необходимо иметь для того, чтобы получить доступ к файлам, находящимся в иерархии ниже этого каталога. В DTEL эти ситуации различаются, — для доступа к каталогу используется привычное "`x`", а для прохождения через каталог при доступе к лежащему под ним файлам применяется специальное право "`d`".

Полномочия на доступ к субъектам из других доменов задаются таким же образом как и полномочия доступа к объектам, но вместо прав доступа указываются виды взаимодействий между процессами. С точки зрения безопасности наиболее важной является операция создания процесса в другом домене, полномочия на осуществление которой описываются правами `exec` и `auto`. Если домен A имеет право доступа `exec` к домену B, то процесс из A может создать процесс в домене B, запустив программу, являющуюся точкой входа домена B, и явно указав при этом, что процесс должен принадлежать домену B. В этом случае программа, осуществляющая запуск процесса в другом домене, должна быть специально ориентирована на использование политики DTE. Для того чтобы обеспечить возможность запуска процессов в другом домене для программ, не рассчитанных на использование DTE, применяется право `auto`, которое означает автоматическое создание процесса в другом домене при выполнении программы, являющейся точкой входа в этот домен. Такой механизм позволяет избежать модернизации системных утилит ОС UNIX и использовать их в том виде, в каком они существуют в любой UNIX-системе. Помимо операций запуска программ полномочия на доступ к доменам могут содержать права на посылку сигналов UNIX (`sigkill`, `sigpause` и т.д.).

Оператор `initial_domain` (исходный домен) назначает домен первому процессу ОС UNIX(`init`).

Оператор `assign` (присваивание) назначает тип одному или нескольким файлам. Оператор присваивания может быть рекурсивным — в этом случае он применяется к каталогу и всем объектам, лежащим в иерархии файловой системы ниже этого каталога. Кроме того, один оператор присваивания может подавлять другой, например, оператор присваи-

вания для `/tmp/foo` подавляет рекурсивный оператор присваивания для `/tmp`.

Кроме того, DTEL поддерживает макроподстановки(конструкция `#define`), позволяющие определять типовые шаблоны для многократного использования в тексте спецификации политики безопасности. Для удобства документирования DTEL поддерживает стандартные соглашения комментариев языка программирования С.

Приведем пример использования основных операторов DTEL в виде описания простой политики безопасности. Сначала определим типы объектов и три домена для прикладных программ:

```
type        unix_t,          /* общесистемные файлы, программы и т.д. */
            specs_t,         /* проектная спецификация */
            budget_t,        /* бюджеты проектов */
            rates_t,         /* ставки зарплаты */
#define     DEF              (/bin/sh),(/bin/csh),(rxd->unix_t)
domain      engineer_d       = DEF, (rwd->specs_t);
domain      project_d        = DEF, (rwd->budget_t), (rd->rates_t);
domain      accounting_d     = DEF, (rd->budget_t), (rwd->rates_t);
```

Данная политика описывает три домена: инженеры(домен `engineer_d`) управляют только проектными спецификациями(тип `specs_t`), руководители проектов(домен `project_d`) следят за ставками зарплаты(тип `rates_t`) и работают над бюджетами проектов(тип `budget_t`), а финансисты(домен `accounting_d`) следят за бюджетами проектов и устанавливают ставки зарплаты. В приведенном примере прикладные домены в качестве программы точки входа используют командные процессоры(`sh,csh`), но в реальной системе вместо них можно запускать прикладные программы, решающие специфические задачи, например программы обработки документов или средства автоматизации проектирования.

Помимо прикладных доменов спецификация политики DTE должна описывать домены для системных процессов, а также механизм инициализации системы. После старта система UNIX запускает начальный процесс, выполняющий программу `/etc/init`. Этот процесс отвечает за создание всех других процессов UNIX, включая процесс регистрации пользователей в системе (`login`), с которого начинаются все сеансы. Политика DTE определяет домены для всех процессов без исключения, устанавливая домен первого процесса, а затем контролируя операции запуска процессов. При запуске нового процесса его домен устанавливается в соответствии с определенными отношениями доступа между доменами. Оператор DTEL `initial_domain` определяет домен первого процесса. Новые процессы наследуют домены своих родителей или, если они явля-

ются точками входа в домен, переходят в другой домен в соответствии с правами `exec` и `auto`.

Дополним предыдущий пример двумя системными доменами и механизмом инициализации, запускающим все системные приложения в домене `system_d`, для которого закрыт доступ к прикладным типам данных :

```
domain            system_d      =(/etc/init), (rwxd->unix_t), (auto->login_d);
domain            login_d       =(/bin/login), (rwxd->unix_t), (exec->engineer_d,
                                  project_d, accounting_d);
initial_domain    =system_d;
```

Оператор `initial_domain` запускает исходный процесс в домене `system_d`, субъекты которого имеют доступ только к общесистемным файлам типа `unix_t`. Этот домен наследуют все системные процессы, за исключением процесса `login`, регистрирующего пользователей в системе. Когда системный процесс из домена `system_d` выполняет программу `/bin/login`, политика DTE в соответствии с режимом доступа `auto` из домена `system_d` в домен `login_d` запускает процесс `login` в домене `login_d`, который специально наделен полномочиями создавать процессы в трех прикладных доменах. Таким образом инициировать прикладные домены может только процесс `login`, который аутентифицирует каждого пользователя, прежде чем запустить для него командный интерпретатор в соответствующем прикладном домене.

Назначим типы объектам системы:

```
assign      -r   -s   unix_t      /; /* тип по умолчанию */
assign      -r   -s   specs_t     /projects/specs;
assign      -r   -s   budget_t    /projects/budget;
assign      -r   -s   rates_t     /projects/rates;
```

Операторы присваивания `assign` однозначным образом назначают тип для каждого файла. Поскольку тип файла определяется его местом в иерархии каталогов, сначала формулируются общие правила, а затем перечисляются исключения. В приведенном примере все файлы корневого каталога имеют тип `unix_t`. Однако в трех подкаталогах тип `unix_t` подавляется типами `specs_t`, `budget_t` и `rates_t`. По умолчанию все файлы наследуют тип каталога, в котором они находятся. Такой подход позволяет без особых трудностей связать типы, число которых обычно невелико, с большим числом файлов, поскольку файлы одного типа имеют тенденцию группироваться в некоторой локальной области иерархии каталогов. Оператор присваивания может быть рекурсивным (с опцией `-r`), тогда он применяется ко всем подкаталогам указанного каталога. Если путь, заданный в одном операторе является префиксом пути, задан-

ного в другом операторе, то оператор, для которого указан самый длинный путь, "подавляет" операторы с более короткими путями для всех файлов, достижимых по этому пути. Назначение типа устойчиво по отношению к изменению имени объекта, поскольку при переименовании файла относящийся к нему оператор присваивания автоматически изменяется, для того чтобы отражать его новое местоположение. Кроме того политика DTE позволяет блокировать попытки создания в указанных областях иерархии объектов других типов(опция `-s`).

Поскольку тип объекта определяется его положением в иерархии каталогов, может случиться так, что файлы, для доступа к которым можно использовать несколько путей, попадают под область действия нескольких операторов назначения типа. Чтобы обнаружить такие ситуации интерпретатор DTEL проверяет, не могут ли несколько операторов присваивания, в которых указаны неперекрывающиеся пути, относиться к одному и тому же файлу. Для каждого такого случая интерпретатор DTEL запрашивает у администратора безопасности, какой из конфликтующих типов должен иметь этот файл, а затем включает в спецификацию дополнительные операторы присваивания, явно назначающие указанный тип. Таким образом гарантируется, что каждому файлу назначен ровно один тип.

Для того, чтобы все программы и утилиты из состава ОС UNIX могли нормально работать, все системные процессы, за исключением процесса входа в систему, запускаются в домене, разрешающем доступ ко всем стандартным объектам UNIX, достижимым из корневого каталога ("/"), которому назначен тип `unix_t`.

Объединим приведенные примеры использования операторов DTEL в законченное описание прикладной политики безопасности:

```
type            unix_t,         /* обычные UNIX файлы, программы и т.д. */
                specs_t,        /* проектная спецификация */
                budget_t,       /* бюджеты проектов */
                rates_t,        /* ставки зарплаты  */

#define         DEFAULT         (/bin/sh),/bin/csh),(rxd->unix_t) /*макро*/

domain          engineer_d      = DEFAULT, (rwd->specs_t);
domain          project_d       = DEFAULT, (rwd-> budget_t), (rd-> rates_t);
domain          accounting_d    = DEFAULT, (rd-> budget_t), (rwd-> rates_t);
domain          system_d        =(/etc/init), (rwxd-> unix_t),(auto-login_d);
domain          login_d         =(/bin/login), (rwxd-> unix_t), (exec->engineer_d
                                project_d, accounting_d);

initial_domain  system_d        /* система начинает работу в этом домене */

assign          -r -s           unix_t;         /* используется по умолчанию */
                                                /* для всех файлов */
assign          -r -s           specs_t,        /projects/specs;
assign          -r -s           budget_t,       /projects/budget;
assign          -r -s           rates_t,        /projects/rates;
```

Эта спецификация определяет три прикладных домена, в каждом из которых обрабатывается данные определенного типа, и два вспомогательных системных домена, в одном их которых работают общесистемные программы, а в другом процесс `login`. Система, построенная в соответствии с этой спецификацией, стартует в домене `system_d`, который наследуется всеми системными процессами, за исключением процесса `login`, который запускается в домене `login_d` с использованием механизма `auto`. Стандартная для ОС UNIX программа `login` модифицирована для реализации политики DTE и после идентификации и аутентификации пользователей запускает программы, являющиеся точками входа в их прикладные домены (`engineer_d`, `project_d` и `accounting_d`), с использованием механизма `exec`. Таким образом каждый сеанс ограничен одним прикладным доменом, в каждом из которых доступ к защищаемым данным, находящимся в трех подкаталогах каталога `/projects`, контролируется в соответствии с определенными правилами. Приведенная спецификация политики безопасности DTE сильно упрощена, и для реального применения ее следует усовершенствовать путем добавления дополнительных персональных доменов, управляющих доступом для каждого пользователя, поддержкой различных доменов для локальных и сетевых сеансов и усилением контроля за различными системными приложениями.

Приведенное описание основных положений политики DTE, обзор возможностей языка DTEL и рассмотренный пример спецификации прикладной политики показывают, что задача усовершенствования защиты ОС UNIX может быть решена без ее существенной модификации, усложнения администрирования и потери совместимости с существующими системами и приложениями. Добиться таких результатов удалось благодаря тому, что архитектура ОС UNIX основана на простых и четко определенных концепциях. В первую очередь это иерархия процессов, связанных отношениями наследственности, и взаимодействующих с ядром и между собой с помощью стандартизованного множества вызовов, а также представление всех системных ресурсов в виде иерархии файлов и каталогов. Использование для описания спецификаций политики DTE специального языка позволяет составлять типовые шаблоны прикладных политик, которые можно использовать в различных UNIX-системах и настраивать с помощью макроподстановок. Отказ от хранения атрибутов безопасности для каждого объекта или субъекта позволил сохранить совместимость с традиционными форматами данных и файловых систем и существенно сократить объемы обрабатываемой информации.

4.1.6. Итоги обзора моделей политик безопасности

В этом разделе мы рассмотрели формальные модели, оказавшие наибольшее влияние на развитие теории информационной безопасности. Необходимо отметить, что за рамками данного обзора остались модели информационных потоков, вероятностные модели и многие другие. Представляя основные положения этих моделей авторы стремились показать, что несмотря на различающиеся подходы, суть всех моделей безопасности одинакова, поскольку они предназначены для решения одних и тех же задач. Целью построения модели является получение формального доказательства безопасности системы, при соблюдении определенных условиям, и а также определение достаточного критерия безопасности.

Огромную роль играет то, каким образом модель отражает устоявшиеся технологи организации обработки информации. Мандатные и дискреционные политики безопасности хорошо соответствуют традиционным механизмам принятым в существующих автоматизированных информационных системах. Для дискреционных моделей права на объекты (файлы) назначаются пользователями, которым они принадлежат, а полномочия процесса определяются идентификатором пользователя, от имени которого он выполняется. Для мандатной модели уровни безопасности объектов соответствуют грифам секретности хранящихся в них документов, а уровни безопасности субъектов определяются исходя из категорий допуска пользователей. Напротив, ролевые политики и политика доменов и типов отражают прикладные политики безопасности, поэтому что для них такого очевидного соответствия не существует. Механизмы их реализации необходимо разрабатывать исходя из условий прикладной задачи, как и методики назначения ролей, полномочий, а также доменов и типов.

Глядя на разнообразие моделей и множество подходов к их реализации многие задаются вопросом, — какие модели лучше других, и какие из них предпочтительнее использовать в том или ином случае? Ответ на этот вопрос, на наш взгляд, состоит в следующем. Безопасность есть успешное противостояние угрозам, поэтому сама модель безопасности не обеспечивает защиту, а только предоставляет основополагающий принцип архитектуры системы, реализация которого и должна обеспечить заложенные в модели свойства безопасности. Следовательно, безопасность системы в равной степени определяется тремя факторами: свойствами самой модели, ее адекватностью угрозам, воздействующим на систему, и тем насколько корректно она реализована. Поскольку при существующем разнообразии теоретических наработок в области теории информационной безопасности выбор модели, адекватной заданным угрозам, не является проблемой, последнее, решающее слово остается за ее реализацией в защищенной системе. Этот вопрос мы подробнее затронем в пятой главе.

4.2. Криптографические методы защиты

Кроме политики безопасности, моделей управления доступом и контроля за его осуществлением существует еще один аспект построения защищенных систем, которого мы еще практически не касались. Действительно, создание защищенной системы невозможно без применения криптографических методов, предоставляющих в распоряжение разработчика средства, обеспечивающие определенные гарантии степени защиты. Данный раздел содержит аналитический обзор некоторых популярных отечественных и зарубежных методов аутентификации и контроля целостности, решающих соответствующие задачи защиты систем обработки информации. Поскольку рассматривать аутентификацию и контроль целостности в отрыве от алгоритмов шифрования невозможно, последним уделяется значительное внимание. Вместе с тем, за рамками данного раздела остался широкий круг вопросов, затрагивающих, в частности, управление ключами, квантовую криптографию и многие другие.

Для того, чтобы сделать излагаемый материал главу понятным как можно более широкому кругу читателей, использование математического аппарата сведено к необходимому минимуму. Учитывая ограниченный объем книги и специфику оценки безопасности криптографических алгоритмов, предполагающую глубокое знание алгебры, теории чисел, теории вероятностей, приводятся только конечные результаты криптоанализа.

4.2.1. Идентификация и аутентификация

В процессе *идентификации* устанавливается взаимно однозначное соответствие между множеством сущностей системы и множеством *идентификаторов*. Идентификация позволяет различать сущности системы при контроле доступа, аудите и т.д.

Аутентификация представляет собой проверку подлинности *идентификаторов*.

В процессе идентификации и аутентификации (рис. 4.3) пользователь или программа (*претендент*) запрашивает доступ у системы (*верификатора*). Сначала осуществляется идентификация. Верификатор требует от претендента предъявить некоторый идентификатор и проверяет принадлежность предъявленного идентификатора *ID* множеству зарегистрированных в системе. В случае корректности идентификатора верификатор выполняет процедуру аутентификации (например, запрашивает пароль), чтобы убедиться, что претендент является именно тем, за кого себя выдает. Допуск претендента в систему разрешается только в случае успешного завершения процедуры аутентификации. В большинстве систем устанавливается еще некоторое пороговое значение для числа попыток предъявления некорректного идентификатора и пароля, при превышении которо-

го все дальнейшие попытки доступа данного претендента к системе блокируются.

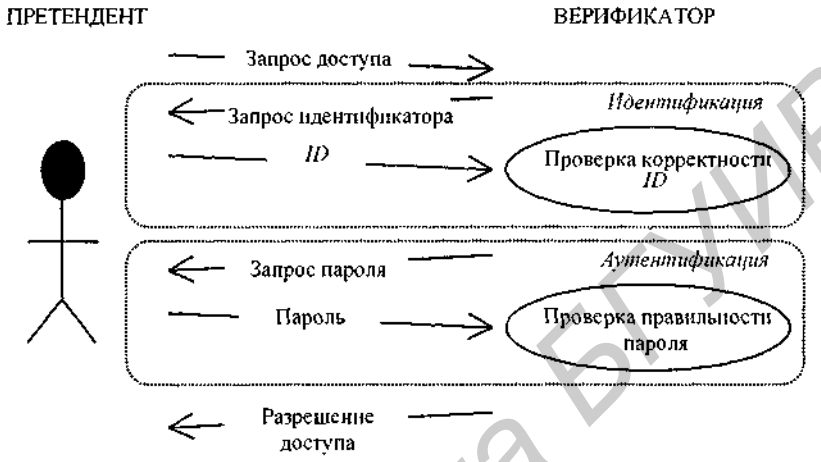


Рис. 4.3. Модель процесса идентификации и аутентификации

В основе методов аутентификации могут лежать, например, следующие принципы:

(1) знание претендентом некоторой секретной информации (пароля);
(2) предъявление претендентом некоторых неизменных характеристик (отпечатков пальцев);
(3) предоставление претендентом доказательства того, что он находится в некотором определенном месте (возможно, в некоторое определенное время);
(4) установление подлинности претендента некоторой третьей стороной, которой доверяет верификатор.

Часто для надежности используются различные комбинации этих принципов.

Различают два основных типа аутентификации:

- *Аутентификация субъекта* решает задачу установления подлинности идентификатора, предъявляемого субъектом взаимодействия (например, пользователя, прикладных процессов и т.п.) и обычно используется при доступе к ресурсам.
- *Аутентификация объекта* устанавливает подлинность идентификатора некоторого объекта. В качестве доказательства подлинности обычно используется подтверждение того, что источником данного

объекта является владелец указанного идентификатора (например, отправитель электронной почты, владелец банковского счета и т.п.).

4.2.1.1. Аутентификация субъекта

Методы аутентификации субъекта подразумевают обмен аутентификационной информацией, имеющий своей целью убедить верификатора в подлинности идентификатора, предъявленного претендентом. Аутентификационная информация обычно защищается с помощью криптографических алгоритмов.

Аутентификация субъекта может быть как *односторонней*, так и *взаимной*. При односторонней аутентификации аутентифицируется только один субъект. При взаимной аутентификации два взаимодействующих субъекта аутентифицируют друг друга. В принципе, взаимную аутентификацию можно осуществить путем объединения двух сеансов односторонней аутентификации. Однако, в этом случае может возникнуть уязвимость к атакам перехвата и повтора[1], даже если при односторонней аутентификации этот недостаток отсутствовал. Кроме того, число сообщений в протоколах взаимной аутентификации можно сделать значительно меньше удвоенного числа сообщений соответствующей односторонней аутентификации.

4.2.1.1.1 Пароли

Классическим средством аутентификации субъекта являются парольные схемы, в которых претендент предъявляет некоторый пароль, а верификатор сравнивает этот пароль с имеющимся у него множеством паролей. Эти схемы не лишены недостатков, наиболее серьезными из которых являются несанкционированный доступ к паролям, хранимым в памяти компьютера, и большая вероятность угадывания пароля. Кроме того, возможен перехват пароля при вводе или при передаче.

Одним из способов защиты пароля при передаче является передача претендентом не самого пароля, а его образа, вычисленного с помощью некоторой *вычислимой в одну сторону*[2] функции h, например, *хэш-функции* (требования к хэш-функциям, используемым для защиты пароля, и некоторые примеры таких функций будут рассмотрены в п. 4.2.5). Претендент, прошедший идентификацию, предъявляет пароль p', вычисляет значение $q' = h(p')$ и посылает q' верификатору (рис. 4.4). Верификатор для каждого ID хранит величину $q = h(p)$.

---

1 При такой атаке нарушитель осуществляет активный перехват и изменение сообщений, передаваемых между претендентом и верификатором, выдавая себя за верификатора перед претендентом и за претендента перед верификатором ("проблема гроссмейстера").

2 Под вычислимой в одну сторону функцией будем понимать такую функцию $h: P \to Q$, что, зная $p \in P$ легко вычислить $q \in Q$, $q = h(p)$, но из q восстановить p трудно.

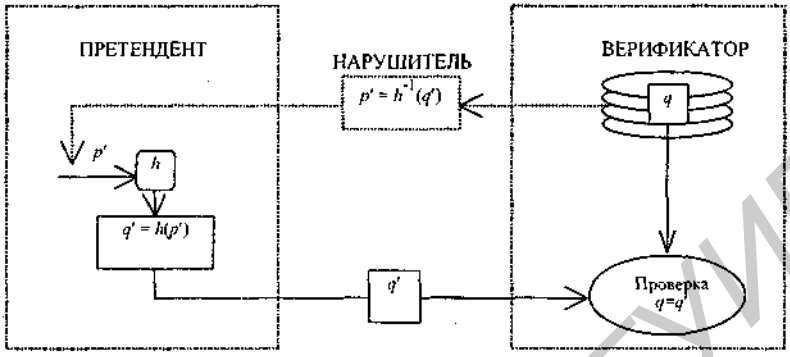


Рис. 4.4. Защита пароля при передаче

Верификатор при выполнении равенства $q = q'$ заключает, что был введен верный пароль, и разрешает доступ. Такая схема обладает определенной степенью защиты, если нарушитель может передавать свою информацию только с терминала претендента. Даже если нарушитель определит q, отследив сеанс успешной аутентификации, ему нелегко будет определить p.

Если предположить, что нарушитель может передавать свою информацию, подключаясь непосредственно к линии связи, то для защиты от компрометации верификатора процедуру вычисления $q' = h(p')$ следует возложить на верификатора (рис. 4.5).

Оба рассмотренных варианта не защищают от случая, когда нарушитель записывает передаваемую претендентом по линии связи информацию и организует повтор. Для защиты от такого рода нарушений может использоваться схема, представленная на рис. 4.6, где функция h_2 зависит от блока неповторяющихся данных *nrb*, в качестве которых может использоваться, например, порядковый номер или время.

4.2.1.1.2 Симметричные методы аутентификации субъекта. Схема Kerberos

Аутентификация субъекта может осуществляться симметричными методами [44], т.е. с применением *симметричных* алгоритмов шифрования (п.4.2.2.1). В этом случае претендент и верификатор используют общий секретный ключ K. Претендент зашифровывает свое сообщение на этом ключе, вводя в открытый или зашифрованный текст некоторое контрольное значение (КЗ). Если верификатору удается успешно расшифро-

вать полученное сообщение и проверить корректность КЗ, то он может быть уверен в подлинности претендента.

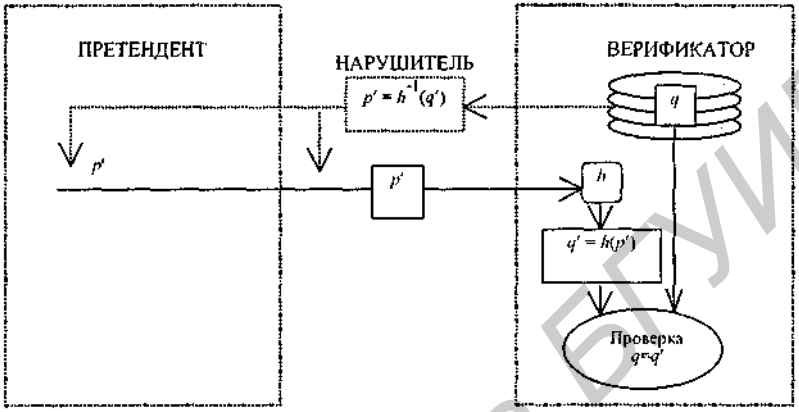


Рис. 4.5. Защита пароля от компрометации верификатора

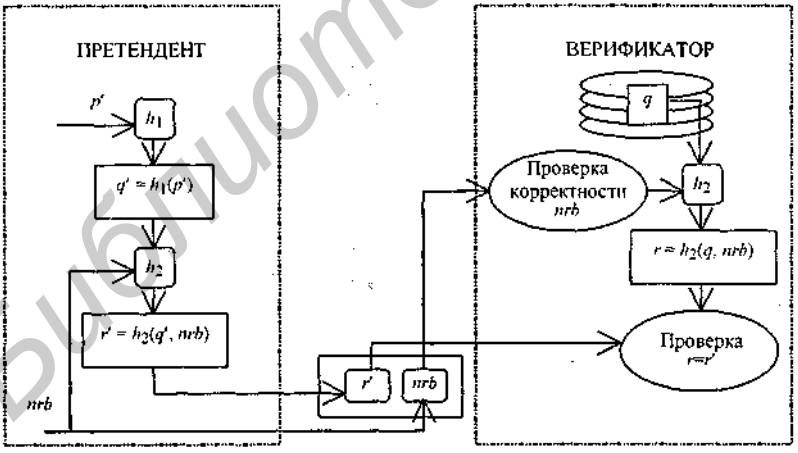


Рис. 4.6. Защита пароля от повтора

Для реализации симметричных методов часто применяют протокол "запрос - ответ", когда верификатор сначала посылает претенденту случайный запрос x, затем претендент зашифровывает этот запрос и возвра-

щает верификатору ответ $y = E_K(x)$, после чего верификатор проверяет соответствие запроса и ответа, т.е. выполнение равенства $x = E_K^{-1}(y)$.

В многопользовательских системах практическое применение симметричных криптографических методов несколько затруднено, поскольку эти методы предполагают хранение каждым верификатором (т.е. каждым хостом, сервером и т.д.) секретного ключа K_i и информации о каждом пользователе, который может воспользоваться ключом K_i. Эта проблема устраняется путем введения в сеанс связи доверенной третьей стороны (*сервера аутентификации*), с которой разделяют секретный ключ каждый пользователь и каждый верификатор. Существуют различные способы взаимодействия с таким сервером. На рис. 4.7 изображены два основных подхода.

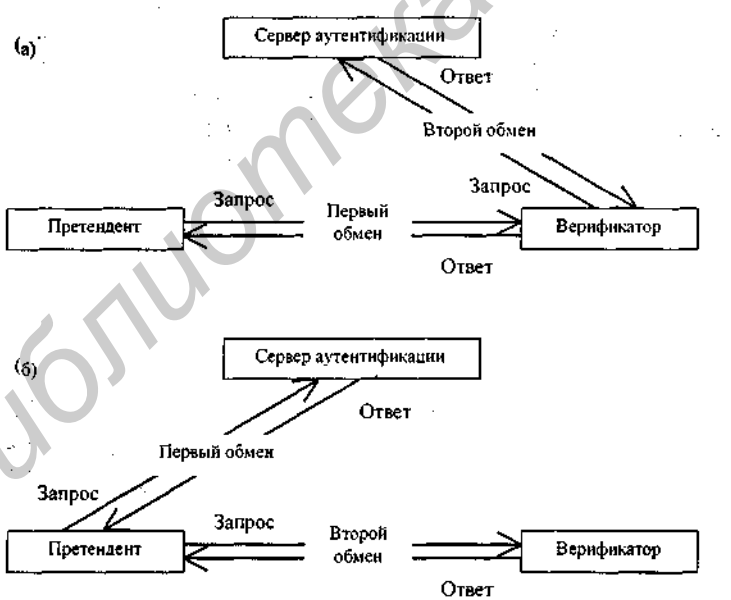


Рис.4.7 Схема аутентификации с доверенной третьей стороной

В схеме рис. 4.7(а) претендент зашифровывает сообщение на своем секретном ключе и посылает шифртекст верификатору. Поскольку ве-

рификатор не знает ключа пользователя, он не может сразу осуществить аутентификацию, а сначала должен обратиться к серверу аутентификации и получить необходимый ключ. Для защиты передаваемых данных верификатор и сервер аутентификации используют общий секретный ключ. После обмена с сервером аутентификации верификатор проводит аутентификацию претендента.

Отличие метода рис. 4.7(б) состоит в том, что претендент сначала устанавливает связь с сервером аутентификации и получает разрешение на доступ, которое затем передает верификатору. Здесь для защиты данных общий секретный ключ используют претендент и сервер аутентификации. Претендент может получить разрешение только в том случае, если он знает этот ключ (такой вариант используется в системе Kerberos).

Схема с доверенной третьей стороной может обеспечить и взаимную аутентификацию.

Типичным примером реализации симметричных методов аутентификации субъекта является схема Kerberos [52, 53], разработанная в Массачусетском технологическом институте (MIT) в середине 80-х годов. Она обеспечивает как одностороннюю, так и взаимную аутентификацию, и может использоваться, в частности, при запросе сетевых ресурсов. Для применяемых в ней симметричных алгоритмов шифрования, например, DES (п. 4.2.2.1.1), разработаны специальные режимы, позволяющие обеспечивать дополнительно целостность передаваемых данных (п. 4.2.3.1.2).

До проведения собственно аутентификационного обмена (рис. 4.8), претендент устанавливает связь с центром распределения ключей, называемом еще сервером Kerberos, который выполняет роль доверенной третьей стороны. По завершении обмена и претендент, и верификатор получают секретный сеансовый ключ, сгенерированный центром распределения ключей. На знании этого ключа и основывается взаимная аутентификация.

В схеме Kerberos выделяют три типа обмена:

1. Во время регистрации претендента проводится *обмен между претендентом и центром распределения ключей*. Претендент отправляет центру распределения ключей открытое сообщение, включающее в себя идентификаторы претендента P и верификатора V и другую необходимую информацию.

Центр распределения ключей сверяет полученные идентификаторы с имеющейся у него базой данных, генерирует для претендента разрешение на доступ к серверу выдачи разрешений или напрямую к верификатору и связанные с этим разрешением данные, зашифровывает их на открытом ключе претендента и возвращает претенденту зашифрованные данные вместе с некоторой открытой контрольной информацией.

Разрешение, выдаваемое центром распределения ключей, представляет собой результат зашифрования на секретном ключе участника y

определенной последовательности данных и имеет вид $tkt_{Py} = E_y\{P, K_{Py}, ltime_{Py}\}$, где K_{Py} - общий сеансовый ключ для P и y, $ltime_{Py}$ - время жизни разрешения tkt_{Py} (т.е. дополнительно предполагается, что в сети действует служба единого времени), а в роли y может выступать либо сервер выдачи разрешений G, либо верификатор V. Содержание разрешения tkt_{Py} могут прочитать только центр распределения ключей и y.

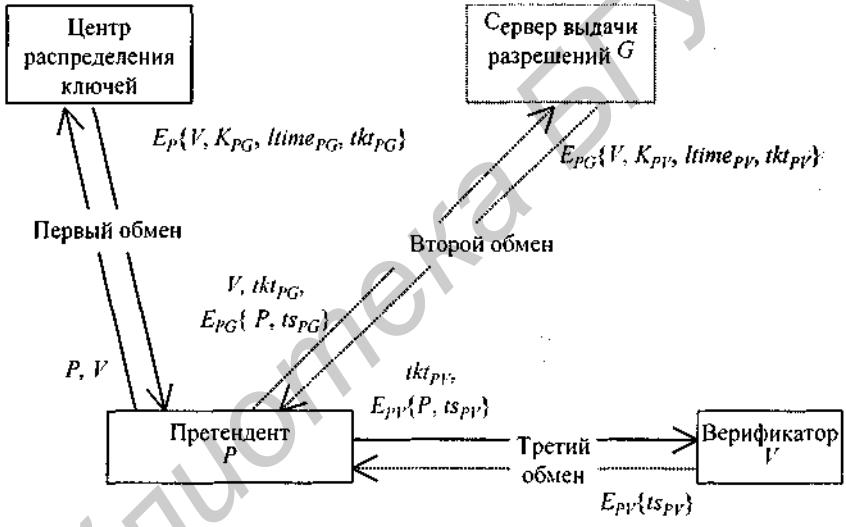


Рис. 4.8. Схема диалоговой аутентификации Kerberos

2. При *обмене между претендентом и сервером выдачи разрешений* секретный ключ претендента не используется, а применяется разрешение tkt_{PG}, полученное претендентом ранее при обмене с центром распределения ключей. В результате обмена с сервером выдачи разрешений претендент получает дальнейшее разрешение на доступ к верификатору.

При этом обмене используется текущая временная метка ts_{PG}, генерируемая претендентом P и позволяющая серверу выдачи разрешений G обнаружить повторные сообщения.

В результате этих двух обменов претендент P получает секретный сеансовый ключ K_{PV} и разрешение на секретную передачу этого ключа верификатору V.

3. *Собственно аутентификационный обмен* представляет собой заключительный обмен между претендентом и верификатором, при котором происходит аутентификация верификатором претендента, а при взаимной аутентификации еще и претендентом верификатора. Здесь используется разрешение, полученное ранее либо при первом, либо при втором обмене.

Запрос, отправляемый претендентом при аутентификационном обмене, позволяет безопасно передать верификатору V ключ K_{PV} (который P и V используют затем для взаимной аутентификации). В запросе передается также временная метка ts_{PV}, гарантирующая верификатору V своевременность запроса и обеспечивающая защиту от повтора. Ответ верификатора посылается только в случае необходимости взаимной аутентификации и показывает претенденту, что V успешно расшифровал присланное разрешение и выделил из него значение ts_{PV}.

4.2.1.1.3 Несимметричные методы аутентификации субъекта

Несимметричные методы аутентификации субъекта [45] подразумевают использование несимметричных (с открытым ключом) криптографических алгоритмов (п. 4.2.2.2). В таких схемах верификатор формирует некоторое сообщение и просит претендента подписать его. Претендент подписывает это сообщение, используя свой секретный ключ (механизм цифровой подписи будет подробно рассмотрен в п. 4.2.4), а затем верификатор проверяет подпись с помощью известного ему открытого ключа подлинного пользователя. Если при проверке подпись окажется правильной, то верификатор может быть уверен, что претендент действительно является тем, за кого себя выдает. Здесь в качестве защиты от атак повтора также можно включить в сообщение неповторяющуюся величину.

С появлением криптосистем с открытым ключом потребность в серверах аутентификации в качестве доверенной третьей стороны практически отпала, поскольку характеристики этих систем естественным образом предотвращают проблему раскрытия ключа претендента по известному ключу верификатора. Тем не менее, для проверки подписи верификаторы должны, как правило, получать сертифицированные открытые ключи. Обычно сертификаты3 на ключи выдаются специальным сервером, который не является непосредственным участником сеанса аутентификации.

---

3 *Сертификат* представляет собой цифровой документ, удостоверяющий принадлежность данного открытого ключа данной сущности. Простейший сертификат включает в себя открытый ключ и имя сущности.

Примером аутентификации субъекта с открытым ключом могут служить схемы, представленные в стандарте CCITT Recommendation X.509 [30].

В этих схемах взаимная аутентификация претендента P и верификатора V может осуществляться в два или три этапа, как это показано на рис. 4.9. Для проведения аутентификации претенденту необходима сертифицированная копия открытого ключа зашифрования, принадлежащего верификатору, а верификатору - сертифицированная копия открытого ключа зашифрования, принадлежащего претенденту. Выдача сертификатов осуществляется специальным сервером.

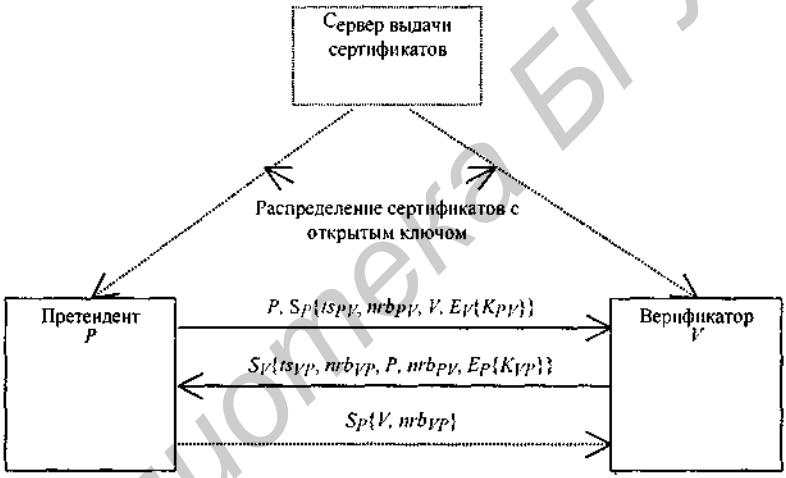


Рис.4.9 Схема диалоговой аутентификации X.509

При двухэтапной аутентификации претендент:

1) генерирует секретный ключ K_{PV}, который затем используется при защите последующих сеансов связи;
2) зашифровывает секретный ключ K_{PV} на открытом ключе верификатора, т.е. вычисляет $E_V\{K_{PV}\}$;
3) генерирует текущую временную метку ts_{PV} (которая может содержать информацию о дате создания и времени жизни сообщения претендента) и блок неповторяющихся данных nrb_{PV}, призванные помочь верификатору в обнаружении повторных сообщений:

4) подписывает с помощью своего секретного ключа последовательность значений ts_{PV}, nrb_{PV}, V, $E_V\{K_{PV}\}$, где V - идентификатор верификатора, и отправляет верификатору блок данных $\{P, S_P\{ts_{PV}, nrb_{PV}, V, E_V\{K_{PV}\}\}$ (P - идентификатор претендента, S_P - данные, подписанные на секретном ключе претендента).

Верификатор, в свою очередь:

1) проверяет правильность подписи претендента, корректность идентификатора V, временной метки и (при использовании эффективной процедуры выработки неповторяющихся данных) блока неповторяющихся данных для защиты от повтора;
2) генерирует секретный ключ K_{VP}, который используется при защите последующих сеансов связи;
3) зашифровывает секретный ключ K_{VP} на открытом ключе претендента, т.е. вычисляет $E_P\{K_{VP}\}$;
4) генерирует текущую временную метку ts_{VP} и блок неповторяющихся данных nrb_{VP}, призванные помочь претенденту в обнаружении повторных сообщений;
5) подписывает с помощью своего секретного ключа последовательность значений ts_{VP}, nrb_{VP}, P, nrb_{PV}, $E_P\{K_{VP}\}$ и отправляет претенденту блок данных $S_V\{ts_{VP}, nrb_{VP}, P, nrb_{PV}, E_P\{K_{VP}\}\}$.

Получив сообщение от верификатора, претендент выполняет проверку подписи верификатора. Для каждого сообщения проверка подписи обеспечивает основную аутентификацию, временная метка и/или неповторяющаяся величина - защиту от повтора сообщения для того же самого верификатора, а идентификатор верификатора, включенный в сообщение, - от повтора одного и того же сообщения для другого верификатора. Пересылка зашифрованных секретных ключей K_{PV} и/или K_{VP} необязательна, но, как правило, рекомендуется для обеспечения симметричной криптографической защиты целостности (п. 4.2.3.1.3) передаваемых данных.

При трехэтапной аутентификации претендент и верификатор выполняют те же самые действия, что и при двухэтапном обмене, однако временные метки здесь не требуются. Претендент проверяет, что переданное ему значение nrb_{PV} совпадает с тем, которое сам он отправлял верификатору. После этого претендент передает верификатору подписанное на своем секретном ключе сообщение $S_P\{V, nrb_{VP}\}$.

Аналогично, получив последнее сообщение, верификатор проверяет, что присланный ему блок rrb_{VP} совпадает с тем, который он отправлял претенденту. Такое расширение двухэтапной аутентификации позволяет защитить от повтора пару "запрос - ответ" в обоих направлениях. Поэтому и не требуются временные метки.

Еще одним примером диалоговой аутентификации с открытым ключом может служить трехэтапная аутентификация [34], представляющая собой совокупность взаимной аутентификации с открытым ключом и ключевого обмена Диффи - Хеллмана (рис. 4.10). Ключ здесь вычисляется над полем F_p, где p - простое число.

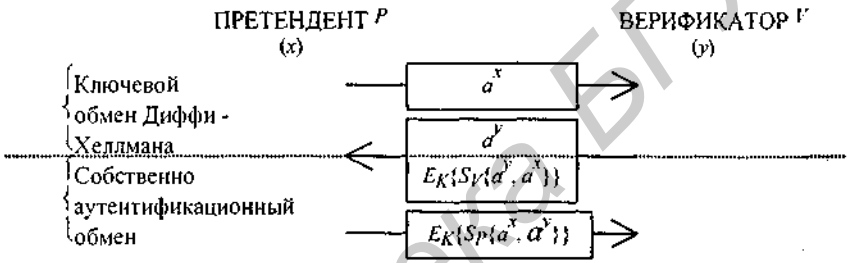


Рис. 4.10 Схема диалоговой аутентификации Диффи - Хеллмана

В аутентификационном обмене участвуют две стороны - претендент P и верификатор V. Пусть x - случайное число, вырабатываемое претендентом, y - случайное число, вырабатываемое верификатором. Пусть a - образующая циклической группы F_p^*, E_K - функция зашифрования на сеансовом ключе $K = a^{xy}$, D_K - функция расшифрования на том же ключе.

1. Претендент вычисляет значение $a^x \bmod p$ и передает его верификатору.
2. Верификатор вычисляет $a^y \bmod p$, с помощью своего секретного ключа создает цифровую подпись $S_V\{a^y, a^x\}$ сообщения $\{a^y, a^x\}$, вычисляет сеансовый ключ $K = (a^x)^y \bmod p$, зашифровывает S_V на этом ключе и отправляет претенденту a^y и $E_K\{S_V\{a^y, a^x\}\}$.
3. Претендент вычисляет сеансовый ключ $K = (a^y)^x \bmod p$, с помощью своего секретного ключа создает цифровую подпись $S_P\{a^x, a^y\}$ сооб-

щения $\{a^x, a^y\}$, зашифровывает ее на ключе K и отправляет верификатору $E_K\{S_P\{a^x, a^y\}\}$.

4. Если верификатору удастся успешно расшифровать $E_K\{S_P\{a^x, a^y\}\}$, т.е. найти $D_K\{E_K\{S_P\{a^x, a^y\}\}\} = S_P\{a^x, a^y\}$ и проверить подпись $S_P\{a^x, a^y\}$, то он может быть уверен, что претендент является тем, за кого себя выдает.
5. Если претенденту удастся успешно расшифровать $E_K\{S_V\{a^y, a^x\}\}$, т.е. найти $D_K\{E_K\{S_V\{a^y, a^x\}\}\} = S_V\{a^y, a^x\}$ и проверить подпись $S_V\{a^y, a^x\}$, то он может быть уверен, что верификатор является тем, за кого себя выдает.

В этой схеме компрометация ключа подписи, необходимого при аутентификации, в отличие от предыдущих схем не приводит к компрометации основных секретных ключей. Использование подписи делает эту схему устойчивой к целому ряду атак перехвата и повтора.

4.2.1.1.4. Доказательства с нулевым разглашением знаний

Для аутентификации субъекта могут применяться и *доказательства с нулевым разглашением знаний* (*zero-knowledge proofs*), суть которых состоит в том, что наличие информации у претендента проверяется без раскрытия этой информации (или ее части) верификатору или третьей стороне.

Как правило, в таких схемах аутентификации верификатор задает претенденту ряд вопросов. Претендент вычисляет ответ на каждый вопрос с помощью имеющейся у него секретной информации. По этим ответам верификатор может с достаточно высокой степенью уверенности установить, что претендент действительно обладает секретной информацией (хотя никакая часть этой информации в ответах фактически не раскрывается). Разные схемы предполагают использование разного числа пар "вопрос - ответ" и имеют разную сложность вычислений, проводимых обеими сторонами.

В качестве примера рассмотрим типичную схему аутентификации на основе задачи о гамильтоновом цикле [39]4, которая общепризнанно является сложной. Пусть претенденту и верификатору известен ориентированный граф G с n вершинами и пусть G имеет гамильтонов цикл H, который известен только претенденту. Покажем, как претендент может доказать, что он знает H, не предъявляя верификатору самого цикла H.

---

4 *Гамильтоновым циклом* в ориентированном графе $G=(V, E)$, где V - множество вершин и E - множество ребер, называется цикл, содержащий все вершины из V [1].

1. Претендент выбирает случайную перестановку π и переставляет узлы в G, т.е. вычисляет граф $G'= \pi(G)$. Затем претендент выполняет независимое преобразование каждого бита матрицы смежности графа G' и посылает преобразованную таким образом матрицу верификатору.
2. Верификатор случайным образом выбирает "0" или "1" (например, подбрасывает монету) и отправляет выбранное значение претенденту.
3. Если претендент получает "0", то он раскрывает верификатору перестановку π и способ шифрования битов матрицы смежности. Верификатор вычисляет $G'= \pi(G)$, зашифровывает матрицу смежности графа G' и проверяет соответствие зашифрованной матрицы и матрицы, полученной на шаге 1. Т.е. верификатор может проверить, что претендент знает перестановку π.

Если претендент получает "1", то он открывает верификатору способ преобразования только тех n битов матрицы смежности (т.е. ребер) графа G', которые задают гамильтонов цикл в G'. Верификатор по структуре графа G' может проверить, что эти ребра действительно составляют гамильтонов цикл.

Неизвестно, какое задание ("0" или "1") претендент получит на шаге 2, поэтому, если он не знает гамильтонова цикла H, то обман раскрывается с вероятностью 1/2. Шаги 1-3 повторяются последовательно r раз. Тогда вероятность нераспознаваемого обмана равна 2^{-r}.

Преобразование битов, выполняемое претендентом на шаге 1, представляет собой функцию $\varepsilon:\{0,1\}\times\Omega\rightarrow\Sigma$ (где мощность множества Σ достаточно велика, а Ω - совокупность случайных чисел заданной длины) такую, что:

- по значению $\varepsilon(b, \omega)$, где $b\in\{0,1\}$, $\omega\in\Omega$, трудно предсказать b;
- невозможно найти ω, ω' такие, что $\varepsilon(0, \omega) = \varepsilon(1, \omega')$.

Примером такого преобразования бита b может служить вычисление хэш-функции (п. 4.2.5) от случайного аргумента, в котором младший бит равен b.

Доказательства с нулевым разглашением знаний являются потенциально более сильными с точки зрения криптографии, чем большинство традиционных криптографических методов. Однако, во многих таких схемах необходима пересылка большого количества информации и/или более сложные протоколы. В частности, может потребоваться несколько обменов для проведения простой односторонней аутентификации.

4.2.1.1.5. Аутентификация пользователей

Как уже отмечалось, парольным методам свойственны некоторые недостатки. Методы, основанные на симметричных и несимметричных алгоритмах и на доказательствах с нулевым разглашением знаний не яв-

ляются их заменой и не используются напрямую для аутентификации пользователей, в частности потому, что длинные случайные векторы секретного ключа довольно трудны для запоминания. Однако, при наличии некоторого устройства взаимного доверия эти типы методов можно комбинировать. В общем виде данный подход выглядит следующим образом: устройство аутентифицирует пользователя, предъявляющего пароль, а затем конечный верификатор аутентифицирует это устройство с помощью криптографических методов. Примерами такого сочетания могут служить ключи генерации паролей и смарт-карты.

В первом случае объединение пользователя, связанного с паролем, и криптографической системы происходит за счет установления взаимно однозначного соответствия между парой "пользователь - пароль" и секретным ключом криптосистемы. Пароль часто защищается с помощью хэш-функции (п. 4.2.5), но, кроме того, результирующее значение определяется специальным образом и используется в качестве ключа криптосистемы. Это значение позволяет криптографической системе аутентификации одинаково обслуживать и пользователя, и автоматизированную систему со встроенным ключом.

Во втором случае процессор карты выполняет роль претендента, взаимодействуя с верификатором либо на близком терминальном устройстве, либо в удаленной, доступной из сети системе. В ранних схемах аутентификации на основе смарт-карт использовались хэш-функции и простые криптосистемы с открытым ключом. Сейчас все большее применение находят доказательства с нулевым разглашением знаний.

4.2.1.2. Аутентификация объекта

В процессе аутентификации объекта (рис. 4.11) (иногда называемой *аутентификацией источника данных*) проверяется подлинность идентификатора, представленного в совокупности с некоторыми данными. Отличие от аутентификации субъекта состоит в том, что в этом случае претенденту не нужно быть активным участником процесса аутентификации. Этот тип аутентификации тесно связан с контролем целостности данных (п. 4.2.3), поскольку верификатор, получивший сообщение, должен быть уверен в том, что информация, посланная претендентом, не была изменена.

Основными криптографическими способами аутентификации объекта являются шифрование симметричным алгоритмом, имитовставка и цифровая подпись.

При использовании шифрования симметричным алгоритмом (п. 4.2.2.1) претенденту и верификатору обычно заранее известен метод вычисления некоторого контрольного значения (КЗ) исходного сообщения и секретный ключ. Для защиты данных при передаче претендент присоеди-

няет к ним КЗ, а затем зашифровывает результат. Верификатор расшифровывает полученные данные и отделяет КЗ от основного текста. В случае успеха проверки корректности КЗ верификатор может быть уверен, что сообщение пришло от подлинного владельца идентификатора.

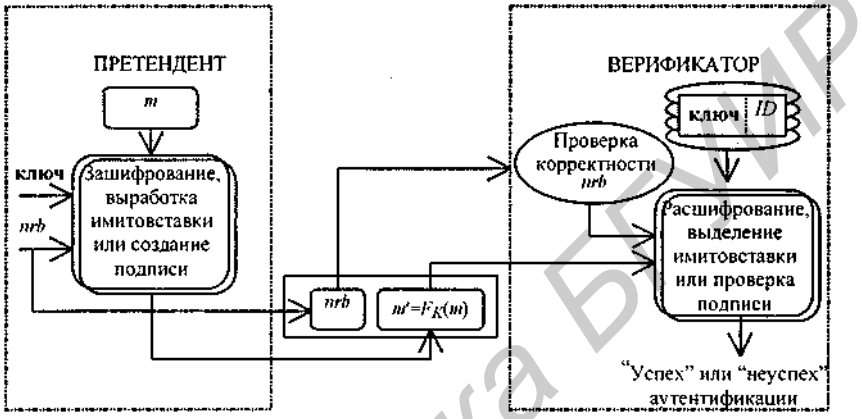


Рис.4.11 Схема аутентификации объекта

Имитовставка (п. 4.2.3.1.1) играет роль КЗ, присоединяемого к открытым (как при шифровании) или к зашифрованным данным.

Симметричные алгоритмы и имтовставка применяются в тех случаях, когда претендент и верификатор доверяют друг другу. Действительно, при использовании этих алгоритмов возможности претендента и верификатора одинаковы: если верификатор заявляет третьей стороне, что он получил сообщение от претендента, а тот отрицает факт отправки сообщения, то ясно, что один из них лжет. Однако кто именно говорит неправду, установить формальными методами невозможно. В таких случаях, т.е. когда необходимо доказать подлинность идентификатора третьей стороне (при условии, что верификатор не имеет возможности изменить текст полученного сообщения), может потребоваться цифровая подпись (п. 4.2.4).

4.2.2. Шифрование

Под *шифрованием* понимают процесс зашифрования или расшифрования. *Зашифрованием* называется процесс преобразования открытых данных (*открытого текста*) в зашифрованные (*шифртекст*) при помощи *шифра*. Шифр представляет собой совокупность обратимых преобразований, осуществляемых с использованием некоторого параметра (*ключа*). *Расшифрованием* называется процесс, обратный процессу за-

шифрования, преобразующий зашифрованные данные в открытые [15]. Операции зашифрования и расшифрования можно описать в виде функций:

$$C = E_K(P),\ P = D_{K^*}(C),$$

где P и C - соответственно открытый и зашифрованный тексты, K - ключ зашифрования, K^* - ключ расшифрования, E_K - функция зашифрования, D_{K^*} - функция расшифрования. Функция D_{K^*} обратна к E_K, так что для любого открытого текста P справедливо $D_{K^*}(E_K(P)) = P$. Это означает, что в результате расшифрования шифртекста получается первоначальная незашифрованная информация. Процесс зашифрования и расшифрования можно представить в виде схемы, представленной на рис. 4.12.

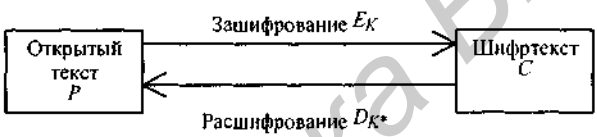


Рис. 4.12 Схема зашифрования - расшифрования

Способность шифра противостоять раскрытию (т.е. недопустимому снижению защитных качеств шифра), называется *стойкостью*. Численно стойкость выражается как сложность наилучшего известного алгоритма, раскрывающего шифр. Поэтому стойкость шифра имеет две составляющих: временную и емкостную сложность.

Общепринятым требованием к современным шифрам является стойкость к анализу на основе известных и специально подобранных открытых текстов (зашифрованные тексты a priori считаются известными).

Различают два типа алгоритмов шифрования: *симметричные* (с секретным ключом) и *несимметричные* (с открытым ключом). Если в симметричных алгоритмах знание ключа преобразования на передающей стороне позволяет легко вычислить ключ преобразования на принимающей стороне и наоборот, то в несимметричных алгоритмах такая вычислимость отсутствует.

4.2.2.1 Шифрование с секретным ключом

В симметричных алгоритмах шифрования источник зашифровывает открытый текст на секретном ключе K ($C = E_K(P)$), а приемник рас-

шифровывает шифртекст на секретном ключе K^* ($P = D_{K^*}(C)$). Обычно $K=K^*$.

При симметричном шифровании информация часто обрабатывается блоками. В современных алгоритмах шифрования обычно используются блоки длиной n не менее 64 бит (например, DES, FEAL, ГОСТ 28147-89).

Выделяют два общих принципа построения шифров: рассеивание и перемешивание [22]. *Рассеиванием* называется распространение влияния одного знака открытого текста на несколько знаков шифртекста, помогающее скрыть статистические свойства открытого текста. Обобщением этого принципа является распространение влияния одного знака ключа на несколько знаков шифртекста, которое препятствует восстановлению ключа по частям. *Перемешиванием* называется использование таких шифрующих преобразований, которые усложняют восстановление взаимосвязи символов открытого и зашифрованного текста, а также ключа и зашифрованного текста.

Шифр должен обеспечивать легкость зашифрования и расшифрования (при известном секретном ключе). Распространенным способом построения шифра является последовательное выполнение несложных обратимых преобразований, в качестве которых чаще всего используются операции сложения, умножения, подстановки и перестановки. Сложение может быть по модулю 2 (DES) или по модулю 2^{32} (ГОСТ 28147-89). Подстановка задает перемешивание, перестановка - рассеивание. При *перестановке* изменяется порядок последовательности битов открытого текста, причем конкретный вид перемешивания определяется секретным ключом (ГОСТ 28147-89) или является неизменным (DES). При *подстановке* совокупность нескольких битов (обычно 4 или 8) открытого текста заменяется совокупностью такого же числа битов, а конкретный вид подстановки определяется секретным ключом. Иногда используется и фиксированный набор подстановок и перестановок, как, например, в DES.

При многократном чередовании простых подстановок и перестановок можно получить стойкий алгоритм шифрования с хорошим результирующим рассеиванием и перемешиванием. Кроме того, для обеспечения стойкости шифра необходимо учесть ряд дополнительных требований, обусловленных появлением в последние годы универсальных эффективных методов криптоанализа.

Для различных длин текстов и различных требований к шифрованию разработан ряд режимов шифрования [20, 29, 59], в том числе:

1) *электронная кодовая книга* (ECB) (в ГОСТ 28147-89 - *режим простой замены*) (рис. 4.13) применяется для шифрования сообщений длиной в один блок (как правило, для шифрования ключей), поскольку здесь все блоки текста шифруются независимо друг от друга на одном и

том же ключе, т.е. $C_i = E_K(P_i)$ при зашифровании, $P_i = D_K(C_i)$ при расшифровании, $1 \le i \le N$;

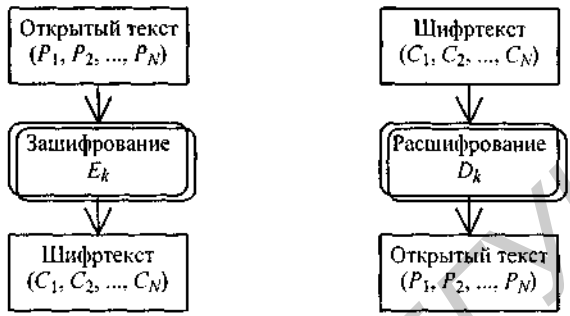


Рис. 4.13 - Режим электронной кодовой книги (ECB)

2) *сцепление блоков шифра* (CBC) (в ГОСТ 28147-89 - *режим гаммирования с обратной связью*) (рис. 4.14) может использоваться для аутентификации данных. В этом режиме открытый текст P разбивается на n-битные блоки $P = P_1P_2...P_N$. Первый блок текста после зашифрования суммируется по модулю 2 с *синхропосылкой* C_0, представляющей собой блок текста, который не несет полезной информации и известен источнику и приемнику данных. На каждом последующем шаге блок шифртекста C_i вычисляется по формуле $C_i = E_K(P_i \oplus C_{i-1})$, $1 \le i \le N$, т.е. зашифровывается значение суммы по модулю 2 блока открытого текста и шифртекста предыдущего блока. Расшифрование проводится аналогично - зная синхропосылку C_0, можно вычислить $P_i = C_{i-1} \oplus D_K(C_i)$, $1 \le i \le N$. Из последней формулы ясно, что если изменить один блок шифртекста, то после расшифрования изменится соответствующий расшифрованный блок и следующий за ним, остальные блоки открытого текста останутся без изменений;

3) *обратная связь по шифртексту* (CFB) (рис. 4.15) предназначена для шифрования отдельных символов. Для нее используется блочный шифр, генерирующий для источника и приемника *псевдослучайную последовательность* битов (ПСП), зависящую от ключа и открытого текста. Размер блоков открытого текста определяется параметром t, $1 \le t \le n$. Открытый текст P разбивается на t-битные блоки $P = P_1P_2...P_N$. n-битный регистр сдвига инициируется некоторым значением синхропосылки S_0. На каждом шаге блок шифртекста C_i вычис-

ляется по формуле $C_i = P_i \oplus R_i$, $1 \le i \le N$, где через R_i обозначены t старших битов шифрограммы $E_K(S_{i-1})$, которые и используются в качестве ПСП. Регистр сдвига модифицируется следующим образом: t старших битов в S_i отбрасываются, а справа присоединяется C_i, т.е. $S_i \equiv (2^t S_{i-1} + C_i) \bmod 2^n$, $1 \le i \le N$. Расшифрование осуществляется аналогично: S_i вычисляются по тем же формулам, в качестве R_i используются t старших битов расшифрованного текста $D_K(S_{i-1})$, а открытый текст P_i восстанавливается как $P_i = C_i \oplus R_i$, $1 \le i \le N$.

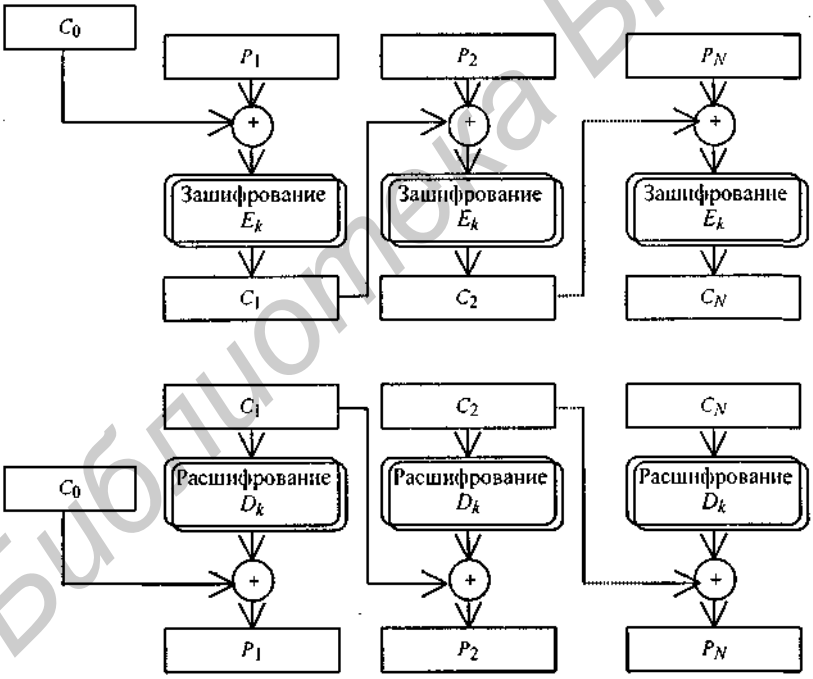


Рис. 4.14· Режим сцепления блоков шифра (CBC)

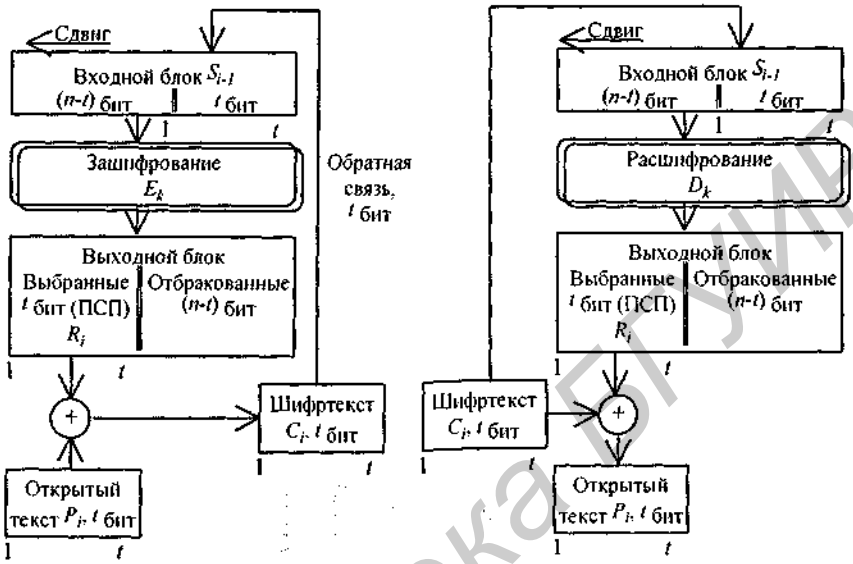


Рис. 4.15 Режим обратной связи по шифртексту (CFB)

4) *обратная связь по выходу* (OFB) (в ГОСТ 28147-89 - *режим гаммирования*) (рис. 4.16) так же, как и режим CFB, использует блочный шифр для генерации ПСП источником и приемником, переменный размер блока t и регистр сдвига, инициируемый синхропосылкой S_0 (в ГОСТ 28147-89 $t=n$), но отличается от предыдущего режима способом изменения информации в регистре сдвига: $S_i \equiv (2^t S_{i-1} + R_i) \bmod 2^n$ [59] или $S_i \equiv E_K(S_{i-1})$ (т.е. без сдвига) [29], $1 \le i \le N$. Кроме того, необходимо при каждом очередном использовании системы менять S_0. Это обусловлено тем, что при использовании одинаковых синхропосылок формируются одинаковые ПСП. Следовательно, если известен открытый текст, зашифрованный с помощью синхропосылки S_0, то может быть вычислен другой открытый текст, зашифрованный на этой же синхропосылке, причем знание ключа для этого не требуется.

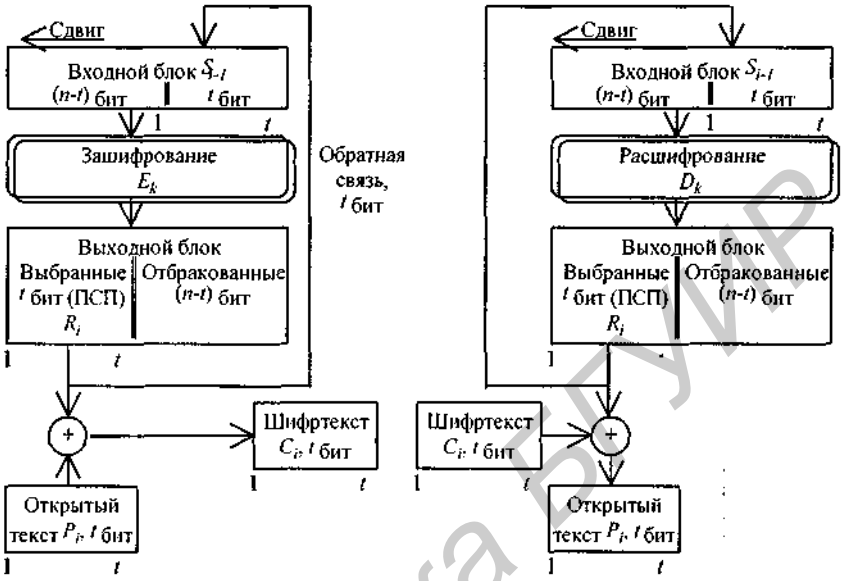


Рис.4.16 Режим обратной связи по выходу (OFB)

4.2.2.1.1. DES

Стандарт шифрования США DES (Data Encryption Standard) [57] разработан в середине 70-х гг. сотрудником корпорации IBM Х.Фейстелем. Алгоритм, лежащий в основе DES, представляет собой блочный шифр, использующий подстановки, перестановки и сложения по модулю 2, с длиной блока 64 бита и длиной ключа 56 бит5. Подстановки и перестановки, используемые в DES, являются фиксированными.

Расширенный ключ, задающий выбор соответствующего подключа на каждом цикле шифрования, фиксирован и определяется следующей последовательностью операций:

- разделение 56-битного секретного ключа на два полублока по 28 бит, c_0 и d_0;
- циклический сдвиг блоков c_0 и d_0 влево на $\sigma[i]$ битов, результатом которого являются полублоки c_1 и d_1, и далее по индукции:

5 Вообще говоря, в DES используется 8-байтный ключ, каждый байт которого состоит из семи битов фактического ключа, непосредственно используемых в алгоритме, а восьмой бит служит контрольным битом четности остальных семи битов.

$$c_{i+1} = \sigma[i+1] \cdot c_i,\ d_{i+1} = \sigma[i+1] \cdot d_i,\ 1 \le i \le 15,$$

(число $\sigma[i]$ зависит от номера итерации i и определяется соответствующей таблицей);

48-битный подключ K_i, i=0, ..., 15, используемый на i-том цикле, выбирается по определенному правилу из конкатенации блоков c_{i+1}, d_{i+1}.

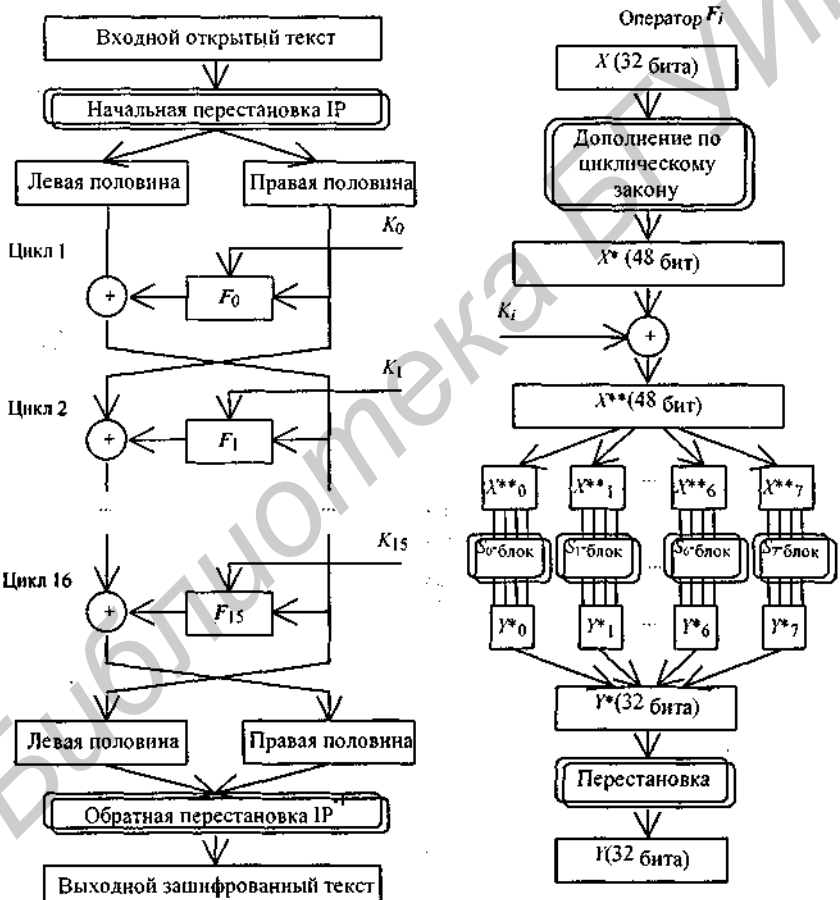


Рис. 4.17. Схема алгоритма DES

Рассмотрим основные этапы алгоритма зашифрования DES (рис. 4.17):

1. К блоку входного текста применяется фиксированная перестановка IP;
2. Для каждого цикла $i = 0, ..., 15$ выполняется операция зашифрования π_{F_i}, при которой:

– 64-битный блок разбивается на левую и правую половины x'' и x' по 32 бита;
– правая половина x' разбивается на 8 тетрад по 4 бита. Каждая тетрада по циклическому закону дополняется крайними битами из соседних тетрад до 6-битного слова;
– полученный 48-битный блок суммируется по модулю 2 с 48 битами подключа, биты которого выбираются на каждом цикле специальным образом из 56 бит ключа, а затем разбивается на 8 блоков по 6 бит;
– каждый из полученных на предыдущем шаге блоков поступает на вход функции фиксированного *S-блока*, которая выполняет нелинейную замену наборов 6-битных блоков тетрадами;
– полученные 32 бита подвергаются фиксированной перестановке, результатом которой является полублок $F_i(x')$;
– компоненты правого зашифрованного полублока $F_i(x')$ суммируются по модулю 2 с компонентами левого полублока x'' и меняются местами, т.е. блок $(x'', F_i(x'))$ преобразуется в блок $(x''+ F_i(x'), x'')$;

1) К блоку текста, полученному после 16 циклов, применяется обратная перестановка IP^{-1}. Результатом является выходной зашифрованный текст.

Итоговое уравнение зашифрования имеет вид (порядок следования операторов - справа налево):

$$DES = IP^{-1} \times \pi_{F_{15}} \times \pi_{F_{14}} \times ... \times \pi_{F_0} \times IP.$$

При расшифровании порядок следования операторов меняется на обратный.

4.2.2.1.2. FEAL

Алгоритм FEAL (Fast Data Encipherment Algorithm) предложен Shimizu и Miyaguchi в 1987 г. [75] как альтернатива DES и ориентирован на 8-разрядный процессор. FEAL имеет длину блока 64 бита и длину ключа 64 бита. Обычно число циклов шифрования N варьируется от 4 до 32. Существуют и обобщенные версии алгоритма с более длинным (128-

битным) ключом и с большим числом циклов. Основные операции в алгоритме FEAL - сложение по модулю 2 и по модулю 256, сдвиг.

64-битный секретный ключ преобразуется в расширенный ключ длиной $2N+16$ байт, K_0, K_1, ..., K_{2N+15}. Зашифрование блока открытого текста происходит в три этапа (рис. 4.18):

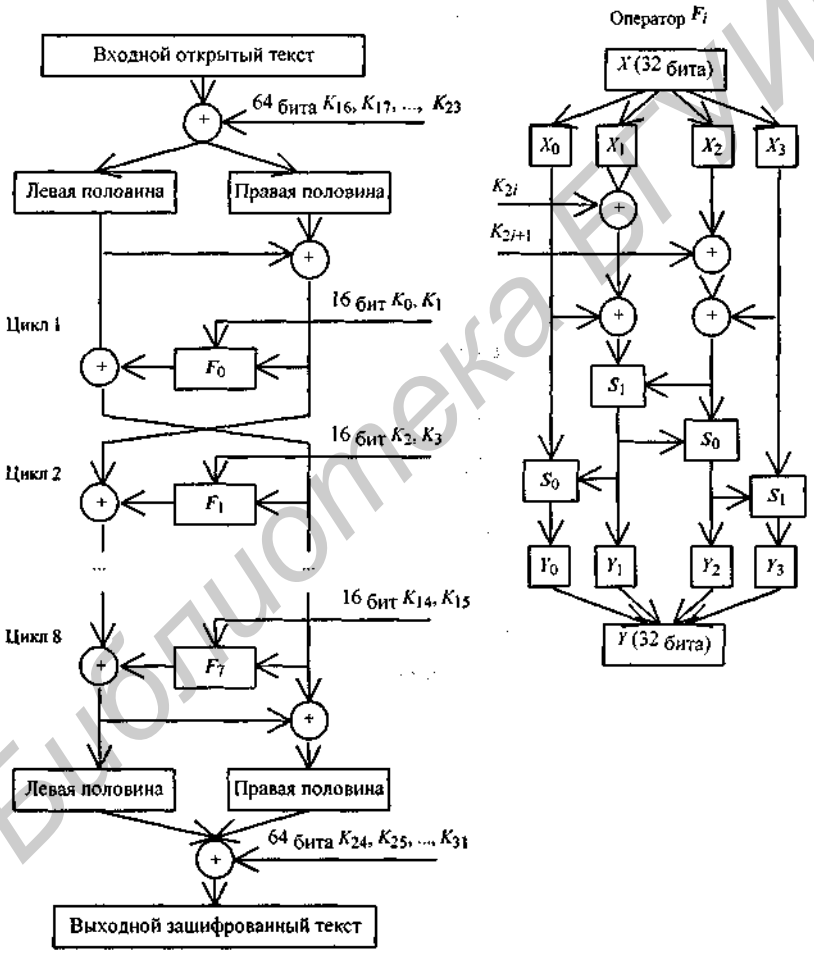


Рис. 4.18 Схема алгоритма FEAL ^cN=8

1. Входной 64-битный блок открытого текста суммируется по модулю 2 с 64 битами расширенного ключа. Результирующий блок разбивается на левую и правую половины (X^0, X^1) по 32 бита.
2. Для каждого цикла выполняется операция зашифрования, при этом на каждом шаге новый 32-битный полублок задается уравнением:

$$X^{i+2} = \boldsymbol{F}_i(X^{i+1}) \oplus X^i,\ i = 0, ..., N\text{-}1,$$

где оператор $\boldsymbol{F}_i$ взаимно однозначен, зависит только от двух байтов расширенного ключа K_{2i} и K_{2i+1}, а его значение $Y = \boldsymbol{F}_i(X)$, где X и Y - 32-битные полублоки, определяется следующим образом:

– 32-битный полублок X разбивается на четыре 8-битных подблока X_0, X_1, X_2, X_3;

– 8-битные подблоки Y_0, Y_1, Y_2, Y_3 32-битного значения Y оператора $\boldsymbol{F}_i$ вычисляются по формулам:

$Y_1 = S_1(X_0 \oplus X_1 \oplus K_{2i}, X_2 \oplus X_3 \oplus K_{2i+1})$

$Y_0 = S_0(X_0, Y_1)$

$Y_2 = S_0(Y_1, X_2 \oplus X_3 \oplus K_{2i+1})$

$Y_3 = S_1(Y_2 \oplus X_3),$

где $S_j(a,b)$ - циклический сдвиг на два бита влево числа $a+b+j$ (mod 256), $j \in \{0,1\}$.

3. Блок выходного зашифрованного текста определяется сложением по модулю 2 64-битного блока (X^N, X^{N+1}), полученного на последней итерации, с 64 битами расширенного ключа.

При расшифровании порядок выполнения операций меняется на обратный, вид оператора $\boldsymbol{F}_i$ сохраняется.

4.2.2.1.3. IDEA

Алгоритм шифрования IDEA (International Data Encryption Algorithm), предложенный в 1991 г. X.Lai и L.Massey [56] в качестве улучшенной версии алгоритма шифрования PES (Proposed Encryption Standard) [55], применяется в системе PGP (Pretty Good Privacy). IDEA представляет собой блочный шифр, в котором 64-битные блоки открытого текста последовательно шифруются на 128-битном ключе.

Для процедуры зашифрования используется пятьдесят два 16-битных подключа, алгоритм генерации которых из секретного ключа включает в себя:

- разбиение 128-битного ключа на восемь 16-битных блоков, образующих первую восьмерку подключей $K_1, K_2, ..., K_8$;
- циклический сдвиг 128-битного ключа на 25 битов влево и получение нового ключа, который также разбивается на восемь 16-битных блоков, образующих следующую восьмерку подключей $K_9, K_{10}, ..., K_{16}$;
- повторение шага 2 до тех пор, пока не будут получены 52 подключа.

Основными операциями алгоритма шифрования являются умножение по модулю $2^{16}+1$, сложение по модулю 2 и по модулю 2^{16}. Для того, чтобы умножение по модулю $2^{16}+1$ было обратимой операцией, нулевой сомножитель рассматривается как 2^{16}, а произведение, равное 2^{16}, заменяется на 0.

64-битный блок входного открытого текста разбивается на четыре 16-битных подблока P_1, P_2, P_3 и P_4. Далее зашифрование выполняется по схеме, изображенной на рис. 4.19. Итоговые четыре блока $C_1, ..., C_4$, сцепляются и образуют 64-битный блок шифртекста. Процесс повторяется последовательно для каждого из N 64-битных блоков открытого текста, пока не будет зашифрован весь текст.

При расшифровании выполняется та же самая последовательность операций над 64-битными блоками шифртекста, но с другим множеством подключей. Подключи расшифрования вырабатываются из подключей зашифрования и являются мультипликативно обратными по модулю $2^{16}+1$ или аддитивно обратными по модулю 2^{16} к этим подключам. В таблице 4.1 приведены подключи расшифрования (по отношению к подключам зашифрования $K_1, ..., K_{52}$) для каждого цикла алгоритма.

Таблица 4.1. Значения подключей расшифрования по отношению к подключам зашифрования в алгоритме IDEA

Цикл 1	K_{49}^{-1}	$-K_{50}$	$-K_{51}$	K_{52}^{-1}	K_{47}	K_{48}
Цикл 2	K_{43}^{-1}	$-K_{45}$	$-K_{44}$	K_{46}^{-1}	K_{41}	K_{42}
Цикл 3	K_{37}^{-1}	$-K_{39}$	$-K_{38}$	K_{39}^{-1}	K_{35}	K_{36}
Цикл 4	K_{31}^{-1}	$-K_{33}$	$-K_{32}$	K_{34}^{-1}	K_{29}	K_{30}
Цикл 5	K_{25}^{-1}	$-K_{27}$	$-K_{26}$	K_{28}^{-1}	K_{23}	K_{24}
Цикл 6	K_{19}^{-1}	$-K_{21}$	$-K_{20}$	K_{22}^{-1}	K_{17}	K_{18}
Цикл 7	K_{13}^{-1}	$-K_{15}$	$-K_{14}$	K_{16}^{-1}	K_{11}	K_{12}
Цикл 8	K_7^{-1}	$-K_9$	$-K_8$	K_{10}^{-1}	K_5	K_6

Подключи завершающего преобразования расшифрования выражаются через подключи завершающего преобразования зашифрования следующим образом:

$$K_1^{-1} \quad -K_2 \quad -K_3 \quad K_4^{-1}$$

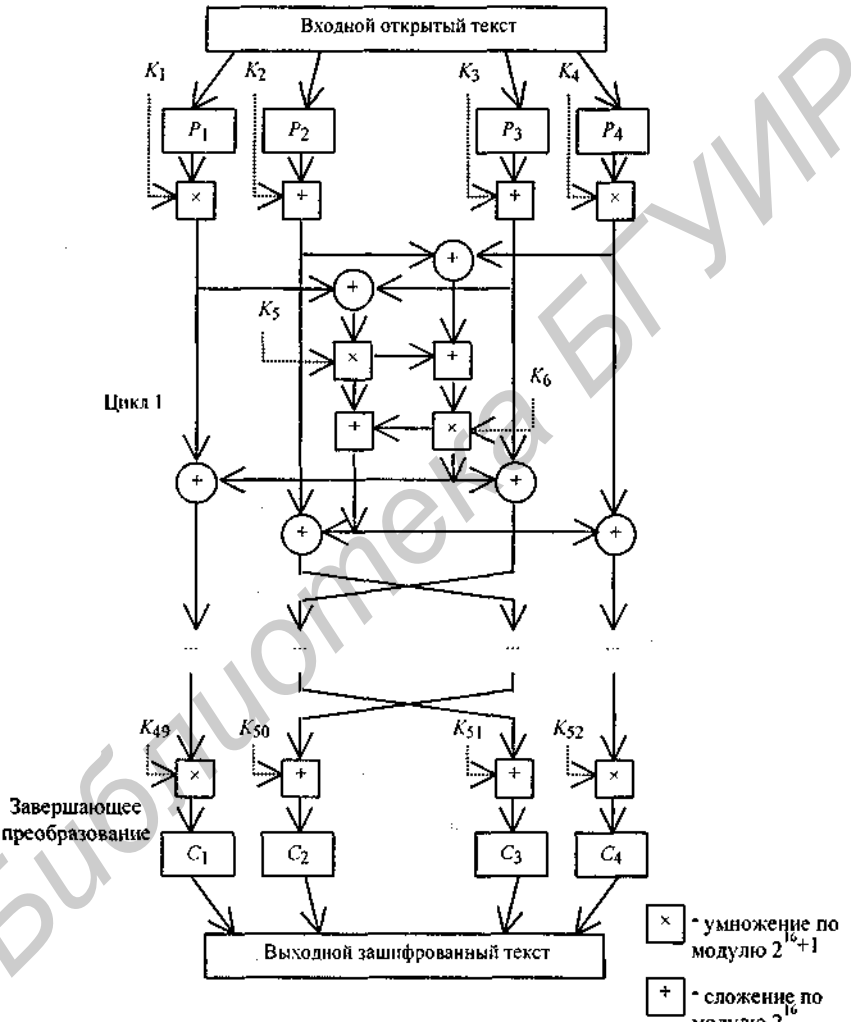


Рис. 4.19 Алгоритм шифрования IDEA

4.2.2.1.4. ГОСТ 28147-89

Отечественный алгоритм блочного шифрования ГОСТ 28147-89 [15] предусматривает три режима шифрования: простой замены, гаммирования и гаммирования с обратной связью, а также режим выработки имитовставки (п. 4.2.3.1.1). В основу всех указанных режимов положен режим простой замены.

Алгоритм шифрования имеет длину блока 64 бита и длину ключа 256 бит. Ключ составлен из восьми 32-битных подблоков K_0, ..., K_7. Основные операции - сложение по модулю 2 и 2^{32}, подстановка, циклический сдвиг.

При реализации алгоритма зашифрования в режиме простой замены (рис. 4.20) открытый текст разбивается на два полублока x' и x'' по 32 бита, к которым применяется оператор циклической итерации F_i, i=0, ..., 31, включающий в себя:

- суммирование по модулю 2^{32} правой половины текста x'' с 32 битами ключа;
- преобразование результата суммирования в блоке подстановки, представляющем собой набор из восьми 4-битных подстановок. Каждая из восьми тетрад, поступающих на вход блока подстановки, заменяется в соответствии со своей подстановкой;
- циклический сдвиг в сторону старших битов.

Результат циклической итерации $F_i(x'')$ суммируется по модулю 2 с полублоком x', а полученные полублоки переставляются местами. Таким образом, блок $(x', F_i(x''))$ преобразуется в блок $(x'', x' + F_i(x''))$.

Всего реализуется 32 цикла, которые различаются используемым подблоком ключа K_i. При зашифровании порядок выбора подблоков ключа следующий:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

Благодаря такому выбору подблоков, итоговый оператор зашифрования не является степенным. При расшифровании подблоки ключа используются в обратном порядке.

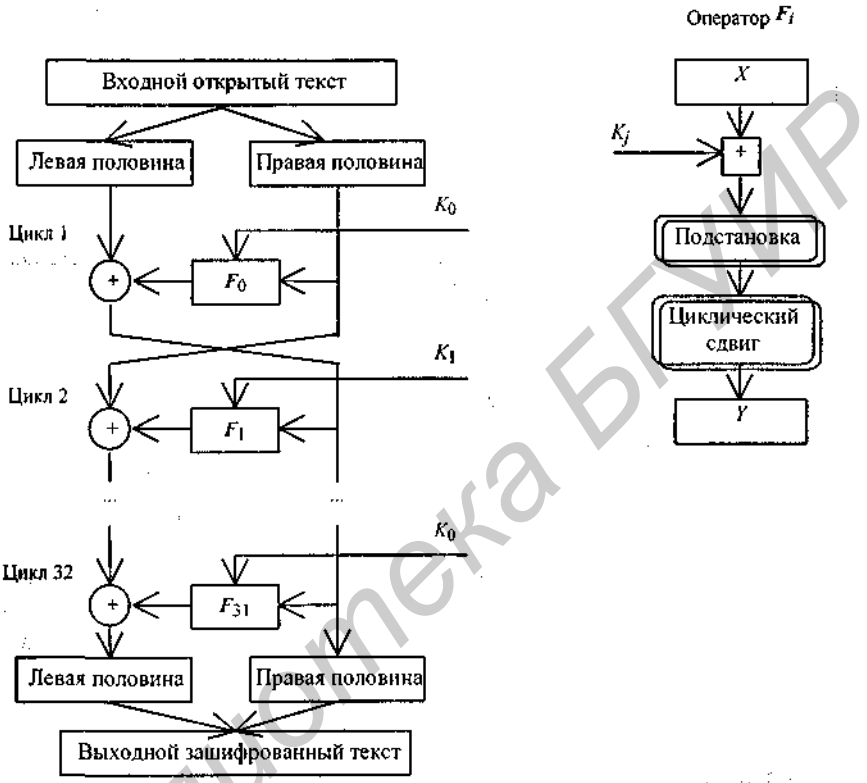


Рис. 4.20. Алгоритм шифрования в режиме простой замены ГОСТ 28147-89

4.2.2.1.5. RC5

Быстрый алгоритм блочного шифрования RC5 [67], предложенный Р.Ривестом в 1994 г., представляет собой параметризованный алгоритм с переменной длиной блока, переменным размером ключа и переменным числом циклов. Блок может иметь длину 32, 64 или 128 бит. Число циклов варьируется от 0 до 255, размер ключа - от 0 до 255 байт.

Секретный b-байтный ($0 \le b \le 255$) ключ KS преобразуется в расширенный ключ $K = K_0 K_1 \ldots K_{2R+1}$, где R - число циклов, а размер W слова K_i может быть 16, 32 или 64 бита. Построение расширенного ключа, кото-

рое может быть проведено на этапе предвычислений, включает в себя следующие операции:

- представление b-байтного ключа в виде последовательности W-битных слов $L=L_0L_1...L_{r-1}$, где $r=8(b-1)/W+1$;
- построение фиксированной (не зависящей от ключа KS) псевдослучайной последовательности $S=S_0S_1...S_{2R+1}$ на основе чисел $e=2.71828...$ (основание натурального логарифма) и $\varphi=1.61803...$ (золотое сечение);
- получение расширенного ключа K из последовательностей L и S по алгоритму, аналогичному алгоритму шифрования, в котором на каждом шаге:

$a \leftarrow (S_i+a+b)<<<3$; $S_i \leftarrow a$; $b \leftarrow (L_j+a+b)<<<(a+b)$; $L_j \leftarrow b$,

где "+" - сложение по модулю 2^W, "$<<<x$" - циклический сдвиг влево на x бит.

Исходными значениями для a, b, i, j являются нули, а число итераций определяется размером ключа KS.

При выполнении алгоритма зашифрования (рис. 4.21):

1. Блок входного открытого текста длины $2W$ бит разбивается на два полублока - $\boldsymbol{A}$ и $\boldsymbol{B}$ по W бит;
2. Первые два слова расширенного ключа K_0 и K_1 суммируются по модулю 2^W с полублоками $\boldsymbol{A}$ и $\boldsymbol{B}$ соответственно;
3. Выполняется циклическая итерация:

$\boldsymbol{A} \leftarrow ((\boldsymbol{A} \oplus \boldsymbol{B}) <<< W_{\boldsymbol{B}}) + K_{2i}$,

$\boldsymbol{B} \leftarrow ((\boldsymbol{B} \oplus \boldsymbol{A}) <<< W_{\boldsymbol{A}}) + K_{2i+1}$, $i = 1, ..., R$,

где "+" - сложение по модулю 2^W, "$<<< W_{\boldsymbol{X}}$" - циклический сдвиг влево на $\boldsymbol{X}$, т.е. сдвиг на $X \pmod W$;

Выходным зашифрованным текстом является конкатенация итоговых значений полублоков $\boldsymbol{A}$ и $\boldsymbol{B}$ после R циклов.

При расшифровании операции выполняются в обратном порядке с тем же расширенным ключом.

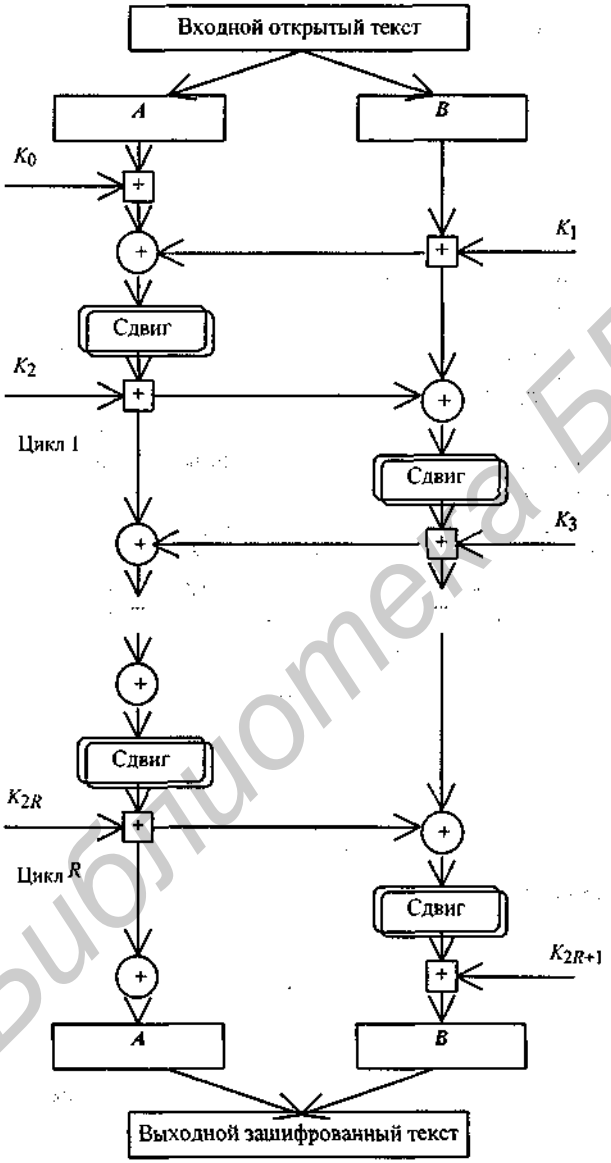


Рис. 4.21 Схема алгоритма RC5

4.2.2.1.6. Blowfish

Алгоритм шифрования Blowfish разработан Б.Шнейером в 1993 г. [71] и предназначен специально для применения в больших микропроцессорах. При создании алгоритма учитывались такие критерии, как:

- скорость (*при шифровании на 32-разрядном процессоре выполняется* 26 элементарных операций на один байт данных);
- компактность (реализация алгоритма требует менее 5 Кбайт оперативной памяти);
- простота (используются только простые операции).

Blowfish представляет собой блочный шифратор с переменной длиной ключа. Размер блока - 64 бита. Основные операции шифрования: подстановка, сложение по модулю 2 и по модулю 2^{32}.

Для построения расширенного ключа длиной 4168 байт из секретного ключа, длина которого не превышает 448 бит, 521 раз применяется сам алгоритм зашифрования. Поэтому оптимально использовать Blowfish в таких приложениях, где не требуется частая смена ключей. Построение расширенного ключа, включающего в себя восемнадцать 32-битных подключей K_1, K_2, ..., K_{18}, и четыре 32-битные подстановки (*S*-блоки)

$$S_1 = (S_{1,0}, S_{1,1},..., S_{1,255}),$$
$$S_2 = (S_{2,0}, S_{2,1},..., S_{2,255}),$$
$$S_3 = (S_{3,0}, S_{3,1},..., S_{3,255}),$$
$$S_4 = (S_{4,0}, S_{4,1},..., S_{4,255}),$$

может быть выполнено на этапе предвычислений.

Зашифрование 64-битного блока входного текста включает в себя (рис. 4.22):

1. Разбиение блока на два 32-битных полублока x' и x''.
2. Выполнение на каждом цикле $i = 1, ..., 16$ следующих операций:

- левый полублок x' суммируется по модулю 2 с соответствующим подключом K_i;
- полученная сумма x разбивается на четыре 8-битных подблока a, b, c, d, каждый из которых является адресом элемента соответствующей подстановки S_j, $1 \le j \le 4$. 32-битный результат обработки полублока оператором F формируется по формуле:

$$F(x) = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d},$$

где "⊕" - сложение по модулю 2, "+" - сложение по модулю 2^{32}.

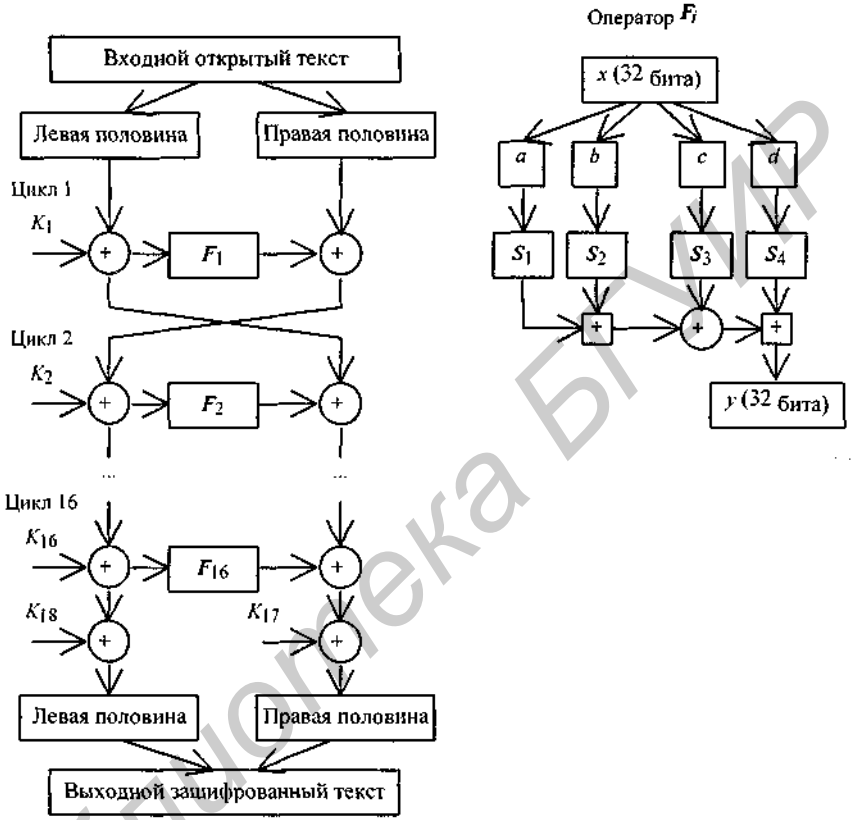


Рис. 4.22 Схема алгоритма

- компоненты левого зашифрованного полублока $F(x')$ суммируются по модулю 2 с компонентами правого полублока x'';
- компоненты левого и правого полублоков меняются местами (на последнем цикле этот шаг не выполняется).

3. После 16 циклов левая и правая половина полученного текста суммируются по модулю 2 с подключами K_{18} и K_{17} соответственно.

Выходным зашифрованным текстом является конкатенация двух полублоков, полученных на шаге 3.

Расшифрование осуществляется аналогично, с обратным порядком использования подключей: $K_{18}, K_{17}, ..., K_1$.

Подключи $K_1, K_2, ..., K_{18}$ *и S-блоки* S_1, S_2, S_3, S_4 *вырабатываются* из секретного ключа *KS* с использованием алгоритма зашифрования, а именно:

(1) начальный массив K_i подключей и *S*-блоков инициируется фиксированной 128-битной строкой (дробной частью числа π=3,14159... в шестнадцатиричном представлении). Например:

K_1 = 243f6a88, K_2= 85a308d3, K_3 = 13198a2e, K_4 = 03707344.

(2) K_1 суммируется по модулю 2 с первыми 32 битами ключа *KS*, K_2 суммируется по модулю 2 со следующими 32 битами ключа *KS* и т.д. для всех разрядов ключа. В случае короткого ключа *KS*, например, длиной 64 бита, для построения массива K_i используется конкатенация ключа вида: *KSKS*, *KSKSKS* и т.д.

(3) 64-битный блок **0**=(0, 0, ..., 0), зашифровывается алгоритмом Blowfish на подключах, полученных на шагах (1) и (2).

(4) K_1 и K_2 заменяются полученным на шаге (3) результатом зашифрования *C0*=Blowfish(**0**).

(5) Шифртекст *C0* зашифровывается алгоритмом Blowfish на модифицированных подключах.

(6) K_3 и K_4 заменяются полученным на шаге (5) результатом зашифрования *C1*=Blowfish(*C0*).

(7) Процесс продолжается циклически до тех пор, пока не будут получены сначала все пары подключей (9 пар), а затем все пары элементов *S*-блоков (512 пар). Таким образом, всего выполняется 521 цикл зашифрования, на каждом из которых происходит модификация подключей.

Полученные массивы подключей и S-блоков используются для шифрования данных.

4.2.2.1.7. Сравнительный анализ представленных симметричных алгоритмов шифрования

Стойкость алгоритма шифрования в немалой степени зависит от его параметров, таких как размер ключа, длина блока входного текста, нелинейность6 подстановки, число циклов шифрования и др.

Блок подстановок в DES допускает хорошую аппроксимацию аффинными преобразованиями. Многие булевы функции, используемые в

6 Здесь под *нелинейностью* подстановки понимается наименьшая из нелинейностей булевых функций, задающих подстановку. *Нелинейностью булевой функции* называется хеммингово расстояние между этой функцией и ближайшей аффинной функцией [29].

подстановках, отличаются от аффинных функций лишь для двух из шестнадцати возможных наборов аргументов, т.е. нелинейность подстановки DES равна двум (максимально достижимая нелинейность 4-битной подстановки равна четырем). Это обусловливает уязвимость DES к дифференциальному и линейному методам криптоанализа. Кроме того, DES обладает свойством дополнения, а именно, если x, z - открытый текст и ключ соответственно, то имеет место равенство $\mathrm{DES}(\bar{x},\bar{z}) = \overline{\mathrm{DES}(x,z)}$, где $\bar{x} = x \oplus \mathbf{1}$ ($\mathbf{1}$ - единичный вектор). При подборе ключа это позволяет нарушителю вдвое сократить объем перебора, т.е. ключ можно искать с точностью до инверсии.

Таблица 4.2. *Параметры представленных алгоритмов шифрования*

Алгоритм шифрования	Размер ключа, бит	Длина блока, бит	Число циклов	Основные операции
DES	56	64	16	подстановка, перестановка, ⊕
FEAL	64, 128	64	≥ 4	сложение по модулю 2^8, циклический сдвиг, ⊕
IDEA	128	64	8	умножение по модулю $2^{16}+1$, сложение по модулю 2^{16}, ⊕
ГОСТ 28147-89	256	64	32	сложение по модулю 2^{32}, подстановка, циклический сдвиг, ⊕
RC5	$8t$, t≤255	32, 64, 128	≤255	сложение по модулю 2^W (W = 1/2 длины блока), циклический сдвиг, ⊕
Blowfish	≤448	64	16	сложение по модулю 2^{32}, подстановка, ⊕

До 1990 г. перебор был наилучшим алгоритмом раскрытия ключа DES. Появление дифференциального метода криптоанализа привело к снижению стойкости сначала "усеченных" версий [24], а затем и полного алгоритма до уровня 2^{37} при требуемом объеме известных блоков открытого текста 2^{36} [26]. В дальнейшем стойкость DES была снижена линейным методом. Следует отметить, что раскрытие ключа дифференциальным методом проводится в предположении, что на каждом цикле используется свой ключ. Поэтому увеличение объема ключа DES не позволяет заметно увеличить стойкость.

В алгоритме FEAL операции сложения по модулю 2 и 256 похожи и различаются только переносом. Поэтому нелинейность оператора S_i здесь мала. Это обусловливает уязвимость FEAL к дифференциальному [25] и линейному [61] методам криптоанализа. Выбором соответствующих открытых текстов можно дополнительно уменьшить влияние переносов.

Стойкость FEAL падала заметно быстрее чем стойкость DES. Ключ 8-циклового алгоритма FEAL раскрывается дифференциальным методом с помощью 2000 подобранных открытых текстов на персональном компьютере всего за 2 минуты [25].

Алгоритм IDEA благодаря использованию операции умножения по модулю $2^{16}+1$, обладающей сильным перемешивающим эффектом, представляется достаточно стойким по отношению к линейному криптоанализу. Стойкость этого алгоритма по отношению к дифференциальному методу криптоанализа не очевидна.

IDEA ориентирован на программную или аппаратную реализацию с использованием встроенного аппаратного умножителя. Однако даже в этом случае умножение по модулю $2^{16}+1$ выполняется программно заметно медленнее, чем сложение, что обусловлено необходимостью выполнения дополнительных операций, кроме собственно умножения 16-битных чисел. Количество машинных тактов для шифрования IDEA в программной реализации при отсутствии специальных мер зависит от вида ключа и шифруемого текста. Поэтому точное измерение длительности шифрования каждого блока позволяет извлечь дополнительную информацию о ключе. Очевидно, это обстоятельство может заметно снизить стойкость IDEA. Кроме того, для этого алгоритма существует перечислимый класс слабых ключей7 [33].

В алгоритме шифрования ГОСТ 28147-89 блок подстановки не фиксирован, как в DES, и является секретным параметром. Ключ гораздо больше - 256 бит, что делает невозможной атаку перебором. Опыт DES показывает, что выбор подстановки решающим образом влияет на стойкость шифра. Механизмы рассеивания в ГОСТ 28147-89 и DES различаются. Если в DES рассеивание достигается перестановкой бит в блоке текста и подстановкой, то в ГОСТ 28147-89 оно осуществляется в основном сложением по модулю 2^{32}, подстановкой и сдвигом. Для повышения стойкости к дифференциальному и линейному методам криптоанализа желательно выбирать экстремальные подстановки с нелинейностью 4 и рассеиванием 1 [41]. Кроме того, наиболее вероятная разность двух выходов подстановки при фиксированной разности входов должна иметь малую вероятность (разности определяются суммой по модулю 2). Однако нахождение таких подстановок сопряжено со значительными трудностями. Число циклов в ГОСТ 28147-89 по сравнению с DES увеличено вдвое.

---

7 *Слабым ключом* для данного алгоритма шифрования называют такой секретный ключ, при использовании которого этот алгоритм проявляет регулярные свойства, имеющие простое описание. Например, для DES существуют четыре ключа такие, что если исходный открытый текст P дважды зашифровать на одном из этих ключей K_i, то в результате опять получится открытый текст: $E_{K_i}(E_{K_i}(P))=P$ [58].

Криптоанализ усеченного 24-циклового ГОСТ 28147-89 без последних восьми циклов показал, что стойкость его превышает 2^{54} для случайного блока подстановки [48].

Алгоритм RC5 по скорости совпадает с DES или превосходит его. Этот алгоритм может быть безопаснее DES, поскольку использует более длинный ключ. RC5 очень прост в реализации, а значения встроенных параметров - размера блока, числа циклов и длины ключа - могут удовлетворять самым разным требованиям обеспечения защиты и производительности. RC5 имеет слабые ключи [48], для которых циклический сдвиг оказывается нулевым, что следует учитывать, в частности, при построении на его основе хэш-функций. В 1995 г. сложность дифференциального и линейного методов криптоанализа 12-циклового RC5 с длиной блока 64 бита превышала сложность перебора [47], однако, согласно результатам криптоанализа [27], для раскрытия ключа 12-циклового RC5 требуется 2^{44} подобранных открытых текстов, что существенно ниже сложности перебора.

Исследования Blowfish показали [77], что для этого алгоритма существуют слабые ключи (*S*-блоки, в которых есть одинаковые слова). При использовании таких ключей дифференциальный криптоанализ позволяет восстановить массив подключей для 8 циклов с помощью 2^{23} выбранных открытых текстов, а для 16 циклов с помощью $3 \cdot 2^{51}$ выбранных открытых текстов. Если слабые ключи не используются, то для 8 циклов восстановить массив подключей можно с помощью 2^{48} выбранных открытых текстов.

В 1998 г. алгоритм Blowfish представлен в качестве кандидата на стандарт шифрования США.

4.2.2.2. Шифрование с открытым ключом

В несимметричных алгоритмах шифрования ключи зашифрования и расшифрования всегда разные. Ключ зашифрования K является несекретным (открытым), ключ расшифрования K^* - секретным, т.е. текст может быть зашифрован с помощью открытого ключа любым источником, а расшифрован лишь приемником, знающим секретный ключ расшифрования.

4.2.2.2.1. Алгоритм шифрования RSA

Широко известная криптосистема с открытым ключом RSA является составной частью многих стандартов, в том числе International Standards Organization (ISO) 9796, Society for Worldwide Interbank Financial Telecommunications (SWIFT), ANSI X9.31, французского стандарта

ETEBAC 5, австралийского AS2805.6.5.3 и др. Эта система была предложена в 1977 г. Р.Ривестом, Э.Шамиром и Л.Адлманом [68]. Принцип ее действия заключается в следующем:

Пусть p и q - два больших простых числа и пусть $n = pq$. Пусть число e, $0<e<n$, такое, что числа e и $\varphi(n)$, где $\varphi(n) = (p-1)(q-1)$, взаимно простые, а d - мультипликативно обратное к нему $d \equiv e^{-1} \bmod \varphi(n)$, то есть $ed \equiv 1 \bmod \varphi(n)$. Числа e и d называются открытым и секретным показателями соответственно. Открытым ключом является пара (n, e), секретным ключом - число d. Множители p и q должны храниться в секрете.

Предполагается, что зная открытый ключ (n, e), трудно вычислить секретный ключ d, однако, если удастся разложить n на множители p и q, то можно узнать d. Таким образом, общая безопасность системы RSA основана на трудности задачи разложения.

Для зашифрования сообщения P источник вычисляет шифртекст $C=P^e \pmod n$ и передает его приемнику. Тот, в свою очередь, расшифровывает полученное сообщение, вычисляя $P=C^d \pmod n$.

4.2.2.2.2. Алгоритм шифрования Эль-Гамаля

В алгоритме шифрования Эль-Гамаля [38] открытым ключом зашифрования является совокупность:

- простое число p;
- образующая g группы F_p^*, порядок которой имеет большой простой делитель;
- экспонента $y = g^x \pmod p$.

Секретным ключом расшифрования служит целое число x, $0<x<p-1$.

Для зашифрования открытого текста $P \in F_p$ источник выполняет следующие действия:

1. Выбирает случайное число k, $0<k<p-1$.
2. Вычисляет $r \equiv y^k \pmod p$.

Шифртекстом является пара (C_1, C_2), где

$$C_1 = g^k \pmod p,$$

$$C_2 = rP \pmod p.$$

Приемник выполняет алгоритм расшифрования:

1. Восстанавливает r с помощью своего секретного ключа x, т.е. вычисляет

$$r = (C_1)^x \pmod p.$$

2. Восстанавливает открытый текст P, вычисляя

$$P = r^{-1} C_2 \pmod{p}.$$

4.2.2.2.3. Алгоритм шифрования Месси - Омуры

В алгоритме шифрования Месси - Омуры [51] открытым ключом зашифрования является простое число p такое, что p-1 имеет большой простой делитель, а секретного ключа нет вообще. Зашифрование и расшифрование сообщения происходят в процессе диалога между источником и приемником.

Источник:

1. Выбирает случайное число e, $0<e<p$-1, взаимно простое с p-1, и вычисляет $d=e^{-1} \pmod{p\text{-}1}$.
2. Зашифровывает исходный открытый текст P, вычисляя
$$C = P^e \pmod{p},$$
и посылает C приемнику.

Получив C, приемник:

1. Выбирает случайное число e', $0<e'<p$-1, взаимно простое с p-1, и вычисляет $d'=(e')^{-1} \pmod{p\text{-}1}$.
2. Вычисляет
$$C' = C^{e'} \pmod{p},$$
и передает C' обратно источнику.

Источник возводит C' в степень d:
$$P' = C'^{d} \pmod{p}$$
и передает P' приемнику.

Приемник восстанавливает исходный открытый текст, вычисляя
$$P = P'^{d'} \pmod{p}.$$

4.2.2.2.4. Сравнительный анализ представленных несимметричных алгоритмов шифрования

Алгоритмы RSA и Эль-Гамаля не требуют диалога и в этом смысле аналогичны симметричным алгоритмам шифрования. Алгоритм Месси - Омуры требует диалога, что накладывает ограничения на область его применения.

В алгоритме Эль-Гамаля шифртекст в два раза длиннее, чем исходный открытый текст. Кроме того, в этом алгоритме при зашифровании вычисляются две экспоненты, а в RSA - только одна, т.е. зашифрование алгоритмом RSA осуществляется в два раза быстрее. В алгоритмах Эль-Гамаля и Месси - Омуры, в отличие от RSA, одинаковые открытые тексты переходят в разные шифртексты, что затрудняет исследование их статистических свойств.

Безопасность алгоритма RSA основана на сложности разложения составного числа, а безопасность протокола Эль-Гамаля - на сложности дискретного логарифмирования.

При использовании в алгоритме RSA малых показателей зашифрования (e) возможно бесключевое чтение, если источник передает разным приемникам близкие открытые тексты [18]. Использование общего секретного ключа двумя пользователями приводит к взаимному раскрытию их секретных ключей.

Для предотвращения бесключевого чтения в протоколе Эль-Гамаля необходимо, чтобы случайные числа k были непредсказуемыми и с большой вероятностью неповторяющимися.

Алгоритмы шифрования Эль-Гамаля и Месси - Омуры допускают обобщение на случай произвольной конечной группы, например, группы точек эллиптической кривой. Однако в этом случае открытый текст должен являться элементом группы. Поэтому алгоритм шифрования должен предусматривать стадию преобразования произвольного текста в элемент группы (при зашифровании) и обратно (при расшифровании).

4.2.2.3. Сравнение симметричных и несимметричных алгоритмов шифрования

Основное преимущество несимметричных алгоритмов перед симметричными состоит в том, что секретный ключ, позволяющий расшифровывать всю получаемую информацию, известен только приемнику. Кроме того, первоначальное распределение ключей в системе не требует передачи секретного ключа, который может быть перехвачен нарушителем. Несимметричные алгоритмы получили новое качество - на их основе строятся протоколы цифровой подписи (п. 4.2.4). Для аутентификации с использованием симметричных алгоритмов часто требуется участие доверенной третьей стороны, которая, как, например, в схеме Kerberos, хранит копии секретных ключей всех пользователей. Компрометация третьей стороны может привести к компрометации всей системы аутентификации. В системах с открытым ключом эта проблема устранена - каждый пользователь отвечает за безопасность только своего секретного ключа.

Симметричные алгоритмы, при обнаружении в них каких-либо слабостей, могут быть доработаны путем внесения небольших изменений, а для несимметричных такая возможность отсутствует.

Симметричные алгоритмы являются значительно более быстрыми, чем алгоритмы с открытым ключом, используемые для шифрования. На практике несимметричные алгоритмы шифрования обычно применяются в совокупности с симметричными алгоритмами: открытый текст зашифровывается симметричным алгоритмом, а секретный ключ этого симметричного алгоритма зашифровывается на открытом ключе несим-

метричного алгоритма. Такой механизм называют *цифровым конвертом* (*digital envelope*). Вот, например, как он реализуется для алгоритмов DES и RSA [70].

Пусть K - секретный ключ алгоритма DES, e и d - соответственно открытый и секретный ключ алгоритма RSA.

Источник зашифровывает:

1. открытый текст P алгоритмом DES на секретном ключе K:
$$C = DES_K(P);$$
2. секретный ключ алгоритма DES алгоритмом RSA на открытом ключе e:
$$C_K = K^e \pmod n$$
и отправляет приемнику пару (C, C_K).

Приемник, в свою очередь:

1. расшифровывает секретный ключ алгоритма DES алгоритмом RSA на секретном ключе d:
$$K = (C_K)^d \pmod n$$
2. восстанавливает открытый текст P, расшифровывая алгоритмом DES шифртекст C с помощью вычисленного на предыдущем шаге секретного ключа K:
$$P = (DES_K)^{-1}(C).$$

4.2.3. Контроль целостности данных

Средства *контроля целостности* защищают данные от случайного или умышленного изменения (в том числе утраты, повтора, переупорядочения или замены).

Для защиты данных от случайных повреждений используются корректирующие коды, для защиты от умышленных искажений - криптографические средства, которые и будут здесь рассмотрены.

4.2.3.1. Способы контроля целостности

Целостность отдельного сообщения обеспечивается имитовставкой, цифровой подписью или зашифрованием, целостность потока сообщений (помогающая предотвратить повтор, переупорядочение или утрату данных) - соответствующим механизмом целостности.

Схема контроля целостности данных подразумевает выполнение двумя сторонами - *источником* и *приемником* - некоторых (возможно, разных) криптографических преобразований данных (рис. 4.23). Источник преобразует исходные данные и передает их приемнику вместе с некоторым приложением, обеспечивающим избыточность шифрограммы.

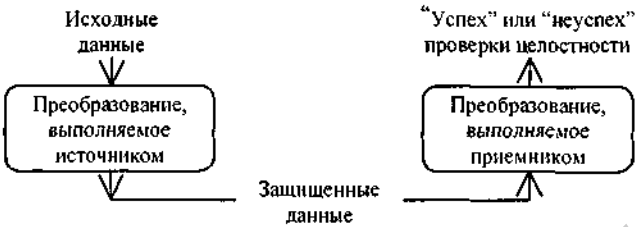


Рис. 4.23 Схема контроля целостности данных

Приемник обрабатывает полученное сообщение, отделяет приложение от основного текста и проверяет их взаимное соответствие, осуществляя таким образом контроль целостности. Контроль целостности может выполняться *с восстановлением* или и *без восстановления* исходных данных.

Конкретное криптографическое преобразование является частью общего протокола взаимодействия. В различных средах целостность передаваемых сообщений может обеспечиваться различными сочетаниями основных механизмов (рис. 4.24). - подписи, зашифрования, контрольного значения (КЗ) и порядкового номера целостности (ПНЦ).

4.2.3.1.1. Цифровая подпись и имитовставка как способы обеспечения целостности

Для обеспечения целостности в текст сообщения часто вводится некоторая дополнительная информация, которая легко вычисляется, если секретный ключ известен, и является трудновычислимой в противном случае. Если такая информация вырабатывается и проверяется с помощью одного и того же секретного ключа, то ее называют *имитовставкой* (в зарубежных источниках используется термин *код аутентификации сообщений* - Message Authentication Code (MAC) - поскольку помимо целостности может обеспечиваться еще и аутентификация объекта. Имитовставкой может служить значение хэш-функции (п. 4.2.5), зависящей от секретного ключа, или выходные данные алгоритма зашифрования в режиме сцепления блоков шифра (CBC) [43].

В отличие от имитовставки для цифровой подписи (п. 4.2.4) обычно используют несимметричный криптографический алгоритм с различающимися ключами создания и проверки подписи, в котором знание открытого ключа проверки не позволяет вычислить секретный ключ создания.

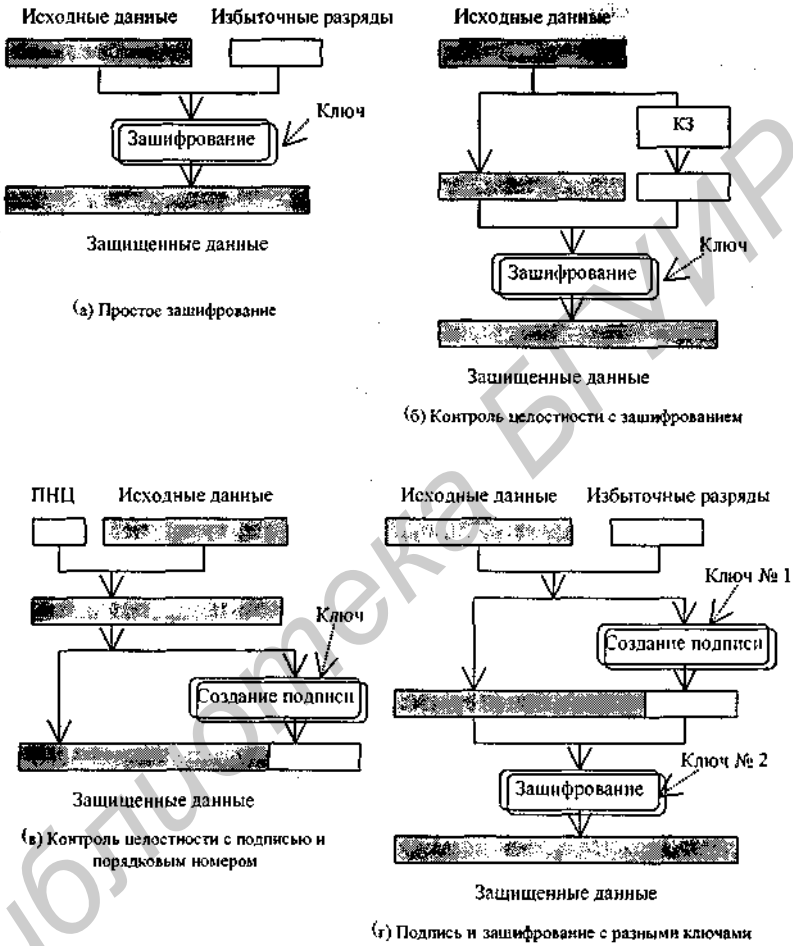


Рис. 4.24 Примеры схем контроля целостности

4.2.3.1.2. Зашифрование как способ обеспечения целостности

Целостность данных можно обеспечить и с помощью их зашифрования симметричным криптографическим алгоритмом при условии, что подлежащий защите текст обладает некоторой избыточностью. Последняя необходима для того, чтобы нарушитель, не зная ключа зашифрования, не

смог создать шифрограмму, которая после расшифрования успешно прошла бы проверку целостности.

Избыточности можно достигнуть многими способами. В одних случаях текст может обладать достаточной естественной избыточностью (например, в любом языке разные буквы и буквосочетания встречаются с разной частотой). В других можно присоединить к тексту до зашифрования некоторое КЗ, которое, в отличие от имитовставки и цифровой подписи, не обязательно должно вырабатываться криптографическими алгоритмами, а может представлять собой просто последовательность заранее определенных символов.

4.2.3.1.3. Контроль целостности потока сообщений

Контроль целостности потока сообщений помогает обнаружить их повтор, задержку, переупорядочение или утрату. Предполагается, что целостность каждого отдельного сообщения обеспечивается зашифрованием, имитовставкой или цифровой подписью. Для контроля целостности потока сообщений можно, например:

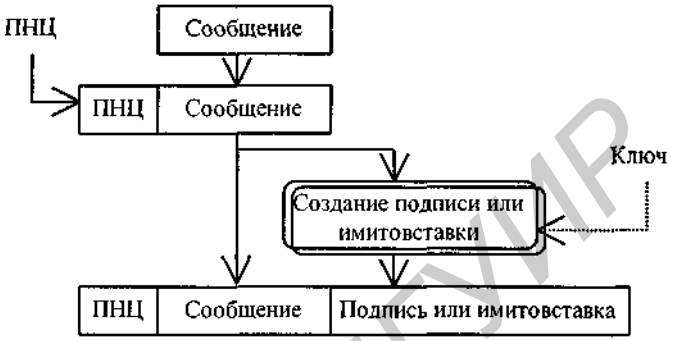
- присвоить сообщению *порядковый номер целостности* (ПНЦ); или
- использовать в алгоритмах зашифрования *сцепление* с предыдущим сообщением.

На рис. 4.25 приведены типичные процедуры, выполняемые источником в этих двух случаях.

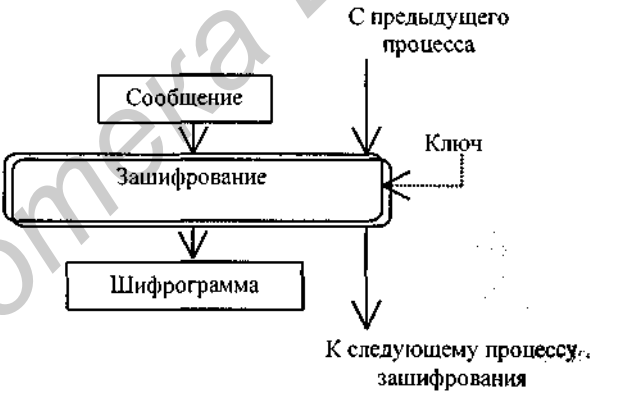
При использовании ПНЦ, который может включать в себя порядковый номер сообщения и имя источника, приемник хранит последний номер принятого сообщения для каждого источника. Для контроля целостности приемник проверяет, например, что номер $ПНЦ_i^{(S)}$ текущего сообщения от данного источника S на единицу больше номера предыдущего сообщения: $ПНЦ_i^{(S)} = ПНЦ_{i-1}^{(S)} + 1$. Если в качестве ПНЦ используется время отправки сообщения, то проверяется, что время отправки и время приема близки друг к другу с точностью до задержки сообщения в канале связи и разности хода часов источника и приемника.

4.2.4. Цифровая подпись

Цифровая подпись в цифровых документах играет ту же роль, что и подпись, поставленная от руки в документах, напечатанных на бумаге: это данные, присоединяемые к передаваемому сообщению, подтверждающие, что владелец подписи (претендент) составил или заверил данное сообщение. Получатель сообщения (верификатор) или третья сторона с помощью цифровой подписи может проверить, что автором сообщения является именно владелец подписи (т.е. аутентифицировать источник



(а) Использование порядкового номера целостности



(б) Сцепление

Рис. 4.25. Механизмы контроля целостности потока сообщений

данных) и что в процессе передачи не была нарушена целостность полученных данных.

Таким образом, при разработке механизма цифровой подписи возникают две основные задачи:

- создание подпись таким образом, чтобы ее невозможно было подделать;
- возможность проверки того, что подпись действительно принадлежит указанному владельцу.

Кроме того, необходимо предотвратить возможность отказа от подписи, т.е. подпись должна быть построена таким образом, чтобы претендент, подписавший сообщение, не смог затем отрицать перед верификатором или третьей стороной факт подписания, утверждая, что подпись подделана.

4.2.4.1. Классические схемы подписи

Обычно, говоря о схеме цифровой подписи, имеют в виду следующую ситуацию:

- Претендент знает содержание сообщения, которое он подписывает.
- Верификатор, зная открытый ключ проверки подписи, может проверить правильность подписи в любое время без какого-либо разрешения или участия претендента.
- Безопасность схемы подписи (т.е. трудность подделки, невозможность отказа от подписи и т.д.) гарантируется некоторыми теоретико-сложностными положениями.

При создании цифровой подписи по классической схеме претендент (рис. 4.26):

1) применяет к исходному тексту хэш-функцию;
2) дополняет (при необходимости) хэш-образ сообщения до длины, требуемой в алгоритме создания подписи; и
3) вычисляет *цифровую подпись* по (дополненному) хэш-образу сообщения с использованием секретного ключа создания подписи.

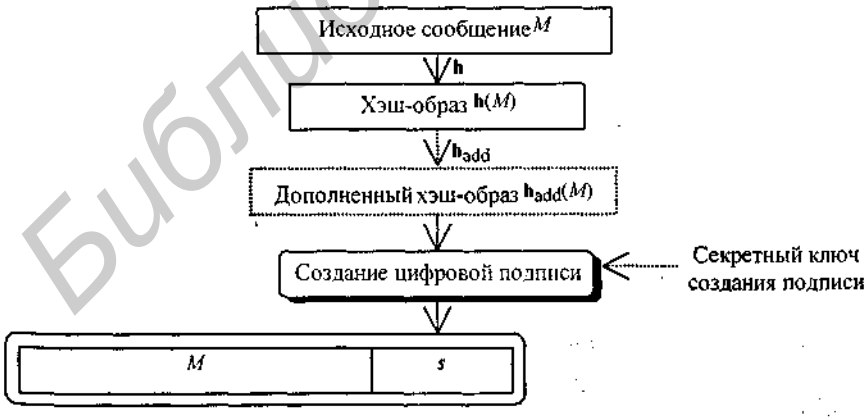


Рис. 4.26. Классическая схема создания цифровой подписи

Верификатор, получив подписанное сообщение, отделяет цифровую подпись от основного текста и выполняет проверку подписи, а именно (рис. 4.27):

1) применяет к тексту полученного сообщения хэш-функцию;
2) дополняет (при необходимости) хэш-образ сообщения до длины, требуемой в алгоритме проверки подписи;
3) проверяет соответствие (дополненного) хэш-образа сообщения полученной цифровой подписи с использованием открытого ключа проверки подписи.

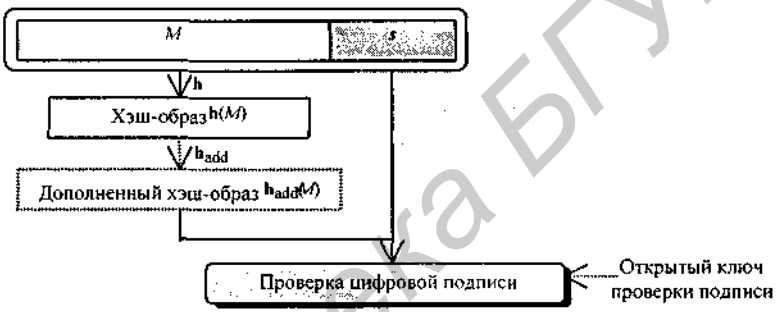


Рис. 4.27 Классическая схема проверки цифровой подписи

Дополнение хэш-образа нужно лишь в том случае, если необходимая длина не обеспечивается алгоритмом хэш-функции. На практике, как правило (например, в стандартах DSS и ГОСТ Р34.10-94), такое дополнение не требуется.

Классические схемы подписи можно разделить по типу вычислимости порядка группы. Безопасность протоколов, использующих группу вычислимого порядка, основана на сложности логарифмирования в группе, безопасность протоколов, использующих группу трудновычислимого порядка, - на сложности вычисления порядка группы.

4.2.4.1.1. Подпись на группе трудновычислимого порядка

Для этого класса подписей используется группа, для которой вычисление порядка эквивалентно труднорешаемой задаче разложения составного числа на простые множители.

4.2.4.1.1.1. Схема подписи RSA

Криптосистема с открытым ключом RSA может использоваться не только для шифрования, но и для построения схемы цифровой подписи.

Для создания подписи сообщения M претендент:

1) вычисляет сжатый образ $r = h(M)$ сообщения M с помощью хэш-функции (на практике для подписи RSA часто используются хэш-функции MD4 и MD5 (п. 4.2.5.1));
2) зашифровывает полученный сжатый образ $h(M)$ на своем секретном ключе d, т.е. вычисляет экспоненту $s \equiv r^d \bmod n$, которая и является подписью;

Для проверки подписи верификатор:

1) расшифровывает подпись s на открытом ключе e претендента, т.е. вычисляет $r' \equiv s^e \bmod n$ и таким образом, восстанавливает предполагаемый сжатый образ r' сообщения M;
2) вычисляет сжатый образ $r = h(M)$ сообщения M с помощью той же самой хэш-функции, которую использовал претендент;
3) сравнивает полученные значения r и r'. Если они совпадают, то подпись правильная, претендент действительно является тем, за кого себя выдает, и сообщение не было изменено при передаче.

4.2.4.1.1.2. Схема подписи Рабина

Вариантом RSA является схема подписи, предложенная в 1979 г. М.Рабином [63], в основу которой положена следующая теорема [55].

Теорема 3.1. *Задача нахождения всех четырех решений квадратичного сравнения* $y^2 \equiv \alpha \pmod{pq}$ *эквивалентна задаче разложения на множители* $n = pq$.

Пусть p и q - большие простые числа, $n = pq$, и пусть сообщение M представлено в виде двоичной последовательности

$$M = (m_0, m_1, ..., m_{t-1}).$$

Для создания подписи претендент:

1) вырабатывает случайное число r в виде двоичной последовательности:

$$r = (r_0, r_1, ..., r_{s-1}).$$

и сопоставляет конкатенации

$$(m_0, m_1, ..., m_{t-1}, r_0, r_1, ..., r_{s-1})$$

целое число $\alpha \in Z_n$ ($0 \le \alpha < n$). Например, сжимает последовательность $(m_0, m_1, ..., m_{t-1}, r_0, r_1, ..., r_{s-1})$ с помощью хэш-функции и находит вычет сжатого образа по модулю n;

2) проверяет, является ли α квадратичным вычетом по модулю n, для чего, например, возводит α в степени $(p-1)/2$ и $(q-1)/2$ или вычисляет символ Якоби $\left(\frac{\alpha}{n}\right)$ (при надлежащем выборе отображения M, $r \to \alpha$ вероятность выбора такого r, для которого соответствующее α будет являться квадратичным вычетом по модулю n, приблизительно равна $1/4$);
3) если α - квадратичный невычет по модулю n, то возвращается на шаг 1 (генерирует новое значение r и вычисляет новое α);
4) если α - квадратичный вычет по модулю n, то вычисляет значение $\beta \in Z_n$ такое, что $\beta^2 \equiv \alpha \pmod n$, а именно:

– находит $\beta_p \equiv \alpha^{1/2} \pmod p$,
– находит $\beta_q \equiv \alpha^{1/2} \pmod q$,
– восстанавливает $\beta \equiv \alpha^{1/2} \pmod n$ по **китайской теореме об остатках**, т.е. вычисляет

$$\beta \equiv \beta_p q(q^{-1} \bmod p) + \beta_q p\,(p^{-1} \bmod q) \bmod (pq).$$

Подписью для сообщения M является пара (r, β).

Для проверки подписи верификатор:

1) возводит число β в квадрат по модулю n:

$$\alpha \equiv \beta^2 \pmod n,$$

и находит двоичное представление числа α, $\alpha = (\alpha_0, \alpha_1, ..., \alpha_{t+s-1})$;

2) если $\alpha_i = m_i$ для $0 \le i \le t-1$ и $\alpha_{i+t} = r_i$ для $0 \le i \le s-1$, то подпись верна, иначе подпись неправильная.

4.2.4.1.2. Поднись на группе вычислимого порядка

В отличие от протоколов RSA и Рабина, данные протоколы могут использовать произвольные группы вычислимого порядка, такие как группа точек эллиптической кривой [49, 51], якобиан гиперэллиптической кривой [50, 14] и др. В качестве примера будет рассмотрена подгруппа простого порядка q мультипликативной группы F_p^* простого поля.

4.2.4.1.2.1. Схема подписи Эль-Гамаля

Протокол подписи Эль-Гамаля [38] (рис. 4.28) строится следующим образом. Секретным ключом создания подписи служит случайное целое число x, $0<x<q$, открытым ключом проверки подписи - совокупность:

- простое число p;
- образующая g подгруппы;
- экспонента $y = g^x \pmod p$.

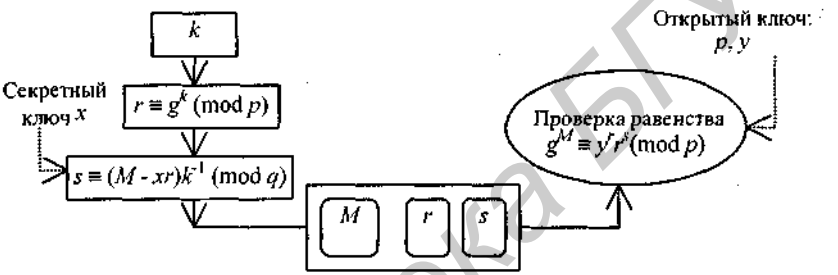


Рис. 4.28 Схема цифровой подписи Эль-Гамаля

Для создания подписи сообщения M, $0 < M < q$, претендент:

1) генерирует случайное число k, $0 < k < q$,
2) вычисляет $r \equiv g^k \pmod p$,
3) находит число s решением уравнения $M \equiv xr + ks \pmod q$:

$$s \equiv (M - xr)k^{-1} \pmod q. \qquad (*)$$

Так как числа k и q взаимно простые, то k обратимо по модулю q, то есть число s всегда существует и оно единственное.

Подписью для сообщения M является пара (r, s).

Для проверки подписи верификатор проверяет выполнение сравнения (*) для экспонент:

$$g^M \equiv y^r r^s \pmod p.$$

При создании подписи в протоколе Эль-Гамаля можно использовать предвычисления. Действительно, числа k и r никак не связаны с сообщением и могут быть вычислены заранее, что позволяет ускорить процедуру создания подписи. Проверка подписи является более трудоемким и требует вычисления трех экспонент.

4.2.4.1.2.2. Схема подписи Шнорра

В протоколе цифровой подписи К.Шнорра [72] (рис. 4.29) ключи создания и проверки подписи - те же, что и в протоколе Эль-Гамаля.

Для создания подписи сообщения M, $0 < M < q$, претендент:

1) генерирует случайное число k, $0 < k < q$,
2) вычисляет $r \equiv g^k \pmod p$,
3) находит $e = h(M, r)$, где h - хэш-функция;
4) вычисляет $s \equiv k - xe \pmod q$.

Подписью для сообщения M является пара (e, s).

Для проверки подписи верификатор

1) вычисляет $r' \equiv g^s (g^x)^e \pmod p$;
2) находит $e' = h(M, r')$, где h - та же хэш-функция, которой пользовался претендент;
3) проверяет выполнение равенства $e' = e$. Если оно выполняется, то подпись верна, иначе - подпись неправильная.

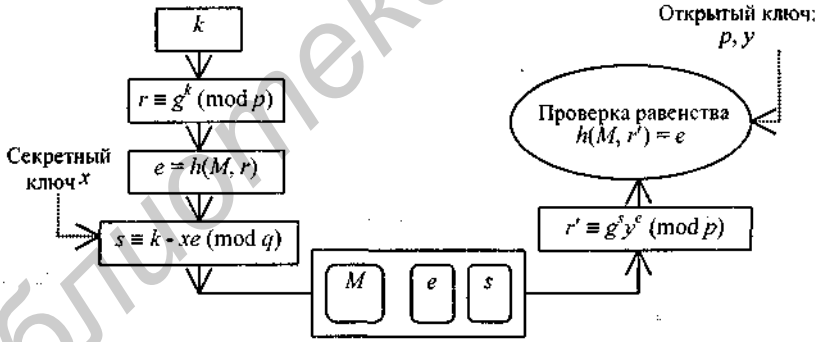


Рис. 4.29 Схема цифровой подписиШнорра

4.2.4.1.2.3. DSS и ГОСТ Р34.10-94

По принципу протокола цифровой подписи Шнорра созданы американский стандарт цифровой подписи DSS (Digital Signature Standard) [60] и отечественный ГОСТ Р34.10-94 [16], основанные на задаче дискретного логарифмирования в подгруппе простого порядка q простого поля характеристики p. В DSS число q имеет длину 160 бит, для создания и проверки подписи используется хэш-функция SHA-1 (п. 4.2.5.2).

В ГОСТ Р34.10-94 длина числа q - 256 бит и используется хэш-функция ГОСТ Р34.11-94).

Параметрами алгоритма цифровой подписи в стандарте DSS являются:

1. p - простое число, $2^{L-1} < p < 2^{L}$, где L кратно 64 и $512 \le L \le 1024$;
2. q - простой делитель числа p - 1, где $2^{159} < q < 2^{160}$;
3. g - образующая циклической группы порядка q;
4. x - случайное целое число, $0 < x < q$;
5. $y = g^{x} \bmod p$.

Целые числа p, q и g являются открытыми и могут быть общими для группы пользователей. Число x - секретный ключ создания подписи, y - открытый ключ проверки подписи.

Для создания подписи сообщения M претендент:

1) генерирует случайное целое число k, $0 < k < q$;
2) вычисляет $r = (g^{k} \bmod p) \bmod q$;
3) находит $e = h(M)$, где $h(M)$ - 160-битное значение хэш-функции SHA-1;
4) вычисляет $s = (k^{-1}(e + xr)) \bmod q$.

Подписью для сообщения M является пара (r, s).

Для проверки подписи верификатор:

1) вычисляет $w = s^{-1} \bmod q$;
2) находит $e = h(M)$, где $h(M)$ - 160-битное значение хэш-функции SHA-1;
3) вычисляет $u1 = (ew) \bmod q$; $u2 = (rw) \bmod q$, $r' = ((g^{u1} y^{u2}) \bmod p) \bmod q$;
4) проверяет выполнение равенства $r' = r$. Если оно выполняется, то подпись верна, иначе - подпись неправильная.

4.2.4.1.3. Сравнительный анализ представленных классических схем подписи

Сложность задачи разложения составного числа и сложность задачи логарифмирования в мультипликативной группе простого поля по отношению к асимптотически наилучшим универсальным алгоритмам разложения и логарифмирования (метод решета числового поля [40]) близки по порядку величины8, если длины характеристики поля p в сис-

8 Логарифмирование и разложение выполняются в два этапа: 1) строится база данных из вспомогательных уравнений; 2) решается система линейных сравнений (в задаче

теме Эль-Гамаля и характеристики кольца n в системе RSA одинаковы. Эта сложность оценивается субэкспонентой $exp(c*(ln\ p)^{1/3}(ln\ ln\ p)^{2/3})$ для $c \approx 1{,}9$. Для задач логарифмирования в якобиане и на группе точек эллиптической кривой вычислимого простого порядка q при правильном выборе кривых сложность логарифмирования асимптотически равна $q^{1/2}$.

Протоколы Эль-Гамаля и Шнорра, а также стандарты шифрования, основанные на этих протоколах, требуют использования непредсказуемых случайных чисел, так как знание хотя бы одного такого числа позволяет решить сравнение создания подписи относительно секретного ключа. Кроме того, эти случайные числа должны быть неповторяющимися. Использование одного и того же случайного числа дважды позволяет решить систему из двух линейных сравнений и найти ключ создания подписи.

Схема цифровой подписи Рабина на самом деле не эквивалентна задаче разложения, так как решение квадратичного сравнения предполагает нахождение всех четырех корней, тогда как при создании подписи вычисляется только один корень (знание четырех корней верификатором немедленно ведет к разложению числа n). Данный протокол подвержен атаке на основе выбранных сообщений (например, извлечение корня из 4 не требует разложения n). Кроме того, возможно нарушение "правильного" порядка вычислений, при котором сначала находится корень из сообщения как целого числа, а потом вычисляются соответствующие добавочные биты. Стандартный метод противодействия таким нарушениям заключается в применении к исходному сообщению хэш-функции.

Проверка подписи в протоколе Эль-Гамаля выполняется медленнее, чем в RSA. Размер подписи в протоколе Эль-Гамаля больше, чем в RSA, однако если использовать вычеты составляющих подписи по модулю порядка группы, как в стандартах DSS и ГОСТ Р34.10-94, то подпись оказывается короче, чем в RSA. В системе DSS, в отличие от RSA, алгоритм создания подписи работает быстрее, чем алгоритм проверки подписи.

логарифмирования) или ищется подмножество линейно зависимых векторов (в задаче разложения). Оптимум алгоритмов логарифмирования и разложения достигается при равенстве сложностей первого и второго этапов. Поскольку сложности решения системы линейных сравнений и нахождения линейно зависимых векторов близки друг к другу, то и сложности логарифмирования и разложения тоже близки друг к другу.

На практике разложение больших чисел выполняется легче, чем логарифмирование. Это объясняется тем, что, с одной стороны, для демонстрации возможностей разложения выбираются числа специального вида (например, a^t+1). С другой стороны, при разложении ищутся линейно зависимые векторы над полем F_2, а при логарифмировании решается система сравнений над полем вычетов по модулю порядка группы.

Проверка подписи в протоколе Шнорра требует вычисления двух экспонент, поэтому создание подписи в этом протоколе выполняется примерно в 1,5 раза более быстрее, чем в протоколе Эль-Гамаля, так как обычно хэш-функция вычисляется быстрее экспоненты.

4.2.4.2. Специальные схемы подписи

В некоторых ситуациях могут потребоваться более слабые теоретико-сложностные положения классической схемы и/или дополнительные специальные свойства. В последние годы было разработано множество *специальных схем цифровой подписи*, удовлетворяющих требованиям различных приложений, в том числе:

- схема подписи "*вслепую*", когда претендент не знает содержания подписанного им сообщения;
- схема *"неоспоримой"* (*undeniable*) подписи, решающая проблему отказа нечестного претендента от правильной подписи;
- схема "*групповой*" подписи, в которой верификатор может проверить, что подписанное сообщение пришло от некоторой группы претендентов, но не знает, кем именно из членов группы оно подписано;
- схема *"разовой"* подписи, позволяющая использовать данный секретный (и открытый) ключ для подписи только одного сообщения;
- схема подписи, в которой проверка осуществляется с привлечением третьей стороны (*designated confirmer*) и др.

Рассмотрим подробно некоторые специальные схемы подписи.

4.2.4.2.1. Схема "неоспоримой" подписи

Эту схему подписи по сравнению с классической схемой можно назвать *диалоговой*, поскольку проверка подписи здесь осуществляется при участии претендента, подписавшего сообщение, и реализуется посредством протокола "запрос - ответ", в течение которого верификатор посылает претенденту ряд запросов. Если ответы претендента на запросы верификатора проходят проверочные тесты, то верификатор может быть уверен, что подпись правильная (вероятность ошибки верификатора обратно пропорциональна экспоненте от длины подписи). В противном случае верификатор может с большой долей уверенности сказать, является подпись неправильной или претендент дает заведомо неверные ответы (т.е. умышленно отказывается от правильной подписи и обманывает верификатора).

Д.Шаум предложил и запатентовал ряд достаточно эффективных и практичных протоколов такой "неоспоримой" (*undeniable*) подписи на основе задачи дискретного логарифмирования в мультипликативной группе простого поля. Простейшая версия протокола требует, чтобы порядок используемой группы был известен всем взаимодействующим сто-

ронам. В другом варианте этого же протокола порядок группы не известен ни одному из участников. И наконец, в наиболее общем случае порядок группы знать кому-либо из участников необязательно, однако, он остается защищенным и в том случае, если кто-нибудь его все-таки узнает.

Рассмотрим здесь общий случай протокола подписи Шаума [32]. Пусть g - образующая группы G простого порядка q (не обязательно известного). Пусть два секретных ключа создания подписи - целые числа x и y, два открытых ключа проверки подписи - g^x и g^{xy}. Подписью для сообщения M, отправляемого претендентом, является элемент s группы G, $s = M^x$.

Для проверки подписи верификатор посылает претенденту запрос, состоящий из двух элементов группы G - $s^C g^{xD}$ и $M^A g^B$, где A, B, C и D выбираются независимо одним и тем же вероятностным алгоритмом из интервала (1, q). Претендент отвечает ему тройкой s^y, $(s^C g^{xD})^y$ и $(M^A g^B)^{xy}$.

Теорема 3.2. *Даже имея компьютер бесконечной вычислительной мощности9, претендент не может с вероятностью большей q^{-1} дать верный ответ для неправильной подписи.*

Получив ответ претендента, верификатор:

1) проверяет, выполняется ли равенство $(s^y)^C (g^{xy})^D = (s^C g^{xD})^y$. Согласно теореме 3.2, он знает, что неправильный ответ будет выявлен с вероятностью $1-q^{-1}$. Таким образом, верификатор на этом этапе убеждается, что первой компонентой тройки действительно является s^y;
2) проверяет, выполняется ли равенство $(s^y)^A (g^{xy})^B = (M^A g^B)^{xy}$. Если оно выполнено, то верификатор может быть уверен, что s действительно имеет вид M^x с вероятностью $1-q^{-1}$ (также по теореме 3.2).

Если равенство на шаге 2 не выполняется, то верификатор посылает второй запрос $M^E g^F$ (где числа E и F выбираются аналогично числам A, B, C и D) и ожидает ответа $(M^E g^F)^{xy}$. Теперь по ответу претендента верификатор может определить, обманывает его претендент, или подпись действительно неправильная.

9 Бесконечная вычислительная мощность компьютера означает, что стойкость данной схемы подписи не зависит от сложности задачи логарифма.

Теорема 3.3. *Даже имея компьютер бесконечной вычислительной мощности, претендент, дав два неправильных ответа для правильной подписи, будет уличен во лжи с вероятностью большей* $1-q^{-1}$.

Таким образом, получив два разных ответа r_1 и r_2, верификатор проверяет выполнение равенства $(r_1(g^{xy})^{-B})^E = (r_2(g^{xy})^{-F})^A$. Согласно теореме 3.3, равенство означает, что подпись неправильная, неравенство - что претендент лжет.

В этом протоколе в качестве циклической группы G простого порядка q может использоваться подгруппа мультипликативной группы конечного поля или группа точек эллиптической кривой, заданной над конечным полем. В любом случае сообщение M не может быть произвольным, а должно являться элементом группы.

4.2.4.2.2. Схема подписи "вслепую"

Схема подписи "вслепую" так же, как и предыдущая, была предложена Д.Шаумом [31]. Эта схема позволяет верификатору получить сообщение, подписанное претендентом, который не знает содержания сообщения.

"Вслепую" могут быть реализованы некоторые классические схемы подписи. Вот как, например, это делается для схемы подписи RSA [70]. Пусть, как и в классической схеме RSA, d - секретный ключ создания подписи, (n, e) - открытый ключ проверки подписи. Пусть M - сообщение, для которого верификатор хочет получить подпись претендента, не раскрывая последнему числа M.

Процедура создания подписи проводится с участием двух сторон - претендента и верификатора. Верификатор генерирует случайное число r, взаимно простое с n, и отправляет претенденту $M'=r^e M \pmod n$. Содержание сообщения M "скрыто" числом r. Претендент возвращает верификатору подписанное сообщение $s'=(M')^d=(r^e M)^d \pmod n$. Так как $s'=rM^d \pmod n$, то верификатор может получить значение подписи $s=M^d$ для сообщения M, вычислив $s= s'r^{-1} \pmod n$.

Проверку подписи верификатор осуществляет, проверяя выполнение равенства $s^e=M \pmod n$.

4.2.4.2.3. Схема подписи с восстановлением сообщения

Схема подписи с восстановлением сообщения (рис. 4.30) [42] может использоваться для подписи коротких сообщений (например, ключей). Подписывание выполняется несимметричным алгоритмом без каких-либо хэш-функций. Ключ зашифрования является секретным ключом создания подписи. Ключ расшифрования является открытым ключом проверки подписи (например, в системе RSA). Исходное сообщение дополняется избыточными битами, включающими в себя признак начала сообщения и число добавленных битов, и зашифровывается на секретном ключе претендента. Восстановление исходного текста по подписи происходит в процессе проверки подписи. Верификатор расшифровывает полученное сообщение и проверяет правильность избыточных битов, выполняя действия претендента в обратном порядке.

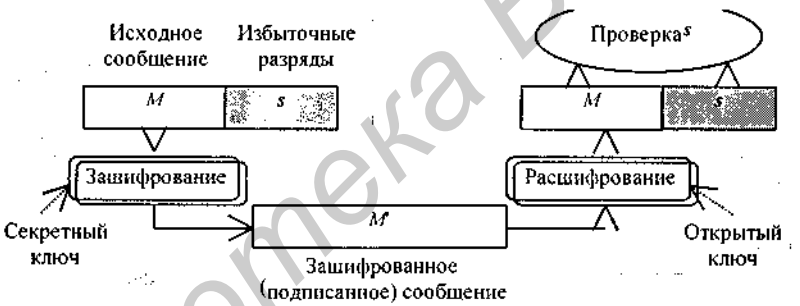


Рис. 4.30 Схема цифровой подписи с восстановлением подписанного сообщения

4.2.5. Хэш-функция

Хэш-функцией в криптографии называется преобразование информации, переводящее строку битов произвольной длины в строку битов фиксированной длины.

Хэш-функция должна обладать двумя основными свойствами:

- для данного значения $h(M)$ должно быть невозможно найти аргумент M. Такая хэш-функция называется *стойкой в смысле обращения* или *стойкой в сильном смысле*;
- для данного аргумента M должно быть невозможно найти другой аргумент M' такой, что $h(M) = h(M')$. Такая хэш-функция называется *стойкой в смысле вычисления коллизий* или *стойкой в слабом смысле*.

Хэш-функция может использоваться:

- для создания сжатого образа сообщения, применяемого в механизме цифровой подписи;
- для защиты пароля;
- для построения кода аутентификации сообщений;
- для контроля соответствия порядка вычислений, проводимых в некотором процессе10.

Отметим, что в первом и третьем случае необходимы хэш-функции, стойкие в смысле вычисления коллизий, а в остальных - стойкие в смысле обращения.

Схема вычисления значения $h(M)$ хэш-функции h для сообщения M обычно включает в себя (рис. 4.31):

- алгоритм вычисления шаговой функции хэширования g;
- итеративную процедуру вычисления хэш-функции h.

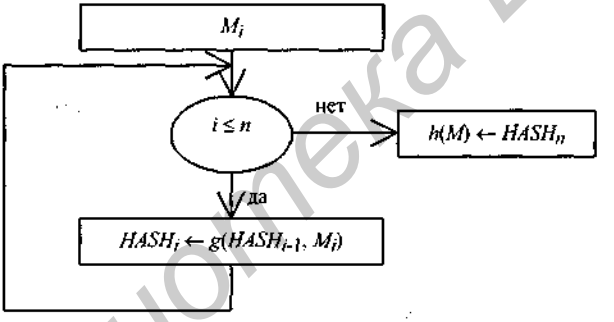


Рис. 4.31 Схема вычисления значения $h(M)$ хэш-функции h для сообщения $M = M_1M_2...M_n$

10 Пусть результатом вычислений являются две величины A и B, причем сначала должна быть найдена A, а из нее B. Однако, при нарушении порядка вычислений можно сначала найти B, а затем A. Для того, чтобы подтвердить правильность порядка вычислений, можно выбрать величину C, найти хэш-образ $h(C)$, а из него найти A. Таким образом, модификация вычислений, результатом которой является тройка (A, B, C), исключает необнаруживаемое нарушение порядка вычислений.

Например, стойкость цифровой подписи может быть снижена по сравнению с общепринятой оценкой, если сначала выбрать неприводимый полином специального вида, а затем для него найти простое число p (такой полином используется в методе дискретного логарифмирования с помощью решета числового поля). Поэтому целесообразно в состав открытого ключа ввести число C такое, что простое число p вычисляется детерминированным алгоритмом с помощью хэш-образа $h(C)$.

Для практических применений хэш-функция должна быть быстро вычислимой. Это достигается применением таких преобразований как шифрование n-битного блока текста, операции модульной арифметики, быстрое преобразование Фурье и др. Ниже будут рассмотрены некоторые наиболее употребительные хэш-функции.

4.2.5.1 MD2, MD4 и MD5

Семейство алгоритмов вычисления хэш-функции MD (Message Digest Algorithm) разработано Р.Л.Ривестом. Хэш-функции в этих алгоритмах преобразуют входное сообщение произвольной длины в сжатый 128-битный образ. Эти алгоритмы могут использоваться, например, в совокупности с несимметричным криптоалгоритмом типа RSA.

Основные операции, используемые в этих алгоритмах - сложение по модулю 2^{32}, циклический сдвиг, логические операции $\oplus$, $\wedge$, $\vee$.

Алгоритм MD2 [46], предложенный в 1989 г., ориентирован на 8-разрядный процессор и отличается от MD4 и MD5 (ориентированных на 32-разрядный процессор) способом дополнения сообщения (с применением 16-байтной контрольной суммы), значением стартового вектора хэширования и использованием 256-байтной перестановки, а обработка одного блока сообщения в нем выполняется за 18 циклов.

В алгоритме MD4 [64, 65], разработанном в 1990 г., обработка одного блока выполняется за 3 цикла, каждый из которых включает в себя 16 операций. На каждом цикле используется своя цикловая функция f_j, $1 \le j \le 3$, где $f_1 = (X \wedge Y) \vee (\bar{X} \wedge Z)$, $f_2 = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$, $f_3 = X \oplus Y \oplus Z$.

Наиболее употребительным является алгоритм вычисления хэш-функции MD5 [66], созданный в 1991 г., который и будет рассмотрен подробно.

Пусть исходное сообщение m задано t-битной строкой $m_1 m_2 ... m_t$.

Строка сообщения m дополняется до длины, сравнимой с 448 по модулю 512 (то есть длине сообщения "не хватает" ровно 64 битов, чтобы быть кратной 512), а именно: к сообщению присоединяется одна "1", а затем необходимое количество нулей. Причем дополнение строки выполняется даже в том случае, если ее длина уже сравнима с 448 по модулю 512 (в этом случае к сообщению просто добавляется 512 бит).

К полученной строке присоединяется 64-битное представление числа t. В случае $t \ge 2^{64}$, используются только младшие 64 бита числа t. Пусть дополненное сообщение $M = M_1 M_2 ... M_n$, где M_i - 16-словный блок с размером слова 32 бита, а n кратно 16.

Стартовый вектор хэширования MD_0 имеет длину 128 бит и представляет собой конкатенацию четырех 32-битных слов $md_0||md_1||md_2||md_3$, где

md_0 =01234567, md_1 = 89abcdef, md_2 = fedcba98, md_3 = 76543210.

В алгоритме MD5 используются четыре цикловые функции f_j, $1 \le j \le 4$, каждая из которых сопоставляет трем 32-битным словам X, Y и Z одно 32-битное слово:

$$f_1(X, Y, Z) = (X \wedge Y) \vee (\bar{X} \wedge Z), f_2(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \bar{Z}),$$
$$f_3(X, Y, Z) = X \oplus Y \oplus Z, f_4(X, Y, Z) = Y \oplus (X \vee \bar{Z}).$$

Все операции в этих функциях выполняются побитно, т.е. каждый выходной бит зависит только от соответствующих ему входных битов.

Вычисление хэш-функции h:

Вход: $M = M_1 M_2 ... M_n$ (n блоков по 512 бит).

Алгоритм: Для всех $i = 1, ..., n$ $MD_i \leftarrow g(MD_{i-1}, M_i)$.

Выход: $h(M) = MD_n$.

При вычислении используется накопитель E, содержащий четыре 32-битных слова A, B, C, D. Исходным заполнением накопителя E являются слова стартового вектора хэширования MD_0.

Обработка 16-словного блока M_i, $1 \le i \le n$, сообщения M осуществляется за 4 цикла, каждый из которых включает в себя 16 шагов.

На каждом шаге j-того цикла $A \leftarrow B + ((A + f_j(B, C, D) + M_i[s] + r) <<< k)$, $1 \le j \le 4$, где $M_i[s]$ - слово, выбранное из M_i, s, r и k - параметры шага; $X <<< k$ - циклический сдвиг слова X влево на k бит; "+" - операция сложения по модулю 2^{32}. Таким образом, на каждом шаге выполняется четыре операции сложения, одна операция сдвига и вычисляется значение одной цикловой функции.

Для каждой итерации MD_i представляет собой конкатенацию текущих значений четырех слов накопителя E. После обработки блока M_n итоговым сжатым образом сообщения будет 128-битная строка из четырех слов $MD_n = A||B||C||D$.

4.2.5.2. SHA-1

В стандарте США на хэш-функцию SHS (Secure Hash Standard) [58], утвержденном в 1995 г., представлена модифицированная версия SHA-1 алгоритма SHA (Secure Hash Algorithm). Этот алгоритм сопоставляет сообщению длиной до 2^{64} бит сжатый 160-битный образ, который используется, в частности, для генерации и проверки цифровой подписи в стандарте DSS.

Алгоритм SHA-1 основан на тех же принципах, что и семейство алгоритмов MD, и создан по образцу MD4. Основные операции этого алгоритма - сложение по модулю 2^{32} и по модулю 2, циклический сдвиг.

t-битная строка битов исходного сообщения $m=m_1m_2...m_t$ дополняется до длины, кратной 512. В случае, если $t \geq 2^{64}$, используются только младшие 64 бита числа t. Дополнение сообщения выполняется так же, как и для MD-функций, то есть сначала к m присоединяется одна "1", затем необходимое количество нулей и, наконец, 64-битное представление числа t - длины исходного сообщения.

Таким образом, дополненное сообщение состоит из n 512-битных блоков, т.е. имеет вид $M = M_1M_2... M_n$, где каждый блок M_i содержит шестнадцать 32-битных слов.

Стартовый вектор хэширования SHA_0 имеет длину 160 бит и представляет собой конкатенацию пяти 32-битных слов $sha_0||sha_1||sha_2||sha_3||sha_4$, где

sha_0 = 67452301, sha_1 = efcdab89, sha_2 = 98badcfe, sha_3 = 10325476,

sha_4 = c3d2e1f0.

В алгоритме SHA-1 используется последовательность функций f_0, f_1,..., f_{79}, каждая из которых сопоставляет трем 32-битным словам X, Y и Z одно 32-битное слово $f_j(X, Y, Z)$, $0 \leq j \leq 79$. Вид функций f_j приведен в таблице 4.3.

Таблица 4.3. Функции f_j в алгоритме SHA-1

$f_j(X, Y, Z) = (X \wedge Y) \vee (\overline{X} \wedge Z)$	$0 \leq j \leq 19$
$f_j(X, Y, Z) = X \oplus Y \oplus Z$	$20 \leq j \leq 39$, $60 \leq j \leq 79$
$f_j(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	$40 \leq j \leq 59$.

Вычисление хэш-функции h:

Вход: $M = M_1M_2...M_n$ (n блоков по 512 бит).

Алгоритм: Для всех $i = 1, ..., n$ $SHA_i \leftarrow g(SHA_{i-1}, M_i)$.

Выход: $h(M) = SHA_n$.

Стандарт SHS предусматривает два альтернативных способа вычисления шаговой функции хэширования g, дающих в результате один и тот же сжатый образ сообщения.

При вычислении используется последовательность 32-битных слов $\{W_i\}$, $0 \le i \le 79$, и два накопителя $\boldsymbol{F}$ и $\boldsymbol{H}$, содержащие по пять 32-битных слов. Слова в первом накопителе обозначены как A, B, C, D, E, во втором - $H_0, ..., H_4$. Исходным заполнением накопителя $\boldsymbol{H}$ являются слова стартового вектора хэширования SHA_0.

Обработка каждого из 512-битных блоков $M_1, M_2, ..., M_n$ осуществляется по следующему алгоритму:

1. Блок M_i разбивается на 16 слов $W_0, W_1, ..., W_{15}$ по 32 бита.
2. Для $j = 16, ..., 79$ $W_j \leftarrow S^1(W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16})$, где операция циклического сдвига $S^n(X)$, $0 \le n < 32$, определяется как $S^n(X) = (X <<< n) \vee (X >>> 32-n)$.
3. Содержимое второго накопителя $\boldsymbol{H}$ переписывается в первый $\boldsymbol{F}$.
4. Для $j = 0, ..., 79$

$\{TEMP \leftarrow S^5(A) + f_j(B, C, D) + E + W_j + K_j;$

$E \leftarrow D;\ D \leftarrow C;\ C \leftarrow S^{30}(B);\ B \leftarrow A;\ A \leftarrow TEMP$

$\}$

где K_j, $0 \le j \le 79$ - фиксированные константы, значения которых определяются параметром j, $TEMP$ - временная переменная, "+" - сложение по модулю 2^{32}.

5. Содержимое первого накопителя $\boldsymbol{F}$ суммируется по модулю 2^{32} с содержимым второго накопителя $\boldsymbol{H}$ и записывается в $\boldsymbol{H}$.

Для каждой итерации SHA_i представляет собой конкатенацию текущих значений пяти слов накопителя $\boldsymbol{H}$. После обработки блока M_n итоговым сжатым образом сообщения будет 160-битная строка из пяти слов $SHA_n = H_0||H_1||H_2||H_3||H_4$.

4.2.5.3. RIPEMD-160

Алгоритм RIPEMD (и его модификации RIPEMD-128, RIPEMD-160) разработан в рамках европейского проекта RIPE (Race Integrity Primitives Evaluation) и включает в себя две параллельные версии алгоритма MD4, различающиеся цикловыми константами.

Хэш-функция RIPEMD-160 [37] преобразует входное сообщение произвольной длины в сжатый 160-битный образ. Основные операции в этом алгоритме такие же, как и в алгоритме MD4, - сложение по модулю 2^{32}, циклический сдвиг, логические операции $\oplus$, $\wedge$, $\vee$.

Пусть исходное сообщение m задано t-битной строкой $m_1m_2...m_t$.

Строка битов сообщения m дополняется точно так же, как и в предыдущем случае. Таким образом, дополненное сообщение M имеет вид $M = M_1M_2...M_n$, где M_i - 32-битный блок, а n кратно 16.

Стартовый вектор хэширования $RIPE_0$ имеет длину 160 бит и представляет собой конкатенацию пяти 32-битных слов $ripe_0||ripe_1||ripe_2||ripe_3||ripe_4$, где

$$ripe_0 = 67452301,\ ripe_1 = efcdab89,\ ripe_2 = 98badcfe,\ ripe_3 = 10325476,\ ripe_4 = c3d2e1f0.$$

В алгоритме RIPEMD-160 используется последовательность цикловых функций f_0, f_1,..., f_{79}, каждая из которых сопоставляет трем 32-битным словам X, Y и Z одно 32-битное слово.

Таблица 4.4. Функции f_j в алгоритме RIPEMD-160

$f_j(X, Y, Z) = X \oplus Y \oplus Z$	$0 \le j \le 15$
$f_j(X, Y, Z) = (X \wedge Y) \vee (\overline{X} \wedge Z)$	$16 \le j \le 31$
$f_j(X, Y, Z) = (X \vee \overline{Y}) \oplus Z$	$32 \le j \le 47$
$f_j(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \overline{Z})$	$48 \le j \le 63$
$f_j(X, Y, Z) = X \oplus (Y \vee \overline{Z})$.	$64 \le j \le 79$

Вычисление хэш-функции h:

Вход: $M = M_1M_2...M_n$ (n блоков по 512 бит).

Алгоритм: Для всех $i = 1, ..., n$ $RIPE_i \leftarrow g(RIPE_{i-1}, M_i)$.

Выход: $h(M) = RIPE_n$.

При вычислении используются три накопителя ***F***, ***F'*** и ***H*** содержащие по пять 32-битных слов. Слова в первом накопителе обозначены как A, B, C, D, E, во втором - A', B', C', D', E', в третьем - H_0, ..., H_4. Исходным заполнением накопителей F и F' являются слова стартового вектора хэширования $RIPE_0$.

Обработка каждого из 512-битных блоков M_1, M_2, ..., M_n осуществляется по следующему алгоритму:

1. Блок M_i разбивается на 16 слов по 32 бита.
2. Для $j = 0, ..., 79$

$\{T \leftarrow E+((A+f_j(B, C, D)+ M_i[s]+ r)<<<k);$
$A \leftarrow E;\ E \leftarrow D;\ D \leftarrow C<<<10;\ C \leftarrow B;\ B \leftarrow T;$
$T \leftarrow E'+((A'+f_{79-j}(B', C', D')+ M_i[s']+ r')<<<k');$
$A' \leftarrow E';\ E' \leftarrow D';\ D' \leftarrow C'<<<10;\ C' \leftarrow B';\ B' \leftarrow T$
$\},$

где $M_i[s]$, $M_i[s']$ - слова, выбранные из M_i, s, r, k, s', r' и k' - параметры шага; $X<<<k$ - циклический сдвиг слова X влево на k бит; "+" - операция сложения по модулю 2^{32}.

3. $T \leftarrow H_1+C+D';\ H_1 \leftarrow H_2+D+ E';\ H_2 \leftarrow H_3+E+ A';$
$H_3 \leftarrow H_4+A+ B';\ H_4 \leftarrow H_0+B+ C';\ H_0 \leftarrow T.$

Для каждой итерации $RIPE_i$ представляет собой конкатенацию текущих значений пяти слов накопителя ***H***. После обработки блока M_n итоговым сжатым образом сообщения будет 160-битная строка из пяти слов $RIPE_n = H_0||H_1||H_2||H_3||H_4$.

4.2.5.4. Хэш-функция на основе быстрого преобразования Фурье

Хэш-функция на основе быстрого преобразования Фурье, предложенная К.Шнорром в 1991 г. [73] и усовершенствованная им же год спустя [75], сопоставляет сообщению произвольной длины 128-битный сжатый образ.

Основные операции в этом алгоритме - дискретное преобразование Фурье и рекурсия полиномов над конечным полем.

Пусть сообщение m задано t-битной строкой $m_1m_2...m_t$.

Строка сообщения m дополняется до длины, кратной 128, а именно: к сообщению присоединяется одна "1", затем необходимое количество

нулей и двоичное представление числа t. Пусть дополненное сообщение $M = M_1M_2...M_n$ состоит из n блоков, каждый из которых имеет длину 128 бит, т.е. состоит из восьми 16-битных слов.

128-битный стартовый вектор хэширования FFT_0 представляет собой конкатенацию восьми 16-битных слов $ffi_0||ffi_1||$ ... $||ffi_7$, где

$$ffi_0 = 0123, ffi_1 = 4567, ffi_2 = 89ab, ffi_3 = cdef,$$
$$ffi_4 = fedc, ffi_5 = ba98, ffi_6 = 7654, ffi_7 = 3210.$$

Вычисление хэш-функции h:

Вход: $M = M_1M_2...M_n$ (n блоков по 128 бит).

Алгоритм: Для всех $i = 0, ..., n$ $FFT_i \leftarrow g(FFT_{i-1}, M_i)$.

Выход: $h(M) = FFT_n$.

При вычислении шаговой функции хэширования g используется вектор 16-битных слов $\{e_j\}$, $0 \le j \le 15$, и два накопителя $E_{0/7}$ и $E_{8/15}$, содержащие по восемь 16-битных слов. Слова в первом накопителе обозначены как e_0, ..., e_7, во втором - e_8, ..., e_{15}. Исходным заполнением накопителя $E_{0/7}$ являются вычеты по модулю 2^{16} слов стартового вектора хэширования FFT_0.

Обработка каждого из 128-битных блоков M_1, M_2, ..., M_n осуществляется по следующему алгоритму:

1. Блок M_i разбивается на 8 слов W_0, ..., W_7 по 16 бит и записывается во второй накопитель $E_{8/15}$, т.е. для $j = 0, ..., 7$ $e_{j+8} \leftarrow W_j$.
2. Для всех $j = 0, ..., 15$ $e_j \leftarrow e_j + e'_{j-1}e'_{j-2} + e_{j-3} + 2^j \pmod p$, где $p=2^{16}+1$; $e' = e$, если $e \neq 0$ и $e' = 1$, если $e=0$ (нижние индексы j, j-1, j-2, j-3 вычисляются по mod 16).
3. $(e_0, e_2, ..., e_{14}) \leftarrow \mathbf{FT}_8(e_0, e_2, ..., e_{14})$;

$(e_1, e_3, ..., e_{15}) \leftarrow \mathbf{FT}_8(e_1, e_3, ..., e_{15})$,

где вектор $(b_0, ..., b_7) = \mathbf{FT}_8(a_0, ..., a_7)$ - результат дискретного преобразования Фурье над полем Z/pZ вектора $(a_0, ..., a_7)$,

$$b_i = \sum_{j=0}^{7} 2^{4ij} a_j \pmod p, \quad 0 \le i \le 7.$$

4. Для всех $j = 0, ..., 15$ $e_j \leftarrow e_j + e'_{j-1}e'_{j-2} + e_{j-3} + 2^j \pmod p$.

5. Содержимое второго накопителя $E_{8/15}$ приводится по модулю 2^{16} и переписывается в первый накопитель $E_{0/7}$, т.е. для $j = 0, ..., 7$ $e_j \leftarrow e_{j+8} \pmod{2^{16}}$.

Для каждой итерации FFT_i представляет собой конкатенацию текущих значений восьми слов накопителя $E_{0/7}$. После обработки блока M_n итоговым сжатым образом сообщения будет 128-битная строка из восьми слов $FFT_n = e_0||e_1|| ... ||e_7$.

Следует отметить, что шаги 2 и 4 в алгоритме вычисления шаговой функции хэширования g можно заменить на:

$$e_j \leftarrow e_j + f(e_0, ..., e_{j-1}, e_{j+1}, ..., e_{15}, j) \pmod{p}, 0 \le j \le 15,$$

где f - произвольная функция, выбор которой может определяться требованиями к обеспечению безопасности конкретного приложения.

4.2.5.5. ГОСТ Р34.11-94

Отечественный алгоритм вычисления хэш-функции ГОСТ Р34.11-94 [17] сопоставляет сообщению произвольной длины 256-битный сжатый образ, который используется, в частности, для генерации и проверки цифровой подписи в стандарте ГОСТ Р34.10-94.

Пусть исходное сообщение m задано t-битной строкой $m_1m_2...m_t$.

Строка сообщения m дополняется необходимым количеством нулей до длины, кратной 256. Пусть дополненное сообщение $M = M_1M_2...M_n$ состоит из n блоков, каждый из которых имеет длину 256 бит.

Параметрами хэш-функции ГОСТ Р34.11-94 являются 256-битный стартовый вектор хэширования $GOST_0 = gost_0||gost_1||gost_2||gost_3||$, на выбор которого ограничения не накладываются, и блок подстановки, используемый в алгоритме шифрования ГОСТ 28147-89.

Вычисление хэш-функции h:

Вход: $M = M_1M_2...M_n$ (n блоков по 256 бит).

Алгоритм: Для всех $i = 0, ..., n$ $GOST_i \leftarrow g(GOST_{i-1}, M_i)$.

Выход: $h(M) = GOST_n$.

При вычислении шаговой функции хэширования используется накопитель H, содержащий четыре 64-битных слова $H_0, ..., H_3$. Исходным

заполнением накопителя $\boldsymbol{H}$ являются слова стартового вектора хэширования $GOST_0$.

Алгоритм вычисления шаговой функции хэширования g для 256-битного блока M_i включает в себя три этапа:

- генерацию четырех 256-битных ключей $K^{(i)}{}_1$, $K^{(i)}{}_2$, $K^{(i)}{}_3$, $K^{(i)}{}_4$;
- зашифрование заполнения накопителя $\boldsymbol{H}$ на этих ключах с помощью алгоритма ГОСТ 28147-89;
- перемешивание результата зашифрования.

На первом этапе блок M_i рассматривается как вектор $\mathbf{m}$ над полем F_2. С помощью четырех различных невырожденных аффинных над F_2 преобразований из этого вектора вырабатываются четыре 256-битных ключа $K^{(i)}{}_1$, $K^{(i)}{}_2$, $K^{(i)}{}_3$, $K^{(i)}{}_4$:

$$K^{(i)}{}_j = \boldsymbol{A}_j\,\mathbf{m} + \mathbf{c}_j,\ j = 1, \ldots, 4,$$

где $\boldsymbol{A}_j$ - блочная матрица, $\mathbf{c}_j$ - вектор сдвига.

На втором этапе каждое из четырех 64-битных слов H_0, H_1, H_2, H_3 накопителя $\boldsymbol{H}$ зашифровывается в режиме простой замены на соответствующем ключе $K^{(i)}{}_j$. Результатом зашифрования является вектор $\mathbf{s} = (s_1, s_2, s_3, s_4)$, где:

$$s_j = E_{K^{(i)}_j}(H_j),\ 1 \le j \le 4.$$

На третьем этапе выполняется перемешивание, представляющее собой композицию невырожденных линейных над F_2 преобразований, применяемую к векторам $\mathbf{h}$, $\mathbf{m}$, $\mathbf{s}$, где $\mathbf{h}$ - представленное в виде вектора содержимое накопителя $\boldsymbol{H}$. Результат перемешивания заносится в накопитель $\boldsymbol{H}$ и является текущим значением $GOST_i$ шаговой функции хэширования g для 256-битного блока M_i.

После обработки блока M_n итоговым сжатым образом сообщения будет 256-битная строка из четырех слов $GOST_n = H_1 \| H_2 \| H_3 \| H_4$.

4.2.5.6. Сравнительный анализ представленных хэш-функций

Сложность нахождения методом Полларда [62] коллизии хэш-функции с длиной значения l бит не может быть больше, чем $O(l*2^{l/2})$. Обычно обращение хэш-функции является более сложной задачей, чем нахождение коллизий, поэтому целесообразно сравнивать хэш-функции

по сложности вычисления коллизий. Скорость алгоритма вычисления хэш-функции существенным образом зависит от используемого вычислителя, поэтому по скорости алгоритмы можно сравнивать лишь при фиксированной вычислительной модели.

Таблица 4.5. Скорость некоторых хэш-алгоритмов, основанных на MD4. Вычисления проводились на Pentium 90МГц [37].

Алгоритм	Скорость	(Мбит/с)
	Ассемблер	Си
MD4	165.7	81.4
MD5	113.5	59.7
SHA-1	46.5	21.2
RIPEMD-160	39.8	19.3

Алгоритм MD2 является самым медленным из семейства алгоритмов MD. Кроме того, если опустить вычисление контрольной суммы, то для этой хэш-функции можно найти коллизии [69].

В 1995 г. H.Dobbertin [35] показал, что для алгоритма MD4 коллизии могут быть найдены за 1 минуту на обычном персональном компьютере.

Алгоритм MD5 является обобщением алгоритма MD4, не требует больших таблиц подстановок и более стоек к атакам, но в то же время приблизительно на 33% медленнее, чем MD4. B. den Boer и A.Bosselaers [28] нашли для MD5 псевдоколлизии[11], а в 1996 г. H.Dobbertin [36] вычислил для MD5 коллизии при стартовом векторе хэширования MD_0=12ac23753b3410425f62b97c4ba763ed за 10 часов на Pentium PC. Кроме того, согласно исследованиям [23] этот алгоритм должен быть уязвим по отношению к дифференциальному методу криптоанализа, использующему разности по модулю 2^{32}.

Хэш-функция стандарта SHS по своей структуре аналогична MD4 и MD5. Она на 25% медленнее, чем MD5, но может быть более безопасной, поскольку производимые ею сжатые образы сообщений на 25% длиннее, чем для функций MD [70].

RIPEMD-160 примерно на 15% медленнее, чем SHA-1, и в 4 раза медленнее, чем MD4 [37]. Однако его "двойная" структура, специфика построения операторов циклического сдвига, параметры, выбранные с учетом недостатков алгоритмов MD4 и MD5, и увеличенный размер хэш-

---

11 Если *коллизия* - это равенство хэш-образов двух разных сообщений M и M', вычисляемых для одного и того же стартового вектора хэширования IV, т.е. $h(IV, M)= h(IV, M')$, то под *псевдоколлизией* здесь понимается равенство хэш-образов двух (не обязательно разных) сообщений M и M', вычисляемых для разных стартовых векторов хэширования IV и IV', т.е. $h(IV, M)= h(IV', M')$ [24].

образа дают основание считать этот алгоритм более стойким, чем другие MD-функции.

Хэш-функция FFT является альтернативой алгоритмам семейства MD и допускает простую программную реализацию операций сложения и умножения как по модулю 2^{16}, так и по модулю 2^8. Применяемое в алгоритме дискретное преобразование Фурье обладает хорошими свойствами перемешивания и рассеивания.

Следует отметить, что сложение по модулю 2^{16}, используемое в алгоритме FFT, очень похоже на сложение по модулю $p=2^{16}+1$. А так как *сложение и умножение полиномов по модулю p* является линейной операцией над полем F_p, целесообразно исследовать стойкость этой функции по отношению к известным методам анализа (например, дифференциальному) для соответствующей метрики.

Нахождение коллизий для хэш-функции FFT требует около 2^{24} вычислений основной функции, которые могут быть проведены на персональном компьютере за несколько часов. Обращение хэш-функции FFT требует 2^{48} вычислений основной функции. Были предложены некоторые простые улучшения, устраняющие слабости функции FFT [76].

Допустимость произвольного стартового вектора в ГОСТ Р34.11-94 позволяет разработчику системы совместить выбор этого вектора с вычислением коллизий [14]. В этом случае стойкость хэш-функции в части вычисления коллизий уменьшается примерно в 2^{32} раз по сравнению с оценкой сложности алгоритма Полларда, что обусловлено свойствами перемешивающего преобразования [19].

4.2.6. Заключение к разделу 4.2

Функциональные характеристики методов идентификации, аутентификации и контроля целостности определяются в основном характеристиками используемых в них криптографических средств шифрования, цифровой подписи, хэш-функций. Если в 70-х годах единственным широко используемым открытым алгоритмом шифрования был DES, то в последние годы в открытой печати появилось множество различных алгоритмов шифрования, цифровой подписи и хэш-функции.

Наметилась тенденция разработки и использования программно ориентированных криптографических алгоритмов, причем во многих международных стандартах (ISO/IEC 9796 [42], ISO/IEC 9797 [43], X.509 [30] и др.), конкретные алгоритмы специально не оговариваются, что дает разработчику системы обеспечения информационной безопасности воз-

можность выбора существующих или вновь появляющихся алгоритмов, в том числе методов управления ключами.

Конечно, прежде всего следует исходить из имеющихся в наличии ресурсов. Быстродействие компьютера, разрядность процессора, объем обрабатываемых данных, время обработки, требования к безопасности, — все это следует учитывать при построении криптографической системы защиты.

Выбирая тот или иной протокол аутентификации, необходимо определить, какая именно аутентификация требуется - односторонняя или взаимная; нужно ли использовать доверенное третье лицо и если да, то какая из сторон - претендент или верификатор - будет с ним взаимодействовать. Протоколы бездиалоговой аутентификации обычно осуществляют еще и контроль целостности данных.

Выбрав протокол аутентификации, необходимо определить его параметры, в том числе:

- *Тип алгоритма*: В зависимости от целей обеспечения безопасности можно использовать симметричные или несимметричные (с открытым ключом) алгоритмы. Алгоритмы с открытым ключом требуют более трудоемких вычислений, чем симметричные алгоритмы сопоставимой стойкости, но, с другой стороны, позволяют обеспечить защиту информации в условиях взаимного недоверия.
- *Конкретный алгоритм*: Помимо приведенных в данной главе, существует еще множество различных алгоритмов шифрования и вычисления хэш-функции, а также протоколов цифровой подписи. Выбор того или иного алгоритма может определяться его криптографической стойкостью (оценка стойкости предполагает изученность криптографических характеристик алгоритма доверенными криптографами), эффективностью и стоимостью реализации.
- *Режим работы*: Многие алгоритмы шифрования допускают несколько режимов работы. На базе основных режимов шифрования создаются новые, позволяющие, например, обеспечивать одновременно аутентификацию источника данных и целостность передаваемых сообщений.
- *Процедуры управления ключами*: Необходимо определить процедуры управления ключами, включая генерацию, распределение, замену и уничтожение ключей. Генерация секретных ключей должна осуществляться с помощью специальных датчиков случайных чисел. Следует отметить, что хорошие статистические свойства датчика не всегда гарантируют случайность формируемой последовательности [21]. Генерация ключей, общих для всех пользователей (например, числа p в схеме подписи или блока подстановки в симметричном криптоалгоритме), позволяет разработчику заложить нераспознаваемую "мину"

(непреднамеренно или умышленно), позволяющую снизить стойкость алгоритма по сравнению с расчетной оценкой. При распределении ключей необходимо обеспечить их аутентичность и секретность, поскольку нарушение аутентичности ключа может привести к нарушению секретности всей системы. При утрате секретного ключа необходимо обеспечить запрет на его использование в системе, при этом замене подлежат как сам секретный ключ, так и соответствующие ему открытые ключи. Уничтожение ключей должно обеспечиваться надежно и исключать их восстановление.

Совместимость используемых алгоритмов: В бездиалоговых протоколах цифровой подписи без восстановления сообщения хэш-функцию следует подбирать так, чтобы размер вырабатываемого ею хэш-образа сообщения согласовывался с размерами соответствующих значений в протоколе подписи, а стойкость хэш-функции не ухудшала стойкость системы подписи. Примерами таких пар хэш-функция/цифровая подпись могут служить MD5/RSA, SHA/DSA, ГОСТ Р34.11-94/ ГОСТ Р34.10-94.

Некоторые факторы, такие как тип алгоритма и процедуры управления ключами, как правило, определяют область применения алгоритма и требуют тщательного предварительного изучения. Другие характеристики могут меняться в соответствии с требованиями, предъявляемыми тем или иным приложением. В протоколах, допускающих различные криптографические алгоритмы, обычно указывается идентификатор конкретного используемого алгоритма и значения параметров, необходимые для этого алгоритма, например, индикаторы режима работы или значение синхропосылки.

Стойкость ключей, используемых для шифрования или подписи, существенно зависит от программной реализации алгоритма. Следует исключить зависимость длительности вычислений от вида ключа, вспомогательного случайного числа (в алгоритмах цифровой подписи) и открытого текста (в алгоритмах шифрования), которая может быть использована для раскрытия ключа (*timing attack*).

Оценки стойкости многих криптографических алгоритмов быстро падают, что обусловлено развитием и взаимным влиянием математики, криптоанализа и вычислительной техники. Появление дифференциального и линейного методов криптоанализа привело к пересмотру защитных свойств целого ряда хэш-функций и симметричных криптоалгоритмов. Сложность задач разложения и дискретного логарифмирования изменилась с возникновением метода решета числового поля. В то же время для алгоритмов на алгебраических кривых стойкость практически не меняется (накладываются лишь незначительные ограничения на принципы выбора

открытых ключей), что позволяет считать группы на этих кривых перспективными с точки зрения построения криптографических алгоритмов.

Указанные обстоятельства повышают роль криптоанализа в обосновании и прогнозировании безопасности выбираемых способов обеспечения идентификации, аутентификации и контроля целостности.

Литература к главе 4

1. M.Harrison, W.Ruzzo, J.Uhlman "Protection in operating systems", Communications of the ACM, 1976.
2. M.Harrison, W.Ruzzo "Monotonic protection systems", Foundation of secure computation, 1978.
3. Ravi S. Sandhu "The Typed Access Matrix Model" Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, May 4-6, 1992, pages 122-136.
4. Leonard J. LaPadula and D. Elliott Bell "Secure Computer Systems: A Mathematical Model", MITRE Corporation Technical Report 2547, Volume II, 31 May 1973.
5. Ciaran Bryce "Lattice-Based Enforcement of Access Control Policies", Arbeitspapiere der GMD (Research Report), Nummar 1020, August 1996.
6. John McLean "Security models", Encyclopedia of software engineering, 1994
7. John McLean "The Specification and Modeling of Computer Security", Computer, 23(1):9-16, January 1990.
8. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman "Role-Based Access Control Models", IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.
9. David Ferraiolo and Richard Kuhn. "Role-based access controls." In 15th NIST-NCSC National Computer Security Conference, pages 554-563, Baltimore, MD, October 13-16 1992.
10. D. Ferraiolo, J. Cugini, and D.R. Kuhn. "Role based access control: Features and motivations." In Annual Computer Security Applications Conference. IEEE Computer Society Press, 1995.
11. Lee Badger, Daniel F. Sterne, David L. Sherman, Kenneth M. Walker, Sheila A. Haghighat "Domain and Type Enforcement UNIX Prototype", USENIX Security Symposium, 1995.
12. L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, S. A. Haghighat, "Practical Domain and Type Enforcement for UNIX," 1995 IEEE Symposium on Security and Privacy, Oakland CA, May 1995.
13. А.Ахо, Дж.Хопкрофт, Дж.Ульман. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
14. А.П.Баранов и др. Математические основы информационной безопасности. Орел: ВИПС, 1997.
15. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. М.: Госстандарт СССР, 1989.
16. ГОСТ 34.10-94. Информационная технология. Криптографическая защита информации. Система электронной цифровой подписи на базе асимметричного криптографического алгоритма. М.: Госстандарт России, 1994.

17. ГОСТ 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования. М.: Госстандарт России, 1994.
18. Дж.Х.Мур. Несостоятельность протоколов криптосистем. ТИИЭР, т. 76, № 5, 1988. С. 94-104.
19. Ростовцев А.Г., Дук Т.Г. Ускоренный метод вычисления коллизий в стандарте на хэш-функцию. Методы и технические средства обеспечения безопасности информации. Тезисы докладов. СПб., 1996. С. 204-205.
20. Смид М.Э., Бранстед Д.К. Стандарт шифрования данных: Прошлое и будущее. ТИИЭР, т. 76, № 5, 1988. С. 43-54.
21. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М.: Наука, 1987.
22. Шеннон К.Э. Теория связи в секретных системах. В кн.: Шеннон К.Э. Работы по теории информации и кибернетике. М.: ИЛ, 1963. С. 333-402.
23. T.A.Berson. Differential Cryptanalysis Mod 2^{32} with Applications to MD5. *Advances in Cryptology - EUROCRYPT '92. LNCS*, v. 658, Springer-Verlag, 1993, pp. 67-76.
24. E. Biham, A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Advances in Cryptology - CRYPTO '90. LNCS*, v. 537, Springer-Verlag, 1991, pp. 2-21.
25. E. Biham, A. Shamir. Differential Cryptanalysis of FEAL and N-Hash. *Advances in Cryptology - EUROCRYPT '91. LNCS*, v. 547, Springer-Verlag, 1991, pp. 1-16.
26. E. Biham. A. Shamir. Differential Cryptanalysis of the Full 16-round DES. *Advances in Cryptology - CRYPTO '92. LNCS*, v. 740, Springer-Verlag, 1993, pp. 487-496.
27. A. Biryukov. Cryptanalysis of RC5. DREI '97 Workshop on Cryptography and Network Security (Abstracts). July 28 - Aug 15, 1997.
28. B. den Boer, A. Bosselaers. Collisions for the Compression Function of MD5. *Advances in Cryptology - EUROCRYPT '93. LNCS*, v. 765, Springer-Verlag, 1994, pp. 293-304.
29. G.Brassard. Modern Cryptology. *LNCS*, v. 325. Springer-Verlag. 1988.
30. CCITT (Consultative Committee on International Telegraphy and Telephony). Recommendation X.509: The Directory - Authentication Framework, 1988.
31. D.Chaum. Blind Signatures for Untraceable Payments. *Advances in Cryptology: Proceedings of Crypto'82*, Plenum Press, 1983, pp. 199-203.
32. D.Chaum, H. van Antwerpen. Undeniable Signatures. *Advances in Cryptology - CRYPTO'89. LNCS*, v. 435, Springer-Verlag, 1990, pp. 205-212.
33. J. Daemen, R. Govaerts, J. Vandewalle. Weak Keys for IDEA. *Advances in Cryptology - CRYPTO '93. LNCS*, v. 773, Springer-Verlag, 1994, pp. 224-231.
34. W. Diffie, P.C. van Oorschot, M.J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, v. 2, 1992, pp. 107-125.
35. H.Dobbertin. Alf Swindles Ann. *CryptoBytes*, v. 1(3), 1995, p. 5.
36. H.Dobbertin. Cryptanalysis of MD5 Compress. http:/www.ioc.ee/home/helger/crypto/ htmls/hash.html.
37. H.Dobbertin, A.Bosselaers, B.Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. *Fast Software Encryption*, Cambridge Workshop, *LNCS*, v. 1039, Springer-Verlag, 1996, pp. 71-82.

38. T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory, IT-31*, 1985, pp. 469-472.
39. U.Feige, A.Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. *Advances in Cryptology - CRYPTO'89. LNCS*, v. 435, Springer-Verlag, 1990, pp. 491-513.
40. D.M.Gordon. Discrete Logarithms in GF(p) Using the Number Field Sieve. *SIAM Journal of Computing*, v. 6, ' 1, Feb 1993, pp. 124-138.
41. H.M.Heys, S.E.Tavares. Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis. *Journal of Cryptology*, v. 9, 1996, pp. 1-19.
42. ISO/IEC 9796. Information Technology - Security Techniques - Digital Signature Scheme Giving Message Recovery, 1991.
43. ISO/IEC 9797:1994. Information Technology - Security Techniques - Data Integrity Mechanism Using a Cryptographic Check Function Employing a Block Cipher Algorithm (http://www.ewos.be/sec.gdim.htm).
44. ISO/IEC 9798-2. Information Technology - Security Techniques - Entity Authentication - Part 2: Mechanisms Using Symmetric Encipherment Algorithms. 1994.
45. ISO/IEC 9798-3. Information Technology - Security Techniques - Entity Authentication Mechanisms - Part 3: Entity Authentication Using a Public Key Algorithm. 1993.
46. B.S. Kaliski Jr. RFC 1319: The MD2 Message-Digest Algorithm. RSA Laboratories, April 1992.
47. B.S.Kaliski, Y.L.Yin, On Differential and Linear Cryptanalysis of the RC5 Encryption Algorithm, *Advances in Cryptology - CRYPTO '95. LNCS*, v. 963, Springer-Verlag, 1995, pp. 171-184.
48. J.Kelsey, B.Schneier, D.Wagner. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. *Advances in Cryptology - CRYPTO '96, LNCS*, v. 1109, Springer-Verlag, 1996, pp. 237-251.
49. N.Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, v. 48, 1987, pp. 203-209.
50. N.Koblitz. Hyperelliptic Cryptosystems. *Journal of Cryptology*, v. 1, 1989, pp. 139-150.
51. N.Koblitz. A Course in Number Theory and Cryptography. Springer-Verlag, 1994.
52. J.T.Kohl. The Use of Encryption in Kerberos for Network Authentication. *Advances in Cryptology - CRYPTO'89. LNCS*, v. 435, Springer-Verlag, 1990, pp. 29-35.
53. J. Kohl, B. Neuman. RFC 1510: The Kerberos Network Authentication Service. *Network Working Group*, 1993.
54. A.Konheim. Cryptography. A Primer. Wiley, 1981.
55. X. Lai, J.L. Massey. A Proposal for a New Block Encryption Standard. *Advances in Cryptology - EUROCRYPT '90. LNCS*, v. 473, Springer-Verlag, 1991, pp. 389-404.
56. X. Lai, J.L. Massey, S. Murphy. Markov Ciphers and Differential Cryptanalysis. *Advances in Cryptology - EUROCRYPT '91. LNCS*, v. 576, Springer-Verlag, 1992, pp. 17-38.
57. National Institute of Standards and Technology (NIST). FIPS Publication 46-2: Data Encryption Standard (DES). Dec 1993.

58. National Institute of Standards and Technology (NIST). FIPS Publication 180: Secure Hash Standard (SHS). May 1993.
59. National Institute of Standards and Technology (NIST). FIPS Publication 181: DES Modes of Operation. Dec 1980.
60. National Institute of Standards and Technology (NIST). FIPS Publication 186: Digital Signature Standard (DSS). May 1994.
61. K.Ohta, K.Aoki. Linear Cryptanalysis of the Fast Data Encipherment Algorithm. *Advances in Cryptology - CRYPTO '94. LNCS*, v. 839, Springer-Verlag, 1994, pp. 12-16.
62. J.M.Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, v. 32, ' 143, 1978, pp. 918-924.
63. M.O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, MIT, 1979.
64. R.L.Rivest. The MD4 Message Digest Algorithm. *Advances in Cryptology - CRYPTO '90. LNCS*, v. 537, Springer-Verlag, 1991, pp. 303-311.
65. R.L.Rivest. RFC 1320: The MD4 Message-Digest Algorithm. Network Working Group, April 1992.
66. R.L.Rivest. RFC 1321: The MD5 Message-Digest Algorithm. Internet Activities Board, April 1992.
67. R.L.Rivest. The RC5 Encryption Algorithm. *Proceedings of the Second International Workshop on Fast Software Encryption*, Springer-Verlag, 1995, pp. 86-96.
68. R.L.Rivest, A.Shamir, L.Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. CACM, v. 21, ' 2, 1978, pp. 120-126.
69. N. Rogier, P. Chauvaud. The Compression Function of MD2 is not Collision Free. *Selected Areas in Cryptography '95*, Ottawa, Canada, May 18-19, 1995.
70. RSA Laboratories. Answers to Frequently Asked Questions About Today's Cryptography. Revision 3.0, RSA Data Security Inc., 1995 (http://www.rsa.com/rsalabs/newfaq/).
71. B.Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). *Fast Software Encryption, Cambridge Security Workshop Proceedings* (Dec 1993), *LNCS*, v. 809, Springer-Verlag, 1994, pp. 191-204.
72. C.P.Schnorr. Efficient Identification and Signatures for Smart Cards. *Advances in Cryptology - CRYPTO '89. LNCS*, v. 435, Springer-Verlag, 1990, pp. 239-251.
73. C.P.Schnorr. FFT-Hashing: An Efficient Cryptographic Hash Function. *Advances in Cryptology - CRYPTO '91*, Rump Session. *LNCS*, v. 576. Springer-Verlag, 1992.
74. C.P.Schnorr. FFT-Hash II, Efficient Cryptographic Hashing. *Advances in Cryptology - EUROCRYPT '92. LNCS*, v. 658, Springer-Verlag, 1993, pp. 41-66.
75. A.Shimizu, S.Miyaguchi. Fast Data Encipherment Algorithm FEAL. *Advances in Cryptology - EUROCRYPT '87. LNCS*, v. 304, Springer-Verlag, 1988, pp. 267-280.
76. S.Vaudenay. FFT-Hash-II is not Yet Collision-Free. *Advances in Cryptology - CRYPTO '92, LNCS*, v. 740, Springer-Verlag, 1993, pp. 587-593.
77. S.Vaudenay. On the Weak Keys of Blowfish. *Fast Software Encryption, Cambridge, LNCS*, v. 1039, Springer-Verlag, 1996, pp. 27-32. (ftp://ftp.ens.fr/pub/reports/liens/liens-95-27.A4.ps.Z)

Долог путь от замысла к свершению.
Жан Батист Мольер

Глава 5. Архитектура защищенных операционных систем

Построенная классификация изъянов защиты (гл. 3) по размещению в вычислительной системе показывает, что основные недостатки защиты вычислительных систем сосредоточены в ОС. Поэтому в данной главе рассмотрим основные принципы построения современных ОС. Основой для данного обзора является анализ принципов построения операционных систем (ОС), сертифицированных в соответствии с требованиями Министерства Обороны США («Оранжевая книга» [2]). В приложении 5 приводится сводная таблица характеристик и параметров ОС, прошедших оценку в соответствии с этими требованиями.

5.1. Обзор архитектур сертифицированных защищенных систем

Прежде чем перейти к рассмотрению общих принципов, используемых при построении защищенных ОС, кратко проанализируем данные, приведенные в Приложении 5. Общее количество созданных и сертифицированных в промежутке между 1985 и 1997 годами можно охарактеризовать следующими данными:

1. МО США сертифицировало одну систему класса C1, 23 системы класса C2, 20 систем класса B1, 7 систем класса B2, 3 системы класса B3 и 2 системы класса A1.
2. Всего было сертифицировано 33 операционные системы, 6 подсистем безопасности, 9 сетевых компонентов и 8 систем управления базами данных.

На основании анализа данных Приложения 5 можно отметить четкие тенденции влияния новых информационных технологий, появившихся в 90-е годы.

В 80-е годы в числе сертифицированных систем доминировали системы, оцененные по классу C2. Исключением являются системы System V/MLS (класс B1), Multics (класс B2) и Secure Communication Processor (A1). В 90-х годах ситуация кардинально меняется, и доля систем, рассчитанных на классы B1-B3, начинает расти и в настоящий момент превалирует над системами класса C, которые прочно обосновались в нише специализированных систем (например систем обработки транзакций). Эти данные свидетельствуют о том, что в настоящее время проектируются

принципиально новые системы и разрабатываются защищенные версии старых с расчетом на реализацию более высокого уровня защиты, чем в предыдущее десятилетие, что наглядно свидетельствует об осознании актуальности проблемы безопасности информационных систем.

Если в восьмидесятые годы защищенные системы разрабатывались в основном для мэйнфреймов, то в 90-е годы наблюдается тенденция к проектированию защищенных систем для персональных компьютеров типа IBM PC (XTS-300, Trusted Xenix, Trusted Mach, Windows NT), что объясняется популяризацией компьютерной обработки данных.

Развитие Internet также оказало существенное влияние на разработку защищенных систем. Развитие сетевых технологий в 90-е годы привело к появлению большого числа сетевых компонентов. Системы, прошедшие сертификацию без учета требований к сетевому программному обеспечению, в настоящее время часто используются в сетевом окружении и даже подключаются к Internet. Однако, это приводит к появлению изъянов и уязвимостей, не обнаруженных при сертификации защищенных вычислительных систем. Классическим примером, иллюстрирующим данную проблему, являются многочисленные атаки на популярную операционную систему Windows NT через сетевые компоненты этой системы.

В 90-е годы при проектировании защищенных систем происходит переход от модульной архитектуры к микроядерной (данная архитектура будет рассмотрена в следующем разделе). Использование микроядерной архитектуры не только облегчает перенос ОС на различные аппаратные платформы, но и позволяет проводить доказательство безопасности системы с применением формальных механизмов.

5.1.1. Общие принципы построения защищенных систем

Опыт построения защищенных систем обработки информации позволяет выделить две возможных технологии, на основе которых могут быть спроектированы защищенные системы, претендующие на сертификацию в соответствии с требованиями не ниже класса С:

1. Проектирование *защищенной системы «с нуля»*. В этом случае безопасность является одной из главных целей разработчиков системы. Данный подход характеризуется тем, что система разрабатывается как единое целое, начиная (хотя это необязательно) от аппаратной части и операционной системы и до приложений пользователя. В качестве примера систем, построенных по данному принципу, можно привести: *A Series MCP/AS с InfoGuard Security Enchancements, VAX/VMS, Primos, MPE/VE, AOS/VS, ConvexOS/Secure, Trusted OS/32, Tandem Guardian 90, OpenVMS VAX, AOS/VS II, Secure Communication Processor, System V/MLS, SEVMS VAX, XTS-200, MVS/SP, XTS-300.*

Однако, вследствие своей трудоемкости данный подход преимущественно применяется производителями, обладающими значительными материальными ресурсами, или поставляющими собственное аппаратное обеспечение.

2. *Доработка существующей системы.* Данный подход состоит в улучшении характеристик некоторой системы-прототипа и доработки ее защиты до требуемого уровня. Минимальное требование к базовой системе — обеспечение поддержку функций защиты на аппаратном уровне, например разделения системных и прикладных программ. В этом случае разработчики защиты существенно ограничены необходимостью совместимости своих продуктов с прототипом, а также функциональными возможностями исходной системы. В качестве примера таких систем можно привести: *Top Secret, UTX/32S, ACF2/VM, MVS/XA с RACF2, VM/SP, HP-UX BLS, Ultrix MLS+, Unisys OS Security, Admahl UTS/MLS, Trusted Irix/B, CX/SX, Trusted Xenix, Windows NT.*

Несмотря на то, что этот подход характеризуется значительно меньшими затратами, попытки внести защиту в незащищенную систему испытывают серьезные трудности, что подтверждается многочисленными провалами попыток доработки операционных систем, не удовлетворяющих требованиям защиты, таких как MS Windows 3.11 и 95 и Novell Netware 3.x .

Для того чтобы быть сертифицированными в соответствии с «Оранжевой книгой», современные системы должны отвечать как соответствующим требованиям защиты (см. глава 2), так и общим функциональным требованиям, которые также существенно влияют на принципы построения защищенных систем. Ведь защищенные системы, как и все остальные современные средства обработки информации, должны быть в зависимости от назначения: многопользовательскими, многозадачными, надежными и масштабируемыми, работать в гетерогенных сетях. В последнее время к системам обработки информации было предъявлено требование распределенной архитектуры. В связи с развитием Internet возрастает значение требований безопасности для глобальных распределенных систем. Поскольку системы на базе UNIX удовлетворяют всем вышеперечисленным требованиям, то не удивительно, что большинство современных защищенных систем ведут свою родословную от старой доброй UNIX, хотя и значительно переработанной в части защиты.

Среди требований «Оранжевой книги» не последнее место занимают требования управления доступом. В следующем параграфе рассмотрим наиболее важные принципы проектирования систем разграничения доступа (как основу безопасности операционной системы) и архитектуру TCB защищенной операционной системы.

5.1.2. Контроль и управление доступом

Основной задачей контроля и управления доступом является ограничение операций, выполняемых зарегистрированными пользователями в системе. Существует два основных механизма управления доступом – дискреционный (произвольный) и мандатный (нормативный). Требования различных стандартов информационной безопасности к управлению доступом рассматривались в главе 2. Проведем анализ основных особенностей механизмов реализации этих требований в современных системах, на примере ОС.

5.1.2.1. Произвольное управление доступом

Требования произвольного управления доступа в соответствии с требованиями «Оранжевой Книги» появляются, начиная с класса С.

Основой реализации произвольного управления доступом является матрица прав доступа, строки которой соответствуют субъектам (пользователи, процессы и т. д.), а столбцы — объектам (файлы, каталоги, процессы и т.д.). В ячейках матрицы содержатся права доступа субъектов к объектам. Пример матрицы доступа приведен на рисунке 5.1.

Файлы / Пользователи	F1	F2	F3	F4	F5
Петров		R			
Иванов		RW	R		
Федоров					RW
Сидоров	C	C	C	C	C

Рис. 5.1. Матрица произвольного управления доступом.
R – права доступа пользователя по чтению;
W – права доступа пользователя по записи;
C – управление доступом для других пользователей;

В зависимости от способа представления матрицы прав доступа в ОС различают несколько способов реализации произвольного контроля доступа. Наиболее распространенными для операционных систем являются списки прав доступа (Access Control List — ACL), биты доступа, «парольная» защита.

«Парольная» защита осуществляется следующим образом: пользователь использует отдельный пароль для доступа к каждому объекту в системе. В большинстве реализаций парольных систем существуют различные пароли для каждого объекта и для каждого типа доступа. Этот механизм был реализован в операционных системах IBM MVS и NOS.

Использование данного метода доставляет пользователю массу неудобств, т.к. запомнить пароли для каждого объекта и типа доступа невозможно, а хранить их в программах и файлах – ненадежно. Существенную проблему для применения парольной защиты представляет собой и необходимость периодической смены паролей.

В подавляющем большинстве современных защищенных ОС используются списки прав доступа и биты доступа. Рассмотрим эти механизмы более подробно.

Списки прав доступа

При реализации произвольного управления доступом с помощью ACL с каждым объектом ассоциируется список пользователей с указанием их прав доступа к объекту. При принятии решения о доступе, соответствующий объекту доступа ACL проверяется на наличие прав, ассоциированных с идентификатором пользователя, запрашивающего доступ, или его группы. Пример реализации списка прав доступа приведен на рис. 5.2.

Для уменьшения размера списков прав доступа могут применяться методы группирования субъектов, использования умолчаний, шаблонов и т. д. Преимуществом данной реализации произвольного контроля доступа является возможность задания прав доступа индивидуально для каждого пользователя. Основными недостатками данного метода являются большие временные затраты на обработку списков по сравнению с битами защиты, а также необходимость разрешения противоречий между отдельными элементами списка.

Биты защиты

Вследствие того, что многие защищенных ОС ведут свое происхождение от UNIX, они реализуют произвольный доступ с помощью механизма битов защиты. При этом вместо списка пользователей, которым разрешен доступ к объекту, с объектом связываются биты защиты. Пример битов защиты приведен на рис. 5.3. В ОС UNIX биты защиты указывают, права доступа для трех категорий пользователей: все пользователи (world), члены группы владельца (group) и владелец объекта (owner). При этом биты защиты может изменять только владелец объекта и администратор.

Рассмотрим типичный алгоритм проверки прав доступа субъекта системы к объекту при использовании механизма битов защиты. Субъект системы обычно ассоциирован с некоторым эффективным идентификатором (EUID), содержащим информацию о пользователе и группе, к которой он принадлежит.

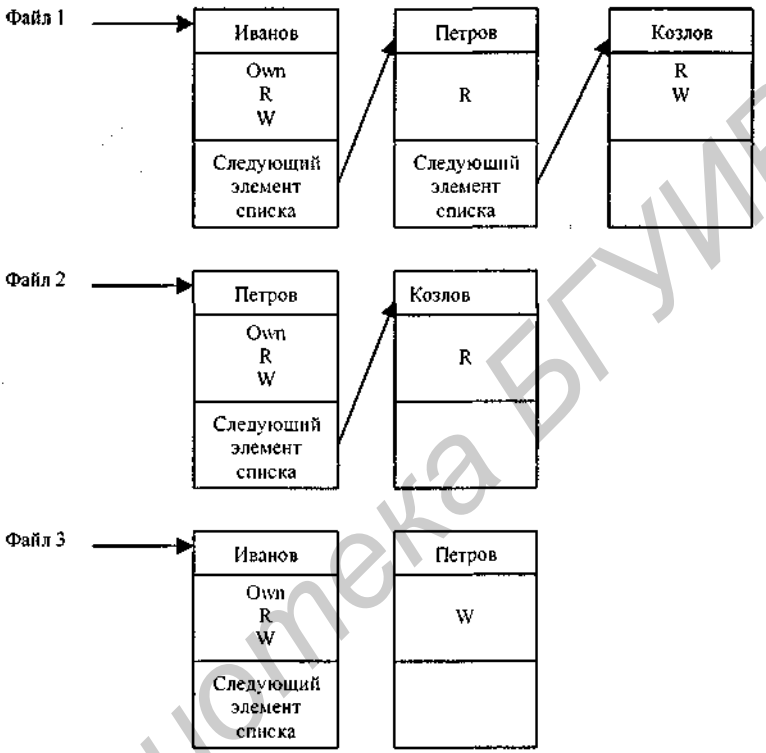


Рис. 5.2. Пример списка контроля доступа к файлам.

Владелец			Группа			Все пользователи		
Чтение	Запись	Выполн.	Чтение	Запись	Выполн.	Чтение	Запись	Выполн.
1	2	3	4	5	6	7	8	9

1…9 - номер бита

Рис. 5.3. Биты защиты UNIX.

При попытке доступа выполняются следующие действия:

1. Проверяется, является ли субъект собственником объекта. Для этого сравниваются значения эффективного идентификатора процесса и идентификатора владельца объекта. Если они равны, то сравниваются полномочия владельца объекта с запрашиваемым типом доступа. Если запрашиваемый тип доступа присутствует в соответствующем поле, то

доступ предоставляется. Если нет, то доступ отклоняется. Если идентификаторы не равны, то осуществляется переход ко второму шагу алгоритма.

2. Проверяется, входит ли субъект в группу владельца. Для этого сравнивается значения эффективного идентификатора процесса и идентификатора группы владельца объекта. Если они равны, то сравниваются полномочия группы владельца объекта с запрашиваемым типом доступа. Если запрашиваемый тип доступа присутствует в соответствующем поле, то доступ предоставляется. Если нет, то доступ отклоняется. Если идентификаторы не равны, то осуществляется переход к третьему шагу алгоритма.

3. В противном случае сравниваются полномочия, предоставленные всем пользователям системы с запрашиваемым типом доступа. Если запрашиваемый тип доступа присутствует в соответствующем поле, то доступ предоставляется. Если нет, то доступ отклоняется.

Недостатком использования механизма битов защиты является неполная реализация произвольного контроля доступа, т.к. доступ к объекту нельзя разрешить или запретить для отдельных пользователей. Сложные комбинации прав доступа могут быть установлены путем создания индивидуальных групп для каждого файла и копированием файлов, но это очень сложно организовать для большого количества пользователей. В современных системах часто используется комбинация списков контроля доступа и битов защиты.

5.1.2.2. Нормативное управление доступом

В отличие от произвольного управления доступом, которое позволяет передавать права от одного пользователя другому, нормативное управление доступом полностью запрещает передачу прав доступа между пользователями. Это позволяет разрешить проблему «троянских коней» в защищенных информационных системах [9, 10, 11]. Нормативное управление доступом основано на модели Белла-Лападула, которая описывает правила документооборота, принятые в правительственных учреждениях США (см. разд. 4.1.4).

Для защиты от угрозы целостности информации в защищенных информационных системах чаще всего используется модифицированный аналог модели Белла и Лападула, так называемая нормативная модель целостности Биба.

Нормативную модель целостности Биба часто называют инверсией модели Белла и Лападула. Это довольно точное название, поскольку основные правила этой модели просто переворачивают правила модели Белла и Лападула. Но при этом в модели рассматриваются уровни целостности, а не уровни безопасности.

В системах, не реализующих модель Биба, для защиты целостности доверенного программного обеспечения, TCB системы (см. следующий параграф), в рамках нормативной модели Белла и Лападула может использоваться следующий прием. TCB системы размещается на нижней ступени иерархии безопасности. Поскольку субъекты и объекты с высокой целостностью находятся внизу иерархии, а компоненты с низкой целостностью - наверху иерархии, то правила «запрет чтения с верхнего уровня» и «запрет записи на нижний уровень» имитируют нормативную модель целостности Биба в структуре модели Белла и Лападула. Примеры реализации модели Белла и Лападула приведены далее.

5.1.2.3. Примеры реализации управления доступом и контроля за его осуществлением в различных ОС

Для иллюстрации методов реализации контроля доступа в защищенных операционных системах, рассмотрим, как практически реализован контроль доступа в системах UTS MLS 2.1.5+, Trusted Xenix версии 3.0 и Windows NT. В двух первых операционных системах реализован произвольный и нормативный контроль доступа. В третьей системе реализовано произвольное управление доступом. Операционная система UTS MLS 2.1.5+ является доработкой традиционной UNIX системы до системы, защищенной по классу B. Система Trusted Xenix версии 3.0 выбрана авторами для рассмотрения, т.к. механизмы защиты, реализованные в ней, удовлетворяют требованиям класса B3, и она функционирует на персональных компьютерах фирмы IBM. Операционная система Windows NT попала в данный список по причине своей популярности.

5.1.2.3.1. Управление и контроль доступа в UTS MLS 2.1.5 +

Управление доступом в операционной системе UTS MLS 2.1.5+ реализовано следующим образом. В описании объекта (в структуре i-node) операционной системы UTS MLS 2.1.5+ (рис. 5.4.) хранится информация о владельце объекта (его идентификатор — UID, указывающий на пользователя, создавшего объект), псевдогрупповая информация (Privs) и права доступа для всех пользователей (other) в системе.

Владелец	Псевдогрупповая информация	Все пользователи

Рис. 5.4. Описатель объекта.

Псевдогрупповая информация указывает на комбинацию группы субъекта с меткой безопасности нормативного контроля доступа. Стандартный механизм UNIX дает поддержку для произвольного разграничения доступа для групп (DGID). Однако, в стандартной версии ОС UNIX отсутствует поддержка нормативного управления доступом. Для того чтобы обеспечить поддержку групп и нормативное управление доступом,

а также совместимость с UNIX в UTS MLS 2.1.5+ вместо групп UNIX используется понятие псевдогрупповой информации, ассоциированной с каждым объектом и процессом.

Каждая комбинация группы и метки безопасности нормативного управления доступом является уникальной и отображается в единственную псевдогрупповую информацию. При этом меняется семантика поля GID. Вместо идентификатора группы в UTS MLS данное поле содержит псевдогрупповую информацию, указывающую на соответствующий элемент в таблице атрибутов безопасности (данная таблица поддерживается TCB ОС UTS). Каждый элемент таблицы атрибутов безопасности (рис. 5.5.) содержит следующую информацию:

- Идентификатор псевдогрупповой информации в таблице (PTI).
- Группу пользователя.
- Метку безопасности нормативного управления доступом.

PTI	Метка нормативного контроля доступа		Группа (DGID)
	Уровень	Категория	

Рис. 5.5. Формат записи таблицы атрибутов безопасности.

Из приведенной таблицы атрибутов безопасности видно, что в ОС UTS метка нормативного контроля доступа является объединением иерархического (упорядоченного) уровня и неиерархической категории. Данная пара определена для каждой метки в таблице атрибутов безопасности.

Нормативное управление доступом в ОС UTS позволяет использовать 256 упорядоченных уровней (самый низкий – 0, самый высокий -255) для задания уровня безопасности субъектов и объектов. Уровень 0 (SYSTEM) присвоен всем элементам, входящим в состав TCB, (что обеспечивает защиту его целостности, см. предыдущий параграф), уровни 1-255 предназначены для пользователей. Всего поддерживается 1024 категории.

Рассмотрим алгоритм принятия решения о доступе в ОС UTS MLS 2.1.5+.

1. Для нормативного контроля доступа псевдогрупповой идентификатор объекта (PTI) используется для получения информации об уровне безопасности объекта, извлекаемой из таблицы атрибутов безопасности. Данная информация сравнивается с помощью правил модели Белла и Лападула с соответствующим уровнем безопасности субъекта, запрашивающего доступ. Кроме проверки соответствия уровня безопасности система проверяет соответствие категорий в метке нормативного доступа.

2. Для произвольного контроля доступа прежде всего сравнивается эффективный идентификатор процесса (EUID), которым отмечается каждый процесс пользователя, с идентификатором владельца объекта (UID). Если они равны, то сравниваются полномочия владельца объекта с запрашиваемым типом доступа. Если запрашиваемый тип доступа присутствует в соответствующем поле, то доступ предоставляется. Если нет, то доступ отклоняется. Если идентификаторы не равны, то сравнивается информация о группе процесса с информацией о группе владельца (DGID), получаемой из таблицы атрибутов безопасности (Priv-таблицы). Если они равны, то сравниваются полномочия группы владельца объекта с запрашиваемым типом доступа. Если запрашиваемый тип доступа присутствует в соответствующем поле, то доступ предоставляется. Если нет, то доступ отклоняется. Если и в данном случае совпадение отсутствует, то сравниваются полномочия хранящиеся в поле «все пользователи» (other) с запрашиваемым типом доступа. Если запрашиваемый тип доступа присутствует в соответствующем поле, то доступ предоставляется. Если нет, то доступ отклоняется.

Субъекту предоставляется доступ к объекту только в том случае, если он одновременно разрешается механизмами нормативного и произвольного контроля доступа. Диаграмма, иллюстрирующая алгоритм принятия решения о доступе в операционной системе UTS MLS 2.1.5+, показана на рис. 5.6.

5.1.2.3.2. Контроль доступа в Trusted Xenix версии 3.0

Произвольное управление доступом в операционной системе Trusted Xenix версии 3.0 обеспечивается битами защиты и списками контроля доступа (ACL). Как и в традиционной UNIX-системе биты защиты хранятся в описателе (i-node) файлов. Всего существует 11 битов защиты. Первые два бита защиты – биты привилегированных приложений (setuid/setgid биты), остальные биты защиты – стандартные для UNIX.

ACL в операционной системе Trusted Xenix хранятся в специальных файлах. Максимальное число элементов в ACL ограничено только максимальным размером файла. Структура описателя каждого файла, доступ к которому контролируется ACL, содержит ссылку на имя и описатель ACL файла, в котором хранятся права доступа к этому файлу. Доступ к ACL-файлам имеет только TCB системы. Записи в ACL файле имеют следующую форму: «пользователь – группа – тип доступа» (“user-id:group-id:access-type”).

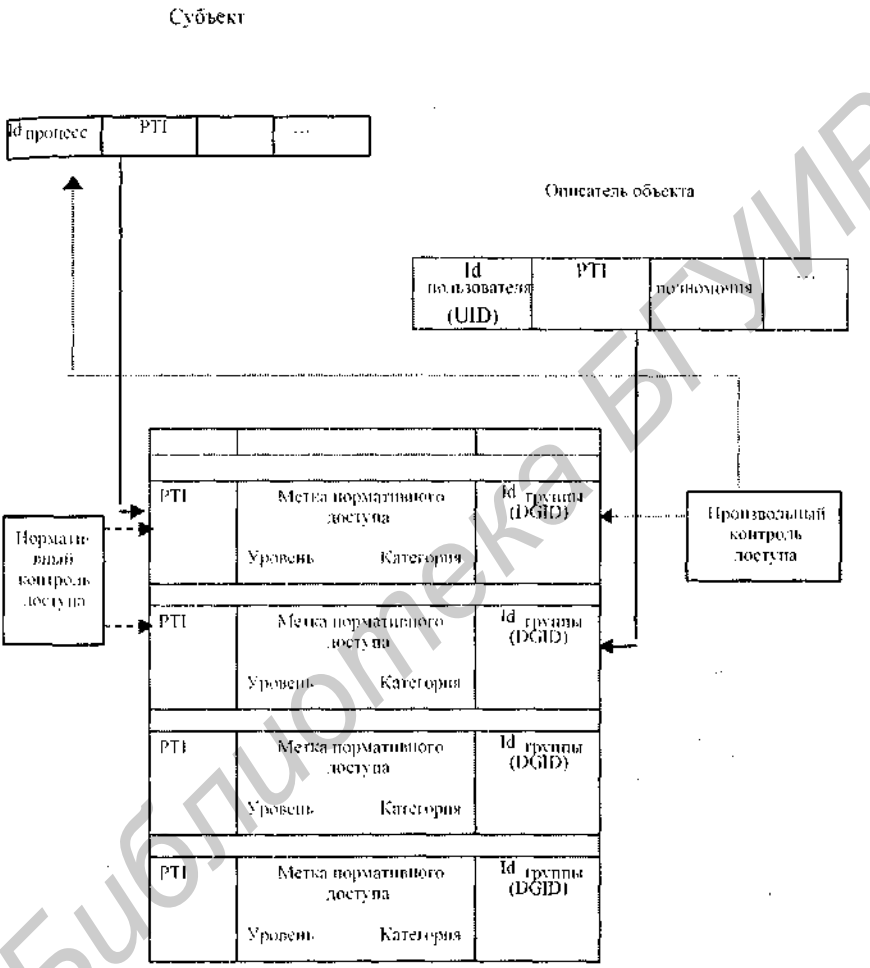


Рис. 5.6. Механизм контроля доступа в UTS MLS 2.1.5+

Механизм принятия решения о предоставлении доступа объединяет стандартный алгоритм с использованием битов защиты с алгоритмом, основанным на ACL. В начале осуществляется проверка битов защиты владельца(user-id) и группы(group-id).

После этого, если для объекта существует ACL, то соответствующий файл сначала проверяется на наличие записей, в которых участвует идентификатор пользователя и его группы. Затем проверяются биты защиты для всех остальных пользователей системы (world).

Нормативное управление доступом опирается на иерархические уровни и неиерархические категории. Решение о доступе принимается на основании модели Белла и ЛаПадула. Данные об уровне безопасности и категории объекта хранятся в описателе объекта. В системе поддерживается 255 уровней безопасности и 64 категории.

Доступ предоставляется, только в том случае, если он одновременно разрешен механизмами нормативного и произвольного контроля доступа.

5.1.2.3.3 Контроль доступа в Windows NT

Windows NT поддерживает произвольное управление доступом с помощью списков прав доступа (ACL). Каждый объект, к которому контролируется доступ, имеет собственный ACL, который состоит из так называемых сущностей контроля доступа (Access Control Entries — ACE). Всего существует три типа ACE. Два из них относятся к управлению доступом: первый разрешает указанный тип доступа (ACE Allowed), а второй — запрещает (ACE Denied). Третий тип используется для аудита. ACE первых двух типов содержит маску доступа, определяющую разрешаемые или запрещаемые типы доступа объекта.

При принятии решения о предоставлении доступа каждая ACE обрабатывается следующим образом:

1. Система сравнивает идентификатор процесса, запросившего доступ к объекту с идентификаторами, присутствующими в ACL объекта. Если в ACL отсутствует упоминание этого идентификатора, то доступ запрещается. Если идентификатор присутствует в ACL, то дальнейшая обработка ACL производится в зависимости от типа ACE. Сперва обрабатываются ACE типа Denied, а затем типа Allowed.
2. Для всех ACE, имеющих тип Denied, запрашиваемый тип доступа сравнивается с указанным в маске ACE. Если хотя бы один тип доступа (чтение, запись и т. д.) присутствует в обеих масках, то доступ запрещается, и дальнейшая обработка не производится. После обработки всех ACE этого типа начинается обработка элементов типа Allowed.
3. Для всех ACE, имеющих тип Allowed, запрашиваемый тип доступа сравнивается с указанным в маске ACE. По результатам сравнения отмечается какие типы запрашиваемого доступа разрешены. Если все запрашиваемые типы доступа встретились в масках ACE, то доступ разрешается, в противном случае доступ запрещается.

5.1.3. Архитектура информационных системы с точки зрения безопасности

Основные механизмы обеспечения безопасности сосредоточены в специально выделенной части системы, получившей название TCB (Trusted Computing Base).

«Оранжевая книга» [2] определяет это понятие следующим образом: «совокупность защитных механизмов в компьютерной системе, включая аппаратное, специальное и программное обеспечение, объединение которого реализует политику безопасности. TCB системы состоит из одного или более компонентов, которые совместно реализуют единую политику безопасности в системе. Способность доверенного программного обеспечения корректно реализовывать политику безопасности зависит только от механизмов TCB и корректной администрации системы».

Рассмотрим основные принципы, используемые при построении TCB систем защиты. Прежде всего, необходимо отметить, что существуют два подхода к построению TCB: в виде единого модуля, и многокомпонентное TCB.

TCB, состоящие из единого модуля, применялись в старых, а в настоящее время используются в небольших системах, в которых функции TCB ограничены исключительно реализацией монитора взаимодействий и других базовых механизмов. В качестве примера TCB, состоящего из единого модуля, будет рассмотрено TCB операционной системы UNISYS OS 1100.

В дальнейшем мы будем рассматривать архитектуру современных операционных систем, которые являются многокомпонентными.

С одной стороны TCB должно быть достаточно простым для проведения анализа с целью доказательства безопасности системы. Но с другой стороны, TCB современных систем включают в себя все множество функций, необходимых для поддержания политики безопасности, и, следовательно, являются достаточно громоздкими и сложными. Ключ к решению данного противоречия лежит в применении таких технологий проектирования архитектуры TCB и входящих в него программных средств, которые бы позволяли проводить декомпозицию TCB и анализировать его по частям. Существуют две технологии проектирования сложных программных систем — вертикальная (иерархическая) и горизонтальная (модульная).

Иерархическое проектирование системы защиты основано на одном из фундаментальных постулатов компьютерной безопасности: «*Механизм защиты должен быть простым, единым и находиться на самом низком уровне системы*». Самым простым решением в этом случае является создание такой многослойной иерархической структуры подсистем защиты, что совокупность каждого слоя с лежащими ниже его образует

TCB для вышележащего слоя. Данная структура системы обладает следующими преимуществами:

1. упрощается структура TCB;

2. доказательство безопасности более высокого иерархического уровня системы может опираться на доказанную безопасность более низких слоев иерархии.

В качестве примера рассмотрим иерархическую декомпозицию ОС UNIX. Данная операционная система обычно подразделяется на два модуля с четко определенными интерфейсами:

1. Ядро ОС, исполняющиеся в привилегированном режиме аппаратного обеспечения (Kernel). Вся деятельность прикладных процессов контролируется этой частью ОС. Ядро подразделяется на подсистемы следующим образом:

- интерфейсы верхнего уровня, обеспечивающие реализацию всех системных вызовов (уровень системных вызовов);
- подсистемы, управляющие аппаратным обеспечением (машинно-зависимый уровень).

2. Компоненты TCB, не относящееся к ядру системы и исполняющиеся без поддержки со стороны аппаратного обеспечения.

Для реализации защищенной системы, как было указано ранее, необходима аппаратная поддержка, образующая основу системы защиты. В рассматриваемых системах обычно используются следующие возможности аппаратной защиты:

- кольца защиты;
- разделение адресных пространств.

Кольца защиты используются для разделения процессов разной степени доверия. В самом привилегированном кольце исполняются процессы, принадлежащие ядру ОС. Приложения пользователя исполняются в самом непривилегированном кольце. В кольцах между ними могут исполняться различные привилегированные процессы, входящие в состав TCB. Разделение адресных пространств используется для разграничения доступа к объектам памяти, принадлежащим различным процессам. Если память разделяется между процессами, то для доступа к ней должны использоваться соответствующие механизмы защиты.

Обычно TCB предоставляет пользователю и прикладным процессам следующие типы интерфейсов:

- командный интерфейс;
- прикладной программный интерфейс системы (API, множество системных вызовов).

Командный интерфейс TCB состоит из множества команд TCB, которые пользователь системы может ввести с помощью устройств ввода (клавиатура, мышь и т.д.). Прикладной программный интерфейс системы

состоит из множества системных вызовов, к которым может обратиться прикладная программа.

Помимо иерархического проектирования ТСВ существуют технологии «горизонтального», или модульного проектирования. Модульное проектирование облегчает декомпозицию системы в ходе ее анализа и прочно вошло в практику создания защищенных систем. Как правило, в современных системах ТСВ представляет собой множество отдельных взаимосвязанных модулей, выполняющих различные функции, относящиеся как к безопасности системы, так и к ее основным функциям. Так как безопасность ТСВ должна быть доказана, в его состав не должны входить «лишние» компоненты, не влияющие на безопасность системы. Совокупность модулей ТСВ определяет так называемый «периметр защиты». Данный периметр определяется интерфейсом, предоставляемым со стороны ТСВ для внешних субъектов. В «периметр защиты» должны входить компоненты системы, отбираемые по следующему принципу: если ограничения, накладываемые на компонент, оказывают влияние на доказательство безопасности системы, то данный компонент включается внутрь периметра безопасности.

Проектирование архитектуры системы, построенной по модульной технологии, проводится в соответствии с «принципом использования», который сформулирован следующим образом: модуль А использует модуль В если и только если,

- функция из модуля А вызывает функцию из модуля В и использует результаты данного вызова;
- функция из модуля В исполняет свою работу корректно.

На основании этого принципа можно определить взаимосвязи между модулями и их влияние друг на друга. Результатом применения этой технологии должно являться определение интерфейса ТСВ, включающее описание множества функций ТСВ, видимых для внешних по отношению к ТСВ субъектов, и способов их вызова, в т. ч. допустимые параметры (их тип и порядок), правила обработки возникающих при некорректном вызове исключительных ситуаций.

Возвращаясь к описанию ТСВ UNIX-системы, можно отметить, что по своей функциональности ядро этой операционной системы подразделяется на несколько логических подсистем. Каждая подсистема реализует множество программных объектов, доступных процессам посредством системных вызовов или через другие подсистемы ядра через внутренние интерфейсы ядра. Код, структуры данных и определение интерфейсов для каждой логической подсистемы определяют модульную структуру ядра.

К основным подсистемам относят следующие:

- управление памятью,
- управление файлами,

- управление процессами,
- драйверы устройств,
- взаимодействие процессов,
- аудит и управление доступом.

Подсистемы ТСВ, не принадлежащие ядру операционной системы, обычно включают следующие логические модули:

- подсистему аутентификации,
- поддержку файловой системы,
- подсистему резервного копирования,
- службу почты,
- поддержку исполнения командных файлов,
- поддержку функций системного администратора,
- сервис принтеров,
- сервис управления терминалами,
- управление экспортом и импортом данных,
- управление данными аудита,
- программы поддержки политики безопасности.

На основании изложенного можно сделать вывод о том, что структурирование системы защиты посредством выявления в ней иерархических и модульных зависимостей не только соответствует «хорошему стилю программирования» но и облегчает процесс тестирования и формального анализа безопасности.

В настоящее время особенно эффективно развивается направление, основанное на использовании методов объектно-ориентированного проектирования и программирования, а также микроядерных операционных систем, концепция которых описана в следующем разделе этой главы.

Рассмотрим различные варианты архитектуры ТСВ на примере современных операционных систем: UTS MLS 2.1.5.+, Trusted Xenix версии 3.0 и Windows NT.

5.1.3.1. Архитектура операционной системы UNISYS OS 1100

Примером реализации ТСВ в виде единого модуля является операционная система UNISYS OS 1100. Модуль ТСВ этой ОС и осуществляет следующие функции:

- управление аппаратными ресурсами, включая инициализацию, обработку прерываний, ввод-вывод, поддержку таймера;
- управление процессором;
- управление сохранением данных;
- распределение заданий между процессорами;
- управление транзакциями;

- аудит;
- управление лентами и файлами;
- контроль доступа;
- автоматическое восстановление.

Взаимодействие с остальными компонентами системы осуществляется с помощью следующих интерфейсов:

1. Запросы к управляющему модулю из других частей управляющего модуля.
2. Командный интерфейс в виде интерпретатора специального языка управления.
3. Команды оператора консоли.

5.1.3.2 Архитектура операционной системы TCB UTS MLS 2.1.5+.

Рассмотрение архитектуры операционной системы UTS MLS 2.1.5+ начнем с вертикальной декомпозиции TCB. На рис. 5.7. показано, как обычное программное обеспечение системы может обращаться к сервису TCB UTS.

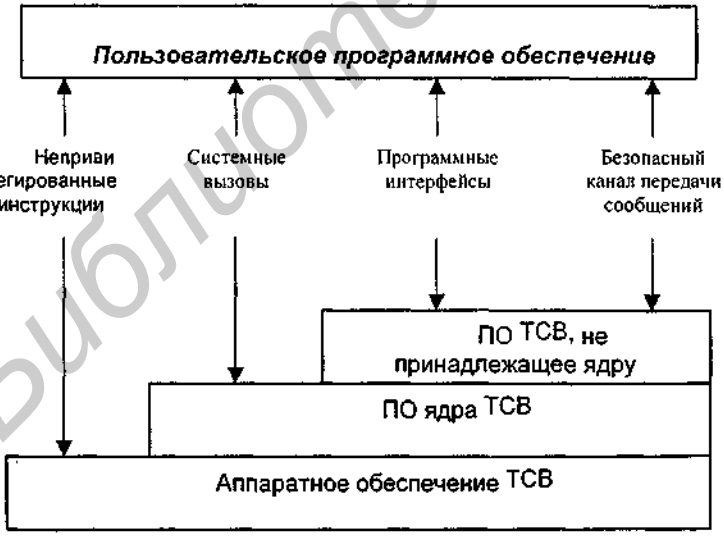


Рис. 5.7. TCB-интерфейс UTS.

Из рисунка видно, что существует четыре возможности обращения к TCB системы:

1. обращение к аппаратной части TCB с помощью доступных инструкций процессора;
2. системные вызовы ядра UTS исполняют вызов супервизора (SVC), который передает управление в ПО TCB ядра. При этом параметры системного вызова копируются из адресного пространства пользователя в адресное пространство ядра одновременно с проверкой их корректности;
3. пользователь может взаимодействовать с ПО TCB, не принадлежащем ядру системы. Необходимо отметить, что не все программное обеспечение TCB, не принадлежащее ядру системы предоставляет пользовательскому ПО свои интерфейсы, например процессы-демоны;
4. процессы могут взаимодействовать с помощью канала безопасных сообщений (SMC).

С точки зрения иерархической декомпозиции TCB UTS /MLS состоит из следующих компонентов:

1. Аппаратное обеспечение.

2. Ядро ОС UTS/MLS, которое выполняется в режиме SUPERVISOR. Это позволяет ядру исполнять привилегированные аппаратные инструкции и осуществлять ввод/вывод. Ядро имеет монолитную структуру и представляет прикладным процессам интерфейс со множеством системных вызовов. При этом само ядро не является процессом. Каждый процесс в системе через соответствующий вызов может переключиться в контекст ядра. В контексте ядра, процесс имеет доступ к стеку и коду ядра. После возвращения из режима ядра, процесс не имеет доступа к контексту ядра.

3. Доверенные программы UTS/MLS, обеспечивающие различные системные службы. К программам этого типа относятся: программы, исполняемые необходимо выполнять с привилегиями root и административные утилиты. Часть TCB, не принадлежащая ядру системы, выполняется в режиме PROBLEM, который не позволяет выполнять непривилегированные инструкции. Это не означает, что часть TCB, не принадлежащая ядру системы, выполняется без привилегий. Все части TCB, не принадлежащие ядру системы, являются доверенными программами и выполняются в своих доменах.

Далее рассмотрим модульную декомпозицию операционной системы UTS MLS 2.1.5+, приведенную на рис. 5.8.

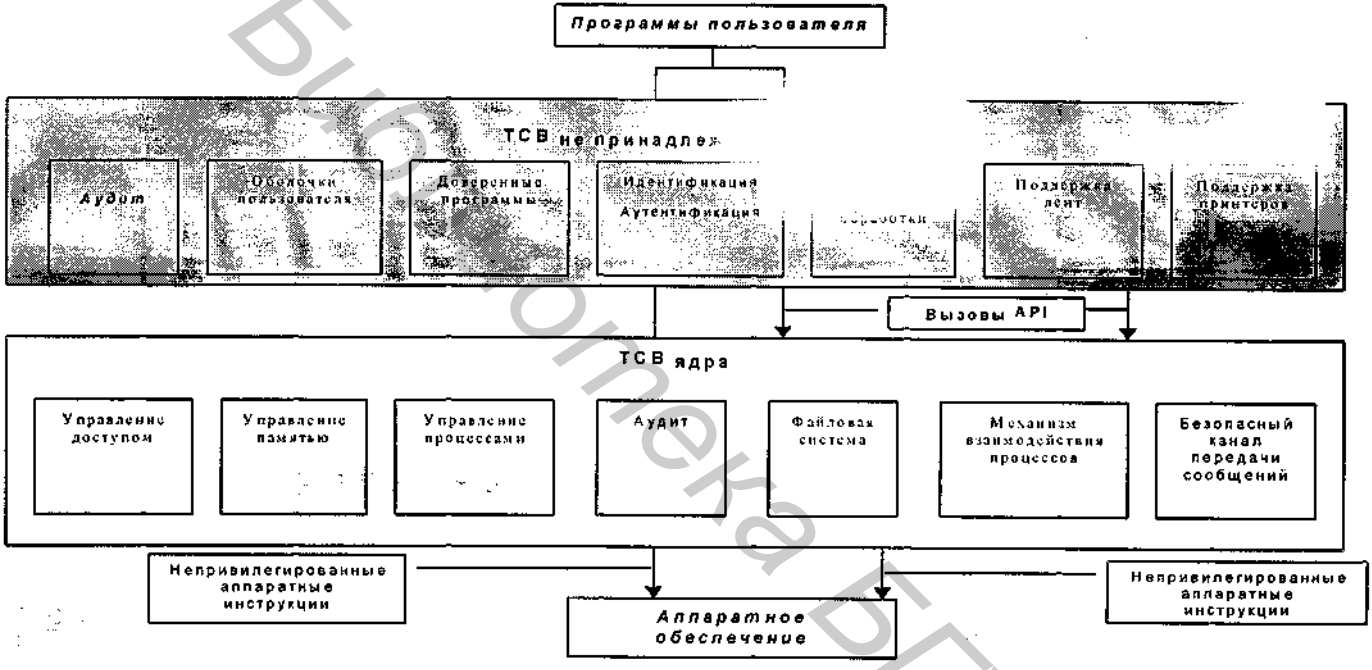


Рис. 5.8. Модульная декомпозиция TCB UTS MLS 2.1.5

Таким образом, иерархическая декомпозиция TCB UTS /MLS может быть дополнена разложением каждого слоя иерархии на модули:
1. Аппаратное обеспечение. Модули отсутствуют.
2. Ядро ОС UTS/MLS состоит из: подсистемы управления памятью, подсистемы управления процессами, файловой системы, механизма взаимодействия процессов, подсистемы разграничения доступом, поддержки аудита в системе, безопасных каналов передачи сообщений.
3. Доверенные программы UTS/MLS включают в себя средства обеспечения идентификации-аутентификации, обработки данных аудита, командный интерпретатор и т. д.

5.1.3.3. Архитектура операционной системы Trusted Xenix версии 3.0

Иерархия компонентов TCB Trusted Xenix версии 3.0 включает следующие четыре слоя:

- аппаратное обеспечение;
- ядро ОС;
- доверенные процессы;
- служебно-административные файлы и базы данных.

Trusted Xenix функционирует на платформе Intel x86 и использует механизмы защиты этой линии процессоров. Как известно, процессор Intel x86 в защищенном режиме обеспечивает 4 кольца привилегий. Программное обеспечение ядра системы Trusted Xenix версии 3.0 функционирует в самом привилегированном кольце, кольце 0. Это позволяет ядру получить доступ к привилегированным инструкциям процессора. Основная часть ядра предоставляет услуги остальному программному обеспечению системы посредством системных вызовов. Часть ядра отвечает за инициализацию системы и обработку прерываний. Ни один компонент ядра не выполняется с уровнем привилегий, отличным от 0.

Все остальное программное обеспечение, как из состава TCB, так и прикладное выполняется на третьем уровне привилегий. К компонентам TCB, не вошедшим в состав ядра относятся утилиты управления безопасностью, которым даны специальные полномочия игнорировать некоторые ограничения.

Рассмотрим модульную декомпозицию компонентов TCB Trusted Xenix, входящих в состав ядра:

- *Подсистема конфигурации* поддерживает системные таблицы, выделяет память для служебных структур данных. Данная подсистема непосредственно не обрабатывает данные, относящиеся к безопасности, но определяет информацию, без которой система просто не может функционировать.

- *Подсистема инициализации* отвечает за инициализацию системы, аппаратных устройств, обработчиков прерываний и таблиц адресного пространства, а также переключение процессора в защищенный режим. Система не реализует функций защиты, за исключением корректной инициализации системы.
- *Подсистема системных вызовов* поддерживает механизм системных вызовов функций ядра. Для обеспечения безопасности осуществляется проверка аргументов системных вызовов, кроме того, данная подсистема осуществляет аудит обращений к ядру.
- *Подсистема безопасности* включает монитор взаимодействий, который поддерживает множество функций, в т. ч. произвольный и нормативный контроль доступа, контроль привилегий и аудит взаимодействий в системе.
- *Подсистема управления процессами* отвечает за переключение процессов, обработку прерываний, контроль времени и функции, обеспечивающие синхронизацию процессов. Кроме того, данная подсистема включает интерфейс для взаимодействия между процессами и средства отладки. Таким образом, эта подсистема реализует следующие функции защиты: изоляция процессов и организация безопасного взаимодействия между ними.
- *Подсистема распределения памяти* отвечает за управление адресными пространствами и свопингом. Из функций защиты к компетенции этой подсистемы относятся: уничтожение содержимого объектов перед повторным использованием, контроль доступа областям памяти, изоляция ядра и процессов, а также реализация повышения привилегий для setuid/setgid программ.
- *Подсистема ввода/вывода* обслуживает функции ввода/вывода и включает драйверы устройств.
- *Файловая подсистема* реализует соответствующие функции работы с файлами: разбор имен файлов, управление каталогами, описателями файлов(i-node), контроль доступа к файлам, управление совместным использованием файлов, монтирование/демонтирование файловых систем и т.д. Для осуществления контроля доступа к файлам данная подсистема обращается к подсистеме безопасности.
- *Подсистема взаимодействия процессов* реализует механизмы межпроцессного взаимодействия, семафоры, очереди сообщений и разделяемые сегменты памяти. Контроль доступа при межпроцессном взаимодействии осуществляется с помощью подсистемы безопасности.
- *Подсистема вспомогательных функций* содержит процедуры, не включенные в перечисленные подсистемы: останов и выгрузка системы, поддержка отладки ядра для разработчиков, мониторинг произ-

водительности ядра и т.д. Данная подсистема не несет ответственности за безопасность.

5.1.3.4 Архитектура операционной системы Windows NT

Рассмотрим вертикальную декомпозиции архитектуры операционной системы Windows NT(рис. 5.9).

Из рисунка видно, что существует три вида операций, доступных для прикладного программного обеспечения:

- непривилегированные инструкции процессора;
- системные вызовы ядра Windows NT;
  (Вызов осуществляется посредством прерывания 2E, которое осуществляет переход из пользовательского режима в режим ядра. При этом параметры вызова копируются из пользовательского адресного пространства в пространство ядра и проверяется их корректность.)
- межпроцессное взаимодействие. С помощью этого механизма прикладные программы могут получить сервис, предоставляемый серверами из состава TCB.

Модульная декомпозиция TCB Windows NT показана на рис. 5.10.

Темным фоном выделено программное обеспечение, зависящее от аппаратной платформы. Из рисунка видно, что TCB Windows NT состоит из следующих частей:

- *ядро системы* исполняется в привилегированном режиме процессора;
- *защищенные серверы* (protected servers) исполняются в пользовательском режиме процессора;
- *средства администрирования* исполняются в пользовательском режиме процессора.

Ядро системы в свою очередь состоит из трех компонентов:

1. Микроядро (Microkernel) отвечает за примитивные объекты ОС, используемых подсистемами Windows NT.
2. Программное обеспечение, скрывающее особенности аппаратной платформы (Hardware Abstraction Level). Данное программное обеспечение обеспечивает переносимость Windows NT с одной аппаратной платформы на другую.
3. Программное обеспечение, отвечающее за функционирование подсистем ввода - вывода, файловых систем, кэша, драйверов устрой, монитора взаимодействий, менеджера процессов, менеджера объектов, менеджера виртуальной памяти, поддержку виртуальных DOS-машин, поддержку взаимодействия между процессами и конфигурацию системы.

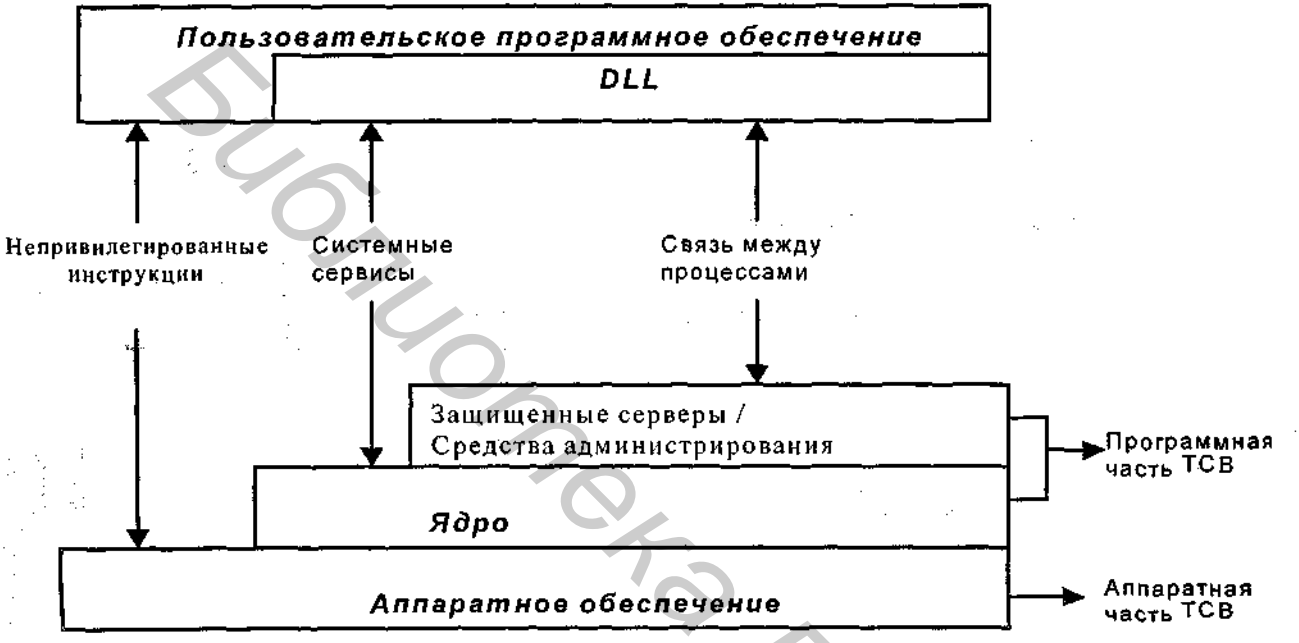


Рис. 5.9. TCB-интерфейс Windows NT.

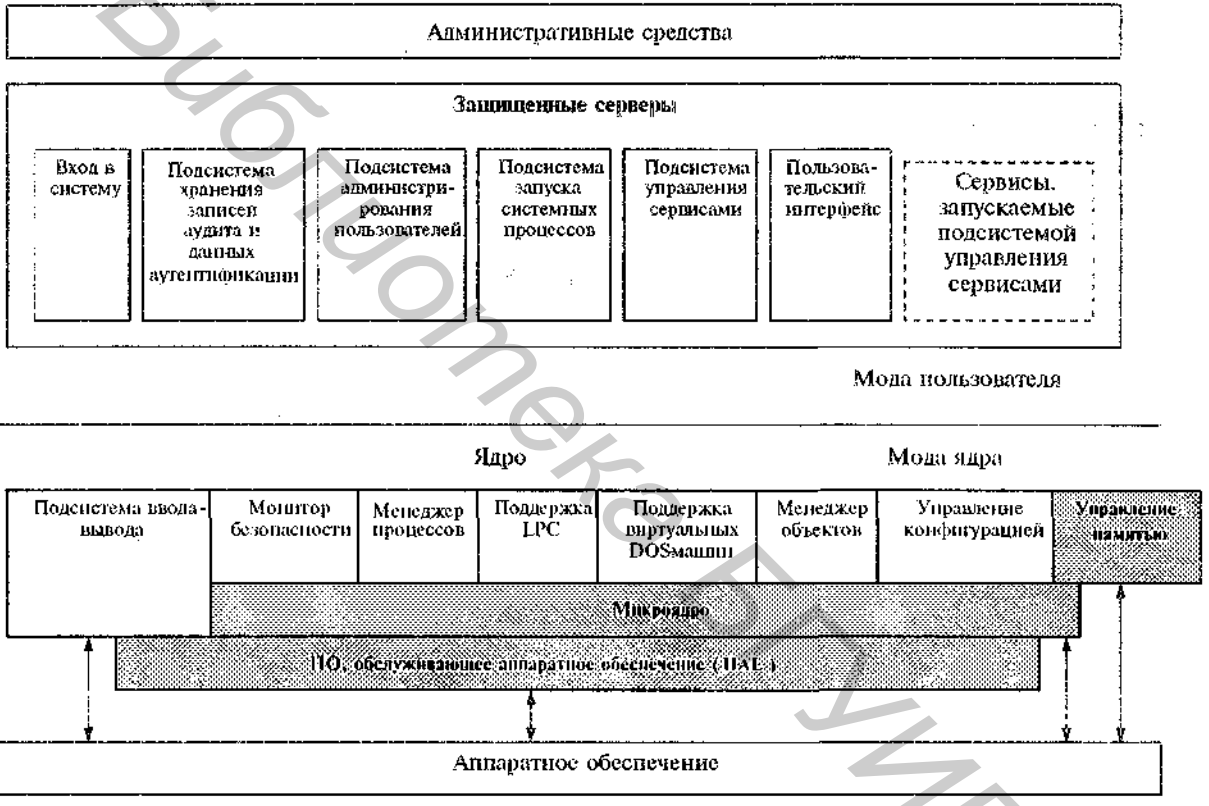


Рис. 5.10. Модульная декомпозиция TCB Windows NT

Набор защищенных серверов включает в себя:

1. Программу аутентификации пользователей (WinLogon).
2. Подсистему хранения журнала аудита и параметров аутентификации.
3. Подсистему администрирования пользователей.
4. Подсистему запуска системных процессов (Session Manager).
5. Подсистему управления драйверами.
6. Пользовательский интерфейс Windows NT (Win32).

Рассмотренные в данном обзоре ОС UTS MLS 2.1.5+, Trusted Xenix 3.0 и Windows NT различаются по функциональным характеристикам, классам защищенности и реализации функций защиты. Однако, их архитектуры во многом похожи. Данное сходство вызвано тем, что принципы построения ТСВ и систем защиты во многом совпадают с принципами «хорошего программирования», в том смысле, что программное обеспечение, составляющее ТСВ системы, должно быть четко структурировано, а его надежность определяет реальный уровень безопасности системы.

5.1.4. Заключение к обзору защищенных систем

На основании анализа архитектуры и характеристик сертифицированных защищенных систем можно сделать следующие выводы:

1. В настоящее время существует два принципиально отличающихся подхода к проектированию защищенных информационных систем. При первом подходе осуществляется разработка защищенной системы с нуля, иногда даже начиная с аппаратной части, для такой системы специально разрабатываются защищенные приложения. Вследствие своей трудоемкости этот подход характерен для производителей, обладающих значительными материальными ресурсами или разрабатывающих собственное аппаратное обеспечение. Второй подход состоит в улучшении характеристик некоторой системы-прототипа и доработки ее защиты до требуемого уровня. Данный подход характеризуется значительно меньшими затратами и применяется для доработки популярных систем общего назначения.

2. Принципы построения защищенных ОС эволюционировали вместе с технологией построения операционных систем и развитием сферы применения компьютерных систем. Защищенные ОС стали разрабатываться для рабочих станций и персональных компьютеров, для сетевых распределенных систем и для систем, подключенных к Internet. Особую роль в эволюции защищенных систем сыграла ОС UNIX, которая наиболее часто служила прототипом для разработки защищенных ОС.

3. Для удовлетворения сертификационным требованиям ОС должна реализовывать основные механизмы управления доступом – дискреционный (произвольный) и мандатный (нормативный). Для реализации произ-

вольного доступа современные защищенные ОС используют как традиционный для UNIX систем механизм битов защиты, так и списки контроля доступа(ACL). Для реализации нормативного контроля доступа как правило используется модель Белла и Лападула. Обеспечение целостности данных в защищенных ОС базируется на нормативной модели целостности Биба.

4. В связи с тем, что в ходе сертификации защищенные ОС подвергаются формальному анализу, подобные системы должны разрабатываться с использованием технологий иерархического и модульного проектирования.

5.2. Создание защищенной операционной системы

Из приведенного обзора следует, что все защищенные ОС по большому счету реализуют примерно один и тот же набор функций защиты — управление доступом и контроль за его осуществлением, идентификация и аутентификация, аудит, прямое взаимодействие и т. д. В этом нет ничего удивительного, т. к. все разработчики ориентируются на соответствующие разделы требований стандартов информационной безопасности (глава 2). Однако, как мы показали в обзоре защищенных систем, способы реализации этих функций, как и обеспечиваемый ими уровень защиты, существенным образом различается от системы к системе даже в рамках одного класса требований.

В этом разделе мы попытаемся достаточно подробно рассмотреть архитектуру защиты ОС и предложить методы реализации функций защиты. Основное внимание будет уделено политике безопасности, управлению доступом и контролю за его осуществлением, также будут затронуты некоторые аспекты реализации аудита, прямого взаимодействия, архитектуры TCB, безопасной инициализации и восстановления после сбоев.

Прежде чем начать построение защищенной системы необходимо определить, какому классу требований стандартов информационной безопасности она должна соответствовать. На современном этапе развития информационных технологий имеет смысл разрабатывать системы отвечающие достаточно жестким требованиям — как минимум должны быть реализованы произвольное и нормативное управление доступом, обеспечивающее защиту от атак с помощью "троянских коней", а структура TCB системы должна позволять применять формальные методы анализа.

Поэтому мы будем ориентироваться на требования безопасности уровня B3 "Оранжевой книги", или примерно соответствующего ему уровня 1B ГТК (см. гл. 2). Требования для этих уровней являются достаточно строгими, т. к. системы этого класса ориентированы на обработку

закрытой информации. Напомним, что эти требования включают поддержку формально определенной модели безопасности, предусматривающую произвольное и нормативное управление доступом и использующую метки безопасности, контроль доступа должен охватывать все субъекты и объекты системы. Кроме того, ТСВ должна быть структурирована с целью исключения из нее подсистем, не отвечающих за реализацию функций защиты, и быть достаточно компактной для эффективного тестирования и анализа. В ходе разработки и реализации ТСВ должны применяться методы и средства, направленные на минимизацию ее сложности. Средства аудита должны включать механизмы оповещения администратора при возникновении событий, имеющих значение для безопасности системы. Требуется наличие средств восстановления работоспособности. Должен осуществляется контроль скрытых каналов утечки информации.

Поскольку представляет интерес только безопасность перспективных систем, будем рассматривать практическую реализацию методов защиты применительно к самой передовой на сегодняшний день технологии построения ОС — технологии микроядра. В отличие от традиционной архитектуры, согласно которой операционная система представляет собой монолитное ядро, реализующие основные функции по управлению аппаратными ресурсами и организующее среду для выполнения пользовательских процессов, микроядерная архитектура распределяет функции ОС между микроядром и входящими в состав ОС системными сервисами, реализованными в виде процессов, равноправных с пользовательскими приложениями.

Микроядро реализует базовые функции операционной системы, на которые опираются эти системные сервисы и приложения. В результате, такие важные компоненты ОС как файловая система, сетевая поддержка и т. д. превращаются в по-настоящему независимые модули, которые функционируют как отдельные процессы и взаимодействуют с ядром и друг с другом на общих основаниях. Это означает, что имевшее место раньше четкое разделение программного обеспечения на системные и прикладные программы размывается, т. к., фактически, между процессами, реализующими функции ОС, и прикладными процессами, выполняющими программы пользователя, нет никаких различий. Все компоненты системы используют средства микроядра для обмена сообщениями, но взаимодействуют непосредственно. Микроядро лишь проверяет законность сообщений, пересылает их между компонентами и обеспечивает доступ к аппаратуре.

Другое нововведение в технологии построения ОС, связанное исключительно с внедрением технологии микроядра, это организация взаимодействий между процессами и ядром с помощью универсального механизма передачи информации — обмена сообщениями, пришедшему на

смену технике системных вызовов. При этом десятки или даже сотни вызовов, различающихся числом и типом параметров, можно заменить несколькими типами сообщений, которые содержат компактные порции информации и могут передаваться от одного обработчика к другому.

На современном этапе развития ОС эта технология является самой перспективной, т. к. позволяет преодолеть самые существенные недостатки существующих систем — отсутствие мобильности, громоздкость, ресурсоемкость. Реализация многих традиционных функций ОС за пределами ядра способствует построению на базе этого ядра операционных систем с недостижимым ранее уровнем модульности и расширяемости.

Мы рассмотрим основные принципы построения защищенных микроядерных операционных систем и методы реализации присутствующих в них функций защиты. В качестве иллюстрации приводятся механизмы защиты наиболее передовой с точки зрения безопасности защищенной ОС — Trusted Mach, разрабатываемая фирмой Trusted Information Systems (TIS) в соответствии с требованиями класса B3 "Оранжевой книги". В качестве примеров реализации средств защиты будут рассмотрены отдельные механизмы Trusted Mach. Однако, следует отметить, что приведенные описания подсистем Trusted Mach ни в коей мере не претендуют на полноту и используются только для того, чтобы послужить иллюстрацией рассматриваемых методов реализации средств защиты операционных систем.

5.2.1. Введение в архитектуру микроядерных операционных систем

Для того чтобы ознакомить читателя с базовым набором понятий, необходимым для изложения деталей реализации средств защиты, мы рассмотрим некоторые понятия и основные принципы построения архитектуры современных микроядерных операционных систем на примере лежащего в основе Trusted Mach микроядра MK++.

5.2.1.1. Основные положения архитектуры микроядерных ОС

В основе архитектуры микроядерных ОС лежат следующие базовые концепции:

- минимизация набора функций, поддерживаемых микроядром, и реализация традиционных функций ОС (файловая система, сетевая поддержка) вне микроядра;
- организация синхронного и асинхронного взаимодействия между процессами исключительно через механизм обмена сообщениями;

- все отношения между компонентами строятся на основе модели клиент/сервер;
- применение объектно-ориентированного подхода при разработке архитектуры и программировании системы.
-

Минимизация функций микроядра дает возможность сконцентрировать в нем код, зависящий от аппаратной платформы, что позволяет повысить переносимость ОС до максимума. Таким образом, микроядро реализует только жизненно важные функции, лежащие в основе операционной системы, являющиеся базисом для всех системных служб, сервисов и прикладных программ.

Использование механизма передачи сообщений позволяет установить единый интерфейс для взаимодействия между всеми компонентами системы, независимо от их уровня и назначения, что дает возможность строить все информационные связи в системе по модели клиент/сервер.

В модели клиент/сервер все компоненты рассматриваются либо как потребители (клиенты), либо как поставщики (серверы) некоторых ресурсов или сервисов. Стандартизованные протоколы предоставления сервиса или ресурсов позволяют серверу обслуживать клиентов независимо от деталей их реализации, что открывает перед разработчиками широкие возможности для построения распределенных систем. Инициатором обмена обычно является клиент, который посылает запрос на обслуживание серверу, находящемуся в состоянии ожидания запроса. Один и тот же процесс может являться клиентом по отношению к одним ресурсам, и быть сервером для других. Данная модель успешно применяется не только при построении ОС, но и при создании программного обеспечения любого уровня. Применение модели клиент/сервер по отношению к ОС состоит в реализации не вошедших в состав ядра компонентов ОС, в виде множества серверов, каждый из которых предназначен для обслуживания определенного ресурса (например, управление памятью, процессами, контроль доступа и т. д.).

Наиболее полно раскрыть преимущества технологии клиент/сервер позволяет применение методов объектно-ориентированного проектирования и программирования. Если каждый сервер обслуживает только один тип ресурсов и представляет его клиентам в виде некоторой абстрактной модель, то такой сервер можно рассматривать как объект, т. к. он обладает всеми необходимыми для этого качествами. Согласно Г. Буч[12] объект должен обладать состоянием, поведением и индивидуальностью. Действительно, каждый сервер выполняется в виде отдельного процесса и поэтому обладает индивидуальностью. Для каждого сервера существует четко определенная модель состояний и переходов между ними. И, наконец, "поведение" каждого сервера однозначно регламентируется протоко-

лом его взаимодействия с клиентами. Соответственно, можно строить модель ОС, построенной по этим принципам, в виде иерархии серверов и моделей представляемых ими ресурсов, а также описывать существующие между ними взаимосвязи с помощью объектных отношений наследования, использования и включения[12]. С точки зрения создания защищенных операционных систем, использование объектно-ориентированного подхода в сочетании с микроядром и технологией клиент/сервер позволяет разработчику реализовать взаимодействие субъектов и объектов, а также контроль за информационными потоками с помощью ограниченного числа простых и понятных механизмов, что облегчает адекватность реализации модели безопасности и позволяет применять формальные методы анализа.

5.2.1.2. Микроядерная архитектура с точки зрения создания защищенных систем

Благодаря принципам, на которых основаны микроядерные ОС, их компоненты функционируют на основе очень небольшого и сравнительно простого набора абстракций, составляющих базис системы и компактно реализованных в микроядре. Можно сказать, что для защищенных систем такая архитектура является оптимальной, т. к. она позволяет достаточно просто и эффективно разрешить целый ряд вопросов, неизбежно возникающих при реализации защищенных систем:

1. Выявление потоков информации в системе. Поскольку все взаимодействия осуществляются исключительно посредством механизма передачи сообщений, очевидно, что контролируя потоки сообщений, можно быть уверенным в том, что контролируются все информационные потоки в системе;

2. Определение субъектов и объектов взаимодействия. Как уже говорилось, все задачи в микроядерных системах связаны между собой отношениями клиент-сервер. Соответственно, субъектом взаимодействия всегда является задача-клиент, а объектом — ресурс, обслуживаемый задачей-сервером;

3. Размещение подсистемы контроля доступа. Поскольку единственным механизмом взаимодействия является передача сообщений, очевидно, что функция контроля за информационными потоками должна быть возложена на ту часть ядра системы, которая реализует этот механизм. Контроль и управление доступом к ресурсам и объектам могут быть реализованы, как в составе серверов, отвечающих за обслуживание этих ресурсов и объектов, так на уровне всей системы в целом. Так как, многие сервера обслуживают однотипные ресурсы и объекты (файлы, устройства и т. д.), то контроль доступа к ним с целью унификации реализуется на глобальном уровне, с помощью Сервера имен, который формирует

глобальное пространство имен системы и организует взаимодействие между клиентами и серверами.

4. Минимизация объема программного кода, отвечающего за контроль доступа. Как следует из предыдущего пункта, в системе существует всего две процедуры, реализующие контроль за осуществлением доступа: на уровне передачи сообщений и глобальная система на уровне именованных объектов. Таким образом, объем программ, корректность функционирования которых критична для безопасности всей системы, сокращен до минимума.

5. Использование объектно-ориентированных технологий программирования. Контроль за потоками сообщений и доступом процессов к ресурсам из глобального пространства имен можно осуществлять на основе унифицированного набора свойств сообщений (источник, приемник) и ресурсов (идентификатор процесса, имя ресурса). За счет этого достигается абстрагирование системы защиты от специфики информационных взаимодействий, но в то же время сохраняется ее гибкость за счет возможности использования в задачах-серверах специализированных механизмов защиты, адаптированных и конкретизированных к ресурсам, которые обслуживают эти сервера.

6. Верификация и анализ защиты. Достигнутая с помощью применения описанных решений простота и компактность средств контроля за осуществлением доступа очевидным образом, за счет структуризации системы (см. 5.1.3) способствует применению формальных методов верификации и анализа программного кода средств защиты.

5.2.1.3. Микроядро как основа для создания защищенной ОС нового поколения —MK++

Микроядро MK++ является объектно-ориентированным ответвлением большого проекта Mach по развитию микроядерных ОС, разрабатываемого известной организацией Open Software Foundation (OSF), и отвечает всем требованиям, предъявляемым к ОС нового поколения (многопоточность, расширяемость, мобильность и т. д.)[13]. Разработчики этого проекта преследовали следующие цели:

- отработать технологии реализации современных требований к операционным системам;
- создать основу для разработки будущих поколений защищенных ОС, рассчитанных на достаточно высокий класс требований безопасности;
- обеспечить возможность работы микроядра и приложений в режиме реального времени;
- предусмотреть максимальное использование параллельности как для приложений, так и для самих компонентов операционной системы в расчете на распределенные и массовопараллельные системы будущего.

- обеспечить переносимость микроядра с одной аппаратной платформы на другую;
- обеспечить совместимость с существующим программным обеспечением.

В рамках данного раздела мы рассмотрим этот проект только с точки зрения создания основы для защищенной ОС.

Поскольку разработчики MK++ стремились создать классическую микроядерную систему, в которой практически все традиционные для ОС функции вынесены за пределы ядра, само микроядро осуществляет только следующий набор функций[13]:

- управление физической аппаратурой (оперативной памятью, процессорами, внешними устройствами и т.д.);
- распределение ресурсов аппаратной платформы между процессами (время процессора, память и т. д.);
- изоляция процессов;
- организация взаимодействия между процессами;
- управление процессами (создание, уничтожение, переключение).

Ядро, таким образом, является своеобразным арбитром, роль которого сводится к поддержанию некоторого набора "правил игры" внутри операционной системы, все остальные традиционные функции ОС должны быть реализованы вне ядра.

Для описания архитектуры микроядерной необходимо ознакомить читателя с набором основных понятий, используемых в микроядерной технологии построения ОС:

Задача. В микроядерных системах это понятие заменяет традиционное для ОС понятие *процесс.* Задача представляет собой обобщение понятия процесс и обозначает набор ресурсов, который образует среду для выполнения *потоков* (см. далее). Эта среда включает в себя:

- изолированное от других задач адресное пространство;
- среду выполнения прикладного процесса;
- атрибуты безопасности;
- средства взаимодействия с ядром;
- средства взаимодействия с другими задачами.

Каждая задача имеет свое собственное пространство имен *портов* (см. ниже). Задача может выступать в роли потребителя ресурсов (клиент), или предоставлять определенные ресурсы другим задачам (сервер). Одна и та же задача может являться одновременно и сервером и клиентом, потребляя ресурсы, контролируемые одними задачами, и предоставляя свои ресурсы другим. Доступ к ресурсам, как и взаимодействие с другими задачами и ядром, осуществляется только с помощью обмена сообщениями через порты.

Поток (Thread) – логически связанный поток выполняемых команд. Каждый поток выполняется в контексте какой-либо *задачи* и может осуществлять непосредственный доступ только к ее среде. Потоки являются основной единицей вычислений и единственными активными элементами в системе. Поток представляет собой последовательность команд, выполняемых в рамках задачи. Его единственным атрибутом является состояние процессора. Все потоки внутри задачи совместно используют адресное пространство и наследуют атрибуты безопасности задачи.

Порт - однонаправленный коммуникационный канал, с помощью которого задачи обмениваются информацией друг с другом и с ядром осуществляя операции посылки и получения *сообщений*. Задачи могут получить доступ к портам только при наличии у них прав на посылку/прием сообщений.

Сообщение - логически связанный набор данных, передаваемый через порт за одно обращение. Для осуществления контроля доступа ядро снабжает все сообщения специальной меткой, идентифицирующей отправителя сообщения.

Пример взаимодействия между сервером, клиентом и микроядром с помощью портов и сообщений показан на рис. 5.11.

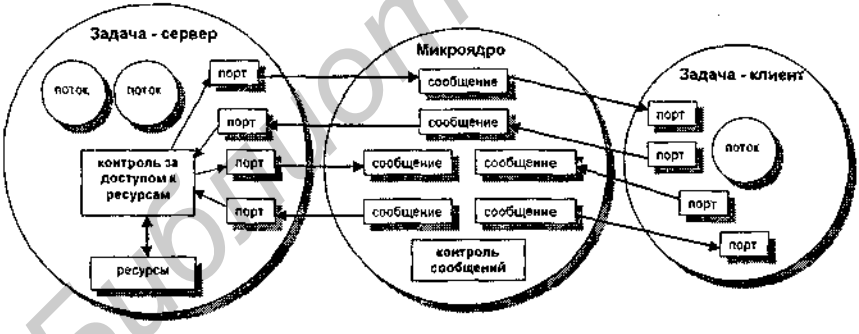


Рис. 5.11 Пример взаимодействия между сервером, клиентом и микроядром.

Рассмотрим внутреннюю структуру MK++, назначение и основные функции составляющих его компонентов[13]. Как видно из рис. 5.12, структура MK++, представляет собой иерархию, в которой каждый уровень опирается на сервисы, реализуемые нижестоящими уровнями, и, в свою очередь, обслуживает уровни лежащие выше.

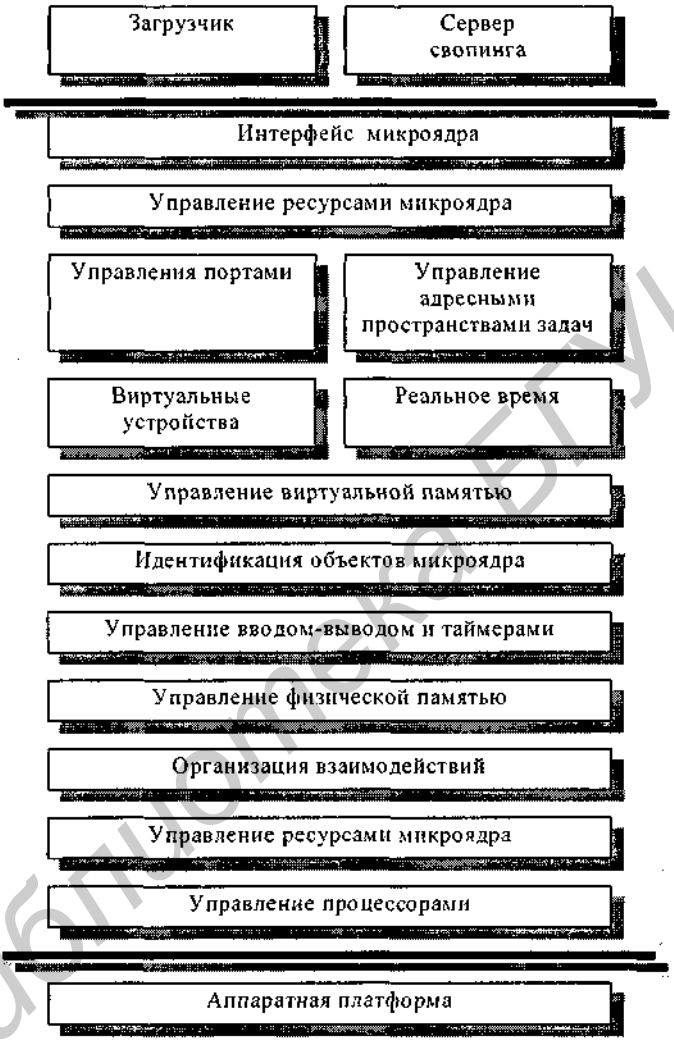


Рис. 5.12 Внутренняя структура микроядра МК++

Для того, чтобы пояснить как функционирует микроядро, рассмотрим в общих чертах тот, сервис, который реализуется компонентами МК++ и некоторыми задачами обеспечивающими важные для функционирования системы функции.

1. Подсистема управления процессорами представляет собой самую низкоуровневую и в наибольшей степени зависимую от аппаратной платформы подсистему во всей структуре микроядра. Подсистема управления процессорами контролирует все переходы между различными режимами работы процессоров, в том числе изменение уровня привилегий и переходы между различными кольцами защиты, а также переключение с одного потока команд на другой и также обработку прерываний и исключений. Данный уровень скрывает от вышестоящих все особенности аппаратной архитектуры, такие как порядок сохранения и восстановления регистров, формат таблиц трансляции адресов, способы управления приоритетами прерываний и т. д.. Таким образом, подсистема управления процессорами реализует следующие функции:

- управление состоянием процессоров;
- обработка аппаратных прерываний и исключений;
- синхронизация совместной работы нескольких процессоров;
- предоставление процессоров в распоряжение потоков и контроль за использованием процессорного времени.

2. Управление ресурсами микроядра. Служебные структуры (сообщения, порты и т. д.) требуют выделения определенных ресурсов (памяти), и при этом должны быть доступны как для приложений, так и из микроядра, что влечет за собой необходимость размещения их в адресном пространства микроядра. Подсистема управления ресурсами памяти микроядра отвечает за выделение памяти в адресном пространстве микроядра, ее освобождение и контроль за использованием.

3. Подсистема организации взаимодействий реализует для вышележащих уровней сервис, позволяющий посылать и принимать элементы данных. Данная подсистема включает в свой состав универсальные механизмы, которые могут быть использованы для передачи абстрактных элементов (непосредственно или с буферизацией в очереди) от некоторого источника к некоторому приемнику. Тип элементов не имеет значения. Самым распространенным примером использования этого сервера является механизм сообщений, обслуживающий информационный обмен между задачами и ядром однако, этот сервис может быть использован и для других целей.

4. Подсистема управления физической памятью отвечает за порядок распределения и использования физической памяти системы. Кроме того, данная подсистема выполняет машинно-зависимые операции трансляции адресов для различных адресных пространств.

5. Подсистема ввода-вывода выполняет низкоуровневый ввод/вывод информации, специфичный для каждого конкретного устройства. Кроме того, эта подсистема управляет аппаратными таймерами, а также осуществляет обработку прерываний от внешних устройств.

6. Подсистема идентификации ставит в соответствие каждой сущности уровня микроядра (задачи, потоки, порты и т. д.) уникальный идентификатор. Идентификатор никогда не используется повторно, даже если объект, на который он ссылается уничтожен. Эта подсистема обеспечивает механизм взаимодействий на уровне всей системы. Все манипуляции с объектами микроядра опираются на этот сервис.

7. Подсистема виртуальной памяти отвечает за отображение виртуальных адресных пространств ядра и задач в физическую память и обеспечивает инициализацию и очистку объектов памяти. Кроме того, эта система реализует совместное использование памяти и осуществляет операцию виртуального копирования фрагментов памяти между адресными пространствами задач.

8. Подсистема реального времени обеспечивает функции работы с сигналами и таймерами и осуществляет доступ к аппаратным часам.

9. Подсистема управления виртуальными устройствами поддерживает абстракции, представляющие их на уровне задач и отображает работу с ними в операции с реальными физическими устройствами. Данная подсистема ответственна за присваивание имен и реализацию операций открытия/закрытия для всех устройств.

10. Подсистема управления адресными пространствами организует виртуальные адресные пространства задач и пользуется сервисом подсистемы виртуальной памяти для манипулирования областями памяти, находящимися в адресном пространстве задач. Эта подсистема также осуществляет низкоуровневое управление доступом к пространствам памяти.

11. Подсистема управления портами отвечает за организацию пространства имен портов для задач и осуществляет отображение имен портов во внутренние идентификаторы микроядра с помощью подсистемы идентификации. Именно эта подсистема реализует контроль за информационными потоками и определяет, обладает ли задача правом на прием сообщений из порта и посылать в него сообщения.

12. Подсистема управления ресурсами обеспечивает высокоуровневые средства управления ресурсами ядра, в частности задачами и потоками. Управление состоит в отслеживании событий, связанных с ресурсами, и манипуляциях с ними, например, создание, завершение и приостановка выполнения потоков. Специальный механизм данной подсистемы позволяет реализовать различные политики управления ресурсами ядра.

13. Интерфейс микроядра определяет границу между выполнением потоков в контексте ядра и в пользовательском контексте. Подсистема управления процессорами осуществляет переключение режимов процессора таким образом, что вход в контекст микроядра из прикладной задачи и выход из него возможен только через интерфейс микроядра.

14. Сервер загрузки находится вне микроядра и представляет собой обычную пользовательскую задачу, создаваемую процессом инициализа-

ции ОС. Целью этой задачи является загрузка и запуск серверов, реализующих системные сервисы, в том числе входящих в состав ТСВ.

15. Сервер свопинга также является обычной задачей. Он обеспечивает сохранение на диске областей виртуальной памяти, вытесненных из физической памяти.

Таким образом в самом микроядре отсутствуют средства, непосредственно отвечающие за реализацию политики безопасности и высокоуровневого управления доступом к информационным ресурсам системы. Однако, MK++ включает два механизма, необходимые для реализации этих функций, а именно: изоляцию задач и контроль за передачей сообщений. Все остальные механизмы защиты, опирающиеся на этот сервис, могут быть реализованы в составе серверов. В совокупности микроядро и эти серверы образуют ТСВ системы.

Теперь, когда мы рассмотрели в общих чертах особенности архитектуры микроядерных ОС вообще и MK++ в частности, можно перейти к изложению принципов и методов построения защищенных микроядерных операционных систем.

В этой главе все внимание сосредоточено на реализации политики безопасности, средствах управления доступом и механизмах контроля за его осуществлением. Сначала мы изложим правила политики безопасности в общем виде, затем рассмотрим механизмы контроля доступа, потом конкретизируем правила политики безопасности для всех видов взаимодействий и, наконец, представим формальную модель безопасности.

5.2.2. Управление доступом

Управление доступом включает в себя как теоретический аспект в виде политики управления доступом, так и практические методы реализации средств контроля за ее осуществлением. Рассмотрим каким образом управление доступом реализуется в микроядерных операционных системах.

В этом разделе будут рассмотрены общие принципы, на которых строится управление доступом в микроядерных ОС, конкретные алгоритмы управления доступом и методы реализации подсистем контроля за его осуществлением доступа в микроядерных ОС. В качестве примеров будут рассмотрены средства контроля доступа Trusted Mach[14]. Первым шагом на пути построения подсистемы управления доступом является определение политики управления доступом

5.2.2.1. Политика управления доступом

Напомним, что основная задача политики управления доступом — сформировать совокупность норм и правил, регламентирующих порядок обработки информации в системе, с целью предотвратить несанкциониро-

ванный доступ к информации, влекущий ее разглашение или модификацию. Управление доступом в операционных системах базируется на традиционной концепции контроля доступа активных компонентов системы (субъектов) к пассивным (объектам). Основным механизмом контроля доступа является монитор взаимодействий, в обход которого осуществить доступ невозможно. Напомним читателю некоторые определения из области информационной безопасности компьютерных систем, и уточним их формулировки для микроядерных ОС.

Пользователями будем называть индивидуумов, имеющих право доступа к системе. Для реализации нормативного управления и контроля доступа с каждым пользователем связан определенный диапазон уровней безопасности. Чтобы пользователь мог получить доступ к системе, его диапазон уровней безопасности не должен быть пустым. После успешной идентификации и аутентификации пользователь может создавать *субъекты*, действующие от его имени. При этом пользователь указывает уровень безопасности этих субъектов, но только в рамках отведенного ему диапазона уровней.

Субъект представляет собой совокупность задач (обычно запущенных одним и тем же пользователем) с одинаковыми атрибутами безопасности. Каждой задаче присваивается специальный идентификатор безопасности, который служит указателем на атрибуты безопасности субъекта, которого представляет эта задача. Другими словами, все задачи с одним и тем же идентификатором безопасности выступают как один субъект и не различаются монитором взаимодействий в ходе осуществления контроля доступа.

Субъекты являются единственными активными элементами системы. Каждому субъекту в момент создания присваивается идентификатор безопасности. Идентификатор безопасности каждого субъекта остается неизменным в течение всего времени существования субъекта и ссылается на атрибуты безопасности субъекта. Как будет показано далее, атрибуты безопасности субъектов включают максимальный уровень безопасности объектов, к которому субъекту разрешено иметь доступ, идентификатор пользователя, от имени которого действует субъект, и список групп, членом которых является этот пользователь.

Взаимосвязь между пользователями, субъектами и атрибутами безопасности для Trusted Mach показана на рис. 5.13 [14].

Объект — это разделяемая область памяти, которая может совместно использоваться несколькими субъектами. Считается, что память используется совместно, в том случае, когда действия одной задачи с этой областью памяти могут быть замечены другой задачей, т. е. через эту область памяти может происходить обмен информацией между двумя задачами. Сервер имен связывает с каждым именованным объектом уникальное имя в виде строки символов, включает его в пространство имен и соз-

дает список прав доступа и метку. Например, в Trusted Mach существуют следующие разновидности объектов:

- файлы;
- *типы;*
- каталоги;
- ссылки;
- соединения;
- -устройства.

При рассмотрении политики безопасности мы рассмотрим специфику управления доступа ко всем типам объектов.

Монитор взаимодействий определяет атрибуты безопасности субъекта и объекта и на основании правил управления доступом, заданных *политикой безопасности, решает разрешать доступ,* или нет.

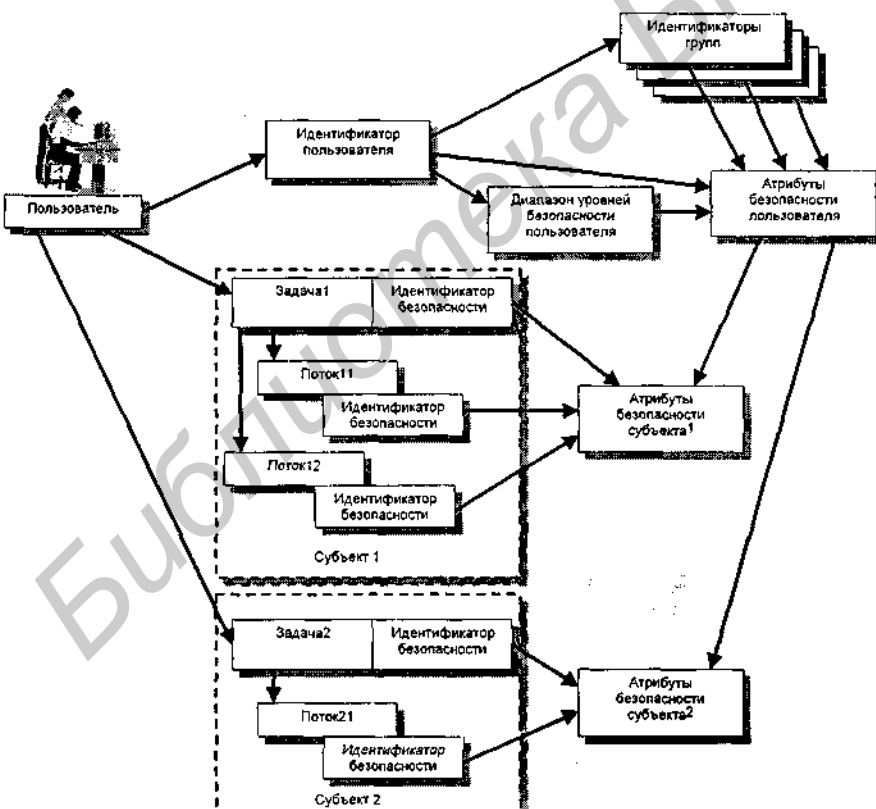


Рис. 5.13 Связь между пользователями субъектами и атрибутами безопасности.

В операционной системе имеет смысл поддерживать пять режимов (прав) доступа к объектам: чтение, запись, добавление, создание объектов и управление доступом. Эти режимы имеют следующий смысл:

- право на чтение подразумевает возможность просматривать, но не модифицировать информацию, содержащуюся в объекте;
- право на запись означает возможность просматривать и модифицировать содержимое объекта;
- право на добавление подразумевает возможность модифицировать содержимое объекта, но без возможности чтения (т. н. "слепая запись");
- право создания объекта означает возможность создавать объект;
- право управления доступом подразумевает возможность уничтожать объект и изменять права доступа к ним в соответствии с политикой произвольного управления доступом.

Как следует из требований, на которые мы ориентируемся, в защищенной ОС должны использоваться две политики управления доступом: нормативного управления доступом на основе присвоения объектам и субъектам меток безопасности и политика произвольного управления доступом.

Политика нормативного управления доступом является определяющей — сначала контроль доступа должен осуществляться в соответствии с ее правилами, и только в том случае, если разрешение на доступ получено, происходит проверка прав доступа в соответствии с политикой произвольного управления доступом. Для реализации нормативного управления доступом в состав атрибутов безопасности каждого объекта и каждого субъекта вводят метку безопасности, которая содержит уровень секретности, принадлежащий полностью упорядоченному множеству уровней. На множестве меток определим отношение доминирования, являющееся отношением частичного порядка. Субъект может осуществлять чтение информации из объекта только в том случае, если метка субъекта доминирует над меткой объекта. Субъект может добавить данные в объект (это не подразумевает возможность чтения), если метка объекта доминирует над меткой субъекта. Как уже говорилось, все задачи, действующие от имени пользователя, обладают его меткой безопасности. Сформулируем правила политики безопасности более строго:

1. Для того чтобы субъект получил к объекту доступ по чтению, уровень безопасности субъекта должен доминировать над уровнем безопасности объекта. Кроме того субъект должен иметь дискреционные полномочия на чтение этого объекта.

2. Для того чтобы субъект получил доступ записи к объекту, уровень безопасности субъекта должен совпадать с уровнем безопасности

объекта. Кроме того субъект должен иметь дискреционные полномочия на запись в этот объект.

3. Для того чтобы субъект получил возможность добавить данные в объект, уровень безопасности объекта должен доминировать над уровнем безопасности субъекта. Кроме того субъект должен иметь дискреционные полномочия на добавление данных в этот объект.

4. Для того чтобы субъект мог осуществлять управление доступом к объекту, субъект и каталог, в котором содержится этот объект, должны иметь один и тот же уровень безопасности. Кроме того, политика произвольного управления доступом должна разрешать субъекту управление доступом для этого объекта.

5. Субъект может создавать объекты, уровень безопасности которых доминирует над его уровнем безопасности. Кроме того, субъект должен иметь дискреционные полномочия на создание объектов этого типа.

Данная политика управления доступом реализует соответствующие требования стандартов информационной безопасности и отражает правила обработки документов, содержащих категорированную информацию.

Для реализации политики произвольного управления доступом расширим состав атрибутов безопасности. Кроме метки безопасности каждому объекту поставим в соответствие список прав доступа, а каждому субъекту — идентификатор управления доступом. Этот идентификатор образуется путем комбинации идентификаторов пользователя и групп, членом которых он является. Каждый пользователь может входить в состав одной или нескольких групп. Каждый субъект, действующий от имени пользователя, обладает идентификатором управления доступом, в соответствии с идентификаторами пользователя и групп, к которым принадлежит пользователь.

Например, в Trusted Mach с каждым объектом ассоциирован список прав доступа (Access Control List - ACL), который содержит имена пользователей и групп с указанием их прав доступа к данному объекту. Элементы списка могут определять как разрешение прав доступа, так и их запрещение. Формат элементов списка имеет следующий вид:

```
<тег>:<идентификатор>:<разрешенные права>:<запрещенные права>
```

Где:

Тег - одно из ключевых слов "user", "group", "all", означающих, соответственно, что данная запись относится к отдельному пользователю, определенной группе, или ко всем пользователям системы;

Идентификатор — в зависимости от значения поля *тег,* либо идентификатор пользователя (в поле *тег* указано "user"), либо идентификатор группы (в поле *тег* стоит "group"), либо символ "*"(поле *тег* содержит "all");

Разрешенные права доступа - список разрешенных прав доступа (*none* означает пустой список);

Запрещенные права доступа - список запрещенных прав доступа (*none* означает пустой список);

Например:

```
user : john : read, write : none
group : programmers : read, write : none
group : users : none : read
all: * : read : none
```

Это означает, что:

- для пользователя john разрешен доступ как по чтению, так и по записи;
- для пользователей группы programmers разрешен доступ как по чтению, так и по записи;
- для пользователей группы users запрещен доступ по чтению;
- для всех пользователей разрешен доступ по чтению.

Как видно даже из приведенного примера элементы списка прав доступа могут противоречить друг другу. Для разрешения конфликтов применяется следующий алгоритм обработки списка прав доступа (см. рис. 5.14):

1. Сначала в списке прав доступа ищутся элементы, для которых в поле <*идентификатор*> указано имя субъекта, запросившего доступ к объекту. Программа контроля доступа просматривает все записи подобного вида и комбинирует из них индивидуальные наборы разрешенных и запрещенных прав доступа. Если хотя бы **одно** из запрашиваемых прав доступа присутствует в списке запрещенных прав, то в доступе отказывается. Таким образом, явное запрещение доступа имеет больший приоритет, чем явное разрешение. Если **все** без исключения запрашиваемые права доступа входят в набор разрешенных прав, то доступ разрешается. Таким образом, индивидуальное управление доступом имеет больший приоритет, чем на уровне групп и шаблонов. Если **ни одно** из запрашиваемых прав не присутствует в наборе запрещенных, но **не все** запрашиваемые права указаны в наборе разрешенных, то вступает в силу управление доступом на уровне групп.

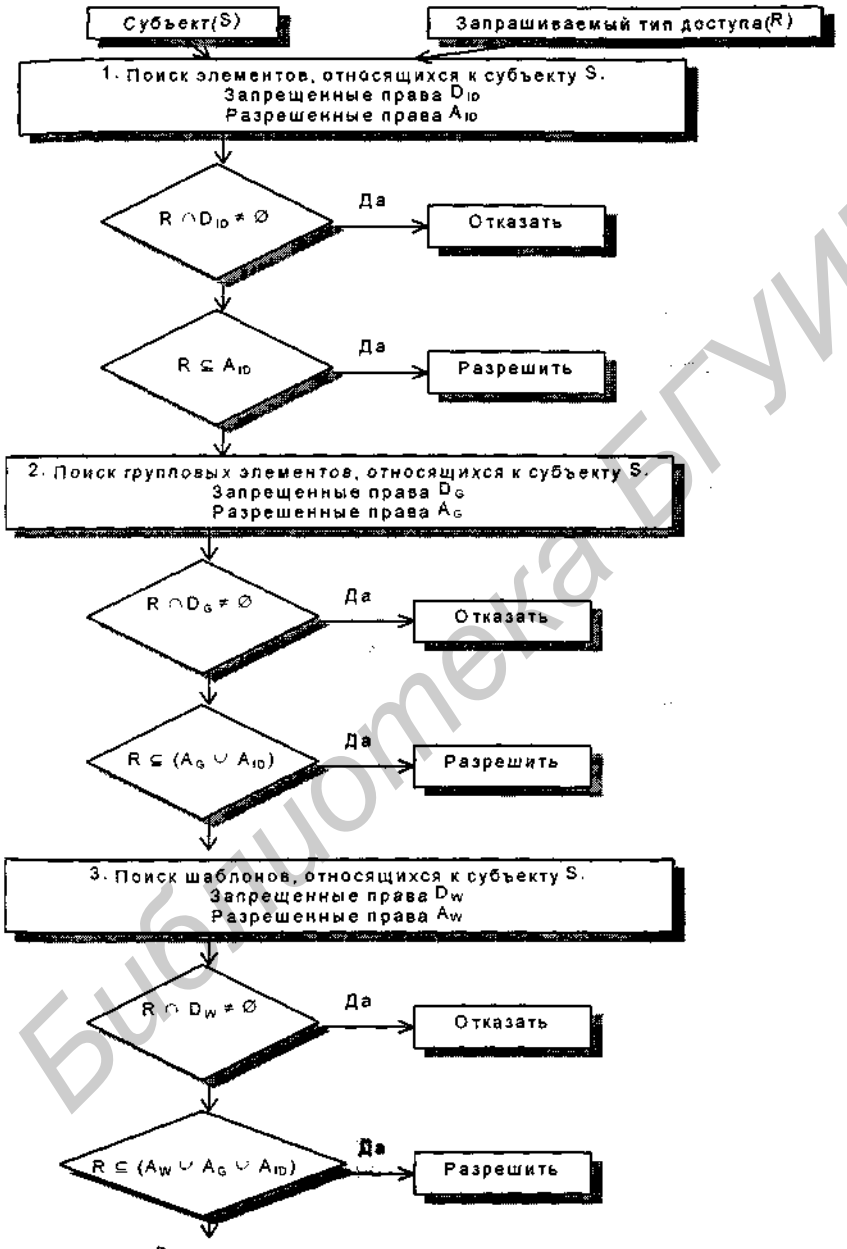


Рис. 5.14 Алгоритм поиска прав в списке прав доступа.

2. В списке прав доступа ищутся элементы, для которых в поле *<идентификатор>* указан идентификатор группы, членом которой является субъект, запросивший доступ к объекту. Программа контроля доступа просматривает все записи подобного вида и комбинирует из них групповые наборы разрешенных и запрещенных прав доступа. К групповому набору разрешенных прав добавляется индивидуальный набор разрешенных прав, полученный на предыдущем этапе. Если хотя бы **одно** из запрашиваемых прав доступа присутствует в списке запрещенных прав, то в доступе отказывается. Таким образом, и для групп явное запрещение доступа имеет больший приоритет, чем явное разрешение. Если **все** без исключения запрашиваемые права доступа входят в набор разрешенных прав, то доступ разрешается. Таким образом, групповое управление доступом имеет больший приоритет, чем на уровне шаблонов. Если **ни одно** из запрашиваемых прав не присутствует в наборе запрещенных, но **не все** запрашиваемые права указаны в наборе разрешенных, то вступает в силу управление доступом на уровне шаблонов.

3. В списке прав доступа ищутся элементы, для которых в поле *<тег>* указан шаблон (all). Программа контроля доступа просматривает все записи подобного вида и комбинирует из них шаблоны разрешенных и запрещенных прав доступа. К шаблону разрешенных прав добавляются индивидуальный и групповой наборы разрешенных прав, полученные на предыдущих этапах. Если хотя бы **одно** из запрашиваемых прав доступа присутствует в шаблоне запрещенных прав, то в доступе отказывается. Для шаблонов также явное запрещение доступа имеет больший приоритет, чем явное разрешение. Если **все** без исключения запрашиваемые права доступа входят в шаблон разрешенных прав, то доступ разрешается. В противном случае, в доступе отказывается.

Таким образом, для того, чтобы субъект получил дискреционные полномочия на доступа к объекту, должны выполняться два условия:

- запрашиваемые права доступа должны отсутствовать во всех элементах списка контроля доступа, применимых к данному субъекту;
- запрашиваемые права доступа должны присутствовать хотя бы в одном элементе списка контроля доступа, который относится к данному субъекту.

Таким образом мы рассмотрели политику управления доступом, которая удовлетворяет достаточно жестким требованиям. Теперь рассмотрим методы практической реализации этой политики для микроядерных ОС.

5.2.2.2. Механизмы контроля за осуществлением доступа

Задача средств контроля за осуществлением доступа — обеспечить выполнение правил политики безопасности. В традиционных ОС кон-

троль доступа осуществлялся на двух уровнях — аппаратном (кольца защиты и изоляция адресных пространств) и общесистемном (контроль доступа к объектам файловой системы на уровне системных вызовов ядра). Преимущества технологий, применяемых при построении микроядерных ОС позволяют осуществлять многоступенчатый контроль за осуществлением доступа на нескольких уровнях взаимодействия. Для микроядерных ОС в иерархии средств контроля доступом можно выделить четыре уровня: аппаратный контроль, контроль на уровне микроядра, общесистемный и прикладной.

При этом высокоуровневые средства в своей работе опираются на базовые, что приводит к упрощению всей системы в целом и повышает гибкость управления доступом. Контроль доступа на нескольких уровнях дает возможность повысить гибкость и эффективность контроля т. к. позволяет:

- осуществлять контроль над информационными потоками между субъектами системы отдельно от контроля и управления доступом к объектам;
- реализовать в рамках одной ОС различные политики безопасности, описывающие управление доступом к объектам разной природы, без конфликтов и противоречий.

Цели, преследуемые на каждом уровне контроля доступа, методы их достижения и компоненты ОС, в которых размещены соответствующие средства, показаны в табл. 5.

Прикладной уровень реализует специализированные политики безопасности для доступа к ресурсам и объектам, которые обслуживают эти серверы. Это очень удобно для реализации специализированных политик безопасности, в частности для управления доступом к базам данных.

Системный уровень содержит средства управления доступом к информационным ресурсам системы, находящимся в глобальном пространстве имен(файлы, устройства, межпроцессное взаимодействие) и реализует механизмы произвольного и нормативного управления доступом.

Уровень микроядра обеспечивает базовые абстракции и механизмы контроля информационных потоков, которые используются системным и прикладным уровнями.

Аппаратной уровень обеспечивает элементарные механизмы защиты, которые невозможно преодолеть программными средствами.

Рассмотрим более подробно реализацию механизмов контроля доступа на различных уровнях архитектуры системы. Поскольку предметом данной главы является построение защищенной операционной системы, основное внимание будет уделено системному уровню и уровню микроядра. На уровне аппаратуры будут рассмотрены только механизмы, необходимые для понимания работы верхних уровней. Уровень приложений

заслуживает отдельного рассмотрения выходит за рамки данной книги. Будем рассматривать уровни в порядке от верхних к нижним, т. к. это позволяет продемонстрировать взаимосвязь между ними и понять каким образом низкоуровневые механизмы реализуют высокоуровневые функции контроля доступа.

Таблица 5. Многоуровневая реализация контроля доступа в микроядерных ОС

Уровень	Цели	Методы	Средства
Прикладной	Реализация специализированных политик безопасности для управления доступа к ресурсам приложений	Управление и контроль доступа к ресурсам в соответствии с правилами политики безопасности	Прикладные сервера, обслуживающие ресурсы
Системный	Реализации общей политики безопасности ОС	Управление атрибутами безопасности субъектов и объектов. Управление и контроль доступа субъектов к объектам в соответствии с правилами политики безопасности системы	Сервер имен
Микроядра	Контроль взаимодействий в системе	Управление потоками сообщений с помощью контроля за доступом к портам	Микроядро
Аппаратный	Изоляция субъектов и поддержка контроля за взаимодействиями	Управление адресными пространствами задач и режимами защиты процессора	Процессор и отдельные компоненты микроядра

5.2.2.2.1. Контроль доступа на уровне операционной системы

Основная задача этого уровня — обеспечить реализацию политики произвольного и нормативного управления доступом и мониторинг взаимодействий. На данном уровне все информационные потоки в системе представляются в виде взаимодействия между субъектами (задачами-клиентами) и объектами (ресурсами, которые обслуживают задачи-серверы). В микроядерных ОС функции монитора взаимодействий реализует Сервер имен, без обращения к которому доступ к объектам невозможен в принципе.

Все субъекты, действующие на этом уровне контроля доступа, представляют собой задачи, действующие от имени аутентифицированного пользователя и обладающие его атрибутами безопасности. Благодаря

разделению адресных пространств задач все субъекты изолированы друг от друга. Объекты этого уровня представляют собой абстракцию ресурсов, доступ к которым осуществляется через соответствующие сервера объектов (сервер файловых систем, сервер устройств и т. д.).

Механизм контроля доступа лучше реализовать отдельно от механизма осуществления доступа для того, чтобы, во-первых, сосредоточить функции контроля доступа в составе одного компонента системы и, во-вторых, унифицировать процедуру контроля доступа для всех типов объектов. Проиллюстрируем данные положения примером работы соответствующих компонентов Trusted Mach, в котором контроль доступа на этом уровне в несколько упрощенном виде можно представить следующим образом [14] (см. рис. 5.15):

Рис. 5.15 Схема осуществления контроля доступа на уровне ОС.

При создании каждой задачи ей присваивается идентификатор безопасности, который указывает на атрибуты безопасности субъекта, с которым ассоциирована данная задача.

Когда задача стремится осуществить доступ объекту, она обращается к Серверу имен с соответствующим запросом, в котором содержатся имя объекта и запрашиваемый тип доступа(1). Отметим, что тип доступа зависит от объекта, но для контроля доступа это не имеет значения. Подробнее специальные типы доступа и операции над объектами будут рассмотрены при подробном изложении политики безопасности.

Контроль доступа. Сервер имен проводит поиск указанного имени в пространстве имен, если указанное имя найдено, определяет идентификатор объекта, которому присвоено это имя, а также идентификатор сервера, который реализует доступ к этому объекту (например, доступ к файлам осуществляет сервер файловых систем). После этого осуществляется контроль доступа в соответствии с изложенными в п. 5.2.2.1 правилами нормативного и произвольного управления доступом. В том случае, если

запрашиваемый тип доступа не противоречит политике безопасности и может быть предоставлен, Сервер имен создает специальный объект — *Дескриптор объекта*, в который помещает идентификатор безопасности запросившего доступ субъекта, идентификатор объекта и типы доступа, разрешенные для данного субъекта к этому объекту(2). После этого Сервер имен назначает задаче-субъекту право записи в порт, представляющий Дескриптор объекта, и передает ему это право(3). Затем Сервер имен посылает серверу, обслуживающему этот объект право на чтение из порта, представляющего Дескриптор объекта(4).

После того, как субъект получил от Сервера имен разрешение на доступ в виде права на доступ к порту, представляющему Дескриптор объекта, он может осуществлять разрешенные ему действия над объектом путем посылки сообщений, содержащих соответствующие команды работы с объектом и их параметров, в порт Дескриптора объекта(5).

Осуществление доступа. Сервер, обслуживающий объект, получает запросы субъекта через Дескриптор объекта, контролирует правомочность запрашиваемых действий над объектом и подлинность субъекта путем сравнения идентификатора отправителя сообщения и типа запрашиваемой операции с информацией указанной в Дескрипторе объекта(6). Если все проверки закончились успешно, сервер осуществляет доступ к объекту(7) и пересылает или получает информацию от субъекта(8).

Представленная схема контроля доступа обладает следующими преимуществами по сравнению с традиционной, принятой в большинстве ОС:

- контроль доступа к объектам осуществляется на основе унифицированного подхода, т. к. механизм контроля доступа не зависит от типа объектов, доступ к которым он контролирует;
- раздельная реализация механизмов контроля доступа и его осуществления позволяет осуществлять централизованное управление доступом, а не растаскивать его по различным подсистемам, и приводит к минимизации объема программ, отвечающих за контроль доступа, что упрощает тестирование и анализ.

5.2.2.2.2. Уровень микроядра

Основное назначение средств контроля доступа уровня микроядра — обеспечить механизмы, с помощью которых могут быть реализованы средства контроля доступа на уровне операционной системы. Основной механизм контроля доступа, на котором базируются все средства вышележащих уровней, — это контроль за доступом к портам, через которые осуществляются все взаимодействия.

Как уже говорилось порты являются универсальным механизмом для осуществления взаимодействий путем посылки сообщений. Фактиче-

ски, порт — это адрес, на который отправляется, или из которого считывается сообщение. С каждым портом ассоциирована очередь сообщений, проходящих через него. Порт не является объектом, а очередь сообщений порта является частью той задачи, которой они передаются. Микроядро снабжает все сообщения, посланные в порты идентификатором отправителя, благодаря чему задача, получившая сообщение всегда знает его отправителя, причем поскольку этот идентификатор присваивается микроядром подделать его невозможно.

В микроядерных системах задачи-сервера используют порты для представления доступа к обслуживаемым ими объектам. Клиенты осуществляют доступ к объектам путем посылки/получения сообщений в/из портов, представляющих объекты. Например, сервер файловых систем предоставляет доступ к объектам, которые он обслуживает — файлам, посредством портов, которые являются представлением файлов для задач-клиентов. Этот сервер обслуживает все порты, представляющие файлы, получает сообщения от клиентов и обрабатывает их запросы. Поскольку для микроядерных систем отношения клиент-сервер охватывают все компоненты системы то все объекты таких систем реализуются серверами, использующими порты в качестве средств, с помощью которых клиенты получают доступ к объектам.

Доступ к портам контролируется правами порта, которыми микроядро наделяет задачи. Для того, контролировать доступ к портам в соответствии с распределением прав портов, ядро хранит для каждой задачи доступное для нее пространство прав портов в своих внутренних структурах, так что задачи не могут их подделать. Права портов могут передаваться от одной задачи к другой, однако микроядро контролирует все передачи прав портов.

Существует два основных права на доступ к портам: право получения позволяет получать сообщения, посланные в этот порт, а право отправления позволяет посылать в порт неограниченное число сообщений. Сообщение может состоять из обычных данных и набора прав порта. Всякий раз, когда задача делает запрос на выделение ей порта, ядро создает порт и назначает задаче право получения для этого порта. После того, как задаче предоставлено право получения, она может создавать для него другие права, например, право отправления.

Кроме того, микроядро обеспечивает контроль за взаимодействием с внешними устройствами ввода-вывода. Доступ к каждому устройству осуществляется через порт — т. е. операция на устройстве реализуется через отправление сообщения на порт, представляющий это устройство.

Например, микроядро MK++ реализует перечисленные функции следующим образом [13]:

- микроядро снабжает все сообщения, посланные от одной задачи к другой идентификатором отправителя сообщения;

- все сообщения, посылаемые микроядром, снабжаются идентификатором микроядра;
- микроядро разрешает задаче доступ к порту только при условии наличия у задачи соответствующих прав на доступ к этому порту;
- микроядро не разрешает задаче отображать объекты в свое адресное пространство, если метка безопасности этого объекта не совпадает с меткой безопасности задачи;
- микроядро запрещает доступ к объекту памяти, если сервер, отвечающий за работу с данным объектом памяти, не поддерживает требуемый режим доступа к объекту;
- микроядро запрещает операции над устройством, если задача, инициировавшая эту операцию, не обладает соответствующим правом для порта, представляющего устройство;
- микроядро запрещает все операции над устройством, если они не входят набор операций, поддерживаемых данным устройством;
- драйверы устройств микроядра поддерживают целостность данных, размещенных на связанных с ними устройствах.

5.2.2.2.3. Аппаратный уровень

В основе любой системы защиты (и тем более защищенной системы) должен лежать некоторый базовый механизм контроля доступа, преодолеть или обойти который не представляется возможным и надежность которого не вызывает сомнений. Наиболее эффективным решением является аппаратная реализация базовых функций контроля доступа, включенная практически во все современные процессоры. На базе этих функций строятся все остальные системы управления доступом, и именно аппаратный контроль обеспечивает надежность и корректность всех механизмов защиты. Базовые механизмы защиты, реализованные в современных процессорах, не слишком отличаются друг от друга (разве что у самых популярных в нашей стране процессоров Intel x86 они несколько усложнены и избыточны, что объясняется непростой историей развития данного семейства микропроцессоров).

Минимально необходимая аппаратная поддержка средств защиты должна включать: поддержку нескольких режимов с различным уровнем привилегий (как минимум два уровня), возможность организации изолированных адресных пространств и управление доступом к сегментам памяти.

1. Режимы с различным уровнем привилегий. Аппаратура должна обеспечивать два или более уровня привилегий для процессов — привилегированный режим для ТСВ и непривилегированный режим для прикладных процессов пользователя. Для функционирования систем защиты необходимо, чтобы существовал по крайней мере один режим, в котором

программы пользователя могли бы работать в контролируемой среде, т.е. там, где они были бы ограничены в возможностях доступа и "физически" не могли бы преодолеть ни одно из наложенных на них ограничений. В микроядерных системах в привилегированном режиме работает только микроядро, объем которого очень невелик, а функции сведены к управлению аппаратурой. Таким образом, саму микроядерную архитектуру ОС можно рассматривать как меру защиты. Однако, привилегированный код должен быть доступен и для непривилегированных программ. Это означает пользовательского процесса существует безопасный способ обращения к программе, работающей в привилегированном режиме. Например, процессоры семейства x86 поддерживают специальный механизм шлюзов межкольцевых вызовов, который позволяет решить эту проблему не нарушая разделение режимов и не оставляя для непривилегированных программ никаких шансов путем обмана или подтасовок получить привилегии.

2. Организация изолированных адресных пространств. Управление и контроль доступа строятся на обеспечиваемой аппаратурой возможностях создания изолированных адресных пространств, в которых функционируют процессы и задачи, представляющие различных субъектов. Для микроядерных ОС изоляция адресных пространств задач позволяет:

- гарантировать невозможность какого-либо взаимодействия задач с различными атрибутами безопасности (в тех случаях, когда такое взаимодействие необходимо, предусматриваются соответствующие механизмы, контролирующие вид взаимодействия и передачу информации в соответствии с политикой безопасности);
- запретить непривилегированным задачам доступ к данным, связанным с безопасностью;
- предотвратить случайный доступ различных компонентов системы (в т. ч. входящих в ТСВ) к информации, принадлежащей другим частям ТСВ, что позволяет реализовать требования стандартов безопасности в части внутреннего структурирования и архитектуры ТСВ.

3. Управление режимами доступа к памяти. Данная функция аппаратуры не участвует в контроля доступа как таковом (он базируется исключительно на изоляции субъектов и кольцах защиты), однако позволяет использовать аппаратные ресурсы более эффективно, без ущерба для безопасности. С помощью этого механизма адресное пространство задачи может быть разделено на отдельные блоки (страницы), каждый из которых может иметь свои собственные, отличные от других атрибуты доступа. Управление атрибутами таких блоков позволяет включать их одновременно в адресные пространства нескольких задач, когда это согласовано с политикой безопасности.

Все перечисленные аппаратные возможности современных процессоров позволяют предоставить каждой задаче отдельную контролируемую среду и организовать эффективный способ передачи информации между задачами тогда и только тогда, когда это позволяет политика безопасности. Все взаимодействия между задачами и контроль за их выполнением осуществляются входящим в ТСВ программным обеспечением, выполняющимся в привилегированном режиме.

5.2.3. Реализация политики безопасности в микроядреных ОС на примере Trusted Mach

Рассмотрим каким образом реализуется в микроядерных ОС рассмотренная в предшествующих разделах политика контроля и управления доступом. Обычно при изложении политики безопасности ограничиваются только очень общими правилами, регламентирующими порядок чтения и записи абстрактных объектов. Этот подход чреват ошибками и проблемами на стадии внедрения моделей безопасности (см. главу 3) в реальные системы, т. к. даже в такой системе как ОС, число типов объектов и количество операций в которой относительно невелико (например, по сравнению с базами данных) существуют операции создания и удаления объектов (файлов), просмотра каталогов и доступ к неименованным объектам. Поэтому на кажется полезным ознакомить читателя с политикой безопасности, описывающий функционирование реальной ОС — Trusted Mach[14,15]. Авторам приходилось сталкиваться со многими системами, но нигде политика безопасности не была так четко проработана, можно сказать что политика безопасности, представленная в такой форме публикуется на русском языке впервые.

Рассмотрим виды объектов, существующих в этой ОС, их атрибуты то как правила политики безопасности регламентируют доступ к объектам различных типов и условия выполнения операций над ними.

5.2.3.1 Типы и атрибуты объектов

Каждый именованный объект в Trusted Mach входит в системное пространство имен и является экземпляром некоторого типа(файл, каталог, тип, ссылка, соединение, устройство). С каждым объектом ассоциирован набор атрибутов, определяющих свойства и тип объектов. Для всех объектов общими атрибутами являются уровень безопасности, список прав доступа и время создания объекта. Атрибуты объектов подразделяются на две группы в зависимости от того как на них отражается изменение содержимого объекта: атрибуты, связанные с содержанием, и атрибуты, не связанные с содержанием. Атрибуты не связанные с содержанием не зависят от изменений объекта, а атрибуты, связанные с содержанием могут меняться в результате модификации содержимого объекта. Тип

объекта, его идентификатор, номер дескриптора (i-node), уровень безопасности и список прав доступа к объекту считаются не связанными с содержанием. Время модификации, время доступа, время последней модификации атрибутов и размер считаются связанными с содержанием. Атрибуты, независящие от содержания, имеют тот же уровень безопасности, что и каталог, в котором создан этот объект. Благодаря этому пользователь, уровень которого позволяет ему прочитать содержимое каталога, может узнать информацию об атрибутах объектов, содержащихся в каталоге, даже если у него нет полномочий на доступ к этим объектам. Ссылка на объект может быть создана только в каталоге, уровень которого совпадает с уровнем каталога, в котором объект был создан. Уровень безопасности атрибутов, несвязанных с содержанием, хранится вместе с этими атрибутами, что позволяет использовать его для контроля в процессе переименования объектов. Атрибуты зависящие от содержания имеют тот же уровень безопасности, что и объект, к которому они относятся.

Пространство имен системы организовано иерархически в виде дерева имен, вершинами которого являются объекты. Внутренние вершины этого дерева (содержащие другие вершины) относятся к типу "каталог". Таким образом каталоги представляют собой особый тип объектов — контейнеров, которые могут содержать другие объекты различных типов.

Каждый тип в свою очередь сам является объектом. Типы бывают двух видов: встроенные и пользовательские. Встроенные типы и операции над относящимися к ним объектам поддерживаются системными компонентами, входящими в состав ТСВ. Файлы, каталоги, типы, символические указатели и устройства являются встроенными типами. Пользовательский тип — это тип, созданный и обслуживаемый прикладными программами, не входящими в состав ТСВ. При создании нового пользовательского типа должны быть заданы следующие параметры типа:

- диапазон уровней безопасности типа, определяющий максимальный и минимальный уровни безопасности, допустимые для объектов данного типа. При создании объектов данного типа их уровень безопасности должен лежать в указанном диапазоне и впоследствии будет изменяться только в его рамках. Диапазон любого пользовательского типа ограничен одним уровнем, что позволяет запретить пользователям создавать прикладные средства, обслуживающие объекты с переменным уровнем безопасности, т. к. подобные объекты нуждаются в усиленном контроле и могут обслуживаться только средствами, входящими в состав ТСВ.
- множество видов доступа, допустимое для объектов данного типа. Это множество состоит из двух групп: общие операции, реализуемые и контролируемые Сервером имен, и частные операции, реализуемые сервером объектов данного типа.

- соответствие между множеством дискреционных прав доступа к данному объекту и общими правами модели нормативного управления доступом. Например, специфичная для каталога операция поиска в нем файла, соответствует чтению этого каталогу.

Наиболее распространенными и типичными объектами являются файлы. Ссылки используются в качестве механизма создания альтернативных имен объектов. Если Сервер имен в ходе поиска объекта встречает ссылку, то он осуществляет интерпретацию содержания этой ссылки и переходит к тому объекту на который она указывает. Внешние устройства управляются Сервером устройств, их уровень безопасности устанавливается администратором системы.

Тип "соединение" является пользовательским типом. Объекты этого типа представляют в пространстве имен объекты взаимодействия процессов и имеют точно такие же атрибуты (список прав доступа, уровень безопасности, время модификации и т.д.), как и любой другой объект.

5.2.3.2 Права доступа к объектам и операции над объектами

Рассмотрим реализацию описанной в п. 5.2.2.1 политики безопасности для объектов всех типов и всех видов операций над ними. Для каждой операции укажем набор условий, которые должны быть выполнены для ее осуществления. Эти условия включают как необходимость наличия у субъекта, инициировавшего операцию, соответствующих прав доступа к объекту, так и соблюдения некоторых дополнительных условий.

5.2.3.2.1. Манипуляции с атрибутами объектов

В Trusted Mach существует определенный набор операций, которые применимы к любому объекту независимо от его типа т. к. предназначены исключительно для манипуляций с атрибутами объектов. Для манипуляций с атрибутами Сервер имен реализует следующие операции:

- чтение атрибутов, связанных с содержанием. Для осуществления этой операции субъект должен обладать полномочиями, позволяющими читать объект — его уровень безопасности должен доминировать над уровнем безопасности объекта и он должен обладать дискреционным правом на чтение объекта.
- изменение атрибутов, связанных с содержанием. Для изменения атрибутов, связанных с содержанием объекта, субъект должен обладать полномочиями, позволяющими ему модифицировать сам объект.
- чтение атрибутов, не связанных с содержанием. Для чтения атрибутов объекта субъект должен иметь возможность осуществлять просмотр каталога, в котором находится объект.

- изменение атрибутов, не связанных с содержанием. Для модификации этих атрибутов уровень безопасности субъекта должен совпадать с уровнем безопасности каталога, содержащего объект и, кроме того, для модификации списка прав доступа субъект должен иметь дискреционные полномочия на управление доступом к объекту.

5.2.3.2.2. Общие операции доступа к объектам

Когда субъект стремиться получить доступ к объекту, он должен явным образом обратиться к Серверу имен, указав имя объекта и желаемый тип доступа. Сервер имен осуществляет контроль доступа и (в том случае, если это не противоречит правилам политики безопасности) субъект получает доступ к портам, через которые будет осуществляться обмен информацией с объектом. Кроме того есть всего три случая, когда Сервер имен осуществляет проверку полномочий субъекта без явного запроса со стороны субъекта. Сначала опишем правила политики контроля доступа при явном запросе со стороны субъекта, а затем расскажем о трех исключительных случаях доступа к объекту без явного запроса.

Чтение

Для того, чтобы субъект получил доступ по чтению к объекту, должны быть соблюдены следующие условия:

- уровень безопасности субъекта должен доминировать над уровнем безопасности объекта;
- список прав доступа объекта должен разрешать субъекту чтение;
- уровень безопасности субъекта должен входить в диапазон уровней безопасности типа, к которому принадлежит объект.

Запись

Для того, чтобы субъект получил доступ по записи к объекту, должны выполняться следующие условия:

- уровни безопасности субъекта и объекта должны совпадать;
- список прав доступа объекта должен разрешать субъекту запись;
- уровень безопасности субъекта должен входить в диапазон уровней безопасности типа, к которому принадлежит объект.

Добавление

Для того, чтобы субъект мог осуществлять добавление данных к объекту, должны быть выполнены следующие условия:

- уровень безопасности объекта должен доминировать над уровнем безопасности субъекта;
- список прав доступа объекта должен разрешать субъекту добавление данных;

- уровень безопасности субъекта должен входить в диапазон уровней безопасности типа, к которому относится объект.

Управление доступом к объектам (изменение списка прав доступа)

Для того, чтобы субъект получил возможность изменять список прав доступа объекта, должны быть выполнены следующие требования:

- уровень безопасности субъекта должен совпадать с уровнем безопасности каталога, в котором содержится объект;
- список прав доступа объекта должен разрешать субъекту управлять доступом к этому объекту.

Создание и уничтожение субъектов и объектов

В системе Trusted Mach субъекты (потоки, задачи) создаются только специальным Сервером субъектов. Доверенный командный процессор обеспечивает соответствие атрибутов безопасности субъектов атрибутам безопасности пользователей, от имени которых действует этот субъект. В том числе, гарантируется, что уровень безопасности субъекта входит в диапазон уровней безопасности пользователя и что дискреционные атрибуты субъекта совпадают с дискреционными атрибутами пользователя. Атрибуты безопасности субъектов не могут изменяться в течение всего времени их жизни, поскольку ни одна из операций не может их изменить.

Субъект может уничтожить другой субъект при условии, они оба имеют один и тот же уровень безопасности и которые принадлежат тому же самому уровню безопасности.

Субъект может создавать объекты и указывать для них уровень безопасности и список прав доступа. Необходимо выполнить лишь два условия: уровень безопасности объекта должен доминировать над уровнем безопасности субъекта, и субъект должен обладать достаточными полномочиями для создания ссылки на объект в каталоге, в котором он хочет его разместить. После того, как объект создан, его уровень безопасности остается неизменным в течение всего времени его существования.

Субъекты могут уничтожать объекты, посылая серверу имен соответствующий запрос на удаление объектов из каталогов. Перед тем как осуществить эту операцию Сервер имен проверяет, имеет ли субъект право удалять объекты в каталоге, содержащем этот объект.

Неявный доступ к объектам

Как уже упоминалось существуют три случая, когда осуществляется контроль доступа без явного запроса со стороны субъекта. Первый случай имеет место, когда Сервер имен осуществляет поиск имени объекта и совершает обход каталогов. При этом Сервер имен проверяет возможность субъекта на осуществление просмотра каталога. Второй случай

имеет место, когда тот же Сервер имен сталкивается в ходе поиска имени с ссылкой. В этом случае проверяется наличие у субъекта права чтения этой ссылки. И наконец, когда субъект посылает запрос на создание объекта определенного типа, Сервер имен проверяет, имеет ли субъект полномочия на создание объектов этого типа.

5.2.3.2.3. Операции доступа, специфичные для конкретных типов объектов

В этом разделе мы рассмотрим операции над объектами, отражающие специфику работы с типом, к которому относятся эти объекты.

Доступ к каталогам

Для работы с каталогами предназначены следующие операции:

- Чтение содержимого каталога — списка имен объектов и их атрибутов, несвязанных с содержанием. Для осуществления этой операции субъект должен иметь полномочия на чтение каталога как обычного объекта.
- Проход каталога в процессе поиска имени объекта. Для осуществления этой операции субъект также должен иметь доступ по чтению к каталогу.
- Создание в каталоге ссылки на объект. Субъект может создавать объекты в каталоге, если он обладает полномочиями, достаточными для осуществления записи в каталог и имеет право создания объектов соответствующего типа. Уровень безопасности каталога должен входить в диапазон уровней безопасности типа. Уровень безопасности объекта должен входить в диапазон уровней безопасности типа. Например, для создания ссылки субъект обладать правом на создании объектом типа "ссылка". Уровень безопасности объекта должен доминировать над уровнем безопасности субъекта.
- Удаление ссылки на объект из каталога. Субъект может удалить объект из каталога, если его полномочия позволяют ему осуществлять запись в каталог.

Управление типом

Субъект может управлять типом (в том числе и правами доступа к типу), если выполнены следующие условия:

- субъект и тип должны иметь одинаковый диапазон уровней безопасности;
- список прав доступа типа должен разрешать субъекту управлять типом.

Так как диапазон уровней безопасности для субъектов, не входящих в состав ТСВ (прикладных программ), состоит из единственного

уровня безопасности, и списки прав доступа для всех встроенных типов (файлов, типов, каталогов, ссылок, устройств) определены таким образом, что управлять типами могут только соответствующие сервера из состава TCB, прикладные программы не могут управлять встроенными типами. Например, управление типом "файл" может осуществлять только сервер файловых систем.

Доступ к соединениям

Для того, чтобы описать взаимодействия между субъектами в тех же терминах, что и взаимодействия между субъектами и объектами в Trusted Mach введен специальный тип объектов — соединения. Соединения дают субъектам возможность обмениваться информацией с другими субъектами напрямую без создания объектов. Контроль за доступом к соединениям позволяет монитору взаимодействий контролировать эти информационные потоки. По существу, соединения представляют собой пассивные компоненты субъектов и имеют те же атрибуты безопасности, что и субъекты, которых они представляют — уровень безопасности соединения автоматически устанавливается таким же, что и уровень безопасности представляемого им субъекта. Соединения создаются и уничтожаются одновременно с созданием и уничтожением субъектов.

Для того, чтобы субъект получил доступ к соединению, т. е. вступил в обмен информацией с другим субъектом, должны выполняться следующие условия:

- субъект и соединение должны обладать одинаковым уровнем безопасности;
- список прав доступа соединения должен разрешать для субъекта запрашиваемый тип доступа.

Доступ к ссылкам

Над ссылками возможны только операции чтения и создания. После того, как ссылка создана, изменять ее содержимое невозможно.

- Чтение содержимого ссылки. Существуют два способа получить содержимое ссылки. Во-первых, субъект может явным образом открыть ссылку, а затем осуществить операцию чтения. Во-вторых, субъект может косвенно получить содержимое символического указателя в процессе поиска имени объекта. Когда Сервер имен встречает в имени объекта ссылку, он определяет, имеет ли субъект право на чтение этой ссылки. Если право на чтение имеется Сервер имен продолжает проход по дереву объектов, в противном случае поиск имени оканчивается неудачей.
- Создание ссылки. Для создания ссылки на объект необходимо иметь полномочия для создания объектов в каталоге, в котором будет размещаться ссылка, право на создание объектов типа "ссылка" и соот-

ветствующие права доступа к объекту, на который будет указывать эта ссылка.

Доступ к внешним устройствам

Поскольку такие объекты как устройства отличаются от объектов других типов (например, файлов) для работы с ними используются специфические операции:

- Получение доступа к порту, представляющему устройство. Порт устройства может быть использован для чтения и записи информации в устройство, в зависимости от имеющихся у субъекта полномочий. Субъектам не разрешается создавать объекты этого типа и переименовывать имеющиеся. Это могут делать только администраторы системы с помощью доверенных средств из состава ТСВ.
- Инициализация устройства. Субъект может осуществлять эту операцию, приводящую устройство в некое начальное состояние, но только в том случае, если он имеет полномочия на запись в данное устройство. На практике эта операция применяется только к портам к клавиатуре, последовательным и параллельным портам. Доступ к таким устройствам как жесткий диск и сетевая карта осуществляют только соответствующие сервера, входящие в состав ТСВ. Всем остальным субъектам доступ к этим устройствам закрыт.
- Получение статуса устройства. Субъект может получить статус устройства, если имеет полномочия на чтение информации из этого устройства.
- Установка статуса устройства. Субъект может установить статус устройства, если он обладает полномочиями на запись информации в это устройство.

5.2.3.2.4. Выводы по реализации политики безопасности Trusted Mach

В данном разделе была рассмотрена неформальная политика безопасности Trusted Mach, однако безопасность всех перечисленных операций при соблюдении указанных условий и правил может быть подтверждена формальной моделью, корректность которой доказана теоретически. Основные положения формальной модели Trusted Mach приведены в Приложении 6.

Реализация политики безопасности в Trusted Mach представляет собой большой шаг вперед по сравнению с традиционными (в первую очередь основанными на Unix) системами, которые рассматривались в разд. 5.1. Введены типы объектов, что позволяет применять более гибкие дискреционные модели управления доступом. Мониторинг доступа к объектам охватывает все отношения доступа за счет расширения множества

объектов с помощью концепции соединений, что стало возможным за счет использования микроядерной архитектуры. Прогрессивным является то, что доступ к объектам контролируется теми серверами, которые реализуют эти операции доступа. Особенно это заметно при контроле доступа к устройствам, т. к. в традиционных системах, где устройства отображались в файлы не существовало возможности контролировать специфические операции, например инициализацию.

С другой стороны целесообразность создания пользовательских типов объектов существенно снижается тем, что диапазон их уровней безопасности ограничен одним единственным уровнем.

В качестве главного недостатка архитектуры контроля доступа в Trusted Mach модно отметить то, что централизованный контроль доступа действует только в отношении запросов, идущих от субъектов к объектам через Сервер имен, а вся ответственность за передачу информации возложена на сервера, обслуживающие объекты. Это приводит к тому, что на практике монитор взаимодействий контролирует не сами информационные потоки, а только запросы субъектов. Однако, это не приводит к нарушению безопасности, т. к. на уровне микроядра действует контроль, основанный на распределении прав портов.

5.2.4. Аудит

В микроядерных операционных системах эту функцию логичнее всего возложить на специальный сервер, который регистрирует и протоколирует все происходящие в системе события, в т. ч. все попытки идентификации и аутентификации пользователей, а также попытки доступа к объектам. Необходимо отметить, что аудит сам по себе не является средством защиты, т. к. непосредственно не противостоит угрозам безопасности, однако анализ протокола аудита позволяет, с одной стороны, определять последствия тех или иных действий пользователей (в том числе нарушений безопасности), а с другой — прямо в процессе функционирования системы выявлять последовательности событий, указывающие то, что система подвергается атаке со стороны нарушителей, и предпринимать необходимые действия для предотвращения таких атак (например, сделать так, чтобы после многократных некорректных попыток идентификации и аутентификации пользователя блокировалось устройство ввода/вывода, через которое производятся эти попытки). Таким образом основной задачей средств аудита является регистрация и учет событий, происходящих в системе.

Разработка средств регистрация и учета событий включает в себя реализацию следующих четырех групп функций:

- отбор событий, подлежащих регистрации;
- регистрация и учет этих событий;

- анализ журнала событий и распознавание угроз;
- реакция на выявленные угрозы.

В самой архитектуре микроядерных ОС заложены богатые возможности по регистрации и учету событий, т. к. взаимодействие всех компонентов системы осуществляется только через механизм передачи сообщений. Это означает, осуществляя регистрацию каждого сообщения, включая идентификаторы отправителя, получателя и передаваемую информацию, мы фактически можем зафиксировать все взаимодействия, имевшие место в системе. На основании такого журнала аудита можно восстановить практически всю историю взаимодействия между задачами и микроядром, имевшую место в системе, включая все информационные потоки между всеми компонентами системы.

В микроядерных ОС с их децентрализованной архитектурой обработки информации отбор событий, которые должны быть зарегистрированы, должен быть возложен на соответствующие сервера, отвечающие за осуществление этих событий. Например, Сервер имен отвечает за события, связанные с попытками доступа к объектам, а Сервер аутентификации за события, связанные с попытками аутентификации. Сервер аудита отвечает только за фильтрацию и запись событий, информацию о которых он получает от других серверов. Поскольку протоколирование всех сообщений может потребовать слишком много времени и дискового пространства (к тому же это далеко не всегда оправдывается) администратор безопасности должен иметь возможность задать набор событий, подлежащих аудиту. Отбор событий может производится в соответствии с критериями, определяемыми администратором как логическая функция на пространстве значений полей регистрационных записей. Например в Trusted Mach каждая запись в протоколе аудита содержит стандартный заголовок, состоящий из следующих полей:

- идентификатор события;
- идентификатор пользователя, инициировавшего событие;
- идентификатор группы пользователя, инициировавшего событие;
- уровень безопасности объекта, с которым связано событие;
- уровень безопасности пользователя, инициировавшего событие;
- терминал, с которым связано событие;
- реакция на событие (запись в журнал аудита или поднятие тревоги).

Соответственно, критериями отбора событий могут служить указанные диапазоны уровней безопасности субъектов и объектов, имя терминала и т. д.

Кроме заголовка записи в протоколе аудита Trusted Mach включают информацию, специфичную для каждого типа событий. Каждому типу событий присвоен определенный идентификатор, который позволяет серверу аудита распознавать типы событий, генерируемых различными сер-

верами. Например, Сервер субъектов генерирует события одного типа — *удаление субъекта*, а сервер имен 14 типов событий, в том числе создание/удаление объектов, установка параметров доступа и т. д.

Типовой набор событий подлежащих регистрации включает в себя:

1. Попытки идентификации и аутентификации. Информация о данном событии включает дату и время, результат попытки и, в случае успеха, полномочия, предоставленные пользователю.

2. Попытки доступа к объектам. Наиболее простой метод отслеживания доступа в микроядерной ОС — регистрация всех обращений с Серверу имен. Поскольку обращение к Серверу имен должно предшествовать любому типу доступа к объекту, протоколирование этих событий дает полную картину того, какие предпринимались попытки осуществления доступа и насколько они были успешны. Для событий этого типа записываются дата и время события, тип объекта, тип запрашиваемого доступа и результат запроса.

3. Создание и удаление объектов. Для этих операций записываются дата и время события, тип объекта, тип операции (создание/удаление) и результат запроса.

4. Модификация параметров системы, связанных с безопасностью. К этой группе событий относятся, во-первых, попытки пользователя модифицировать списки прав доступа, попытки изменения имен объектов, изменения параметров сеанса работы с системой. Кроме того, сюда же относятся попытки администратора остановить систему, изменить параметры аудита, добавить и удалить пользователей, изменить их атрибуты безопасности и т.д. Кроме стандартных параметров записи о событиях этого типа содержат информацию о том, каким образом, и какой параметр должен был быть модифицирован, а также результат операции.

Доступ к файлам, содержащим протокол аудита может контролироваться стандартными механизмами управления доступом. Администратор безопасности может также выбрать процедуру которая должна быть инициирована в том случае, когда дисковое пространство исчерпано.

Система аудита должна позволять администратор безопасности задавать действия, которые должны быть автоматически предприняты в случае наступления того, или иного события. Например, регистрация соответствующей записи в протоколе, посылка сообщения определенному пользователю или приостановка выполнения задач пользователя-нарушителя.

5.2.5. Прямое взаимодействие

Прямое взаимодействие является одной из важнейших составляющих защищенных систем, т. к. недостатки в этой сфере могут свести на

нет все достижения защиты и предоставить нарушителю широкие возможности для проникновения в систему. Пользователи должны быть уверены, как в том, что они взаимодействуют действительно с TCB системы, так в том, что введенные ими команды будут обработаны корректным образом и приведут именно к тем результатам, на которые они рассчитывают.

Например, в Trusted Mach за реализацию прямого взаимодействия с TCB пользователей и администратора отвечают Сервер виртуальных терминалов и два командных процессора — доверенный командный процессор администратора (Trusted administrator shell — TASH), а для остальных пользователей просто доверенный командный процессор (Trusted shell — TSH). Сервер виртуального терминала обеспечивает правильность и корректность отображения данных на мониторе и ввода с клавиатуры. Доверенные процессоры обеспечивают вход и выход пользователей из системы, запуск прикладных программ, изменение параметров безопасности (например, пароля).

Доверенный командный процессор вместе с сервером виртуального терминала реализуют прямое взаимодействие пользователя с TCB, без использования промежуточных звеньев. Схема взаимодействия представлена на рис. 5.16.

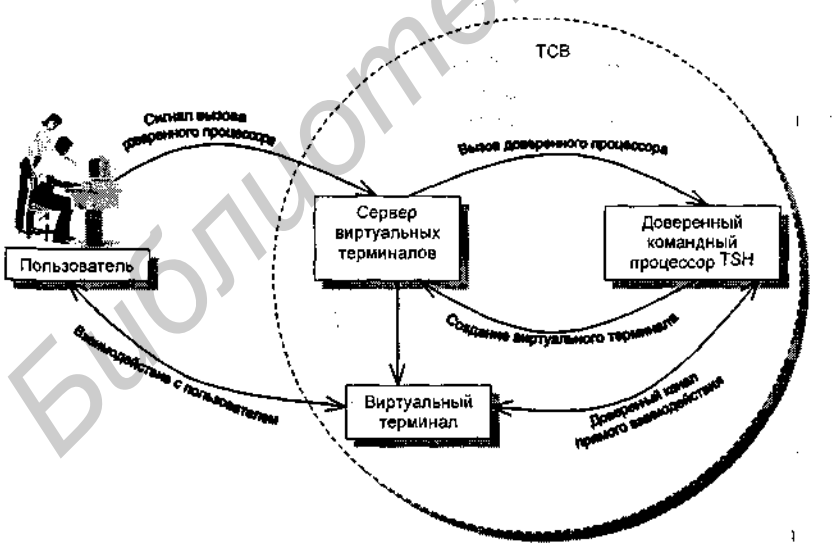


Рис. 5.16 Схема организации прямого взаимодействия пользователя с TCB.

Пользователь, стремящийся установить соединение с доверенным командным процессором с целью входа в систему или запуска прикладных программ, вводит с терминала сигнал вызова доверенного командно-

го процессора, — это может быть специальная клавиша на клавиатуре, или даже специальное устройство, подающее сигнал непосредственно в компьютер. Сервер виртуальных терминалов распознает этот сигнал и посылает доверенному командному процессору соответствующее сообщение, в котором указывает идентификатор терминала, с которого поступил сигнал. Доверенный командный процессор устанавливает соединение с этим терминалом и выводит на него одно из двух фиксированных и известных пользователю приглашений: подсказку для входа в систему или меню, позволяющее осуществлять просмотр и изменение атрибутов безопасности объектов, запускать прикладные программы, управлять сессиями работы и вызывать доверенный командный процессор администратора. Таким образом, хотя при установлении прямого взаимодействия используются только компоненты, входящие в ТСВ, пользователь имеет возможность запускать приложения, не входящие в ТСВ, и быть уверенным в том, что будут вызваны именно те программы, которые он имел в виду. При завершении сеанса работы с системой доверенный командный процессор уничтожает виртуальный терминал и все данные о сеансе.

Кроме того доверенный командный процессор отвечает за идентификацию и аутентификацию пользователей. Когда пользователь пытается войти в систему, он предъявляет доверенному командному процессору через канал прямого взаимодействия свой идентификатор, пароль и требуемый уровень безопасности. Доверенный командный процессор проверяет корректность пароля и принадлежит ли запрашиваемый уровень диапазону уровней, разрешенному для этого пользователя. В случае успешной проверки доверенный командный процессор разрешает пользователю работу с системой и запускает от его имени процессы с соответствующим идентификатором субъекта и атрибутами безопасности.

Доверенный командный процессор администратора функционирует аналогично доверенному командному процессору, за исключением того, что он позволяет запускать утилиты, осуществляющие администрирование системы. С помощью этих утилит администраторы системы осуществляют управление безопасностью.

5.2.6. Управление безопасностью

Управление безопасностью в защищенных ОС осуществляет соответствующим образом подготовленный персонал — администраторы безопасности. В Trusted Mach обязанности администраторов распределены по четырем ролям. Ролью в данном случае называется идентификатор пользователя, которому соответствует субъект, способный выполнять жестко ограниченное множество команд. Роли администраторов безопасности Trusted Mach упорядочены по степени доверия и называются следую-

щим образом: "Доверенный Программист", "Администратор безопасности", "Аудитор", "Оператор безопасности".

С помощью этого механизма Trusted Mach обеспечивает разделение обязанностей для привилегированных пользователей таким образом, что ни из них не имеет доступа ко всем привилегированным операциям. Привилегированные операции сгруппированы по ролям, и каждый привилегированный пользователь может выполнять только операции, доступные для выделенной ему роли, т. е. только те, которые необходимы для выполнения порученной ему работы. Это сокращает размеры ущерба, который может нанести системе привилегированный пользователь, как случайно, так и преднамеренно. Взаимоотношения между ролями приведены на рис. 5.17.

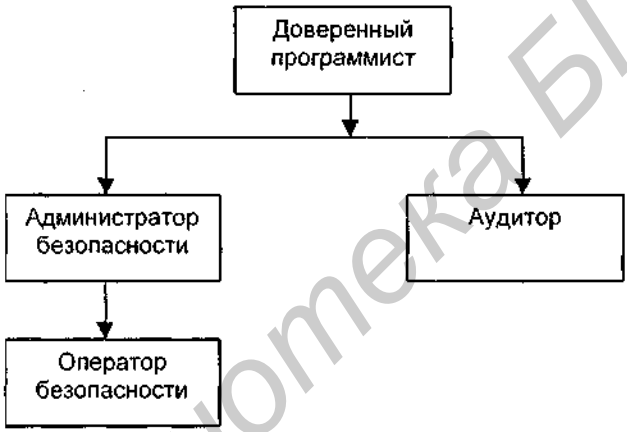


Рис. 5.17. Иерархия административных ролей.

В представленной иерархии каждая вышестоящая роль включает в себя все возможности нижележащих ролей. Наиболее привилегированной является роль "Доверенный программист". Ему предоставляется наибольший набор полномочий, но он несет и самую большую ответственность, т. к. может вносить любые исправления в систему. Поэтому эта роль доступна только в специальном однопользовательском режиме (аналог Single User Mode ОС Unix). Остальные роли доступны через доверенный командный процессор администратора в рабочем порядке. Рассмотрим функции, которые возложены на каждого администратора в соответствии с его ролью:

Только "Доверенный программист", в случае необходимости, может:

- осуществлять управление конфигурацией периферийных устройств, файловой системы и основными параметрами ОС в целом;

- создавать бюджеты "Администратора безопасности" и "Аудитора", устанавливать параметры системы, влияющие на ее безопасность;
- осуществлять администрирование ТСВ в чрезвычайных ситуациях, например, проводить восстановление файлов, входящих в ТСВ, тестирование и наладку внешних устройств, установку новых версий системы.

Роль "Администратора безопасности" предназначена для выполнения следующих действий:

- управления аутентификацией, включая установку начальных паролей и параметров их устаревания;
- управления соответствием между идентификаторами уровней безопасности и их символьным представлением;
- задания пределов изменений и начальных значений уровней безопасности пользователей;
- правления параметрами устройств и задания их уровней безопасности в пределах установленных "Доверенным программистом";
- создания и изменения бюджетов пользователей и групп;
- администрирования файловой системы, редактирование списков прав доступа;
- архивирования, восстановления из резервных копий файловой системы, мониторинга ее состояния;
- управления и мониторинга ОС в целом;
- уничтожения, в случае необходимости, процессов и задач, нарушающих штатный режим работы системы.

Набор функций, доступных роли "Аудитор" включает в себя:

- задание набора типов событий, подлежащих регистрации и учету;
- выделение ресурсов, используемых под журнал аудита, установка его предельного размера;
- проверка журнала аудита и его анализ с помощью соответствующих средств.

Роль "Оператор безопасности" позволяет осуществлять следующие действия:

- загрузку системы и задание пароля в ходе загрузки;
- редактирование информации о пользователях;
- монтирование/демонтирование файловых систем;
- подключение/отключение терминалов и принтеров;
- создание резервных копий, восстановление файловой системы, мониторинг ее состояния;
- мониторинг всей системы в целом;
- останов системы.

Наличие такого набора административных ролей позволяет эффективно разделить задачи по администрированию системы между несколькими пользователями-администраторами и распределить между ними ответственность за обеспечение безопасной работы системы в соответствии со степенью доверия.

5.2.7. Архитектура TCB системы

Как было показано в разделе 5.1.3 архитектура TCB является краеугольным камнем всей системы защиты или защищенной системы. Проанализируем структура TCB Trusted Mach, показанную на рис. 5.18

Структура TCB Trusted Mach не является однородной, в ее состав (рис. 5.18) входят микроядро, доверенные сервера, командные процессоры и доверенные средства администрирования. Взаимосвязи между компонентами TCB, основанные на функциональной зависимости образуют иерархию (на рисунке функциональные зависимости между компонентами показаны стрелками, за исключением транзитивных).

Функциональную зависимость между компонентами разработчики Trusted Mach понимают следующим образом: компонент A зависит от компонента B, если компонент B может выполнить действия, которые повлекли бы за собой нарушения в работе компонента A. Такой подход увеличивает степень структуризации системы, способствует более ясной реализации модели безопасности, упрощает программирование, тестирование, и многие другие аспекты проектирования и разработки системы. Несмотря на что, что все сервера, входящие в состав TCB, являются доверенными, полномочия каждого из них ограничены до минимально необходимых путем соответствующего распределения прав доступа к портам, настройкой уровней безопасности и списков прав доступа серверов.

Любое предоставление полномочий доверенному серверу должно быть обосновано его разработчиком как необходимое для выполнения функций, возложенных на этот сервер. Благодаря этим мерам каждый из доверенных серверов имеет право доступа лишь к минимально необходимому множеству объектов. Например, сервер аутентификации не может получить доступ к журналу аудита, а для сервера системного сервиса недоступен файл паролей. Сервер имен не может непосредственно обращаться к разделам диска, в которых сервер файловых систем размещает файлы, хотя должен иметь доступ к тем областям диска, в которых хранятся атрибуты файлов.

Другим уровнем структурирования компонентов TCB в Trusted Mach служит модульный и объектно-ориентированный подход, применяющийся при реализации компонентов. Каждый компонент TCB состоит из нескольких модулей, взаимосвязи и взаимодействие между которыми поддерживаются с помощью механизмов языка программирования C++.

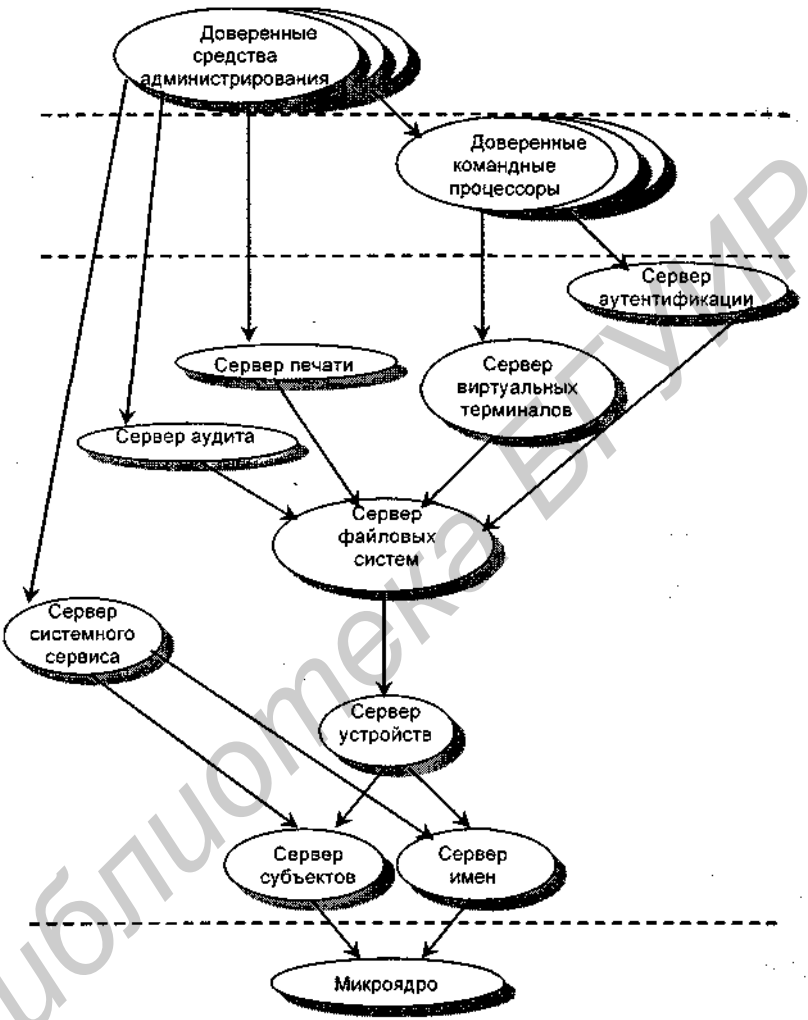


Рис. 5.18. Структура TCB Trusted Mach и зависимости между ее компонентами.

Язык C++ позволяет обеспечить на уровне синтаксиса контроль за изоляцией модулей и способами их взаимодействия, а также определить правила экспорта и импорта операций и переменных из одного модуля в другой. Согласно принципу наименьших привилегий модули должны обладать только полномочиями, необходимыми для выполнения своих функций, но не более того. Модули должны импортировать и экспортировать только те конструкции, которые им необходимы для выполнения возложенных на них задач. Необходимость импорта и экспорта тех или

иных функций, переменных или типов должна быть обоснована разработчиком модуля. Ограничение возможностей модуля (даже на уровне синтаксиса языка программирования) в точности рамками его конкретного назначения уменьшает возможность выполнения этим модулем нежелательных действий. Кроме того, явное предоставление модулю необходимого набора полномочий, подтверждает правильность понимания разработчиком назначения данного модуля. При контроле взаимодействий между модулями применяются те же принципы что и при распределении полномочий между компонентами TCB, но контроль осуществляется на уровне компилятора и синтаксиса C++.

5.2.8. Безопасность в процессе инициализации

В соответствии с теорией информационной безопасности, для того чтобы система была безопасной, необходимо, как минимум, обеспечить безопасность ее начального состояния. Безопасность начального состояния системы определяется соблюдением соответствующих мер в ходе инициализации системы. В Trusted Mach в соответствии с принципом модульной организации TCB каждый элемент несет ответственность за безопасность своей инициализации.

Перед началом первого этапа инициализации — загрузкой ядра его целостность контролируется проверкой соответствия контрольных сумм. После загрузки ядра и инициализации его структур, запускается специальный сервер MachInit, который отвечает за инициализацию остальной части системы. В соответствии с заданной конфигурацией системы этот сервер запускает сервера, входящие в состав TCB (см. рис. 5.18), а также создает порты для взаимодействия между этими серверами и распределяет права доступа к портам в соответствии с принципом минимальных привилегий. Конфигурация системы содержит порядок инициализации серверов и все предварительные условия, которые должны быть выполнены перед запуском каждого сервера (например, Сервер файловых систем должен быть запущен только после успешной инициализации Сервера имен). Прежде чем осуществить запуск серверов, не входящих в состав TCB, MachInit должен получить от всех доверенных серверов подтверждения того, что их инициализация прошла успешно и безопасность не нарушена, в противном случае инициализация системы будет либо повторена, либо приостановлена.

Как уже говорилось, каждый сервер несет ответственность за безопасность своей инициализации. Серверы, обслуживающие данные, которые продолжают существовать и после остановки системы (например сервер файловых систем), должны проверять корректность этих данных с точки зрения безопасности. Например, сервер файловых систем должен проверить целостность служебных структур файловых систем (каталоги,

таблицы распределения блоков диска), чтобы удостовериться в их корректности. Когда сервер, входящий в состав TCB успешно завершает свою инициализацию, он посылает MachInit соответствующее сообщение. Если в ходе инициализации сервера возникли проблемы, то в зависимости от ситуации он может предпринять следующие действия:

- устранить проблему (если это возможно) и продолжить работу;
- завершить работу и послать MachInit сообщение с требованием проведения повторной инициализации всей системы;
- остановить систему и подождать вмешательства администратора.

В том случае, если микроядро само сталкивается с неустранимыми проблемами в ходе инициализации, процесс загрузки может быть повторен, либо система будет остановлена до вмешательства администратора. В любом случае проблемы, с которыми столкнулись серверы или ядро, и предпринятые ими действия протоколируются.

В процессе работы Trusted Mach непрерывно контролирует состояние TCB и может обнаружить возникшие проблемы. Как и в случае с инициализацией, каждый сервер отвечает за устранение нарушений и сбоев в своей работе. Если проблема может вызвать нарушение безопасности, то система может предпринять тот же набор действий, что и в случае неудач в ходе инициализации.

5.2.9. Тестирование

Для любой программной системы основным методом обеспечения надежности является тестирование. В процессе разработки защищенных систем используются два вида тестирования: функциональное тестирование компонентов, входящих в состав TCB, и анализ возможностей преодоления защиты.

Функциональное тестирование преследует своей целью выявить все отклонения от спецификаций у функций интерфейса TCB и основано на применении стандартных технологий тестирования программ.

Анализ возможностей по преодоления защиты состоит в том, что независимая группа экспертов пытается найти и использовать ошибки в архитектуре и реализации средств защиты с целью проникновения в систему и нарушения безопасности. При этом используется известный опыт использования известных общих недостатков в защите операционных систем (глава 3). Эксперты делают предположения о возможных недостатках архитектуры и реализации систем защиты. Затем проводятся исследования системы, проектной документации, исходных текстов и на основании результатов анализа создаются тестовые программы, пытающиеся использовать предполагаемые недостатки защиты для нарушения безопасности. Поскольку в ходе анализа возможностей по преодоления защиты используются те же самые средства и даже больший объем ин-

формации, чем тот, что находится в распоряжении предполагаемого нарушителя, неспособность экспертов в ходе испытаний нарушить защиту является объективным показателем того, что предполагаемый нарушитель также не сможет этого сделать.

5.2.10. Выявление скрытых каналов передачи информации

Одним из самых существенных скрытых недостатков защищенных систем является наличие скрытых каналов утечки информации. На практике чрезвычайно трудно обнаружить и полностью исключить все скрытые каналы. Скрытые каналы представляют собой использование субъектом косвенных методов доступа к информации, которая недоступна ему в соответствии с политикой безопасности. Скрытые каналы бывают двух типов: с использованием временных параметров и областей памяти. Каналы с использованием областей памяти могут возникнуть при использовании общих переменных в различных модулях. На практике, как правило, невозможно полностью устранить совместное использование общих переменных, т. к. это противоречит назначению ОС — разделению ресурсов аппаратуры между прикладными программами и организации их совместной работы. В скрытых каналах с использованием времени информация кодируется последовательностью или длительностью определенных событий, что выявить также чрезвычайно сложно. Однако, стандарты информационной безопасности предъявляют жесткие требования к этому этапу разработки защищенных систем — для соответствия классу B3 требований «Оранжевой книги» необходимо выявить все каналы с использованием памяти и контролировать их полосу передачи, а также выявить и определить ширину полосы передачи каналов с использованием времени.

Для выявления скрытых каналов при проектировании и реализации Trusted Mach применяется метод матрицы общих ресурсов. Этот метод позволяет выявить общие переменные и оценить возможность их использования для организации скрытых каналов. В случае выявления канала разрабатываются меры по сокращению его пропускной способности и устанавливается контроль за его использованием путем аудита. Анализ скрытых каналов с использованием времени осуществляется путем выявления обращений к таймерам и использования результатов этих обращений, т. к. все временные каналы связаны с использованием значением текущего времени.

5.3. Заключение к главе 5

В этой главе мы рассмотрели как строится защита современных операционных систем. По мнению авторов защищенная операционная система должна лежать в основе каждой системы обработки информации, которая претендует безопасность, поскольку исторически процесс разви-

тия информационных технологий сложился так, что именно в рамках операционной системы сосредоточены все базовые механизмы, которые в конечном счете определяют границы возможностей системы. Этот тезис подтверждается также тем, что среди сертифицированных продуктов операционные системы составляют подавляющее большинство. Поскольку реализация политики безопасности является ключевым моментом при создании защищенной ОС, авторы сочли необходимым сосредоточить основное внимание на этом вопросе.

Литература к главе 5

1. "List of Products evaluated by the NSA's Trusted Products", Mitre Corporation, 1995.
2. "Trusted Computer System Evaluation Criteria", Department of Defense, 1985.
3. "A guide to understanding discretionary access control in trusted systems", National Computer Security Center, 1987.
4. "Trusted Network Interpretation", National Computer Security Center, 1987
5. "Trusted Database Management System Interpretation", National Computer Security Center, 1991.
6. "Final Evaluation Report. Admahl Corporation UTS MLS 2.1.5+", National Computer Security Center, 1994.
7. "Final Evaluation Report. Trusted Information Systems Inc. Trusted Xenix version 3.0 ", National Computer Security Center, 1992.
8. "Final Evaluation Report. Microsoft Inc. Windows NT Workstation and Server Version 3.5 with U.S. Service Pack 3", National Computer Security Center, 1996.
9. D. Elliott Bell и Leonard J. LaPadula, "Secure Computer Systems:Mathematical Foundations", MITRE Technical Report 2547, Volume I, 1973
10. Leonard J. LaPadula и D. Elliott Bell,"Secure Computer Systems:A Mathematical Model", MITRE Technical Report 2547, Volume II, 1973
11. John McLean, "Security Models", Encyclopedia of Software Engineering, Wiley Press, 1994.
12. Буч Г. Объектно-ориентированное проектирование с примерами применения. // М.: Конкорд, 1992 г.
13. MK++ Kernel High Level Design. Open Software Foundation, Inc. Revision 2.2 January 12 1996.
14. Trusted Mach System Architecture. Technical report TIS TMACH Edoc--0001--97A, Trusted Information Systems, Inc, March 19, 1997.
15. Trusted Mach Mathematical Model. Technical report TIS TMACH Edoc-0017-96B, Trusted Information Systems, Inc, October 31, 1996.

Заключение

Создавая данную книгу авторы поставили своей целью осветить все основные аспекты создания защищенных систем, сформировать у читателя фундамент понятий, терминов и определений, позволяющий ему свободно ориентироваться в этой области, научить его грамотно формулировать задачи защиты информации и показать направления поиска способов их решения. Поэтому было решено ограничить содержание данной книги изложением теоретических основ информационной безопасности и результатами авторских исследований в этой области. Весь представленный в ней материал послужил базой осуществляемых авторами разработок в области создания защищенных систем. При изложении теоретических основ архитектуры защищенных систем и базовых алгоритмов работы средств защиты наибольшее внимание уделялось тем направлениям, которые обычно недостаточно освещаются в литературе. Среди выбранных направлений, занимающих центральное место в данной книге, необходимо выделить:

- современные зарубежные стандарты информационной безопасности, являющиеся, по сути, энциклопедией концептуальных понятий этой области;
- результаты исследования причин возникновения уязвимостей, позволяющие оценить на практике эффективность тех или иных решений;
- анализ архитектур сертифицированных защищенных систем.

Особое внимание было уделено различным аспектам построения защищенных операционных систем, что связано со спецификой нашей работы, поскольку уже в восьми шести лет мы занимаемся как анализом уровня защищенности современных ОС и технологий их построения, так и разработкой архитектуры защищенной ОС и созданием входящих в ее состав средств защиты.

Особенность нашей позиции в отношении защищенных информационных систем состоит в том, что под защищенной системой мы понимаем не просто совокупность средств обработки информации и средств защиты, а специальную систему обработки информации, обладающую такой архитектурой и поддерживающую такую организацию информационного процесса, которые позволяют обеспечивать как соответствие системы требованиям и критериям стандартов и корректную реализацию всех моделей и методов защиты, так и противостояние реально существующим угрозам. Разработкой технологии создания таких систем авторы занимаются уже в течение ряда лет.

При этом у читателя (особенно у специалиста по безопасности) может возникнуть вопрос: возможно ли в принципе создание защищенных систем, отвечающих указанному критерию, и существует ли вообще общая методология и технология их построения?

Конечно, о технологии проектирования защищенных систем нельзя говорить в том же смысле, в каком это понятие употребляют специалисты по системному проектированию. Общепринято считать, что теоретическая база проектирования любого объекта может быть сформулирована только тогда, когда построен алгоритм предсказания свойств проектируемого объекта на основе некоторой модели, описывающий его поведение и взаимодействие с окружающим миром. В этом случае проектирование сводится к нахождению такого варианта структуры объекта и значений его параметров, которые в соответствии с используемой моделью придают объекту свойства, близкие к заданным. Использование такого метода возможно только при наличии развитого аппарата моделирования всех свойств проектируемого объекта.

Применение такого подхода к проектированию защищенных систем не представляется возможным, поскольку безопасность обработки информации является неоднозначным критерием, а всестороннее моделирование систем обработки информации сталкивается с большими сложностями. Поэтому реализовать на практике традиционную схему проектирования в этом случае представляется весьма проблематичным. Тем не менее, авторы, опираясь на результаты своих исследований в области теории информационной безопасности и на свой практический опыт создания защищенных систем, оптимистичны в своей позиции и надеются, что использование предлагаемых ими подходов позволит построить такую технологию и преодолеть указанные трудности.

Авторы разрабатывают свою технологию построения защищенных систем в процессе создания специализированной защищенной операционной системы, что позволяет на практике проверить эффективность этой технологии и оценить перспективы ее применения. Описание собственно технологии создания защищенных систем и методов ее применения и полученных результатов авторы сознательно отложили до завершения разрабатываемого ими проекта.

Авторы надеются, что изложенные в данной книге теоретические положения, обзоры существующих решений, результаты аналитических исследований и анализа архитектуры защищенных систем послужат вкладом в дело построения безопасных информационных технологий.

Приложение 1.
Ранжированные функциональные требования "Федеральных критериев безопасности информационных технологий"

Данное приложение представляет собой ранжированный перечень функциональных требований, содержащийся в соответствующем разделе "Федеральных критериев". Данное изложение отражает только принципиальную суть требований и не претендует на перевод стандарта как руководящего документа.

Описание требований следует приведенным в разд. 2.8 требованиям в порядке возрастания уровня обеспечиваемой защиты. Описание каждого раздела требований начинается с его краткого описания и обзора предусмотренных уровней ранжирования, идентификаторы уровней сохранены в том виде, в каком они присутствуют в первоисточнике. Описание требований на каждом уровне приводится в виде дополнений и изменений по сравнению с предыдущем уровнем.

1. Идентификация и аутентификация

Идентификация и аутентификация принадлежат к основным компонентам политики безопасности. Отсутствие или низкая надежность процедур идентификации и/или аутентификации не позволяет противостоять атакам неавторизованных субъектов путем предотвращения их регистрации в системе и отказа в получении доступа к ее ресурсам. Надежность механизмов идентификации/аутентификации напрямую влияет на уровень безопасности всей системы в целом. При необходимости эти процедуры могут быть усилены совместным применением нескольких механизмов идентификации и аутентификации.

Требования идентификации и аутентификации ранжируются на основании уровня предоставляемых возможностей. Уровень **IA-1** включает только аутентификацию пользователей и предназначен для простейших систем, типа систем контроля доступа в помещения, в которых кроме идентификации и аутентификации реализована только функция регистрации входа. На уровне **IA-2** предполагается наличие специальных атрибутов (признаков), идентифицирующих конкретного субъекта и позволяющих выполнить его авторизацию (предоставить ему соответствующие права для работы в системе). Этот уровень наиболее широко используется в операционных системах, в которых существуют атрибуты, определяющие степень привилегированности субъектов и уровень конфиденциальности объектов. Данные возможности расширяются на уровне **IA-3** путем регламентации принципов обработки результатов аутентификации, а также требованиями к хранению, защите и предоставлению информации о результатах идентификации и аутентификации пользователей. Этот уровень используется в системах с четко определенной политикой управления доступом. На уровне **IA-4** происходит дальнейшее расширение требований, заключающееся в предоставлении возможностей установки специальных механизмов идентификации и аутентификации и их назначения для индивидуального пользователя и/или группы пользователей. На уровне

IA-5 требуется реализация нескольких независимых механизмов аутентификации пользователей.

Уровень **IA-1**. Минимальная идентификация и аутентификация.

1. TCB должна обеспечивать возможность идентификации каждого пользователя до начала выполнения им любых действий. TCB должна устанавливать индивидуальные полномочия пользователя в соответствии с его уникальным идентификатором. TCB должна обеспечивать возможность установления соответствия каждого регистрируемого в системе события с идентификатором инициировавшего его пользователя.

2. TCB должна использовать механизм аутентификации по паролю для проверки подлинности соответствия пользователя и его идентификатора.

3. TCB должна обеспечивать защиту данных, используемых для идентификации и аутентификации, с целью предотвращения доступа к ним неуполномоченных пользователей.

Уровень IA-2. Идентификация, аутентификация и авторизация.

1. *Без изменений.*

2. *Изменение.* Используемые для аутентификации данные должны включать информацию, необходимую как для проверки подлинности пользователя (например, пароль), так и для реализации политики безопасности (атрибуты пользователей, имена групп и ролей, уровни привилегированности субъектов и конфиденциальности объектов, временные интервалы и т. д.). Эти данные должны использоваться TCB для контроля действий пользователя в соответствии с действующей в системе политикой безопасности.

3. *Без изменений.*

Уровень **IA-3**. Идентификация и аутентификация с контролем исключительных ситуаций.

1. *Без изменений.*

2. *Без изменений.*

3. *Дополнение.* TCB должна обеспечивать выполнение процедуры аутентификации вне зависимости от результата проведения процедуры идентификации (например, требовать пароль даже в случае ввода несуществующего имени пользователя).

TCB должна прекращать выполнение процедуры входа в систему в случае неудачного проведения идентификация и/или аутентификации пользователя последовательно произошедших заданное администратором число раз. В этом случае TCB должна оповестить администратора, зафиксировать данное событие в системном журнале и приостановить дальнейшее выполнение процедуры входа в систему на время, определенное администратором, или отказать пользователю в доступе вообще.

4. TCB должна обеспечивать хранение, защиту и предоставление информации о статусе и атрибутах всех активных пользователей, зарегистрированных в системе и о бюджетах всех пользователей, существующих в системе.

Уровень **IA-4**. Назначение процедур идентификации и аутентификации.

1. *Дополнение.* TCB должна обеспечивать возможность идентификации

каждого привилегированного субъекта.

2. *Дополнение.* ТСВ должна обеспечивать возможность внедрения новых механизмов аутентификации (например на основе электронных карт или биометрических характеристик), дополняющих уже существующие. ТСВ должна обеспечивать возможность назначения различных механизмов аутентификации каждому пользователю в зависимости от его атрибутов.

3. *Без изменений.*

4. *Без изменений.*

Уровень **IA-5**. Комбинированное применение независимых механизмов идентификации и аутентификации.

1. *Без изменений.*

2. *Дополнение.* К каждому пользователю должны применяться два или более независимых механизма аутентификации. Аутентификация считается успешной, если все назначенные пользователю механизмы аутентификации подтвердили соответствие его идентификатора. ТСВ должна обеспечивать возможность назначения пользователю механизмов аутентификации в соответствии с атрибутам политики безопасности.

3. *Без изменений.*

4. *Без изменений.*

2. Регистрация пользователя в системе

Управление регистрацией пользователя в системе позволяет соблюсти требования политики аудита с помощью учета места, времени, способа и режима подключения пользователя к системе. Кроме того, управление регистрацией обеспечивает гарантию того, что зарегистрированный пользователь будет нести ответственность за выполняемые действия.

Процесс регистрации пользователя в системе должен управляться и контролироваться ТСВ. Должны быть четко определены условия, при которых возможно создание субъекта или субъектов, представляющих данного пользователя. Эти условия должны определяться на основании значений атрибутов пользователя, определяющих его статус, принадлежность к группе, возможные роли, степень полномочий, период разрешенного для работы времени, доступный ему сервис и ресурсы системы.

Требования к процедуре регистрации в системе ранжируются на основании обеспечиваемых возможностей защиты. Требования уровня **SE-1** включает в себя только простейшие возможности по управлению регистрацией в системе в соответствии с принадлежностью пользователя к определенной группе или выполнения им конкретной роли, а также степени его привилегированности. Этот уровень может использоваться в большинстве систем, поддерживающих выделенную самостоятельную процедуру регистрации пользователя. Системы, не реализующие отдельно такой процедуры в явном виде, используют для этой цели механизмы идентификации и аутентификации (по сути, идентификация и аутентификация могут считаться начальными процедурами управления регистрацией пользователя в системе). Уровень **SE-2** дополнен возможностями учета времени и места регистрации пользователя в систему. На уровне **SE-3** у пользователя появляются воз-

можности блокировать (приостанавливать) и возобновлять сеанс работы с системой.

Уровень **SE-1**. Базовый механизм управления регистрацией пользователя в системе.

1. Перед началом выполнения процедуры регистрации пользователя в системе (процедуры *login*) ТСВ должна соответствующим сообщением предупреждать пользователя о недопустимости попытки неавторизованного проникновения в систему и о возможных последствиях подобных действий.

2. Перед началом сеанса работы (до разрешения регистрации) ТСВ должна выполнить процедуры идентификации и аутентификации пользователя. Если в ТСВ предусмотрена поддержка одновременного подключения нескольких независимых сеансов работы пользователя, то должен быть реализован механизм контроля количества одновременных подключений и не превышения их максимального числа.

3. ТСВ должна осуществлять регистрацию только в соответствии с подтвержденными аутентификацией атрибутами пользователя. Условия предоставления доступа к системе определяются на основании атрибутов пользователя в рамках политики безопасности. Если эти условия не определены явным образом, используются условия, принятые по умолчанию (например, успешная аутентификация пользователя).

4. ТСВ должна включать в себя защищенные механизмы, при помощи которых уполномоченный пользователь (администратор) может управлять атрибутами пользователей, используемыми при их регистрации в системе. Должны быть определены условия, при которых непривилегированный пользователь может ознакомиться с собственными атрибутами, но не изменить их.

5. После успешной регистрации пользователя в системе ТСВ должна предоставлять ему следующую информацию (без возможности ее модификации):

- место, время, режим, терминал или порт последней успешной регистрации пользователя;
- количество неуспешных попыток регистрации с использованием его идентификатора, зафиксированное между последним и текущим сеансом работы.

6. ТСВ должна обеспечивать блокирование (приостановку) или прерывание и завершение сеанса работы пользователя по истечению устанавливаемого администратором интервала отсутствия активности со стороны пользователя.

Уровень **SE-2**. Управление регистрацией в системе с учетом времени и места подключения.

1. *Без изменений.*

2. *Без изменений.*

3. *Дополнение.* ТСВ должна реализовывать механизм разрешения/запрещения регистрации в системе на основании контроля допустимого периода времени регистрации. Условия контроля периода времени должны быть определены для времени суток, дней недели и отдельных календарных дат.

ТСВ должна реализовывать механизм разрешения/запрещения регистрации в системе на основании контроля местоположения пользователя (используемых терминалов или портов). При необходимости должны быть также определены

условия удаленного подключения пользователей по каналам связи.

4. *Без изменений.*

5. *Без изменений.*

6. *Без изменений.*

Уровень **SE-3**. Блокировка и восстановление сеанса работы пользователя.

1. *Без изменений.*

2. *Без изменений.*

3. *Без изменений.*

4. *Без изменений.*

5. *Без изменений.*

6. *Дополнение.* ТСВ должна поддерживать механизм блокировки и возобновления сеанса работы пользователя по его команде, или по истечению заданного администратором интервала отсутствия активности со стороны пользователя. Этот механизм должен обеспечивать:

- очистку экрана терминала для предотвращения возможности считывания с него информации и блокировку клавиатуры и других средств ввода информации;
- запрет на любые действия в системе с использованием заблокированных средств ввода-вывода информации на время приостановки сеанса пользователя;
- выполнение процедур идентификации и аутентификации пользователя перед возобновлением сеанса работы.

3. Обеспечение прямого взаимодействия с ТСВ

Функциональные требования, обеспечивающие прямое взаимодействие с ТСВ, ранжируются в зависимости от допустимой сферы применения и обеспечиваемых возможностей защиты. Минимальный уровень **ТР-1**, предполагает наличие гарантированного канала взаимодействия пользователя с ТСВ, используемого для выполнения процедуры регистрации в системе. На уровне **ТР-2** прямое взаимодействие с ТСВ обеспечивается не только для процесса регистрации, но и для других процессов, требующих обеспечения гарантированно достоверного канала взаимодействия между пользователем, выполняющим действия, критичные с точки зрения безопасности (например администрирование атрибутов безопасности), и ТСВ. На уровне **ТР-3** обеспечивается гарантированно достоверный канал взаимодействия между пользователем и доверенными приложениями, позволяющими ему работать с критичной информацией.

Уровень **ТР-1**. Обеспечение прямого взаимодействия с ТСВ для процедуры регистрации в системе.

ТСВ должна обеспечивать гарантированно достоверный канал взаимодействия с пользователем в ходе процесса его идентификации и аутентификации во время регистрации в системе. Инициация прямого взаимодействия с ТСВ должна осуществляться только со стороны пользователя.

Уровень **ТР-2**. Обеспечение прямого взаимодействия пользователя с ТСВ.

Дополнение. ТСВ должна обеспечивать гарантированно достоверный канал для всех видов взаимодействий с пользователем (регистрация в системе, изменение атрибутов защиты и т.д.). Данный канал должен предусматривать возмож-

ность инициации как со стороны пользователя, так и ТСВ, и должен быть изолирован от других аналогичных каналов, обладать уникальными характеристиками.

Уровень **ТР-3**. Обеспечение прямого взаимодействия пользователя с доверенными приложениями.

Дополнение. ТСВ должна обеспечивать гарантированно достоверный канал прямого взаимодействия пользователя с доверенными приложениями (например, для ввода или вывода критичной с точки зрения безопасности информации).

4. Регистрация и учет событий в системе (аудит)

Требования к регистрации и учету событий ранжируются в зависимости от предоставляемых возможностей отбора подлежащих регистрации событий, мощности средств анализа журнала событий и степени мониторинга действий пользователя. Требования к аудиту подразделяются на четыре группы:

- защита и управление доступом к системного журналу событий;
- определение множества подлежащих регистрации событий;
- фиксация и хранение зарегистрированных событий в журнале;
- анализ журнала событий и формирование отчетов.

Уровень **AD-1** включает минимальные требования к аудиту, которым должны следовать все системы в той мере, в какой они реализуют соответствующие функции защиты. На уровне **AD-2** эти требования усиливаются как за счет расширения множества типов регистрируемых событий, так и за счет добавления новых функций по управлению процессом регистрации и учета. На уровне **AD-3** появляются требования к наличию доверенных средств аудита, предоставляющих возможности выборочного анализа определенных типов событий, упрощающих взаимодействие с оператором за счет использования графического представления данных и т.п. Уровень **AD-4** характеризуется введением требования выявления критичных с точки зрения безопасности событий и объявления тревоги в случае их обнаружения. На уровне **AD-5** требуется обеспечить такой же контроль в режиме реального времени (осуществлять в режиме реального времени обнаружение попыток нарушений безопасности).

Уровень **AD-1**. Минимальный аудит.

1. ТСВ должна обеспечивать возможность создания, хранения, ведения журнала аудита, содержащего регистрацию обращений к защищенным объектам. ТСВ должна обеспечивать защиту журнала от несанкционированного доступа, изменения или уничтожения. ТСВ должна предоставлять доступ к журналу только авторизованным пользователям.

2. ТСВ должна обеспечивать регистрацию в журнале аудита следующих типов событий:

- использование средств идентификации и аутентификации;
- создание и удаление объектов;
- доступ к объектам, помещение объектов в доступную пользователю область, запуск программ;
- действия, предпринятые операторами и администраторами, ответственными за безопасность.

Для поддержки в системе политики обеспечения работоспособности и контроля за распределением ресурсов должны регистрироваться попытки несанкционированных запросов на выделение ресурсов и попытки получения доступа к ресурсам, предоставленным другим субъектам.

При поддержки в системе нормативного управления доступом ТСВ должна иметь возможность осуществлять регистрацию и учет изменений меток, классифицирующих уровень информации. Если нормативное управление доступом применяется для контроля потоков информации между субъектами, ТСВ должна иметь возможность регистрировать в журнале аудита события, которые потенциально могут использоваться для организации скрытых каналов передачи информации.

3. Для каждого регистрируемого события в журнал аудита заносятся:

- дата, время и тип события;
- идентификатор пользователя, инициировавшего событие;
- результат выполнения действия, соответствующего событию (успешное завершение или отказ).

При запросах на доступ к объектам или их удалении должны также регистрироваться имя и атрибуты объекта.

4. Администратор должен иметь возможность выбора регистрируемых событий и действий для каждого пользователя или объекта на основании соответствующих атрибутов политики безопасности.

Уровень **AD-2**. Базовые требования к аудиту.

1. *Без изменений.*

2. *Изменение.* ТСВ должна иметь возможность регистрировать следующие типы событий:

- использование механизмов идентификации, аутентификации и регистрации пользователя в системе;
- события, связанные с управлением доступом, относящиеся к определенному пользователю, субъекту, объекту или их атрибутам политики безопасности;
- создание, удаление субъектов и объектов, осуществление доступа, передача и отзыв прав доступа, изменение атрибутов политики безопасности, назначение и отзыв привилегий;
- действия, выполняемые операторами и администраторами, ответственными за безопасность, привилегированные операции, такие как модификация элементов ТСВ, настройка ТСВ, изменение параметров ТСВ и системных привилегий, изменение атрибутов пользователей, изменение состава и типов регистрируемых в журнале аудита событий.

Должен быть определен минимальный неизменяемый состав регистрируемых событий и их параметров. ТСВ должна содержать средства защиты и управления множеством регистрируемых событий и их параметров и предоставлять доступ к ним только администратору.

3. *Без изменений.*

4. *Дополнение.* В ТСВ должны присутствовать защищенные средства запуска и остановки процесса аудита. Доступ к этим средствам, равно как и к средствам просмотра журнала аудита, должен быть разрешен только администратору.

ТСВ также должна включать средства управления аудитом, доступные только для администратора, которые позволяют осуществлять:

- создание и удаление журнала аудита, контроль и изменение его размеров;
- форматирование и упаковка записей журнала аудита;
- обеспечение целостности журнала аудита при сбоях и отказах системы.

Уровень **AD-3**. Развитые средства аудита.

1. *Без изменений.*
2. *Без изменений.*
3. *Без изменений.*
4. *Дополнение.* В ТСВ должны присутствовать специально разработанные средства контроля целостности журнала аудита, а также средства контроля целостности заданного множества регистрируемых событий.
5. Средства просмотра журнала аудита должны предоставлять авторизованному пользователю возможность ознакомления с данными аудита и их проверки. Данные средства должны быть защищены от несанкционированного доступа.

ТСВ также должна иметь средства обработки журнала аудита, позволяющие осуществлять выборочный анализ:

- действий одного или нескольких пользователей;
- действий над одним или несколькими объектами или ресурсами;
- всех или подмножества исключительных ситуаций;
- действий, ассоциированных с заданными атрибутами политики безопасности субъектов и объектов.

Средства просмотра журнала аудита должны предусматривать возможность работы параллельно со штатным функционированием системы.

Уровень **AD-4**. Обнаружение попыток нарушения безопасности.

1. *Без изменений.*
2. *Дополнение.* ТСВ должна содержать средства мониторинга событий, возникновение которых может означать угрозу нарушения безопасности. Эти средства должны незамедлительно оповещать администратора системы и останавливать (прекращать) выполнение вызвавшего это событие процесса или всей системы.
3. *Без изменений.*
4. *Без изменений.*
5. *Без изменений.*

Уровень **AD-5**. Выявление попыток нарушения безопасности в режиме реального времени.

1. *Без изменений.*
2. *Без изменений.*
3. *Без изменений.*
4. *Без изменений.*
5. *Дополнение.* ТСВ должна обеспечивать возможность регистрации событий и выявления попыток нарушения безопасности в режиме реального времени и оповещать о них администратора. Эта возможность должна реализовываться специальным механизмом мониторинга событий, критичных с точки зрения политики безопасности.

5. Политика управления доступом (произвольное и нормативное управление доступом)

Требования к реализации политики произвольного управления доступом могут быть ранжированы в зависимости от базы применения политики (ко всем субъектам и объектам системы, к выбранным подмножествам субъектов и объектов, в зависимости от атрибутов безопасности субъектов и объектов) и предоставляемых средств управления доступом (возможность управлять распространением прав доступа, возможность организации контроля доступа к объектам пользователей только с их разрешения). Кроме того, эти требования можно ранжировать в зависимости от уровня абстракции, на котором рассматриваются субъекты (пользователь, группа, роль) и объекты (область памяти, файл, запись в файле).

Для ранжирования требований нормативного управления доступом могут использоваться те же самые критерии, однако описание уровня рассмотрения субъектов и объектов в этом случае должно производится более точно. Поскольку нормативное управление доступом основано на контроле информационных потоков, должны контролироваться соответствующие атрибуты субъектов (например, состояние процесса) и объектов (размер, режим доступа и т. д.).

Требования к управлению доступом ранжированы по четырем уровням. На уровне **АС-1** устанавливаются минимальные требования к реализации политики управления доступом и допускается осуществление управлением доступа только по отношению к некоторому подмножеству субъектов и объектов, а также ограниченные возможности управления атрибутами безопасности. На уровне **АС-2** требования к политике безопасности расширяются в сторону возможности одновременного применения нескольких политик управления доступом и средств управления экспортом и импортом объектов. На уровне **АС-3** управление доступом должно поддерживаться для всех субъектов и объектов. Если реализована политика нормативного управления доступом, она должна использовать все атрибуты безопасности объектов и субъектов. На данном уровне также требуется проводить назначение прав доступа к объектам на основании их типа. На следующем уровне **АС-4** управление доступом расширяется путем добавления атрибутов времени и местоположения. Появляется возможность задания прав пользователей в зависимости от их принадлежности к определенной группе или выполнения ими определенной роли. В дополнение к требованиям контроля создания и уничтожения объектов вводится контроль за ресурсами и наследованием атрибутов. Предполагается, что этот уровень будет использоваться в системах, где требуется точно определенное управление доступом.

Уровень **АС-1**. Минимальное управление доступом.

1. Задание множества атрибутов безопасности объектов и субъектов. ТСВ должна задавать и поддерживать атрибуты безопасности субъектов и объектов. Атрибуты субъекта должны включать индивидуальный и групповой идентификаторы пользователя, представленного этим субъектом. Атрибуты объекта должны включать набор возможных прав доступа к этому объекту (чтение, запись, выполнение).

2. Управление атрибутами безопасности объектов и субъектов. ТСВ должна определять правила назначения и изменения атрибутов безопасности субъектов

и объектов и обеспечивать их безусловное выполнение. Эти правила должны быть основаны на следующих положениях:

- субъект может разрешать доступ к объекту для другого субъекта, только в том случае, если он сам обладает этим правом доступа;

- пользователи должен иметь возможность устанавливать режим совместного использования объектов и управлять им основе индивидуальных и групповых атрибутов субъектов;

- пользователи должны обладать средствами для контроля за процессом распространения и передачи прав доступа и иметь возможность ограничивать его.

Если для разных подмножеств субъектов и объектов определены различные правила управления атрибутами безопасности, то их реализация должна быть согласованной и не противоречить политике безопасности.

3. Управление доступом субъектов к объектам. ТСВ должна определять правила назначения полномочий (авторизацию) с целью управления доступом субъектов к объектам и обеспечивать их соблюдение. Эти правила должны быть основаны на атрибутах субъектов и объектов и обеспечивать защиту объектов от несанкционированного доступа.

Правила назначения полномочий должны охватывать четко определенное подмножество субъектов и объектов, а также принадлежащих им атрибутов безопасности. Должна быть обеспечена возможность применения различных правил назначения полномочий для различных групп субъектов и объектов, в этом случае реализация этих правил должна быть согласованной и не противоречить политике безопасности

4. Контроль за созданием и уничтожением объектов и субъектов. ТСВ должна контролировать создание и уничтожение субъектов и объектов, а также повторное использование объектов. Все полномочия на доступ к объекту, должны быть отозваны перед его уничтожением и предоставлением занимаемых им ресурсов в распоряжение системы. Вся содержащаяся в нем информация, в том числе и зашифрованная, должна быть уничтожена.

5. Инкапсуляция объектов. Если в ТСВ поддерживается механизм инкапсуляции объектов, он должен обеспечивать:

- авторизацию доступа к инкапсулированным объектам;

- возможность создания пользователем инкапсулированных объектов и подсистем;

- средства доступа к инкапсулированным объектам.

Уровень **АС-2**. Базовые механизмы управления доступом.

1. *Дополнение.* Если одновременно поддерживается несколько политик управления доступом, атрибуты безопасности субъектов и объектов для каждой политики должны быть определены отдельно. Атрибуты субъектов и объектов должны точно отражать уровень их конфиденциальности и целостности.

2. *Дополнение.* Правила управления атрибутами безопасности субъектов и объектов должны регламентировать назначение атрибутов в ходе импорта и экспорта объектов. В том числе управлять импортом в систему неклассифицированной информации, не имеющей атрибутов безопасности, и экспортом из системы информации, обладающей атрибутами безопасности.

3. *Дополнение.* Если одновременно поддерживается несколько политик управления доступом, правила назначения полномочий доступа должны быть определены отдельно для каждой политики. ТСВ должна обеспечивать корректность совместного применения политик безопасности, в том числе, совместное применение правил авторизации каждой политики.

4. *Без изменений.*

5. *Без изменений.*

Уровень **АС-3**. Расширенное управление доступом.

1. *Дополнение.* ТСВ должна незамедлительно сообщать пользователю о любых изменениях атрибутов ассоциированных с ним субъектов, повлекших за собой изменение уровня привилегированности пользователя. Пользователю должна быть предоставлена возможность запросить у ТСВ текущие значения атрибутов безопасности ассоциированных с ним субъектов.

ТСВ должна поддерживать назначение атрибутов безопасности всем подключенным к системе физическим устройствам (например, максимальные и минимальные уровни конфиденциальности). Эти атрибуты должны использоваться ТСВ для отражения особенностей функционирования данных устройств, обусловленных их физическими параметрами.

2. *Без изменений.*

3. *Изменение.* Правила назначения полномочий доступа должны быть определены для всех субъектов и объектов, которые прямо или косвенно доступны субъектам.

Если применяется нормативное управление доступом, то правила назначения полномочий доступа должны учитывать все атрибуты субъектов и объектов.

4. *Без изменений.*

5. *Без изменений.*

Уровень **АС-4**. Точно определенная политика управления доступом.

1. *Дополнение.* В состав атрибутов безопасности субъектов должны входить показатели времени и местоположения, позволяющие дополнительно аутентифицировать ассоциированного с данным субъектом пользователя.

2. *Дополнение.* Правила управления атрибутами безопасности субъектов и объектов должны обеспечивать задание для каждого объекта списка индивидуальных субъектов и групп субъектов с указанием *их прав доступа к данному объекту.* Кроме того, правила управления атрибутами безопасности субъектов и объектов должны обеспечивать задание для каждого объекта списка индивидуальных субъектов и групп субъектов, не имеющих прав доступа к данному объекту.

Эти правила также должны обеспечивать возможность управления атрибутами в зависимости от времени и места — предоставление *или отзыв прав доступа* могут быть осуществлены в определенный момент времени и продлиться заданный период, а также зависеть от местоположения объектов и субъектов.

3. *Дополнение.* Правила назначения полномочий доступа должны включать возможность использования атрибутов времени и места осуществления доступа.

4. *Изменение.* ТСВ должна определять и поддерживать правила контроля за созданием и уничтожением субъектов и объектов, позволяющие указать для каждого субъекта и объекта:

- полномочия, требуемые для их созданию и уничтожения;
- процедуру повторного использования объекта;
- ресурсы, требуемые для их создания и размещения;
- устанавливаемые по умолчанию значения атрибутов созданных субъектов и объектов, и, если требуется, правила наследования атрибутов.

Правила создания и уничтожения субъектов и объектов должны определяться на основании их типов. Если для различных типов субъектов и объектов определены разные правила их создания и уничтожения, то должно быть показано, что их совокупность реализует политику безопасности, принятую в системе. Если одновременно используется несколько политик безопасности, правила создания и уничтожения субъектов и объектов должны быть определены для каждой политики.

5. *Без изменений.*

6. Контроль скрытых каналов

Для осуществления контроля за скрытыми каналами требуется присутствие в ТСВ программных, аппаратных и специальных средств, позволяющих обнаруживать скрытые каналы и ограничивать возможности их использования путем их полной ликвидации или минимизации пропускной способности. Ранжирование данной группы требований проводится на основе типов контролируемых скрытых каналов и предоставляемых возможностей контроля (аудит, минимизация пропускной способности, ликвидация).

Скрытые каналы в зависимости от способа кодирования информации подразделяются на два типа: с использованием памяти и с использованием времени. В первом случае для кодирования передаваемой информации используется либо область памяти (например, установление характерных признаков в имени и атрибутах файла или зарезервированные поля в заголовке сетевого пакета). Во втором случае, информация кодируется определенной последовательностью и длительностью событий, происходящих в системе (например, с помощью модуляции интервалов обращения к устройствам, введения задержек между приемом и посылкой сетевых пакетов и т.д.).

Уровень **ССН-1** затрагивает только скрытые каналы, использующие память и ограничивается контролем их использования. На уровне **ССН-2** добавляются требования минимизации пропускной способности и исключения возможностей использования скрытых каналов для штатного режима эксплуатации системы. Уровень **ССН-3** требует полного подавления всех типов скрытых каналов.

Уровень **ССН-1**. Контроль скрытых каналов, использующих память.

1. ТСВ и привилегированные приложения должны содержать функции контроля использования скрытых каналов, использующих память. Эти функции должны позволять идентифицировать источник и приемник скрытого обмена информацией и способ использования скрытого канала.

2. Функции ТСВ и привилегированных приложений, осуществляющие контроль скрытых каналов должны быть определены для каждого скрытого канала и присутствовать в типовой конфигурации системы. Если для некоторых скрытых каналов функции контроля отсутствуют, должно быть проведено доказательство

невозможности их использования для нарушения безопасности.

Уровень **CCH-2**. Контроль и ограничение пропускной способности скрытых каналов, использующих память.

1. *Дополнение.* Должно обеспечиваться ограничение пропускной способности или полное подавление скрытых каналов, использующих память. Пропускная способность каждого скрытого канала должна контролироваться администратором.

2. *Дополнение.* Функции ТСВ и привилегированных положений, обеспечивающие ограничение пропускной способности или полное подавление скрытых каналов, должны присутствовать в типовой конфигурации системы. Если для некоторых скрытых каналов функции ограничения пропускной способности или полного подавления отсутствуют, должно быть проведено доказательство невозможности их использования для нарушения безопасности.

Уровень **CCH-3**. Контроль и ограничение пропускной способности скрытых каналов, использующих время.

1. *Без изменений.*

2. *Дополнение.* Функции контроля, ограничения пропускной способности и подавления скрытых каналов должны в полной мере распространяться и на скрытые каналы, использующие время.

7. Контроль за распределением ресурсов

Данные требования являются частью политики обеспечения работоспособности системы и позволяют контролировать использование ресурсов системы. Ранжирование проводится по отношению к множеству управляемых ресурсов (т.е. подмножества ресурсов с ограниченным распределением) и функциональным возможностям средств управления.

Уровень **AR-1** определяет базовые требования к контролю за предоставлением ресурсов в терминах ограниченного подмножества системных ресурсов, субъектов и объектов. Уровень **AR-2** расширяет область применения средств контроля за предоставлением ресурсов до множества всех системных ресурсов с одновременным введением контроля за попытками монопольного захвата ресурсов и их доступностью для всех субъектов. На уровне **AR-3** к этим требованиям добавляется управление распределением ресурсов на основании приоритета субъекта и введение фиксированных квантов, которыми распределяются ресурсы.

Уровень **AR-1**. Ограничение при распределении ресурсов.

ТСВ должна обеспечивать возможность ограничения множества субъектов и объектов, доступных пользователю одновременно. ТСВ должна контролировать распределение определенного подмножества системных ресурсов таким образом, чтобы ни один пользователь не мог нарушить работу другого пользователя путем захвата такого количества ресурсов системы, при котором другие пользователи не могут осуществлять доступ к объектам и субъектам. Для всех субъектов, объектов и ресурсов должны быть определены ограничения на время и количество использования, а для ресурсов — атрибуты, обозначающие их количество.

Уровень **AR-2**. Полный контроль за распределением ресурсов.

Дополнение. ТСВ должна контролировать распределение системных ресурсов таким образом, чтобы ни один пользователь не мог сделать любой системный ресурс недоступным для других пользователей, или ограничить возможности ТСВ по обслуживанию других пользователей, путем захвата ресурсов или осуществления манипуляций с ТСВ.

Уровень **AR-3**. Распределение ресурсов на основании приоритетов.

Дополнение. ТСВ должна обеспечивать возможность распределения ресурсов на основании специально выделенных атрибутов, поставленных в соответствие каждому субъекту. ТСВ должна осуществлять распределение ресурсов в первую очередь субъектам, обладающим более высоким приоритетом. Все ресурсы должны выделяться блоками определенного размера (квантами).

8. Политика управления безопасностью

Ранжирование требований к средствам управления безопасностью основано на множестве управляемых параметров и уровне предоставляемых возможностей. Уровень **SM-1** содержит минимальные требования по управлению безопасностью. Уровень **SM-2** является базовым и предназначен для применения в большинстве систем. На уровне **SM-3** надежность механизма управления обеспечивается за счет разделения ролей администратора и оператора системы и применения более широкого набора средств управления безопасностью. На уровне **SM-4** требуется наличие доверенных средств управления безопасностью и введение контроля за администрированием системы.

Уровень **SM-1**. Минимальное управление безопасностью.

1. ТСВ должна содержать доверенные средства установки и настройки собственных конфигурационных параметров и инициализации критических внутренних структур данных перед заданием атрибутов безопасности пользователей и администраторов.

2. ТСВ должна поддерживать доверенные средства просмотра и редактирования параметров политики безопасности.

3. ТСВ должна включать доверенные средства просмотра, редактирования и удаления параметров регистрации пользователей и их бюджетов. Эти параметры должны включать уникальный идентификатор пользователя, его имя и служебное положение. Данные средства должны позволять администратору приостанавливать и возобновлять действие идентификаторов пользователей и их бюджетов.

4. ТСВ должна содержать доверенные средства контроля функционирования системы и состояния системных ресурсов. Эти средства должны обеспечивать подключение и отключение внешних устройств, съемных носителей информации, резервное копирование и восстановление объектов, эксплуатацию и тестирование программных и аппаратных компонентов ТСВ, запуск и остановку системы.

5. Средства управления безопасностью должны быть доступны только для администратора системы.

Уровень **SM-2**. Базовые механизмы управления безопасностью.

1. *Дополнение.* Средства управления безопасностью должны учитывать

различие между режимами штатного функционирования и технического обслуживания системы, и поддерживать управление безопасностью и в том и в другом режиме. Режим технического обслуживания системы должен позволять проводить восстановление после сбоев и запуск системы.

2. *Дополнение.* В состав параметров политики безопасности должны входить параметры идентификации, аутентификации, регистрации в системе и параметры управления доступом как для системы в целом, так и для каждого отдельного пользователя.

ТСВ должна позволять администратору определять политику идентификации и аутентификации для всех пользователей системы(период смены паролей, их длину и сложность). Средства управления параметрами политики безопасности должны позволять ограничивать:

- максимальный период отсутствия активности со стороны пользователя;
- максимальное время работы пользователя в системе;
- максимальное число последовательно осуществленных безуспешных попыток регистрации в системе.

Если в системе обеспечивается поддержка политики обеспечения работоспособности, ТСВ должна поддерживать механизм управления доступностью системных ресурсов посредством введения квот и ограничений на объем потребляемых ресурсов.

3. *Дополнение.* ТСВ должна содержать средства для однозначной идентификации каждого параметра политики безопасности. Кроме того должна быть предусмотрена возможность получения списка атрибутов безопасности для каждого пользователя и списка пользователей, ассоциированных с каждым атрибутом безопасности.

Должна обеспечиваться возможность управления атрибутами политики безопасности субъектов, включая привилегии, атрибуты произвольного и нормативного управления доступом, а также централизованного контроля, назначения и снятия атрибутов политики безопасности.

4. *Без изменений.*

5. *Без изменений.*

Уровень **SM-3**. Управление безопасностью в соответствии с политикой безопасности.

1. *Дополнение.* Режим технического обслуживания системы должен включать средства инициализации параметров идентификации, аутентификации, регистрации в системе и назначения полномочий администратора системы.

2. *Дополнение.* В случае совместного использования нескольких методов аутентификации ТСВ должна предоставлять администратору возможность определять методы аутентификации пользователей в зависимости от соответствующих атрибутов политики безопасности.

Если ТСВ поддерживает одновременно несколько сеансов для одного пользователя, администратор должен иметь возможность ограничить число одновременных регистраций для каждого пользователя в зависимости от его атрибутов безопасности.

ТСВ должна позволять администратору ограничивать регистрацию для

пользователя с определенным идентификатором или с определенного терминала после заданного количества неуспешных попыток регистрации с помощью этого идентификатора или терминала.

3. *Дополнение.* ТСВ должна автоматически приостанавливать полномочия пользователей в случае, если они не использовались в течении заданного администратором периода времени. ТСВ также должна обеспечивать автоматическое возобновление приостановленных полномочий по истечении указанного администратором времени.

4. *Дополнение.* ТСВ должна поддерживать разделение функций оператора и администратора. Функции оператора должны ограничиваться управлением внешними устройствами.

5. *Без изменений.*

Уровень **SM-4**. Расширенное управление безопасностью.

1. *Без изменений.*

2. *Без изменений.*

3. *Дополнение.* ТСВ должна содержать доверенные средства администрирования системы, осуществляющие контроль:

- конфигурации системы и регистрации пользователей;
- корректности инсталляции системы;
- отсутствия в ТСВ посторонних программ и данных.

ТСВ должна включать средства контроля безопасности начального состояния ТСВ после инициализации или восстановления.

ТСВ должна включать средства контроля соответствия между пользователями, субъектами, представляющими их в системе, и назначенным им атрибутам безопасности.

4. *Дополнение.* Средства контроля функционирования системы должны поддерживать разделение ролей администратора безопасности и аудитора, контролирующего администрирование. ТСВ должна выполнять заданные администратором действия только после их регистрации в журнале аудита. Не влияющие на безопасность системы действия администратора должны быть строго ограничены для обеспечения эффективного управления безопасностью.

5. *Без изменений.*

9. Мониторинг взаимодействий

Ранжирование требований к мониторингу взаимодействий производится по отношению к области применения мониторинга и степени детализации взаимодействий. На уровне **RM-1** мониторинг взаимодействий ограничивается только заданными подмножествами субъектов и объектов, обращения к которым контролируются политикой управления доступом. На уровне **RM-2** мониторинг взаимодействий должен применяться для всех субъектов и объектов. Уровень **RM-3** увеличивает степень детализации с помощью мониторинга атрибутов безопасности и статуса объектов, субъектов и ресурсов. Уровень **RM-4**, предназначенный для использования в системах, где действуют привилегированные процессы, предусматривает поддержку модели мониторинга взаимодействий привилегированных процессов.

Уровень **RM-1**. Мониторинг взаимодействий для заданных подмножеств субъектов и объектов.

1. ТСВ должна осуществлять мониторинг всех взаимодействий, в которых участвуют субъекты, объекты и ресурсы, включенные в спецификацию ТСВ. Мониторинг должен обеспечивать контроль взаимодействий в соответствии с политикой безопасности.

2. Мониторинг взаимодействий должен осуществляться для заданного подмножества субъектов, объектов и ресурсов, находящихся под контролем политики безопасности системы, а также для обращений к их атрибутам безопасности (правам доступа, уровням конфиденциальности, ролям, квотам и т.д.).

3. Мониторинг взаимодействий привилегированных субъектов должен осуществляться в соответствии с атрибутами безопасности этих субъектов.

Уровень **RM-2**. Мониторинг взаимодействий для всех субъектов и объектов.

1. *Без изменений.*

2. *Дополнение.* Мониторинг взаимодействий должен осуществляться для всех объектов, субъектов и ресурсов, и их атрибутов безопасности.

3. *Без изменений.*

Уровень **RM-3**. *Мониторинг взаимодействий и контроль атрибутов безопасности.*

1. *Без изменений.*

2. *Дополнение.* Требования мониторинга взаимодействий обращений к атрибутам субъектов, объектов и ресурсов распространяются на полное множество атрибутов (состояние, размер, режим использования).

3. *Без изменений.*

Уровень **RM-4**. Мониторинг взаимодействий привилегированных субъектов.

1. *Без изменений.*

2. *Без изменений.*

3. *Дополнение.* Мониторинг взаимодействий привилегированных субъектов должен осуществляться на основе модели безопасности и мониторинга, определенной для этих субъектов.

10. Логическая защита ТСВ

Ранжирование требований логической защиты ТСВ проводится на основе их возможностей по обеспечению безопасности ТСВ. Уровень **LP-1** содержит основные требования к изоляции ТСВ. На уровне **LP-2** эти требования расширяются за счет введения средств контроля целостности структур данных ТСВ и исключения влияния на состояние ТСВ со стороны непривилегированных пользователей. Эти требования призваны сделать невозможным применение злоумышленником средств проникновения в ТСВ. На уровне **LP-3** вводится требование синхронности контроля целостности ТСВ.

Уровень **LP-1**. Базовые средства изоляции ТСВ.

ТСВ должна функционировать внутри собственного домена, изолирован-

ного от остальных компонентов системы. Изоляция домена ТСВ должна обеспечивать защиту от внешних воздействий, модификации программ или данных ТСВ.

1. Изоляция компонентов ТСВ должна включать:

- изоляцию адресного пространства ТСВ от адресного пространства непривилегированных субъектов таким образом, чтобы они не могли получить доступ по чтению и записи к программам и данным ТСВ;

- взаимодействие между доменом ТСВ и остальными компонентами системы должно осуществляться таким образом, чтобы неконтролируемый обмен информацией с ТСВ был невозможен;

- параметры, передаваемые в ТСВ, должны контролироваться на допустимость их значений или принадлежность к адресному пространству ТСВ.

2. Для обеспечение надежности изоляции ТСВ права доступа к объектам, переданным в ТСВ в качестве параметров, должны проверяться на соответствие требуемым, а обращения к объектам ТСВ со стороны средств, обеспечивающих изоляцию, контролироваться монитором пересылок.

Уровень **LP-2**. Изоляция и контроль целостности ТСВ.

1. *Без изменений.*

2. *Дополнение.* Средства защиты ТСВ также должны осуществлять контроль целостности структур данных ТСВ и предотвращать влияние на них со стороны непривилегированных пользователей.

3. Контроль целостности структур данных ТСВ с помощью вычисления функции-инварианта, определенной на множестве переменных, объектов и функций, должен осуществляться до и после любого обращения к ТСВ.

4. Для предотвращение влияния на ТСВ со стороны непривилегированных пользователей необходимо обеспечить, чтобы любое обращение пользователя к ТСВ не приводило к нарушениям в обработке запросов остальных пользователей.

Уровень **LP-3**. Изоляция и синхронный контроль целостности ТСВ.

1. *Без изменений.*

2. *Без изменений.*

3. *Без изменений.*

4. *Дополнение.* Защита ТСВ должна обеспечивать синхронность функций контроля целостности.

5. Синхронность функций контроля целостности означает, что действия, основанные на результатах проверки целостности, осуществляются непосредственно после завершения процесса проверки, и между этими событиями состояние ТСВ измениться не может.

11. Физическая защита ТСВ

Ранжирование требований физической защиты ТСВ производится на основе обеспечиваемого уровня защиты, то есть возможности предвидеть, обнаруживать и предотвращать атаки на систему на физическом уровне.

На уровне **PP-1** от средств обеспечения физической защиты требуется применение административных мер и контроля среды функционирования. На уровне **PP-2** выдвигается требование к устройствам распознавать попытки физи-

ческого вмешательства в их работу. Уровень **PP-3** требует наличия средств противодействия атакам на конфиденциальность и целостность системы, а также изменениям в среде функционирования.

Уровень **PP-1**. Административные меры и контроль среды функционирования.

1. Должны быть определены административные меры и параметры физического контроля среды функционирования, необходимые для обеспечения защиты TCB.

2. Должны иметься и надлежащим образом применяться средства и устройства, необходимые для осуществления физического контроля за компонентами TCB.

Уровень **PP-2**. Обнаружение атак на физическом уровне.

1. *Без изменений.*

2. *Дополнение.* Средства и устройства, осуществляющие физический контроль, должны обеспечивать однозначное обнаружение физического воздействия на TCB. Эти устройства должны быть надежны и устойчивы к непосредственному физическому воздействию.

Уровень **PP-3**. Противодействие атакам на физическом уровне и неблагоприятным изменениям в среде функционирования.

1. *Без изменений.*

2. *Дополнение.* Должны иметься средства противодействия атакам на физическом уровне, их характеристики должны соответствовать требованиям политики безопасности. Для обеспечения конфиденциальности эти устройства должны противостоять попыткам кражи и исследования компонентов TCB с помощью физического воздействия, подслушивания, перехвата и анализа излучений. Для обеспечения целостности эти устройства должны противодействовать несанкционированному изменению состава аппаратного обеспечения, нарушению его функционирования, а также воздействиям на хранящуюся в системе информацию механическими или электромагнитными методами. Для обеспечения работоспособности системы эти устройства должны противодействовать возникновению ситуаций, затрудняющих обслуживание пользователей (вибрации, вода, огонь и другие формы физических воздействий).

12. Самоконтроль TCB

Требования самоконтроля TCB ранжируются на основе перечня контролируемых элементов (аппаратное, программное и специальное обеспечение) и возможностей механизмов проверки (периодичность, длительность, глубина проверки).

На уровне **SC-1** представлены минимальные требования к самоконтролю TCB, предполагается, что их выполнения будет достаточно для большинства коммерческих приложений. Уровень **SC-2** содержит расширенные требования, включающие в себя тестирование при включении питания, загрузке системы, а также управляемые оператором средства тестирования, применяемые для периодического контроля функционирования элементов TCB. На уровне **SC-3** появляются тре-

бования к тестированию программных компонентов ТСВ. На уровне **SC-4** требуется, чтобы тестирование осуществлялось регулярно на протяжении всего периода функционирования системы.

Уровень **SC-1**. Минимальный самоконтроль.

ТСВ должна включать аппаратные и/или программные средства периодического контроля целостности и корректности функционирования собственных аппаратных и специальных компонентов.

Уровень **SC-2**. Базовые механизмы самоконтроля.

Дополнение. В состав средств контроля должны входить: тестирование при включении питания, тестирование в ходе загрузки системы и средства тестирования, управляемые оператором.

Тесты при включении питания должны осуществлять проверку всех аппаратных и специальных компонентов ТСВ, включая оперативную память, шины, соединения, разъемы, управляющие контроллеры, процессор, адаптеры дисковых накопителей и других устройств, коммуникационные порты, консоль и клавнатуру. Эти тесты также должны осуществлять проверку всех компонентов, используемых для выполнения тестов при загрузке системы, и тестов, управляемых оператором.

Тесты при загрузке системы должны включать в себя проверку компонентов центрального процессора (арифметические и логические устройства, математический сопроцессор, устройство декодирования инструкций, контроллер прерываний, кэш, буфер трансляции адресов, внутренние и внешние шины), а также системной шины, контроллеров оперативной памяти и устройств, используемых в ходе тестов, управляемых оператором, и при удаленном тестировании системы.

Выполняемые оператором тесты должны обеспечивать однократное или многократное тестирование компонентов ТСВ и системы в целом, регистрацию результатов тестирования, и, в случае выявления неисправности, выполнять специальные процедуры локализации неисправностей и уведомлять оператора.

Уровень **SC-3**. Тестирование программных средств.

Дополнение. Должны иметься управляемые и конфигурируемые программные или специальные средства тестирования целостности и корректности программных компонентов ТСВ — программ, данных и носителей информации. Эти средства должны включать проверку контрольных сумм и другие механизмы контроля.

Уровень **SC-4**. Регулярное тестирование программных средств.

Дополнение. Тесты контроля целостности и корректности функционирования программных компонентов ТСВ должны выполняться при каждом изменении содержания или структуры этих компонентов, возникающих при сбоях и отказах, произошедших из-за действий непривилегированных субъектов.

13. Инициализация и восстановление ТСВ

Требования инициализации и восстановления безопасного состояния ТСВ ранжируются по отношению к уровню предоставляемых возможностей: ручное восстановление (уровни **TR-1** и **TR-2**), автоматическое (уровень **TR-3**), обнаруже-

ние потерь объектов пользователей (уровень **TR-4**), минимизация потерь объектов (уровень **TR-5**).

Уровень **TR-1**. Минимальные требования восстановления и инициализации.

1. ТСВ должна содержать механизмы, обеспечивающие гарантированное восстановление безопасного состояния ТСВ после сбоев без нарушения функций защиты.

Уровень **TR-2**. Базовые средства восстановления и инициализации ТСВ.

1. *Без изменений.*

2. В случае невозможности автоматического восстановления и реинициализации безопасного состояния, ТСВ должна переходить в особое состояние, в котором доступ может осуществляться только с помощью специальных административных процедур, с помощью которых можно осуществить восстановление ТСВ вручную, без нарушений функций защиты.

Уровень **TR-3**. Автоматическое восстановление и инициализация ТСВ.

1. *Без изменений.*

2. *Изменение.* ТСВ должна включать средства автоматического восстановления безопасного состояния ТСВ после ошибок и сбоев. Эти средства должны по возможности исключать потерю системных и пользовательских объектов. Должны быть определены требования, или правила политики безопасности, позволяющие подтвердить безопасность ТСВ после восстановления.

Уровень **TR-4**. Обнаружение потерь объектов.

1. *Без изменений.*

2. *Дополнение.* ТСВ должна включать специальную контрольную функцию восстановления, способную обнаруживать повреждение или разрушение объектов в результате сбоев, и предупреждать об этом пользователей.

Уровень **TR-5**. Минимизация потерь объектов.

1. *Без изменений.*

2. *Дополнение.* Все вызываемые извне функции и операции ТСВ должны быть атомарными, то есть, либо завершаться полным выполнением указанных действий, либо, при возникновении сбоев, сохранять исходное состояние используемых объектов, субъектов и ресурсов. Применение атомарных функций позволяет минимизировать искажения и потери объектов при сбоях системы.

14. Ограничение привилегий при работе с ТСВ

Требования ограничения привилегий при работе с ТСВ ранжируются на основе детализации описания привилегий, ассоциированных с отдельными функциями или группами функций ТСВ (уровень **PO-1**), с отдельными компонентами (модулями) ТСВ и административными ролями (**PO-2**), с отдельными операциями (**PO-3**) и динамически изменяющихся в ходе выполнения операций (**PO-4**).

Уровень **PO-1**. Назначение привилегий для выполнения функций ТСВ.

1. Должны быть определены привилегии, необходимые для выполнения отдельных функций ТСВ или их групп. Также должны быть определены привиле-

гированные функции и объекты ТСВ, такие как файлы регистрации пользователей, файлы паролей, файлы содержащие уровни безопасности пользователей и объектов, списки ролей пользователей, файлы журнала аудита.

2. Должен быть назначен минимальный уровень привилегий, необходимый и достаточный для осуществления доступа к установленным привилегированным функциям и объектам ТСВ.

Уровень **PO-2**. Назначение привилегий доступа к компонентам ТСВ.

1. *Дополнение.* Должна обеспечиваться возможность назначения необходимых привилегий действиям, выполняемым в ТСВ привилегированными пользователями (администраторами).

2. *Изменение.* Всем функциям и компонентам (модулям) ТСВ должна быть поставлен в соответствие минимальный уровень привилегий, необходимый и достаточный для их доступа к ним.

3. Должна быть обеспечена поддержка реализации привилегий модулей ТСВ с помощью низкоуровневых процедур и механизмов.

Уровень **PO-3**. Назначение привилегий для выполнения отдельных операций.

1. *Без изменений.*

2. *Дополнение.* Должны быть определены привилегии, необходимые для выполнения отдельных операций ТСВ. Для каждой операции должен быть назначен минимальный уровень привилегий, необходимый и достаточный для ее выполнения.

3. *Изменение.* Должна быть обеспечена поддержка реализации привилегий для отдельных операций ТСВ с помощью низкоуровневых процедур и механизмов.

Уровень **PO-4**. Динамическое назначение привилегий для выполнения отдельных операций.

1. *Без изменений.*

2. *Дополнение.* Установленные привилегии должны использоваться всеми функциональными компонентами ТСВ для контроля и ограничения распространения ошибок в работе механизмов защиты и предоставления полномочий, которые могут повлечь за собой нарушение политики безопасности. Должны быть определены функции ТСВ, позволяющие при необходимости динамически повышать привилегии отдельных операций ТСВ (но не выше заданной границы) и автоматически их понижать по завершению выполнения соответствующих операций. Эти меры должны ограничивать использование высокопривилегированных операций ТСВ, потенциально предоставляющих пользователю возможности использования этих привилегий для нарушения политики безопасности.

3. *Без изменений.*

15. Простота использования ТСВ

Ранжирование требований данной группы отражает имеющиеся возможности управления конфигурацией ТСВ. На уровне **EU-1** сформулированы общие требования, отражающие необходимость наличия развитых средств управления

безопасностью системы, вместо использования обычных редакторов для модификации параметров безопасности или содержимого файлов регистрации пользователей. Требования к функциональности средств администрирования расширяются на уровне **EU-2** посредством введения возможности установки атрибутов безопасности по умолчанию для некоторых субъектов и объектов и наличия средств, позволяющих приложениям обеспечить собственную защиту и защиту своих объектов от несанкционированного использования. На уровнях **EU-3** и **EU-4** требования усиливаются и расширяются за счет увеличения множеств субъектов и объектов, на которые они распространяются для стандартной и полной конфигурации системы.

Уровень **EU-1**. Простота управления безопасностью.

1. TCB должна обеспечивать поддержку функций администрирования. Должна быть предусмотрена возможность задания значений параметров безопасности по умолчанию для средств администрирования.

Уровень **EU-2**. Простота разработки приложений.

1. *Дополнение.* TCB должна обеспечивать автоматическую установку атрибутов безопасности по умолчанию для определенных субъектов и объектов, и возможность модификации этих атрибутов.

2. TCB должна предусматривать четко определенный программный интерфейс взаимодействия со всеми принятыми в системе политиками безопасности для поддержки приложений, которые могут обеспечить поддержку этих политик безопасности на прикладном уровне. TCB должна предоставлять пользователю возможность понижения полномочий используемых приложений.

Уровень **EU-3**. Простота использования стандартной конфигурации системы.

1. *Изменение.* TCB должна обеспечивать автоматическую установку атрибутов безопасности по умолчанию для определенных субъектов, объектов и служб, присутствующих в стандартной конфигурации системы, а также возможность модификации этих атрибутов.

2. *Без изменений.*

Уровень **EU-4**. Простота использования полной конфигурации системы.

1. *Изменение.* TCB должна обеспечивать автоматическую установку атрибутов безопасности по умолчанию для всех субъектов, объектов и служб системы, а также возможность модификации этих атрибутов.

2. *Без изменений.*

Приложение 2. Ранжированные требования "Канадских критериев безопасности компьютерных систем"

В настоящем приложении содержится перечень функциональных требований и требований к адекватности реализации "Канадских критериев безопасности компьютерных систем". Данное изложение отражает только принципиальную суть требований и не претендует на перевод стандарта как руководящего документа.

Описание каждого раздела требований начинается с его краткого описания и обзора предусмотренных уровней ранжирования. Идентификаторы уровней сохранены в том виде, в каком они присутствуют в первоисточнике. Специальный нулевой уровень зарезервирован для систем, реализующих данные требования в степени недостаточной для соответствия какому-либо уровню. Некоторые уровни функциональных критериев зависят от других, и для того, чтобы удовлетворить требованиям этих уровней, необходимо соблюсти не только приведенные в них требования, но и требования сопряженных разделов функциональных критериев и критериев адекватности реализации в рамках указанных уровней. Для таких уровней после текста требований указываются идентификаторы уровней, сопряженных с ними.

1. Критерии конфиденциальности

Критерии конфиденциальности регламентируют защиту ресурсов компьютерной системы от несанкционированного доступа. В качестве средств обеспечения конфиденциальности рассматриваются контроль скрытых каналов, произвольное и нормативное управление доступом и контроль за повторным использованием ресурсов.

1.1 Контроль скрытых каналов

Контроль скрытых каналов позволяет выявить присутствие в системе потоков информации, которые не могут контролироваться другими средствами защиты. Ранжирование требований производится по четырем уровням в зависимости от степени анализа наличия скрытых каналов и возможностей по контролю и подавлению.

Уровень CC-0. Недостаточный контроль скрытых каналов.

Данный уровень зарезервирован для систем с примитивными возможностями контроля скрытых каналов, которые не удовлетворяют требованиям более высоких уровней.

Уровень CC-1. Анализ скрытых каналов.

Должно быть проведено исследование наличия в компьютерной системе скрытых каналов. Каждый обнаруженный в аппаратных, программных или специ-

альных средствах системы скрытый канал утечки информации должен быть документирован.

Максимальная пропускная способность каждого скрытого канала должна быть указана в документации. Кроме того, для скрытых каналов, которые могут быть использованы совместно, должна быть указана максимальная пропускная способность их совокупности.

Сопряженные уровни: **Т3**.

Уровень **СС-2**. Контроль скрытых каналов.

Дополнение. ТСВ должна позволять осуществлять контроль за использованием заданного множества скрытых каналов.

Сопряженные уровни: **Т3,WA-1**.

Уровень **СС-3**. Подавление скрытых каналов.

Изменение. Каждый выявленный скрытый канал должен быть ликвидирован.

Сопряженные уровни: **Т3**.

1.2 Произвольное управление доступом

Произвольное управление доступом позволяет администраторам и авторизованным пользователям управлять потоками информации от объектов системы к пользователям. Ранжирование требований проводится на основании возможностей механизма контроля и степени его детализации.

Уровень **CD-0**. Недостаточное произвольное управление доступом.

Уровень зарезервирован для систем с наличием отдельных элементов произвольного управления доступом, но не удовлетворяющих требованиям более высоких уровней.

Уровень **CD-1**. Минимальные требования к произвольному управлению доступом.

ТСВ должна обеспечивать *реализацию политики произвольного управления* доступом по отношению к заданному подмножеству объектов компьютерной системы.

Предоставление доступа к объекту должно производится на основании значений тэгов объекта и процесса, запросившего доступ.

Управление параметрами доступа должно выполняться средствами ТСВ на основании тэга пользователя.

Теги защищенных объектов должны назначаться при их создании или инициализации.

Правила назначения тегов при экспорте-импорте объектов должны являться составной частью политики произвольного управления доступом.

Сопряженные уровни: **CR-1,WI-1**.

Уровень **CD-2**. Базовая политика произвольного управления доступом.

Изменение. Предоставление доступа к объекту должно производится на основании значений тэгов объекта и пользователя, ассоциированного с процессом, запросившем доступ.

Дополнение. Политика произвольного управления доступом должна предусматривать наличие частично определенной матрицы доступа для тегов всех пользователей и тегов защищенных объектов.

Сопряженные уровни: **CR-1,WI-1**.

Уровень **CD-3**. Гибкая политика произвольного управления доступом.

Изменение. Политика произвольного управления доступом должна предусматривать наличие полностью определенной матрицы доступа для тегов всех пользователей и тегов защищенных объектов.

Сопряженные уровни: **CR-1,WI-1**.

Уровень **CD-4**. Расширенная политика произвольного управления доступом

Изменение. Предоставление доступа к объекту должно производится на основании значений тэгов объекта, процесса, запросившего доступ, и пользователя, ассоциированного с этим процессом.

Изменение. Политика произвольного управления доступом должна предусматривать наличие полностью определенной матрицы доступа для тегов всех пользователей и процессов и тегов защищенных объектов.

Сопряженные уровни: **CR-1,WI-1**.

1.3 Нормативное управление доступом

Нормативное управление доступом, также как и произвольное управление доступом, позволяет администраторам и авторизованным пользователям управлять потоками информации от объектов системы к пользователям. Ранжирование требований проводится на основании степени их детализации, множества защищаемых объектов и предоставляемых возможностей по управлению доступом.

Уровень **CM-0**. Недостаточное нормативное управление доступом.

Уровень зарезервирован для систем с примитивными возможностями нормативного управления доступом, не удовлетворяющих требованиям более высоких уровней.

Уровень **CM-1**. Минимальные требования к нормативному управлению доступом.

TCB должна обеспечивать реализацию политики нормативного управления доступом по отношению к заданному подмножеству объектов компьютерной системы.

Предоставление доступа к объекту должно производится на основании значений тэгов объекта и процесса, запросившего доступ.

Управление параметрами доступа должно осуществляться средствами TCB только администратором и авторизованными пользователями.

Теги защищенных объектов должны назначаться при их создании или инициализации.

Правила назначения тегов при экспорте-импорте объектов должны являться составной частью политики нормативного управления доступом.

Сопряженные уровни: **CR-1,IS-1**.

Уровень **CM-2**. Базовая политика нормативного управления доступом.

Изменение. Предоставление доступа к объекту должно производится на основании значений тэгов объекта и пользователя, ассоциированного с процессом, запросившем доступ.

Сопряженные уровни: **CR-1**,**IS-1**,**WI-1**.

Уровень **CM-3**. Гибкая политика нормативного управления доступом.

Изменение. Политика нормативного управления доступом должна применяться ко всем объектам компьютерной системы.

Сопряженные уровни: **CR-1**,**IS-1**,**WI-1**.

Уровень **CM-4**. Расширенная политика нормативного управления доступом.

Изменение. Предоставление доступа к объекту должно производится на основании значений тэгов объекта, процесса, запросившего доступ, и пользователя, ассоциированного с этим процессом.

Сопряженные уровни: **CR-1**,**IS-1**,**WI-1**.

1.4 Повторное использование объектов

Контроль за повторным использованием объектов позволяет обеспечить безопасное использование разделяемых объектов, одновременно или последовательно доступных нескольким процессам. Контроль должен предотвращать сохранение в разделяемых объектах остаточной информации после завершения их использования одним процессом и перед предоставлением доступа к ним другому процессу.

Уровень **CR-0**. Недостаточное нормативное управление доступом.

Уровень зарезервирован для систем с примитивными возможностями контроля за повторным использованием объектов.

Уровень **CR-1**. Безопасное повторное использование объектов.

TCB должна обеспечивать политику безопасного повторного использования объектов. Эта политика должна применяться ко всем объектам, допускающим совместное использования.

Все полномочия доступа к разделяемому объекту должны быть отозваны перед предоставлением его в повторное использование.

Вся информация, содержащаяся в разделяемом объекте, должна быть уничтожена перед предоставлением его в повторное использование.

2. Критерии целостности

Критерии целостности определяют возможности компьютерной системы по обеспечению собственной целостности и целостности обрабатываемой и хранимой в ней информации. Критерии целостности предусматривают поддержку доменов целостности, произвольного и нормативного управления целостностью, разделения ролей, обеспечение физической целостности компонентов компьютерной системы, а также средств отката и самотестирования.

2.1 Домены целостности

Поддержка доменов целостности позволяет ТСВ осуществлять защиту собственных компонентов и контролировать целостность и осуществлять управление защищенными объектами.

Уровень **IB-0**. Недостаточная поддержка доменов целостности.

Зарезервирован для систем, осуществляющих поддержку доменов целостности в недостаточной степени и не удовлетворяющих требованиям более высоких уровней.

Уровень **IB-1**. Домены целостности ТСВ.

ТСВ должна поддерживать политику обеспечения целостности доменов, предусматривающую идентификацию доменов (как доменов ТСВ, так и остальных) и средства их изоляции.

ТСВ должна обеспечивать защиту собственных доменов от внешних воздействий.

Уровень **IB-2**. Полный контроль доступа.

Дополнение. Архитектура ТСВ должна гарантировать, что доступ к сервису средств безопасности и к защищенным объектам возможен только посредством ТСВ.

2.2 Произвольное управление целостностью

Произвольное управление целостностью позволяет администраторам и авторизованным пользователям управлять потоками информации от пользователей к объектам системы. Ранжирование требований проводится на основании возможностей механизма контроля и степени его детализации.

Уровень **ID-0**. Недостаточное произвольное управление целостностью.

Уровень зарезервирован для систем с наличием отдельных элементов произвольного управления целостностью, но не удовлетворяющих требованиям более высоких уровней.

Уровень **ID-1**. Минимальные требования к произвольному управлению целостностью.

ТСВ должна обеспечивать реализацию политики произвольного управления целостностью, по отношению к заданному подмножеству объектов компьютерной системы.

Предоставление доступа к объекту должно производится на основании значений тэгов объекта и пользователя.

Управление параметрами доступа должно выполняться средствами ТСВ на основании тэга пользователя.

Теги защищенных объектов должны назначаться при их создании или инициализации.

Правила назначения тегов при экспорте-импорте объектов должны являться составной частью политики произвольного управления целостностью.

Сопряженные уровни: **CR-1,WI-1**.

Уровень **ID-2**. Базовая политика произвольного управления целостностью.

Изменение. Предоставление доступа к объекту должно производится на основании значений тэгов объекта и процесса, запросившем доступ.

Дополнение. Политика произвольного управления целостностью должна предусматривать наличие частично определенной матрицы доступа для тегов всех процессов и тегов защищенных объектов.

Сопряженные уровни: **CR-1,WI-1**.

Уровень **ID-3**. Гибкая политика произвольного управления целостностью.

Изменение. Политика произвольного управления целостностью должна предусматривать наличие полностью определенной матрицы доступа для тегов всех процессов и тегов защищенных объектов.

Сопряженные уровни: **CR-1,WI-1**.

Уровень **ID-4**. Расширенная политика произвольного управления целостностью.

Изменение. Предоставление доступа к объекту должно производится на основании значений тэгов объекта, процесса, запросившего доступ, и пользователя, ассоциированного с этим процессом.

Изменение. Политика произвольного управления целостностью должна предусматривать наличие полностью определенной матрицы доступа для тегов всех пользователей и процессов и тегов защищенных объектов.

Сопряженные уровни: **CR-1,WI-1**.

2.3 Нормативное управление целостностью

Нормативное управление целостностью, также как и произвольное управление целостностью, позволяет администраторам и авторизованным пользователям управлять потоками информации от пользователей к объектам системы. Ранжирование требований проводится на основании степени их детализации, множества защищаемых объектов и предоставляемых возможностей по управлению доступом.

Уровень **IM-0**. Недостаточное нормативное управление целостностью.

Уровень зарезервирован для систем с примитивными возможностями нормативного управления целостностью, не удовлетворяющих требованиям более высоких уровней.

Уровень **IM-1**. Минимальные требования по нормативному управлению целостностью.

ТСВ должна обеспечивать реализацию политики нормативного управления целостностью по отношению к заданному подмножеству объектов компьютерной системы.

Предоставление доступа к объекту должно производится на основании значений тэгов объекта и пользователя.

Управление параметрами доступа должно осуществляться средствами ТСВ только администратором и авторизованными пользователями.

Теги защищенных объектов должны назначаться при их создании или инициализации.

Правила назначения тегов при экспорте-импорте объектов должны являться составной частью политики нормативного управления целостностью.

Сопряженные уровни: **CR-1,IS-1,WI-1**.

Уровень **IM-2**. Базовая политика нормативного управления целостностью.

Изменение. Предоставление доступа к объекту должно производится на основании значений тегов объекта и процесса, запросившем доступ.

Сопряженные уровни: **CR-1,IS-1**.

Уровень **IM-3**. Гибкая политика нормативного управления целостностью.

Изменение. Политика нормативного управления целостностью должна применяться ко всем объектам компьютерной системы.

Сопряженные уровни: **CR-1,IS-1**.

Уровень **IM-4**. Расширенная политика нормативного управления целостностью.

Изменение. Предоставление доступа к объекту должно производится на основании значений тегов объекта, процесса, запросившего доступ, и пользователя, ассоциированного с этим процессом.

Сопряженные уровни: **CR-1,IS-1,WI-1**.

2.4 Физическая целостность

Критерии физической целостности определяют периметр ТСВ регламентируют возможности средств физической защиты ТСВ. Задачей средств обеспечения физической целостности является выявление, ограничение и предотвращение несанкционированного физического доступа к внутренним элементам компьютерной системы, предотвращение несанкционированного использования, модификации или подмены ее физических компонентов. Требования к средствам физической защиты ранжируются в зависимости от типа обеспечиваемой защиты и средств, которые необходимо потратить злоумышленнику на ее преодоление.

Уровень **IP-0**. Недостаточная физическая целостность.

Зарезервирован для систем с минимальными возможностями обеспечения физической целостности, не удовлетворяющих требованиям более высоких уровней.

Уровень **IP-1**. Базовые требования по обеспечению физической целостности.

ТСВ должна поддерживать политику обеспечения физической целостности, определяющую периметр физической защиты ТСВ и множество компонентов компьютерной системы, к которым данная политика применяется.

Периметр физической защиты ТСВ должен быть защищен с помощью специальных средств, позволяющих обнаруживать несанкционированное использование, физический доступ, модификацию или подмену защищенных компонентов.

Уровень **IP-2**. Регулярная физическая целостность.

Дополнение. ТСВ должна включать средства обнаружения или противодействия попыткам проникновения через все входы, расположенные на периметре физической защиты ТСВ.

Дополнение. В случае преодоления заграждений периметра безопасности ТСВ должна выполнить действия, предотвращающие нарушение политики безопасности, принятой в компьютерной системе.

Уровень **IP-3**. Расширенная физическая целостность.

Дополнение. Экстремальные ситуации, возникающие вследствие стихийных явлений, не должны приводить к разрушению защищенных компонентов компьютерной системы или должны вызывать предусмотренную реакцию ТСВ, направленную на предотвращение нарушения политики безопасности системы.

Уровень **IP-4**. Полная физическая целостность.

Дополнение. Все компоненты компьютерной системы должны быть защищены механизмами обнаружения и противодействия попыткам несанкционированного использования, физического доступа, изменения или подмены таким образом, чтобы физическое вмешательство в их работу было невозможным, или вызывало предусмотренную реакцию ТСВ, направленную на предотвращение нарушений политики безопасности.

2.5 Возможность осуществления отката

Откат обеспечивает возможность отмены последовательности осуществленных действий и возвращение объектов компьютерной системы к исходному состоянию. Ранжирование критериев этого раздела производится в зависимости от уровня конкретизации объектов и множества операций, которые могут быть отменены.

Уровень **IR-0**. Недостаточные возможности осуществления отката.

Зарезервированным для систем с примитивными возможностями отката, не удовлетворяющих требованиям более высоких уровней.

Уровень **IR-1**. Ограниченные возможности осуществления отката.

ТСВ должна поддерживать политику осуществления отката — возврата к предыдущему для определенного множество объектов.

Политика осуществления отката должна обеспечиваться автоматическими средствами, предоставляющими администратору и авторизованным пользователям возможность отмены действий заданного множества операций, над защищенными объектами за определенный период времени, и возврата этих объектов в состояние, в котором они находились до выполнения этих операций.

Сопряженные уровни: **WI-1**.

Уровень **IR-2**. Расширенные возможности осуществления отката.

Изменение. Автоматические средства обеспечения отката должны позволять отменять действия всех операций с защищенными объектами.

Сопряженные уровни: **WI-1**.

2.6 Разделение ролей

Разделение ролей позволяет распределить ответственность за действия в системе, ограничить возможный ущерб безопасности системы в случае злоупотребления предоставленными правами и установить максимально допустимый предел полномочий пользователей и администратора. Требования ранжируются в зависимости от степени детализации.

Уровень **IS-0**. Недостаточное разделение ролей.

Зарезервирован для систем с примитивными возможностями разделения ролей, не удовлетворяющих требованиям более высоких уровней.

Уровень **IS-1**. Базовые требования по разделению ролей.

ТСВ должна поддерживать политику разделения ролей, регламентирующую административные и неадминистративные роли пользователей и соответствующие им действия и функции.

ТСВ *должна обеспечивать разделение административных и неадминистративных функций.*

Пользователи должны получать возможность выполнения административных действий только находясь в роли администратора.

Сопряженные уровни: **WI-1**.

Уровень **IS-2**. Административное разделение ролей.

Дополнение. Политика разделения ролей должна предусматривать наличие как минимум двух отдельных административных ролей: администратор безопасности системы и администратор системы, не имеющий возможностей по управлению безопасностью.

Дополнение. Действия, доступные для выполнения в конкретной роли, *должны быть минимизированы по своему составу и включать только* те, которые необходимы для реализации функций, соответствующих данной роли.

Изменение. Пользователи должны иметь возможность выполнения функций любой роли, а не только административной, только находясь в этой роли.

Дополнение. ТСВ должна гарантировать, что пользователю, находящемуся в определенной роли, доступны те, и только те функции и действия, которые предусмотрены этой ролью.

Сопряженные уровни: **WI-1**.

Уровень **IS-3**. Разделение ролей на основе привилегий пользователей.

Дополнение. Политика разделения ролей должна предусматривать наличие множества различных ролей пользователей, различающихся уровнями привилегий.

Сопряженные уровни: **WI-1**.

2.7 Самотестирование.

Самотестирование ТСВ обеспечивает корректность выполнения операций и надежное функционирование элементов компьютерной системы. Требования ранжируются в зависимости от возможностей механизмов самоконтроля своевременно обнаруживать *некорректное функционирование компонентов компьютерной системы.*

Уровень **IT-0**. Недостаточное самотестирование.

Зарезервирован для систем с примитивными возможностями самотестирования, не удовлетворяющих требованиям более высоких уровней.

Уровень **IT-1**. Базовое самотестирование.

ТСВ должна поддерживать *политику самотестирования*, определяющую возможности системы по периодическому контролю корректности функционирования ТСВ.

Состав тестов и методика проведения тестирования должны быть описаны в руководстве по управлению безопасностью системы.

Уровень **IT-2**. Самотестирование при загрузке или запуске.

Дополнение. ТСВ должна выполнять набор контролирующих тестов в процессе запуска или загрузки системы для проверки правильности функционирования критичных компонентов.

Уровень **IT-3**. Самотестирование в процессе работы.

Изменение. ТСВ должна обеспечивать выполнение контроля правильности функционирования критичных компонентов не только при запуске или загрузке, но и периодически в процессе функционирования компьютерной системы.

3. Критерии работоспособности

Критерии работоспособности регламентируют работу средств, обеспечивающих доступность компьютерной системы и ее ресурсов для авторизованных пользователей. В качестве мер обеспечения работоспособности рассматриваются контроль за распределением ресурсов системы, обеспечение устойчивости системы к отказам и сбоям, обеспечение живучести и восстановления системы в условиях выхода из строя ее компонентов.

3.1 Контроль за распределением ресурсов

Контроль за распределением ресурсов позволяет ТСВ управлять использованием компьютерной системы. Требования ранжируются в зависимости от контролируемых ресурсов и предоставляемых возможностей.

Уровень **AC-0**. Недостаточный контроль за распределением ресурсов.

Зарезервирован для систем с примитивными возможностями *контроля за* распределением ресурсов, не удовлетворяющих требованиям более высоких уровней.

Уровень **AC-1**. Нормированное распределение ресурсов.

ТСВ должна поддерживать политику управления распределением ресурсов компьютерной системы, позволяющую установить нормы (квоты) на предоставление пользователям заданного подмножества ресурсов системы.

Изменение квот на выделение ресурсов может производится только администраторами и уполномоченными пользователями посредством ТСВ.

Сопряженные уровни: **IS-1**.

Уровень **AC-2**. Предотвращение отказов в обслуживании.

Изменение. Политика управления распределением ресурсов должна охватывать все ресурсы компьютерной системы.

Дополнение. Нормы и квоты на потребление ресурсов должны быть заданы таким образом, чтобы ни один пользователь не мог захватить все ресурсы системы

и сделать невозможным доступ остальных пользователей к сервису TCB и защищенным объектам.

Сопряженные уровни: **IS-1**.

Уровень **AC-3**. Разграничение ресурсов.

Изменение. Политика управления распределением ресурсов должна позволять устанавливать ограничения на потребление ресурсов как для отдельных пользователей так и для групп пользователей.

Изменение. Соответственно, нормы и квоты на потребление ресурсов должны быть заданы таким образом, чтобы не только один пользователь, но и группа пользователей не могла захватить все ресурсы системы и сделать невозможным доступ остальных пользователей к сервису TCB и защищенным объектам.

Сопряженные уровни: **IS-1**.

3.2 Устойчивость к отказам и сбоям

Устойчивость к отказам и сбоям позволяет обеспечивает работоспособность системы и доступность ее ресурсов при выходе из строя отдельных компонентов. Требования ранжируются в зависимости от обеспечиваемых возможностей замены компонентов без нарушения функционирования.

Уровень **AF-0**. Недостаточная устойчивость к отказам и сбоям.

Зарезервирован для систем с примитивными возможностями обеспечения устойчивости к отказам, не удовлетворяющих требованиям более высоких уровней.

Уровень **AF-1**. Замена отдельных компонентов в ходе функционирования.

TCB должна поддерживать политику обеспечения устойчивости к сбоям и отказам, для заданного множества компонентов, замена которых не требует прерывания функционирования системы.

Администратор или авторизованные пользователи должны иметь возможность осуществления замены защищенных компонентов.

Сопряженные уровни: **IS-1**,**AR-1**.

Уровень **AF-2**. Полная замена компонентов системы.

Изменение. Политика обеспечения устойчивости к отказам и сбоям должна охватывать все компоненты компьютерной системы и обеспечивать их замену без прерывания функционирования.

Сопряженные уровни: **IS-1**,**AR-1**.

3.3 Живучесть

Живучесть (robustness) системы характеризует ее возможности сохранять работоспособность и доступность ресурсов системы после отказа некоторых из ее компонентов. Фактически свойство живучести определяет возможность частично неисправных компонентов продолжить выполнять свои функции до их замены или прерывания функционирования системы. Требования ранжируются в зависимости количества неисправностей, при наличии которых сохраняется работоспособность системы, и от множества ресурсов, доступных в условиях выходе из строя компонентов системы.

Уровень **AR-0**. Недостаточная живучесть.

Зарезервирован для систем с примитивными возможностями обеспечения живучести, не удовлетворяющих требованиям более высоких уровней.

Уровень **AR-1**. Надежность системы при выходе из строя ограниченного множества компонентов

TCB должна поддерживать политику обеспечения живучести, определяющую набор защищенных компонентов системы и множество неисправностей этих компонентов, при возникновении которых система сохраняет работоспособность и продолжает функционировать.

Выход из строя отдельного защищенного компонента не должен нарушать доступность ресурсов системы, но в худшем случае может привести к деградации ее функциональных возможностей.

Должны быть четко определены неисправности, сбои и отказы, возникновение которых приводит к деградации функциональных возможностей системы или к отказам в обслуживании.

Система должна иметь средства оповещения администратора о выходе из строя защищенных компонентов.

Сопряженные уровни: **IS-1**.

Уровень **AR-2**. Надежность системы при выходе из строя любых компонентов системы с деградацией функциональных возможностей.

Изменение. Политика обеспечения живучести должна применяться по всем компонентам системы, а не только к их заданному набору.

Сопряженные уровни: **IS-1**.

Уровень **AR-3**. Надежность системы без нарушений ее функционирования.

Изменение. Выход из строя отдельного защищенного компонента не должен нарушать доступность ресурсов системы или приводить к деградации ее функциональных возможностей.

Сопряженные уровни: **IS-1**.

3.4 Восстановление

Средства восстановления позволяют вернуть TCB в безопасное состояние после отказов или сбоев. Требования ранжируются в зависимости от степени автоматизации процесса восстановления.

Уровень **AY-0**. Недостаточное восстановление.

Зарезервирован для систем с примитивными возможностями восстановления, не удовлетворяющих требованиям более высоких уровней.

Уровень **AY-1**. Ручное восстановление.

TCB должна поддерживать политику восстановления безопасного состояния, определяющую множество нарушений функционирования системы, после возникновения которых возможно восстановление состояния системы без нарушений политики безопасности.

После нарушения функционирования системы TCB должна обеспечить переход системы в специальное состояние временного останова, в котором только

администратор и соответствующим образом авторизованных пользователи могут выполнить действия по восстановлению нормального функционирования системы.

Должна быть регламентирована процедура ручного восстановления нормального функционирования системы без нарушения принятой политики безопасности.

Должны быть определены нарушения в работе системы, в случае возникновения которых необходима полная переустановка системы.

Сопряженные уровни: **IS-1**.

Уровень **AY-2**. Автоматическое восстановление.

Дополнение. После нарушения функционирования системы TCB должна определить какие автоматические процедуры могут быть использованы для восстановлению нормаьного функционирования системы.

Дополнение. Если это возможно, TCB должна выполнить автоматические процедуры и восстановить нормальное функционирование системы.

Изменение Если автоматическое восстановление невозможно, TCB должна обеспечить переход системы в специальное состояние временного останова, в котором только администратор и соответствующим образом авторизованных пользователи могут выполнить действия по восстановлению нормального функционирования системы.

Сопряженные уровни: **IS-1**.

Уровень **AY-3**. Селективное автоматическое восстановление.

Изменение. В случае, когда после нарушения функционирования системы не требуется ее полная перестановка, TCB должна осуществить автоматическое восстановление без потери доступности системных ресурсов, в худшем случае допускается деградация функциональных возможностей системы.

Сопряженные уровни: **IS-1**.

4. Критерии аудита.

Критерии аудита регламентируют работу средств, позволяющих установить ответственность пользователей за события в системе. Аудит обеспечивается средствами регистрации и учета, идентификации и аутентификации, а также прямого взаимодействия с TCB.

4.1 Регистрация и учет событий в системе

Регистрация и учет событий в системе позволяют выявить потенциально опасные действия пользователей. Требования ранжируются в зависимости от степени их детализации, сложности процесса анализа событий и возможности выявлять потенциальные угрозы безопасности.

Уровень **WA-0**. Недостаточная регистрация и учет событий в системе.

Зарезервирован для систем с примитивными возможностями регистрации и учета, не удовлетворяющих требованиям более высоких уровней.

Уровень WA-1. Регистрация и учет событий в системе.

TCB должна поддерживать политику регистрации и учета событий в системе, определяющую множество событий, подлежащих регистрации в журнале аудита.

ТСВ должна осуществлять минимальный контроль событий, влияющих на безопасность системы, и предоставлять журнал аудита посредством специального защищенного механизма для других компонентов компьютерной системы.

Журнал аудита должен содержать информацию о дате, времени, месте, типе и результате каждого регистрируемого события.

Журнал аудита должен содержать информацию, позволяющую идентифицировать пользователей, процессы и объекты, участвовавшие в зарегистрированных событиях.

Сопряженные уровни: **WI-1**.

Уровень **WA-2**. Регистрация и учет событий в системе и защита журнала аудита.

Изменение. ТСВ должна осуществлять минимальный контроль событий, влияющих на безопасность системы, поддерживать журнал аудита и обеспечивать его защиту от несанкционированного доступа, модификации и уничтожения.

Дополнение. Средства просмотра журнала аудита должны быть доступны администратору и авторизованным пользователям и обеспечивать поддержку проверки зарегистрированных событий.

Сопряженные уровни: **IS-1,WI-1**.

Уровень **WA-3**. Регистрация и учет событий в системе, защита журнала аудита и оповещение администратора.

Дополнение. ТСВ должна осуществлять мониторинг событий или их совокупности, возникновение которых является признаком возможного нарушения политики безопасности.

Дополнение. ТСВ должна обеспечить возможность незамедлительного оповещения администратора в случае возникновения угроз безопасности, а при постоянном их появлении прекратить выполнение операции, вызвавшей это событие, с минимальными последствиями для функционирования системы.

Сопряженные уровни: **IS-1,WI-1**.

Уровень **WA-4**. Детальная регистрация и учет событий в системе.

Изменение. ТСВ должна осуществлять детальный контроль событий, влияющих на безопасность системы, поддерживать журнал аудита и обеспечивать его защиту от несанкционированного доступа, модификации и уничтожения.

Изменение. Средства анализа журнала аудита должны быть доступны администратору и авторизованным пользователям и обеспечивать поддержку анализа зарегистрированных событий.

Сопряженные уровни: **IS-1,WI-1**.

Уровень **WA-5**. Выявление вторжений.

Дополнение. ТСВ должна осуществлять контроль попыток нарушения безопасности и вторжения в систему в режиме реального времени в соответствии с принятой в системе политики безопасности.

Сопряженные уровни: **IS-1,WI-1**.

4.2 Идентификация и аутентификация

Идентификация и аутентификация позволяют ТСВ проверить подлинность пользователей, пытающихся получить доступ к системе и ее ресурсам. Ранжиро-

вание требований выполняется в зависимости от функциональности возможностей *механизмов идентификации и аутентификации*.

Уровень **WI-0**. Недостаточная идентификация и аутентификация.

Зарезервирован для систем с примитивными возможностями идентификации и аутентификации, не удовлетворяющих требованиям более высоких уровней.

Уровень WI-1. Внешняя идентификация и аутентификация.

TCB должна поддерживать политику идентификации и аутентификации, определяющую набор атрибутов, ассоциированных с пользователем, и соответствующие механизмы контроля и управления этими атрибутами.

Каждый пользователь должен иметь уникальный идентификатор.

TCB должна содержать защищенный механизм, позволяющий получить идентификатор пользователя и осуществить его аутентификацию с помощью внешних средств, перед предоставлением ему возможности выполнения любых *действий в системе*.

Уровень **WI-2**. Индивидуальная идентификация и аутентификация.

Изменение. TCB должна содержать защищенный механизм, позволяющий *получить идентификатор пользователя и осуществить его аутентификацию* с помощью собственных средств TCB, перед предоставлением ему возможности выполнения любых действий в системе.

Дополнение. TCB должна обеспечивать защиту информации, применяемой для аутентификации, от несанкционированного доступа, модификации и уничтожения.

*Уровень **WI-3**. Множественная идентификация и аутентификация.*

Изменение. TCB должна содержать два и более защищенных механизма, позволяющих получить идентификатор пользователя и осуществить его аутентификацию с помощью собственных средств TCB, перед предоставлением ему возможности выполнения любых действий в системе.

4.3 Прямое взаимодействие с TCB

Прямое взаимодействие с TCB (Trusted Path) обеспечивает возможность непосредственного, конфиденциального взаимодействия между пользователем и TCB. Требования ранжируются в зависимости от гибкости механизмов, обеспечивающих прямое взаимодействие с TCB и возможности пользователя инициировать взаимодействие с TCB.

Уровень **WT-0**. Недостаточное прямое взаимодействие с TCB.

Зарезервирован для систем с примитивными возможностями прямого взаимодействия с TCB, не удовлетворяющих требованиям более высоких уровней.

Уровень **WT-1**. Базовые механизмы прямого взаимодействия с TCB.

TCB должна поддерживать политику обеспечения прямого взаимодействия с TCB, предусматривающую наличие соответствующих средств создания защищенных каналов взаимодействия пользователя с TCB.

Прямое взаимодействие с TCB должно использоваться для начальной идентификации и аутентификации пользователя.

Взаимодействие с TCB может быть инициировано только со стороны пользователя.

Сопряженные уровни: **WI-2**.

Уровень **WT-2**. Усовершенствованные средства прямого взаимодействия с TCB.

Изменение. Прямое взаимодействие с TCB должно использоваться для начальной идентификации и аутентификации пользователя, а также всегда инициироваться пользователем при необходимости передачи информации от пользователя к TCB, или от TCB к пользователю.

Изменение. Прямое взаимодействие с TCB может быть инициировано как со стороны пользователя, так и TCB. Инициация прямого взаимодействия с пользователем со стороны TCB должно однозначно идентифицироваться пользователем и требовать подтверждения с его стороны.

Сопряженные уровни: **WI-2**.

Уровень **WT-3**. Полное обеспечения прямого взаимодействия с TCB.

Изменение. Прямое взаимодействие с TCB должно использоваться для начальной идентификации и аутентификации пользователя, а также всегда инициироваться пользователем или TCB при необходимости передачи информации от пользователя к TCB, или от TCB к пользователю.

Сопряженные уровни: **WI-2**.

5. Критерии адекватности реализации

Критерии адекватности реализации регламентируют требования к процессу разработки и реализации компьютерной системы, позволяющие определить адекватность реализации политики безопасности и, в конечном счете, отражают степень доверия к системе. Критерии адекватности охватывают все стадии и аспекты создания и эксплуатации системы и включают разделы, относящиеся к архитектуре системы, среде разработки (процесс разработки и управление конфигурацией), контролю процесса разработки (разработка спецификаций, архитектуры, создание рабочего проекта и реализация), поставке и сопровождению, документации (руководство по безопасности для пользователя и руководство администратора безопасности) и тестированию безопасности. Предусмотрено восемь уровней адекватности реализации(**T0-T7**). С ростом номера уровня происходит конкретизация, дополнение и усиление требований без изменения их структуры. Критерии адекватности реализации политики безопасности представляют собой наиболее объемную и детально проработанную часть Канадских критериев.

*5.0 Уровень **T-0**. Недостаточная адекватность реализации.*

Зарезервирован для систем с недостаточным уровнем обеспечения адекватности, не удовлетворяющих требованиям более высоких уровней.

*5.1 Уровень **T-1**.*

5.1.1 Архитектура.

TCB должна обеспечивать поддержку принятой в компьютерной системе политики безопасности.

5.1.2 Среда разработки.

Процесс разработки.

Разработчик компьютерной системы должен применять четко определенную технологию разработки и строго придерживаться ее принципов.

Управление конфигурацией в процессе разработки.

В ходе создания системы разработчик должен использовать средства управления конфигурацией.

Средства управления конфигурацией должны контролировать все изменения, вносимые в ходе разработки в аппаратное и специальное обеспечение, в исходные тексты и объектный код программ, а также в документацию.

Средства управления конфигурацией должна обеспечивать полное соответствие между комплектом документации и текущей версией ТСВ.

5.1.3 Контроль процесса разработки.

Разработка спецификаций.

Разработчик обязан описать все функциональные возможности компьютерной системы в виде функциональных спецификаций.

Функциональные спецификации должны содержать неформальное описание реализуемой политики безопасности, включающее описание всех функций защиты, реализованных в ТСВ.

Разработка архитектуры.

Разработчик обязан составить неформальное описание архитектуры компьютерной системы.

Описание архитектуры компьютерной системы должно включать описание всех компонентов ТСВ.

Описание архитектуры должно включать интерфейс взаимодействия ТСВ с остальными компонентами компьютерной системы.

Описание архитектуры должно включать описание всех функций защиты, реализованных аппаратными, программными и специальными компонентами ТСВ.

Разработчик должен обеспечить полное соответствие между архитектурой системы и политикой безопасности.

Создание рабочего проекта.

Разработчик обязан составить неформальное описание рабочего проекта ТСВ.

Рабочий проект должен содержать описание всех компонентов ТСВ и подробно описывать механизмы функционирования всех функций, критичных с точки зрения безопасности.

Должны быть описаны назначение и параметры интерфейсов компонентов ТСВ, критичных с точки зрения безопасности.

Реализация.

Для данного уровня требования этого раздела не предъявляются.

5.1.4 Поставка и сопровождение.

Разработчик в комплекте с системой должен предоставлять средства ее инсталляции, генерации и запуска.

Разработчик должен определить и документировать все параметры компьютерной системы, используемые для ее настройки системы в ходе инсталляции, генерации и запуска.

5.1.5 Документация.

Руководство по безопасности для пользователя.

Разработчик должен включить в состав документации на компьютерную систему руководство по безопасности для пользователя в виде обзора или раздела в общей документации либо отдельного руководства. Руководство по безопасности для пользователя должно содержать описание возможностей компьютерной системы по обеспечению безопасности и принципов работы рядового пользователя со средствами защиты.

В руководстве пользователя также должно быть описано взаимодействие между отдельными подсистемами обеспечения безопасности.

Руководство администратора безопасности.

Разработчик должен включить в состав документации на компьютерную систему руководство администратора безопасности в виде обзора или раздела в общей документации либо отдельного руководства. Руководство администратора безопасности должно содержать описание возможностей администрирования средств защиты.

Руководство администратора безопасности должно содержать описание взаимодействия отдельных средств защиты с точки зрения их администрирования.

Руководство администратора безопасности должно содержать описание процесса инсталляции, генерации и запуска системы с точки зрения обеспечения безопасности.

Руководство администратора безопасности должно содержать описание всех параметров компьютерной системы, используемых для ее настройки системы в ходе инсталляции, генерации и запуска, с точки зрения обеспечения безопасности.

5.1.6 Тестирование безопасности.

Разработчик должен предоставить методику тестирования безопасности системы, сценарий проведения испытаний и средства тестирования. Полнота набора тестов безопасности системы должна быть обоснована.

Разработчик должен представить доказательства проведения тестирования безопасности системы в виде подробного описания результатов проведенных тестов в соответствии с методикой тестирования.

5.2 Уровень T-2.

5.2.1 Архитектура.

Без изменений.

5.2.2 Среда разработки.

Процесс разработки.

Без изменений.

Управление конфигурацией в процессе разработки.

Без изменений.

5.2.3 Контроль процесса разработки.

Разработка спецификаций.

Дополнение. Функциональные спецификации должны включать неформальное описание модели безопасности.

Дополнение. Разработчик должен обеспечить полное соответствие между моделью безопасностью и реализованной политикой безопасности и показать, что модель безопасности полностью обеспечивает политику безопасности.

Разработка архитектуры.

Изменение. Разработчик должен обеспечить полное соответствие между архитектурой системы и моделью безопасности.

Создание рабочего проекта.

Изменение. Рабочий проект должен содержать описание всех компонентов ТСВ и подробно описывать механизмы функционирования всех функций ТСВ.

Изменение. Должны быть описаны назначение и параметры интерфейсов всех компонентов ТСВ.

Дополнение. Разработчик должен обеспечить полное соответствие между архитектурой системы и рабочим проектом.

Реализация.

Для данного уровня требования этого раздела не предъявляются.

5.2.4 Поставка и сопровождение.

Без изменений.

5.2.5 Документация.

Руководство по безопасности для пользователя.

Без изменений.

Руководство администратора безопасности.

Без изменений.

5.2.6 Тестирование безопасности.

Дополнение. Разработчик должен исправить или нейтрализовать руженные в ходе тестирования ошибки, после чего провести повторное тестирование ТСВ для подтверждения того, что обнаруженные ошибки ликвидированы, и при этом не внесены новые.

5.3 Уровень ***T-3.***

5.3.1 Архитектура.

Дополнение. ТСВ должна быть структурирована в виде набора независимых компонентов.

5.3.2 Среда разработки

Процесс разработки.

Дополнение. Разработчик должен указать использованные в ходе разработки стандарты программирования и показать, что исходные тексты программного обеспечения соответствуют этим стандартам.

Дополнение. Разработчик должен указать использованные в ходе разработки языки программирования, и внести в документацию все зависимости программного обеспечения от реализации языков программирования и используемых компиляторов.

Управление конфигурацией в процессе разработки.

Изменение. Средства управления конфигурацией должны контролировать все изменения, вносимые в ходе разработки в аппаратное и специальное обеспечение, в исходные тексты и объектный код программ, а также в документацию и в компиляторы, используемые для трансляции исходных текстов.

5.3.3 Контроль процесса разработки.

Разработка спецификаций.

Изменение. Функциональные спецификации должны включать полуформальное описание модели безопасности.

Разработка архитектуры.

Изменение. Разработчик обязан составить полуформальное описание архитектуры компьютерной системы.

Создание рабочего проекта.

Без изменений.

Реализация.

Разработчик обязан предоставить исходные тексты заданного подмножества компонентов ТСВ.

Разработчик должен обеспечить полное соответствие между рабочим проектом и реализацией ТСВ.

5.3.4 Поставка и сопровождение.

Дополнение. Должны быть использованы технические, физические и организационные меры для контроля адекватности поставляемой потребителю копии ТСВ дистрибутивной копии.

5.3.5 Документация.

Руководство по безопасности для пользователя.

Без изменений.

Руководство администратора безопасности.

Без изменений.

5.3.6 Тестирование безопасности.

Без изменений.

5.4 Уровень Т-4.

5.4.1 Архитектура.

Дополнение. Компоненты ТСВ, критичные с точки зрения безопасности, должны быть защищены от воздействия со стороны незащищенных компонентов с помощью средств защиты, реализованных аппаратной платформой компьютерной системы.

5.4.2 Среда разработки.

Процесс разработки.

Дополнение. Должны быть описаны все физические, организационные и другие меры, используемые для защиты компьютерной системы и документации в ходе их разработки.

Управление конфигурацией в процессе разработки.

Изменение. Должны использоваться автоматизированные средства управления конфигурацией, контролирующие все изменения, вносимые в ходе разработки в аппаратное и специальное обеспечение, в исходные тексты и объектный код программ, а также в документацию и в конфигурацию компиляторов, используемых для трансляции исходных текстов.

Дополнение. Средства управления конфигурацией должны обеспечивать трансляцию и сборку исходных текстов ТСВ и осуществлять сравнение версий ТСВ для подтверждения внесенных изменений.

Дополнение. Средства управления конфигурацией должны обнаруживать несоответствия между версиями различных компонентов ТСВ и автоматически разрешать эти проблемы.

5.4.3 Контроль процесса разработки.

Разработка спецификаций.

Изменение. Функциональные спецификации должны включать формальное описание модели безопасности.

Изменение. Разработчик должен обеспечить и продемонстрировать полное соответствие между моделью безопасностью и реализованной политикой безопасности и продемонстрировать, что модель безопасности полностью обеспечивает политику безопасности.

Разработка архитектуры.

Изменение. Описание архитектуры должно включать интерфейс взаимодействия ТСВ с остальными компонентами компьютерной системы с указанием исключительных ситуаций и сообщений об ошибках.

Создание рабочего проекта.

Изменение. Разработчик обязан составить полуформальное описание рабочего проекта ТСВ.

Реализация.

Без изменений

5.4.4 Поставка и сопровождение.

Без изменений.

5.4.5 Документация.

Руководство по безопасности для пользователя.

Без изменений.

Руководство администратора безопасности.

Без изменений.

5.4.6 Тестирование безопасности.

Изменение. Разработчик должен исправить все обнаруженные в ходе тестирования ошибки, после чего провести повторное тестирование ТСВ для подтверждения того, что обнаруженные ошибки ликвидированы, и при этом не внесены новые.

Дополнение. Разработчик должен провести тестирование в виде попыток несанкционированного проникновения в систему и доказать, что система относительно успешно противостоит атакам.

*5.5 Уровень **Т-5.***

5.5.1 Архитектура.

Дополнение. Разработчик должен, по возможности, исключить из ТСВ не критичные с точки зрения безопасности компоненты и обосновать свой выбор.

Дополнение. При проектировании ТСВ разработчик должен применять технологии, позволяющие минимизировать ее сложность. В основе структуры ТСВ должен лежать законченный концептуально простой механизм защиты с четко определенной семантикой. Этот механизм должен играть основную роль в обеспечении внутренний структуры ТСВ и всей системы. Архитектура ТСВ должна использовать принципы модульности, абстракции и инкапсуляции внутренних объектов. Каждый компонент должен быть спроектирован с использованием принципа наименьших привилегий.

5.5.2 Среда разработки.

Процесс разработки.

Без изменений.

Управление конфигурацией в процессе разработки

Без изменений.

5.5.3 Контроль процесса разработки.

Разработка спецификаций.

Без изменений.

Разработка архитектуры.

Изменение. Разработчик должен продемонстрировать полное соответствие между архитектурой системы и моделью безопасности.

Создание рабочего проекта.

Без изменений.

Реализация.

Изменение. Разработчик обязан предоставить все исходные тексты ТСВ.

5.5.4 Поставка и сопровождение.

Без изменений.

5.5.5 Документация.

Руководство по безопасности для пользователя.

Без изменений.

Руководство администратора безопасности.

Без изменений.

5.5.6 Тестирование безопасности.

Изменение. Разработчик должен провести тестирование в виде попыток несанкционированного проникновения в систему и доказать, что система успешно противостоит атакам.

5.6 Уровень Т-6.

5.6.1 Архитектура.

Без изменений.

5.6.2 Среда разработки.

Процесс разработки.

Изменение. Должны быть описаны все физические, организационные и другие меры, используемые для защиты компьютерной системы и документации в ходе их разработки и применяемых в процессе разработки инструментальных средств.

Управление конфигурацией в процессе разработки.

Изменение. Должны использоваться автоматизированные средства управления конфигурацией, контролирующие все изменения, вносимые в ходе разработки в аппаратное и специальное обеспечение, в исходные тексты и объектный код программ, а также в документацию, а также в конфигурацию компиляторов, используемых для трансляции исходных текстов, и инструментальных средств разработки.

Дополнение. Должны быть использованы технические, физические и организационные меры для защиты от несанкционированной модификации или унич-

тожения дистрибутивной копии ТСВ или дистрибутивных копий всех материалов, используемых для построения ТСВ.

5.6.3 Контроль процесса разработки.

Разработка спецификаций.

Без изменений.

Разработка архитектуры.

Изменение. Разработчик обязан составить формальное описание архитектуры компьютерной системы.

Изменение. Разработчик должен доказать полное соответствие между архитектурой системы и моделью безопасности.

Создание рабочего проекта.

Изменение. Разработчик должен продемонстрировать полное соответствие между архитектурой системы и рабочим проектом.

Реализация

Без изменений.

5.6.4 Поставка и сопровождение.

Дополнение. Процесс дистрибуции компонентов компьютерной системы должен быть защищен с помощью соответствующих средств, обеспечивающие адекватность поставляемой потребителю копии ТСВ дистрибутивной копии.

5.6.5 Документация.

Руководство по безопасности для пользователя.

Без изменений.

Руководство администратора безопасности.

Без изменений.

5.6.6 Тестирование безопасности.

Без изменений.

5.7 Уровень Т-7.

5.7.1 Архитектура.

Без изменений.

5.7.2 Среда разработки.

Процесс разработки.

Без изменений.

Управление конфигурацией в процессе разработки.

Без изменений.

5.7.3 Контроль процесса разработки.

Разработка спецификаций.

Без изменений.

Разработка архитектуры.

Без изменений.

Создание рабочего проекта.

Изменение. Разработчик обязан составить формальное описание рабочего проекта ТСВ.

Изменение. Разработчик должен доказать полное соответствие между архитектурой системы и рабочим проектом.

Реализация.

Изменение. Разработчик должен продемонстрировать полное соответствие между рабочим проектом и реализацией ТСВ.

5.7.4 Поставка и сопровождение.

Без изменений.

5.7.5 Документация.

Руководство по безопасности для пользователя.

Без изменений.

Руководство администратора безопасности.

Без изменений.

5.7.6 Тестирование безопасности.

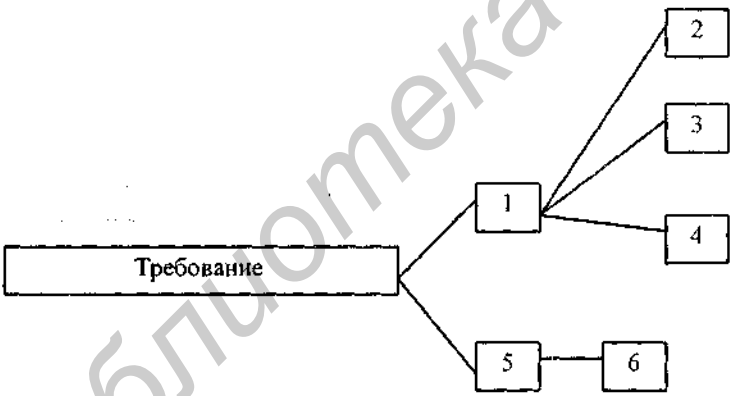
Без изменений.

Приложение 3. Ранжированные требования "Единых критериев безопасности информационных технологий"

Это приложение содержит ранжированный перечень функциональных требований и требований адекватности, содержащихся в "Единых критериях". Данное изложение отражает только суть требований и не претендует на перевод стандарта как нормативного документа.

Для того чтобы отобразить иерархическое, частично упорядоченное ранжирование требований, присущее "Единым критериям", будем представлять их в виде таблиц, в которых сравнимые требования образуют столбцы, а несравнимые — строки.

Рассмотрим соответствие таблиц и иерархии требований на следующем примере. Графическое представление иерархии требований приведено на рисунке, представленном ниже, и соответствует следующей ниже таблице:



Требование			
1			5
2	3	4	6

В этой таблице каждое из несопоставимых между собой требований 2, 3 и 4 обеспечивает больший уровень безопасности чем требование 1. Ни одно из них несопоставимо с требованиями 5 и 6, образующими параллельную шкалу.

Функциональные требования

1. Аудит

Автоматическое реагирование на попытки нарушения безопасности
1. Средства защиты должны реагировать на попытки нарушения безопасности

Регистрация и учет событий	
1. Регистрация заданного множества событий и задание учетной информации для событий каждого типа	2. Регистрация и учет событий и идентификация инициировавших их пользователей

Анализ протокола аудита	
1. Выявление потенциально опасных событий на основе контроля за диапазонами учетных параметров на основе фиксированного множества правил	
2. Статическое распознавание вторжений на основании анализа профилей работы пользователей	3. Динамическое распознавание сигнатур элементарных атак на основании простых эвристик
	4. Распознавание комплексных атак на основе сложных эвристик

Доступ к протоколу аудита		
1. Предоставление доступа к протоколу аудита для ограниченного набора авторизованных пользователей	2. Защита протокола аудита от неавторизованных пользователей	3. Выборочное управление доступом к протоколу аудита

Отбор событий для регистрации и учета
1. Определение множества подлежащих аудиту событий на основании заданного набора атрибутов

Протокол аудита	
1. Выделение ресурсов под протокол аудита, защита протоколов от неавторизованной модификации или удаления	3. Предотвращение потери записей протокола аудита в случае уменьшения объема ресурсов, отведенных под протокол аудита до определенного предела
2. Гарантированная доступность протокола аудита	4. Предотвращение потерь записей аудита в случае исчерпания ресурсов, отведенных под протокол аудита

2. Подтверждение приема/передачи информации

Предотвращение отречения от факта передачи информации
1. Подтверждение факта передачи информации по требованию
2. Автоматическое подтверждение факта передачи информации

Предотвращение отречения от факта получения информации
1. Подтверждение факта получения информации по требованию
2. Автоматическое подтверждение факта получения информации

3. Криптография

Управление ключами			
1. Генерация ключей заданного размера по определенным алгоритмам в соответствии со специальными стандартами	2. Распределение ключей способами, определенными в специальных стандартах	3. Осуществление доступа к ключам с использованием методов, определенных в специальных стандартах	4. Уничтожение ключей с использованием методов, определенных в специальных стандартах

Криптографические средства
1. Выполнение криптографических операций с использованием ключей заданного размера и определенных алгоритмов в соответствии со специальными стандартами

4. Защита информации

Политики управления доступом
1 Управление доступом для ограниченного множества операций и объектов
2 Управление доступом для полного множества объектов, субъектов и операций. Любая операция, осуществляемая любым субъектом должна контролироваться по крайней мере одной политикой управления доступом.

Средства управления доступом
1 Управление доступом на основании атрибутов безопасности или именованных групп атрибутов с явными разрешением или отказом в доступе

Аутентификация информации
1 Аутентификация информации, содержащейся в объекте доступа
2 Аутентификация информации, содержащейся в объекте доступа с идентификацией субъекта, осуществляющего аутентификацию

Экспорт информации из системы	
Экспорт информации без атрибутов безопасности	Экспорт информации вместе с атрибутами безопасности

Политики управления информационными потоками
1. Управление информационными потоками для ограниченного множества операций и потоков
2. Управление доступом для полного множества потоков, субъектов и операций. Любая политика управления потоками должна контролировать все операции в системе. Все потоки в информации в системе должны контролироваться по крайней мере одной политикой управления информационными потоками

Средства управления информационными потоками		
1. Управление информационными потоками на основании атрибутов безопасности информации и субъектов, между которыми происходит обмен информацией	3. Контроль скрытых информационных потоков	6. Мониторинг скрытых информационных потоков и ограничение их пропускной способности
2. Управление информационными потоками с на основании иерархически упорядоченных атрибутов безопасности, присвоенных всем информационным потокам и образующих решетку	4. Частичный запрет скрытых информационных потоков	
	5. Полный запрет скрытых информационных потоков	

Импорт информации	
1. Импорт информации без атрибутов безопасности	2. Импорт информации вместе с атрибутами безопасности

Защита информации при передаче по внутренним каналам	
1. Базовые средства защиты передаваемой информации	3. Контроль целостности передаваемой информации
2. Передача данных с различными атрибутами безопасности по раздельным каналам	4. Применение различных методов контроля целостности в зависимости от атрибутов безопасности
Уничтожение остаточной информации	
1. Уничтожение остаточной информации для определенного подмножества объектов при их создании или удалении	
2. Уничтожение остаточной информации для всех объектов при их создании или удалении	

Откат	
1. Ограниченные возможности осуществления отката для определенного подмножества операций на заданное число шагов	2. Расширенные возможности осуществления отката для всех операций на заданное число шагов

Контроль целостности информации в процессе хранения
1. Обнаружение нарушений целостности информации в процессе хранения.
2. Обнаружение нарушений целостности информации в процессе хранения и определение реакции на обнаружение ошибки

Защита внутрисистемной передачи информации при использовании внешних каналов
1. Защита информации при направлении во внешний канал

Целостность внутрисистемной передачи информации при использовании внешних каналов	
1. Обнаружение нарушений целостности при передаче информации	2. Восстановление информации получателем
	3. Повторная передача информации

5. Идентификация и аутентификация

Реакция на неудачные попытки аутентификации
1. Средства идентификации и аутентификации должны прекращать попытки установления сеансов работы с системой после установленного числа неудачных попыток аутентификации и приостанавливать обслуживание средств, задействованных в ходе этих попыток

Атрибуты безопасности пользователей
1. Индивидуальное назначение атрибутов безопасности пользователей

Аутентификационные параметры	
1. Проверка качества аутентификационных параметров в соответствии с заданными критериями	2. Автоматическая генерация аутентификационных параметров и проверка их качества в соответствии с заданными критериями

Аутентификация пользователей					
1. Обязательность аутентификации пользователей	3. Механизм аутентификации должен распознавать и предотвращать использование подделанных аутентификационных параметров или их дубликатов	4. Использование одноразовых аутентификационных параметров	5. Использование множественных механизмов аутентификации, используемых в зависимости от ситуации	6. Применение механизмов повторной аутентификации для выполнения установленного множества операций	7. Минимизация информации предоставляемой пользователю в процессе прохождения процедуры аутентификации
2. Невозможность осуществления действий, контролируемых средствами защиты, без успешного прохождения процедуры аутентификации					

Идентификация пользователей
1. Обязательность идентификации пользователей
2. Невозможность осуществления действий, контролируемых средствами защиты, без успешного прохождения процедуры идентификации

Соответствие пользователей и субъектов
1. Присвоение субъектам, действующим от его имени пользователя, его атрибутов безопасности

6. Управление безопасностью

Управление средствами защиты
1. Управление авторизованными пользователями средствами защиты

Управление атрибутами безопасности		
1. Управление авторизованными пользователями атрибутами безопасности	2. Контроль корректности значений атрибутов безопасности	3. Корректная инициализация атрибутов безопасности определенными значениями

Управление параметрами и конфигурацией средств защиты		
1. Управление параметрами и конфигурацией средств защиты авторизованными пользователями	2. Выполнение заданных действий в случае выхода параметров функционирования средств защиты за установленные пределы	3. Автоматический контроль корректности конфигурации и параметров средств защиты

Отзыв атрибутов безопасности
1. Отзыв атрибутов безопасности в соответствии с установленными правилами

Ограничение времени действия атрибутов безопасности
1. Назначение времени действия атрибутов безопасности авторизованными пользователями

Административные роли	
1. Использование административных ролей для управления безопасностью	3. Предоставление ролевых полномочий по запросу пользователя
2. Использование упорядоченного набора административных ролей для управления безопасностью	

7. Конфиденциальность работы в системе

Анонимность пользователей
1. Анонимность субъектов, представляющих интересы пользователей
2. Анонимность идентификаторов пользователей для средств защиты

Использование псевдонимов	
1. Контроль действий анонимных пользователей с помощью псевдонимов	
2. Установление личности пользователя по псевдониму	3. Назначение псевдонимов в соответствии с заданными правилами

Анонимность сеанса работы в системе
1. Невозможность установления инициатора операций, осуществляемых в системе

Защита от мониторинга сеансов работы с системой		
1. Защита операций, происходящих в системе, от мониторинга	3. Запрещение средствам защиты запрашивать у пользователя конфиденциальную информацию	4. Мониторинг работы системы и использования ресурсов только авторизованными пользователями
2. Рассредоточение критической информации между различными компонентам средств защиты		

8. Надежность средств защиты

Тестирование аппаратно-программной платформы
1 Проверка корректности функционирования аппаратно-программной платформы

Защита от сбоев
1. Сохранение безопасного состояния в случае возникновения сбоев

Готовность средств защиты к обслуживанию удаленных клиентов
1. Обеспечение готовности средств защиты к обслуживанию удаленных клиентов с заданной вероятностью

Конфиденциальность передаваемой информации при работе с удаленными клиентами
1. Обеспечение конфиденциальности информации, передаваемой между средствами защиты и удаленными клиентами

Целостность передаваемой информации при работе с удаленными клиентами
1. Обнаружение искажений информации, передаваемой между средствами защиты и удаленными клиентами
2. Обнаружение искажений информации, передаваемой между средствами защиты и удаленными клиентами, и их исправление

Защита внутренних каналов информационного обмена между средствами защиты	
1. Базовые средства защиты информационного обмена между средствами защиты	
2. Разделение трафика информационного обмена между средствами защиты и трафика прикладных средств	3. Контроль целостности информации при взаимодействии средств защиты

Физическая защита	
1. Пассивное обнаружение атак на физическом уровне	3. Активное противодействие атакам на физическом уровне
2. Уведомление администратора при обнаружении атаки на физическом уровне	

Безопасное восстановление после сбоев	
1. Ручное восстановление после сбоев	4. Восстановление после сбоев путем осуществления отката в безопасное состояние
2. Автоматическое восстановление после сбоев	
3. Автоматическое восстановление после сбоев с минимизацией потерь информации	

Распознавание повторных передач информации и имитации событий
1. Распознавание повторных передач информации и имитации событий и реагирование в соответствии с установленными правилами

Мониторинг взаимодействий
1. Мониторинг всех взаимодействий в системе

Разделение доменов
1. Выделение отдельного домена для средств защиты
2. Выделение отдельных доменов для процедур, осуществляющих мониторинг взаимодействий и реализующих указанные политики безопасности
3. Выделение отдельных доменов для всех процедур, осуществляющих мониторинг взаимодействий и реализующих любые политики безопасности

Синхронизация
1. Подтверждение приема информации
2. Синхронизация состояния участников взаимодействия в ходе обмена информацией

Время
1. Использование средствами защиты надежного таймера

Согласованность обмена информацией между средствами защиты
1. Корректность преобразования информации при передаче между средствами защиты

Репликация информации, используемой средствами защиты
1. Контроль согласованности копий информации, используемой средствами защиты

Самотестирование средств защиты
1. Самотестирование средств защиты по запросу, в процессе загрузки и функционирования. Проверка целостности кода и данных средств защиты

9. Контроль за использованием ресурсов

Отказоустойчивость
1. Обеспечение работоспособности системы на заданном уровне в случае возникновения указанных сбоев
2. Обеспечение нормальной работы системы в случае возникновения указанных сбоев

Распределение ресурсов на основе приоритетов
1. Распределение ограниченного подмножества ресурсов системы на основе приоритетов
2. Распределение всех ресурсов системы на основе приоритетов

Квотирование ресурсов
1. Ограничение на потребление пользователями ресурсов системы с помощью квот
2. Ограничение на потребление пользователя ресурсов системы с помощью квот и резервирование для пользователя гарантированного множества ресурсов

10. Контроль доступа к системе

Ограничения на использование атрибутов безопасности
1. Ограничение множества атрибутов безопасности, используемых пользователем в рамках одной сессии

Ограничение числа одновременных сеансов
1. Ограничение числа одновременных сеансов
2. Ограничение числа одновременных сеансов в зависимости от атрибутов безопасности пользователей

Блокировка сеанса работы с системой		
1. Автоматическая блокировка сеанса работы после указанного периода неактивности	2. Блокирование сеанса пользователем	3. Автоматическое завершение сеанса работы по истечении заданного периода неактивности

Объявления, предупреждения, приглашения и подсказки
1. Демонстрация объявлений, предупреждений, приглашений и подсказок перед началом сеанса работы с системой

Протокол сеансов работы пользователей
1. Регистрация и демонстрация пользователям протокола сеансов их работы и попыток входа в систему

Управление сеансами работы с системой
1. Запрещение установки сеанса работы с системой на основе заданного множества правил

11. Прямое взаимодействие

Прямое взаимодействие между средствами защиты
1. Прямое взаимодействие между компонентами между средствами защиты различных продуктов

Прямое взаимодействие с пользователями
1. Прямое взаимодействие с пользователями для указанного набора ситуаций или по требованию пользователя

Требования адекватности

1. Управление проектом

Средства управления проектом
1. Применение автоматизированных средств управления проектом
2. Полная автоматизация управления проектом и контроля версий

Управление версиями
1. Нумерация версий
2. Идентификация компонентов
3. Контроль целостности версий
4. Авторизация разработчиков при обновлении версий
5. Контроль целостности и подлинности дистрибутива системы

Конфигурация проекта
1. Основные компоненты проекта(алгоритмы, исходные тексты, тесты, документация)
2. Включение в состав конфигурации проекта обнаруженных ошибок и уязвимостей
3. Включение в состав конфигурации проекта инструментальных средств разработки

2. Дистрибуция

Поставка
1. Регламентированная процедура поставки
2. Обнаружение искажений в процессе поставки
3. Защита от искажений в процессе поставки

Установка, настройка, запуск
1. Регламентированные процедуры установки, настройки, запуска
2. Протоколирование процесса установки, настройки, запуска

3. Разработка

Общие функциональные спецификации
1. Неформальные спецификации для средств защиты
2. Неформальные спецификации для всех интерфейсов средств защиты
3. Полуформальные спецификации для средств защиты

4. Формальные спецификации для средств защиты

Архитектура защиты
1. Описание архитектуры защиты
2. Соответствие архитектуры защиты политике безопасности
3. Полуформальное описание архитектуры защиты
4. Соответствие полуформального описание архитектуры защиты политике безопасности
5. Формальное описание архитектуры защиты и доказательство ее соответствия политике безопасности

Форма представления продукта на сертификацию
1. Описания реализации ограниченного подмножества средств защиты
2. Полное описание реализации всех средств защиты
3. Структурированное описание реализации всех средств защиты

Структура средств защиты
1. Модульность
2. Иерархичность
3. Минимизация сложности

Частные спецификации средств защиты
1. Неформальные частные спецификации средств защиты
2. Полуформальные частные спецификации средств защиты
3. Формальные частные спецификации средств защиты

Соответствие описаний различного уровня
1. Неформальное подтверждение соответствия
2. Полуформальное подтверждение соответствия
3. Формальное доказательство соответствия

Политика безопасности
1. Неформальное описание политики безопасности
2. Полуформальное описание политики безопасности
3. Формальная модель политики безопасности

4. Документация

Руководства администратора
1. Администрирование средств защиты

Руководства пользователя
1. Использование средств защиты

5. Процесс разработки

Безопасность среды разработки
1. Применение мер безопасности в ходе разработки
2. Подтверждение достаточности мер безопасности, применяемых в ходе разработки

Исправление ошибок и устранение уязвимостей
1. Исправление выявленных ошибок и устранение уязвимостей
2. Регулярное исправление ошибок и устранение уязвимостей
3. Гарантированное исправление обнаруженных ошибок и устранение выявленных уязвимостей

Технология разработки
1. Определенная разработчиком технология разработки
2. Стандартизованная технология разработки
3. Технология разработки, позволяющая оценить разрабатываемый продукт

Средства разработки
1. Использование определенного набора средств разработки
2. Использование основных средств разработки, отвечающих определенным стандартам
3. Использование только средств разработки, отвечающих определенным стандартам

6. Тестирование

Полнота тестирования
1. Обоснование полноты тестирования
2. Анализ полноты тестирования
3. Строгий анализ полноты тестирования

Глубина тестирования
1. Архитектура
2. Функциональные спецификации
3. Реализация

Методика тестирования
1. Функциональное тестирование и протоколирование результатов тестов
2. Тестирование в соответствии с определенной методикой

Независимое тестирование
1. Готовность продукта к независимому тестированию
2. Избирательное независимое тестирование
3. Полное независимое тестирование

7. Анализ защиты

Анализ скрытых каналов
1. Поиск и документирование скрытых каналов
2. Поиск скрытых каналов на основе определенных методов
3. Исчерпывающий поиск скрытых каналов

Анализ возможностей неправильного использования средств защиты
1. Анализ руководств по администрированию
2. Подтверждение полноты руководств по администрированию и безопасности их применения
3. Независимый анализ возможностей неправильного использования средств защиты

Анализ стойкости средств защиты
1. Оценка стойкости средств защиты

Анализ продукта на наличие уязвимостей
1. Выявление уязвимостей разработчиком продукта
2. Независимый анализ уязвимостей
3. Систематический анализ уязвимостей на основе заданных методик
4. Исчерпывающий анализ уязвимостей

Для представления результатов классификационного анализа "Единые критерии" содержат семь стандартизованных уровней адекватности (п. 2.10.4.2).

Распределение требований адекватности по уровням представлено в таблице, в ячейках которой указаны номера категорий требований адекватности, которым должен удовлетворять продукт для присвоения ему соответствующего уровня адекватности.

Требования адекватности	Номера уровней адекватности						
	1	2	3	4	5	6	7
1. Управление проектом							
Средства управления проектом				1	1	2	2
Управление версиями	1	2	3	4	4	5	5
Конфигурация проекта			1	2	3	3	3

2. Дистрибуция							
Поставка		1	1	2	2	2	3
Установка, настройка, запуск	1	1	1	1	1	1	1
3. Разработка							
Общие функциональные спецификации	1	1	1	2	3	3	4
Архитектура защиты		1	2	2	3	4	5
Форма представления продукта на сертификацию				1	2	3	3
Структура средств защиты					1	2	3
Частные спецификации средств защиты				1	1	2	2
Соответствие описаний различного уровня	1	1	1	1	2	2	3
Политика безопасности				1	2	2	3
4. Документация							
Руководства администратора	1	1	1	1	1	1	1
Руководства пользователя	1	1	1	1	1	1	1
5. Процесс разработки							
Безопасность среды разработки			1	1	1	2	2
Исправление ошибок и устранение уязвимостей							
Технология разработки				1	2	2	3
Средства разработки				1	2	3	3
6. Тестирование							
Полнота тестирования		1	2	2	2	3	3
Глубина тестирования			1	1	2	2	3
Методика тестирования		1	1	1	1	2	2
Независимое тестирование	1	1	2	2	2	2	3
7. Оценка уязвимости							
Анализ скрытых каналов					1	2	2
Анализ возможностей неправильного использования средств защиты			1	2	2	3	3
Анализ стойкости средств защиты		1	1	1	1	1	1
Анализ продукта на наличие уязвимостей		1	1	2	3	4	4

Приложение 4.
Статистика нарушений безопасности компьютерных систем

(по материалам Carl E. Landwehr, Alan R. Bull, John P. McDermott, and William S.Choi. *A Taxonomy of Computer Security Flaws, with Examples*)

Данная подборка материалов не в коей мере не может претендовать на исчерпывающее описание всех известных нарушений безопасности ВС и приведена здесь исключительно для иллюстрации таксономии причин возникновения ИЗ и ее обеспечения практическими примерами. Приведенные данные охватывают широкий спектр как различных типов вычислительных систем, так и различных видов нарушений безопасности, что необходимо для выявления основных закономерностей возникновения и проявления ИЗ.

Все эти примеры отражают реально имевшие место нарушения безопасности и реализованные атаки на вычислительные системы. В каждом примере указывается тип вычислительной системы, в которой имело место нарушение безопасности, кратко описываются ее особенности и слабые стороны, использованные для осуществления атаки, суть нарушения безопасности и его последствия. Примеры расположены в хронологическом порядке. Группам примеров, относящихся к одной вычислительной системе, предшествует краткое описание ее структуры и принципов функционирования, позволяющее понять реализованную в ней концепцию защиты.

Индексы примеров соответствуют системе обозначений, принятой в [1] (см. литературу к главе 3). В начале обозначения присутствует префикс, определяющий тип ВС, за которым следует порядковый номер примера для данной системы. В приложение вошли не все приведенные в [1] случаи, а только наиболее характерные представители различных классов, в связи с чем некоторые номера пропущены. В главе 3 отражены результаты, полученные при анализе всей доступной информации, в том числе и не включенной в данное приложение.

Система IBM 360/370

В архитектуре систем IBM 360/370 существуют два состояния процессора — режим выполнения прикладной программы, в котором запрещено выполнение подмножества привилегированных команд (загрузка *PSW*, инициализация ввода/вывода и т.п.), и режим супервизора, в котором возможно выполнение любых команд. Попытка выполнения прикладным процессом привилегированной команды вызывает прерывание, вызов привилегированных команд из прикладных (непривилегированных) процессов осуществляются посредством специального вызова — Supervisor Call (SVC).

Оперативная память разделена на страницы, каждая из которых ассоциирована с 4-х битным ключом доступа. Обычно для отведенных пользователю страниц памяти значение ключа доступа равно 8, в то время как значение ключей

для областей памяти, используемых самой системой, лежит в интервале от 0 до 7. Процесс, выполняющийся в области памяти с ненулевым значением ключа, имеет неограниченный доступ к другим областям памяти с тем же ключом доступа. Кроме того, процесс может читать области памяти с другими значениями ключей, если только они не помечены как защищенные. Попытка записи в область памяти с другим значением ключа приводит к возникновению прерывания. Процессы операционной системы выполняются в областях памяти со значением ключа, равным 0. Такие процессы имеют неограниченный доступ ко всем областям памяти вне зависимости от их ключей и статуса защищенности.

Подсистема ввода/вывода состоит из т. н. каналов, которые по сути представляют собой специализированные программируемые микрокомпьютеры, осуществляющие обмен данными между памятью и внешними устройствами. С каналом ввода/вывода может быть связана специальная программа, которая выполняется процессором ввода/вывода и осуществляет управление обменом данными между оперативной памятью и внешним устройством. Такие программы после их инициализации функционируют независимо от основного процессора и имеют неограниченный доступ к оперативной памяти. Таким образом, управление процессом ввода/вывода и контроль доступа возлагается на программу, управляющую каналом ввода/вывода.

В многозадачной системе MVS имеется опция разделения времени Time Sharing Option (TSO), которая позволяет одновременно нескольким пользователям выполнять команды с интерактивных терминалов. В MVS существует категория привилегированных процессов, объединенных понятием авторизованные программы(Authorized Program Facility — APF). Эти программы занимают области памяти с ключом доступа равным 7. Им предоставлены возможности, недоступные остальным прикладным процессам, в частности, перевод процессора в режим супервизора. Данные процессы рассматриваются как расширение операционной системы и считаются гарантированно безопасными (trustworthy).

Система IBM 370 является развитием системы 360 и представляет собой монитор виртуальных машин. На ее основе Министерством обороны США разработана система KVM/370, обеспечивающая более высокий уровень защиты информации. Университетом штата Мичиган разработана система MTS (Michigan Terminal System), специально предназначенная для работы как в пакетном, так и в интерактивном режиме.

Индекс: I1. Система: IBM 360.

Источник информации: A. S. Tanenbaum. Operating System Design and Implementation. Prentice-Hall, Englewood Cliffs, NJ, 1987.

В системе IBM 360 имеется возможность нарушения контроля доступа к файлам. В этой системе для обращения к некоторым файлам требуется ввести соответствующий пароль, причем процедура проверки правильности пароля организована следующим образом: в систему вводится имя файла и пароль, а затем, в случае корректности пароля, осуществляется открытие файла. Однако существует возможность подмены имени файла между проверкой подлинности пароля и открытием файла. Для этого можно использовать фоновый процесс, который имеет возможность записи в системные области памяти, например, процесс обмена с на-

копителями на магнитной ленте. Пользователь выдает запрос на доступ к файлу, пароль которого ему известен. Система проверяет пароль и разрешает доступ. Однако, после проверки, но до открытия файла, фоновый процесс может заменить имя файла, что приведет к получению доступа не к тому файлу, для которого проверялся пароль, а к другому, несмотря на то, что пароля для него пользователь не знает.

Индекс: I4. Система: IBM VM/370.

Источник информации: C. R. Attanasio, P. W. Markstein, R. J. Phillips "Penetrating at Operating System: a study of VM/370 integrity", IBM System Journal, 1976, pp.102-116.

При программировании каналов ввода/вывода на этапе трансляции управляющая программа подвергается статическому анализу на допустимость используемых инструкций и обращений к областям памяти, однако при этом предполагается, что каждая команда имеет фиксированную длину и выровнена по границе слова. Фактически такие программы могут включать в себя команды переменной длины, не обязательно выровненные по границе слова, что при наличии команды перехода на произвольный адрес позволяет организовать передачу управления "в середину" команды. Этот трюк приводит к выполнению других команд, состоящих из фрагментов исходных, и не подвергавшихся трансляции и проверке. Соответствующим образом составив подобную программу, пользователь может запустить параллельный основным вычислениям процесс (как было отмечено выше программы ввода/вывода имеют доступ к любым областям памяти) и осуществить неконтролируемый доступ к информации.

Индекс: I6. Система: IBM MVS (TSO).

Источник информации: R. Paans, G. Bones. "Surreptitious security violation in the MVS operating system", Security.IFIP/Sec, Holland, 1983, pp. 95-101.

Существует возможность использования TSO для запуска привилегированного процесса. Для этого необходимо запустить из TSO фоновый процесс и вызвать любую программу, входящую в APF. Фоновый пользовательский процесс сможет обнаружить факт начала выполнения APF-процесса по изменению значению ключа общей области памяти (для привилегированных процессов оно меньше 8). С этого момента пользовательский фоновый процесс является привилегированным, т. к. обе программы будут выполняться в одном и том же адресном пространстве. После этого он может остановить APF-процесс и получить все привилегии и возможности управления системой.

Индекс: I7. Система: IBM MVS.

Источник информации: R. Paans, G. Bones. "Surreptitious security violation in the MVS operating system", Security, IFIP/Sec , Holland, 1983, pp. 101-105.

Коммерческие приложения, такие как СУБД, часто должны устанавливаться в систему как APF и выполняться как привилегированные. Считается, что такие приложения являются доверенными, и не используют предоставленные привилегии для нарушения защиты и игнорирования установленных требований безопасности. Однако в ряде случаев (в первоисточнике приведен ряд примеров) доверенные приложения предоставляют пользователю возможность использования пре-

доставленных им привилегий, что дает ему возможность нарушать безопасность системы. Данный пример перекликается с имеющейся в ОС Unix возможностью запустить процесс с правами супервизора (пример U9).

Индекс: I9 Система KVM/370.

Источник информации: M. Schaefer, B. Gold, R. Linde, and J. Schield, "Program Confinement in KVM/370". Proc. of ACM National Conf., Oct., 1977.

Система KVM/370 выделяет каждой виртуальной машине квант времени физического процессора. Причем виртуальная машина может использовать весь отведенный ей квант, или освободить процессор раньше срока. С учетом того, что виртуальная машина имеет доступ к часам реального времени, существует возможность организовать скрытый канал передачи информации от одной виртуальной машины к другой путем кодирования информации величиной временного интервала, использованного виртуальной машиной.

Индекс: MT1. Система: MTS.

Источник информации: B. Hebbard et al. "A penetration analysis of the Michigan Terminal System", ACM SIGOPS Operating Systems Review 14, Jan. 1980, pp. 7-20.

Пользователь может посредством манипулирования значениями параметров системных вызовов заставить систему изменить ряд критичных параметров, и тем самым отключить механизмы управления безопасностью и получить полный контроль над системой. Некоторые подпрограммы ядра системы, осуществляющие модификацию переданных им параметров, используют для их передачи механизм косвенной адресации. В каждой функции ядра перед осуществлением каких-либо действий проверяется принадлежность полученного параметра-адреса пространству пользовательского процесса, в противном случае запрос пользователя отвергается. Однако, пользователь может обойти этот контроль посредством задания параметра, который содержит адрес, принадлежащий самой области передачи параметров. При этом можно добиться того, что в результате вызова параметры будут модифицированы, и в них окажутся адреса важных переменных из системной области, что дает возможность пользователю изменить их при помощи других системных вызовов.

Операционная система Multics

Операционная система Multics изначально применялась на специально разработанных компьютерах GE-645 фирмы General Electric. Позднее под управлением Multics функционировали компьютеры серии HIS 6180. Аналогично системам IBM 360/370 аппаратное обеспечение этих компьютеров поддерживало два режима работы: т. н. *master* режим, в котором являлись допустимыми все команды, и *slave* режим, в котором определенные команды были запрещены. Система обеспечения безопасности включала в себя 8 колец защиты, которые в компьютерах GE-645 были реализованы программно, а на платформе HIS 6180 — аппаратно. Кольцо 0 являлось наиболее привилегированным, предполагалось, что в нем может размещаться только код операционной системы.

Индекс: MU1. Система: Multics.

Источник информации: A. S.Tanenabum. "Operating System Design and Implementation". Prentice-Hall, Englewood Cliffs, NJ, 1987.

Система Multics в режиме пакетной обработки информации при считывании с перфокарт не поддерживала идентификацию/аутентификацию пользователя. Это позволяло любому пользователю записывать файлы с информацией, введенной с перфокарт, в любые каталоги. Поскольку пути поиска команд многих пользователей включали их собственные каталоги, это давало возможность внедрить в систему троянскую программу.

Индекс: MU2. Система: Multics.

Источник информации: P. A. Karger, R. R. Schell. "Multics Security Evaluation: Vulnerability Analysis", ESD-TR-74-193, Vol II, June 1974.

В случае, когда программа, выполняющаяся в менее привилегированном кольце защиты, передает параметры программе, находящейся в более привилегированном кольце, последняя должна удостовериться, что вызвавшая ее программа имеет права доступа на чтение или запись переданных ей параметров. Поскольку разделение колец защиты в системах на базе компьютеров GE-645 было реализовано программно, эту функцию осуществляла специальная процедура. Однако, данная процедура неправильно обрабатывала один из видов косвенной адресации системы GE-645, и проверка прав доступа не всегда осуществлялась корректно, что позволяло непривилегированным программам получить доступ к данным, обрабатываемым в привилегированных кольцах защиты.

Индекс: MU3. Система: Multics.

Источник информации: P. A. Karger, R. R .Schell. "Multics Security Evaluation: Vulnerability Analysis", ESD-TR-74-193, Vol II, June 1974.

В ранних версиях Multics регистр указателя стека (*sp*) мог быть модифицирован только в режиме *master*. Когда Multics получила широкое распространение, это было признано неудобным, и в систему были внесены изменения, допускавшие изменение регистра *sp* во всех режимах. Однако в системе остался не исправлен ряд фрагментов кода, предполагавших, что модификация регистра *sp* возможна только в режиме *master*. Это нарушило интерфейс между *master* и *slave* режимами и позволило использовать эти фрагменты кода для осуществления несанкционированного доступа.

Индекс: MU9. Система: Multics.

Источник информации: P. A. Karger, R. R.Schell. "Multics Security Evaluation: Vulnerability Analysis". ESD-TR-74-193, Vol II, June 1974.

С помощью программы тестирования аппаратно реализуемых механизмов защиты Multics (авторы назвали ее Subverter) была выявлена аппаратная ошибка в системе GE-645. Ошибка заключалась в следующем: если исполняемая команда вызывала команду, находящуюся в самом начале другого сегмента, которая использовала индексный регистр, но не устанавливала базу для индексации, то эта команда выполнялась без какого-либо контроля со стороны аппаратных средств. Благодаря этой возможности пользователь мог легко получить контроль над всей системой.

Операционная система Burroughs 6700

Механизмы защиты ОС Burroughs были основаны на том, что пользователь не мог создать собственную прикладную программу, кроме как с использованием специально разработанных доверенных компиляторов с языков высокого уровня, которые осуществляли контроль за его действия уже на стадии компиляции.

Индекс: B1. Система: Burroughs 6700

Источник информации: A.L.Wilkinson et al. "A penetration analysis of a Burroughs large system", ACM SIGOPS Operating Systems Review 15, Jan. 1981, pp. 14-25.

В системах Burroughs 6700 аппаратный контроль доступа к памяти осуществлялся с помощью проверки соответствия используемых программой адресов с диапазоном, задаваемым значениями регистров границ, которые программы самостоятельно устанавливали для себя. Данные регистры для каждой программы можно установить однократно без возможности их последующего изменения. Пользователь, написавший программу, которая могла бы управлять значениями данных регистров, получил бы полный контроль над системой. Для предотвращения такой ситуации система содержала соответствующий механизм, состоявший в том, что могли выполняться только программы, созданные доверенными компиляторами, которые гарантировали не нарушающее защиту использование регистров границ. Каждому файлу в системе был поставлен в соответствие определенный тип. Системный загрузчик проверял соответствие программ типу файлов, создаваемых доверенным компилятором. Устанавливать тип файла непривилегированному пользователю было запрещено.

Таким образом, пользователь принципиально мог создать файл, который содержал бы исполняемый код, переустанавливающий значения регистров границ и забиравший на себя управление системой, но до тех пор, пока этому файлу не был назначен соответствующий тип, он не мог быть загружен в память и выполнен.

В системе присутствовала утилита копирования файлов с/на магнитную ленту, в которой была допущена ошибка. Эта утилита позволяла изменять тип файлов, находящихся на ленте. Это означало, что пользователь мог создать файл, содержащий соответствующий код, сбросить его на ленту, поменять его тип и скопировать его обратно, после чего запустить на выполнение и получить контроль над системой.

Система Univac 1108

Компьютеры Univac 1108, представлявшие собой высокопроизводительные мейнфрейма, в 70-х годах обеспечивали вычислительными ресурсами исследовательские лаборатории и университеты. Их основная память была поделена на два банка, каждый из которых состоял из совокупности 512-байтных элементов. Адресное пространство программ также состояло из двух банков: банк I (банк инструкций) содержал код программы, банк D (банк данных) — данные. Аппаратные средства защиты были организованы таким образом, что программы могли либо использовать оба банка как для чтения, так и для записи, либо установить запрет на запись в оба банка.

Индекс: UN1. Система: Univac 1108/Exec 8.

Источник информации: D. Stryker."Subversion os a "Secure" Operating System", NRL Memorandum Report 2821, June 1974.

Операционная система мейнфреймов Univac — Exec 8 поддерживала одновременное использование несколькими пользователями ряда системных утилит (редакторы, компиляторы и СУБД) с помощью их реализации в виде процессов с повторным вхождением(re-entrant process, или REPs). Эти процессы хранили данные пользователей в принадлежащих им D-банках и совместно использовали одну копию исполняемого код, находящуюся в общем I-банке, который был защищен от записи.

Кроме того, Exec 8 содержала схему обработки ошибок, которая позволяла любой программе перехватывать прерывание, генерируемое при возникновении ошибки (например, при делении на ноль или выходе за границы памяти). Когда установленная пользователем процедура обработки ошибок получала управление, она получала доступ к контексту возникновения ошибки. Системные процессы должны были устанавливать свои собственные процедуры обработки ошибок, однако многие процессы типа REP этого не делали. Это позволяло злоумышленнику установить собственную программу обработки ошибок, создать ошибочную ситуацию в REP-процессе (например, организовав для него D-банк слишком маленького размера), и перехватить прерывание. Получившая управление пользовательская программа обработки ошибок получала возможность доступа как по чтению, так и по записи к банкам I и D процесса типа REP. Это означало, что она могла модифицировать его код, который имел доступ к D-банкам многих пользователей — например, внедрить троянского коня, копирующего чужие данные. Несмотря на то, что подобная троянская программа будет работать только до тех пор, пока существует соответствующий процесс, возможность даже временного получения злоумышленником информации от процесса типа REP является чрезвычайно опасной, т. к. дает ему неограниченный доступ к данным всех остальных пользователей этого процесса.

Системы DEC PDP-10 и VAX

Системы DEC PDP-10 были компьютерами средней производительности, ставшими стандартным средством поддержки распределенной интерактивной обработки данных в 70-х годах. Эти системы функционировали под управлением ОС TENEX, разработанной фирмой BBN. Компьютеры DEC VAX могли функционировать под управлением операционной системы VMS, Unix-подобной системы Ultrix и под управлением операционных систем, разработанных самой DEC. В системе VMS имелся т. н. файл авторизации, в котором находились записи о правах и привилегиях пользователей. Пользователь, который получил доступ к этому файлу, получал неограниченные возможности управления системой.

Индекс: DT1. Система: TENEX.

Источники информации: A. S. Tanenabum. "Operating System Design and Implementation". Prentice-Hall, Englewood Cliffs, NJ, 1987., и R. P. Abbott et al, "Security Analysis and Enhancenments of Computer Operating Systems, Final Report of RISOS Project", National Bureau of Standarts NBSIR-76-1041, Apr. 1976, pp.49-50.

Как и многие другие ОС, TENEX использовала систему паролей для контроля доступа к файловой системе. Процедура проверки правильности пароля была реализована таким образом, что позволяла злоумышленнику подобрать пароль за небольшое количество попыток. Проверка пароля осуществлялась символ за символом, причем выборка символов осуществлялась из области памяти пользовательского процесса, а как только обнаруживался неверный символ — проверка прекращалась. Так как пользователь контролировал область памяти, откуда считывались символы пароля, он мог использовать для радикального ускорения процесса перебора механизм подкачки виртуальных страниц памяти. Для этого вариант пароля помещался на границу виртуальной страницы таким образом, что подбираемый символ размещался в ее последнем байте, а следующая страница отсутствовала в физической оперативной памяти. После этого, если при проверке пароля возникала ситуация подгрузки страницы, это означало что символ подобран верно(процедура проверки обращалась к следующему символу только в том случае, если текущий был верен), в противном случае, процесс подбора этого символа продолжался. После подбора первого символа можно было перейти к подбору второго и т. д. Таким образом, вместо того, чтобы проверять для подбора пароля, состоящего из N символов алфавита мощностью M, M^N комбинаций, злоумышленник мог проверить всего M*N комбинаций. Это полностью дискредитировало всю парольную систему защиты данной ОС.

Индекс: D1. Система: DEC VMS.

Источник информации: "VMS code patch eliminates security breach", Digital Review, June 1, 1987, p. 3.

Данный случай представляет особый интерес, т. к. система DEC VMS была тщательным образом исследована на предмет наличия изъянов в средствах защиты. В новой версии системы был добавлен системный вызов, предоставлявший авторизованным пользователям возможность модификации файла авторизации. Проверка прав инициировавшего запрос пользователя на модификацию этого файла выполнялась на основе анализа содержимого этого же файла. Поэтому в ходе обработки этого запроса ОС первым делом открывала файл авторизации и считывала из него соответствующую информацию. Если пользователь не имел соответствующих прав, выполнение запроса прекращалось. Однако, при задании определенных параметров несмотря на то, что неавторизованный пользователь получал отказ, файл авторизации оставался открытым и доступным для пользователя. Злоумышленник, знавший об этом, мог присвоить себе все права по управлению системой.

Операционная система Unix

Операционная система Unix разрабатывалась лучшими специалистами в области создания ОС с одной целью — обеспечить удобную интерактивную среду обработки данных на компьютерах малой производительности. Поэтому на ранних стадиях разработки вопросам безопасности уделялось относительно мало внимания. Unix поддерживает иерархическую файловую систему с контролем доступа, основанном на понятии "владельца" файла. С каждым файлом связаны идентификаторы владельца и группы, а также атрибуты доступа. Эти идентификаторы

можно изменить с помощью команд chown, chgrp. Атрибуты файла определяют права доступа к нему. Все пользователи подразделяются на три категории:

- владелец файла, его идентификатор совпадает с идентификатором владельца файла;

- члены группы владельца, их идентификатор группы совпадает с идентификатором группы файла;

- прочие — все остальные.

С помощью назначения соответствующих атрибутов доступа можно регламентировать доступ (чтение, запись, выполнение) для каждой категории пользователей. *Изменить права доступа может только владелец файла либо root.* Пользователь с идентификатором root является администратором системы и имеет неограниченные полномочия. Его действия не ограничиваются механизмами контроля доступа.

Все пользователи системы зарегистрированы в специальном файле */etc/passwd*. В нем указаны имена пользователей, их идентификаторы и пароли в зашифрованном виде. Всем пользователям, кроме *root*, разрешен доступ к этому файлу только по чтению. Если злоумышленник получил возможность записи в этот файл, он может создать пользователя с полномочиями *root* и получить полный контроль над системой.

Когда пользователь запускает процесс, как правило, ему присваивается идентификатор этого пользователя, что дает процессу возможность действовать от имени данного пользователя и с его правами доступа. Этому механизму недостает гибкости, поэтому для того, чтобы пользователи могли, например, осуществлять модификацию файла */etc/passwd* в пределах относящейся к ним записи, был введен атрибут *SUID*. Наличие у программы этого атрибута означает, что при ее запуске соответствующий процесс будет иметь идентификатор и права не запустившего его пользователя, а пользователя, являющегося владельцем файла, содержащего программу. Как правило, этот атрибут устанавливается у системных утилит владельцем которых является *root*, требующих для своей работы его полномочий. Например, утилита *setpass*, позволяющая пользователю изменять свой пароль, должна иметь возможность осуществлять запись в файл */etc/passwd*. При этом безопасность системы полностью зависит от корректности реализации подобных программ, поскольку они позволяют любому пользователю выйти за рамки своих полномочий.

Индекс: U2. Система: Unix.

Источник информации: A. S. Tanenbaum. Operating System Design and Implementation. Prentice-Hall. Englewood Cliffs. NJ. 1987.

Присутствующая во многих версиях Unix утилита *lpr*, помещает файл в очередь печати, а при указании опции "-r", еще и удаляет его. В ранних версиях Unix при использовании указанной опции проверка наличия полномочий на удаление заданного файла у пользователя, вызвавшего утилиту *lpr*, отсутствовала. Это позволяло пользователю удалить любой файл, в том числе и файл */etc/passwd*, после чего ни один пользователь не мог войти в систему.

Индекс: U3. Система: Unix.

Источник информации: A. S. Tanenbaum. Operating System Design and Implementation. Prentice-Hall, Englewood Cliffs, NJ, 1987.

В ряде версий Unix программа создания каталогов *mkdir* имела атрибут *SUID*, а ее владельцем был *root*. Создание каталога происходило в две стадии. Сначала посредством специального системного вызова *mknod* для нового каталога выделялись необходимые ресурсы и его владельцем назначался *root*. После этого с помощью другого системного вызова *chown* владельцем нового каталога назначался пользователь, вызвавший *mkdir*. Поскольку эти два этапа не были реализованы как атомарные операции, у пользователей имелась возможность стать владельцем любого каталога и файла в системе, в том числе и, файла */etc/passwd*. Для этого было достаточно запустить *mkdir* как фоновый процесс, и после выполнения первой стадии – создания каталога, приостановить его. После этого пользователь удалял созданный каталог и под тем же именем создавал ссылку (*link*) на файл паролей */etc/passwd*. Затем пользователь инициировал возобновление выполнения утилиты *mkdir*, которая делала пользователя владельцем созданного каталога. Однако, т. к. каталог предварительно был подменен ссылкой на файл */etc/passwd*, пользователь становился его владельцем. Далее, в соответствии со своими полномочиями владельца, он мог изменять этот файл, получая таким образом полный контроль над системой.

Индекс: U4. Система: Unix.

Источник информации: A. V. Discolo, "4.2 BSD Unix security", Computer Science Department, University of California, Santa-Barbara, Apr. 1985.

Во многих версиях Unix команда *sendmail* позволяла неавторизованному пользователю прочитать любой файл в системе. Эта программа могла считывать свои параметры из файла, указанного пользователем, а в том случае, когда формат файла не соответствовал синтаксису команд *sendmail*, выводила на экран каждую строку, в которой встретилась ошибка.

Проверка полномочий пользователя на доступ к файлу параметров не осуществлялась, т. к. *sendmail* имела атрибут *SUID*, а ее владельцем был *root*, так что пользователь легко мог ознакомиться с содержанием любого файла, просто указав его в качестве файла параметров программы *sendmail* (очевидно, что вероятность соответствия строк интересующего пользователя файла синтаксису, принятому в *sendmail*, крайне мала).

Индекс: U5. Система: Unix.

Источник информации: M.Bishop. "Security problems with the UNIX operating system", Computer Science Department, Purdue University, West Lafayette, Indiana, March 1982

Неправильное использование ограничений доступа к каталогам электронной почты приводит к возникновению возможности преодоления системы защиты. В ряде версий Unix почтовая программа после добавления нового сообщения к файлу, в котором хранятся поступившие сообщения ("почтовый ящик" — *mbox*), назначает владельцем этого файла пользователя, на имя которого поступило сообщение. При этом не производится никаких проверок его атрибутов. В дополнение к этому многие системы настроены таким образом, что каталоги, содержащие

электронную почту пользователей, доступны по записи для всех пользователей. Это означает, что любой пользователь может удалить "почтовый ящик" пользователя *root* и заменить его копией командного интерпретатора с установленным атрибутом *SUID* и доступным для выполнения любым пользователем. После этого пользователь может послать на имя *root* любое сообщение. Программа, обслуживающая почту, добавит это сообщение в конец файла — "почтового ящика" и назначит ему нового владельца - *root*. Таким образом, в системе появится командный интерпретатор, с установленным атрибутом *SUID*, владельцем *root*, и доступный для выполнения любому пользователю.

Индекс: U7. Система: Unix.

Источник информации: M. Bishop. "Security problems with the UNIX operating system", Computer Science Department, Purdue University, West Lafayette, Indiana, March, 1982.

В Unix имеется утилита *uux*, реализующая удаленное выполнение ограниченного множества команд и программ. Командная строка, содержащая имя и параметры программы, которую необходимо выполнить, передается программой *uux* на удаленную машину, где осуществляется ее анализ и проверка на принадлежность запускаемой программы к множеству доступных. Если анализ и проверка завершились успешно, порождается соответствующий процесс. В процедуре анализа командной строки содержалась ошибка, которая позволяла выполнять программы, не входящие в множество допустимых.

Разбор командной строки был организован следующим образом: утилита *uux* читала первое слово командной строки (имя команды), проверяла его, а затем пропускала (как аргументы и опции команды) все последующие символы до первого разделителя команд (признака конца команды). Затем производился разбор следующей команды и т. д. После окончания разбора вся строка целиком передавалась командному интерпретатору для выполнения. Но множество проверяемых разделителей команд было неполным, символы "|", "^", ";" учитывались, а "&" и "`" — нет. Таким образом, команды, следующие за неизвестным программе *uux* разделителем, выполнялись, но не проверялись на принадлежность к множеству допустимых. Следовательно, пользователь мог осуществить удаленный запуск любых программ и выполнение любых команд в обход контроля *uux*.

Индекс: U10. Система: Unix.

Источник информации: E. H. Spafford. "Crisis and Aftermath", Comm. of the ACM 32, June 1989, pp. 678-687.

Во многих версиях ОС Unix программа *sendmail* распространялась с установленной по умолчанию отладочной опцией, позволявшей неавторизованным пользователям проникать в удаленные системы. Отладочный режим позволяет посылать сообщения, снабженные программой-получателем, которая запускается на удаленной машине и осуществляет прием сообщения. Эта возможность использовалась разработчиками для отладки программы и в коммерческой версии была оставлена по ошибке. Злоумышленник мог указать в качестве программы-приемника командный интерпретатор, а в текст сообщения включить соответствующие команды, что фактически превращало его в пользователя удаленной системы, не-

смотря на то, что он не был в ней зарегистрирован. Известный вирус Р. Морриса использовал эту возможность для проникновения в удаленные системы.

Индекс: U11. Система: Unix.

Источник информации: D. Gwyn. "UNIX-wizard digest", Vol.6 No. 15, Nov. 1988.

Команда *chfn* системы Unix предоставляет пользователям возможность изменить имя, под которым они зарегистрированы в системе. Поскольку имя пользователя хранится в файле */etc/passwd*, эта программа должна иметь возможность записи в этот файл. Для этого она имеет атрибут *SUID*, а ее владельцем является *root*. Само по себе изменение имени не несет никакой угрозы, но команда *chfn* не осуществляла проверку длины заданной пользователем строки символов. Пользователь, знающий об этом, мог создать очень большой входной буфер, при попытке записи которого окажется, что он не помещается на место прежней записи, и в файл */etc/passwd* будет записана пустая строка. Пустая строка в этом файле интерпретируется как наличие в системе пользователя с пустым именем и паролем, который обладает полномочиями *root*. Таким образом злоумышленник мог создать для себя идентификатор, обладающий максимальными привилегиями.

Индекс: U12. Система: Unix (4.3 BSD on VAX).

Источник информации: J. A. Rochlis, M. W. Eichin, "With microscope and tweezers: the worm from MITs perspective", Comm. ACM 32, June 1989, pp. 689-699.

Программа-демон *fingerd*, имеющаяся в каждой Unix системе, предназначена для передачи удаленным пользователям информации о пользователях локальной системы. Ошибка в этой программе позволяла неавторизованному пользователю запускать на удаленной машине любой процесс. Данная программа содержала в себе фрагмент кода примерно следующего вида:

```
{
 char buffer[100];
 ...
 gets(buffer);
 ...
}
```

Проверка переполнения буфера не осуществлялась. Так как буфер размещался в стеке, то создав в нем фрагмент кода, и вызвав переполнение стека, можно заменить адрес возврата из процедуры таким образом, что управление будет передаваться на этот код. Посредством формирования соответствующего кода злоумышленник мог вынудить систему выполнить любую необходимую ему команду, в том числе запустить командный интерпретатор. Эта возможность наряду с отладочной опцией команды *sendmail* использовалась вирусом Р. Морриса.

Индекс: U14. Система: Unix(SunOS).

Источник информации: J. Purtilo, RISKS-FORUM Digest, Vol. 7 No.2, June 1988.

Процесс-демон *rpc.rexd* обеспечивает в системе Unix поддержку удаленного выполнения программ. В реализации данной программой механизма идентификации присутствовала ошибка. Когда приходил запрос с удаленной системы на выполнение процесса этот демон определял идентификатор пользователя, ини-

цирующего запрос. Если это был пользователь *root*, запрос отвергался, т. к. *root* удаленной системы не имеет полномочий на локальной. Если это был не *root*, *rpc.rexd* проверял легальность для локальной системы идентификатора пользователя удаленной системы, и, в случае наличия в системе пользователя с таким идентификатором, исполнял от его имени указанный процесс. Это означало, что пользователь, имеющий полномочия *root* в одной системе, мог создать пользователя с идентификатором, совпадающим с идентификатором пользователя удаленной системы, и запустить в удаленной системе процесс от имени и с полномочиями этого пользователя. Таким образом, злоумышленник, получивший контроль над одной из систем, мог получить доступ и к другим системам.

Персональные компьютеры

Распространенное системное программное обеспечение персональных компьютеров (ПК) характеризуется отсутствием средств защиты. В первую очередь это касается таких популярных систем как DOS, Windows, Windows 95. Системы, при разработке которых было уделено хотя бы некоторое внимание проблеме защиты информации, выходят за рамки персональных и, как правило, служат в качестве серверных ОС, предоставляющих пользователям тот или иной сервис. В качестве примера можно назвать Novel Netware, Windows NT и многочисленные версии Unix для ПК (SCO, Linux, Solaris, QNX). Наиболее широко распространена информация о вирусных атаках и механизмах функционирования вирусов, использующих отсутствие в ОС, применяемых на ПК, даже минимальной защиты. В качестве примера можно упомянуть публикацию авторов, посвященную этой теме[19] (литература к главе 3). В силу доступности информации по этому вопросу приводить здесь многочисленные факты нарушения безопасности в системах на базе ПК представляется нецелесообразным. Приведенные в таблицах третьей главы индексы примеров, связанных с нарушениями информационной безопасности ПК (IN1, PC1-PC4, MA1, MA2, CA1, AT1) соответствуют индексам из [1] (литература к гл. 3).

Приложение 5.
Операционные системы, сертифицированные в соответствии с требованиями "Оранжевой книги"

Приведенная в конце раздела таблица (табл. П.5.1)создана на основании материалов сайта Национального центра компьютерной безопасности США (National Computer Security Center -– NCSC, http://www.radium.ncsc.mil), посвященного поддержке программы сертификации защищенных систем (Trusted Product Evaluation Program). В таблице присутствуют не только операционные системы, но и сетевые компоненты, подсистемы безопасности и системы управления базами данных. Все эти продукты успешно прошли сертификацию на тот или иной класс защиты в соответствии с требованиями МО США ("Оранжевая книга"). При этом сетевые компоненты оценивались не только в соответствии с требованиями Оранжевой книги, но и со специальными требованиями МО США для сетевых компонентов [4]. В ходе сертификации систем управления базами данных учитывались специальные требования МО США для защиты баз данных [5].

В табл. П.5.1 используются следующие обозначения:

ОС – операционная система;

СК – сетевой компонент;

БД – система управления базами данных;

ПС – подсистема безопасности.

В графе «платформа» для подсистем безопасности и баз данных указана ОС, на которой функционирует сертифицированное приложение.

AOS/VS II версии 3.01

1. Назначение и аппаратное обеспечение.

AOS/VS II – ОС, предназначена для компьютеров семейства Data General's ECLIPSE MV. Оцененная версия исследовалась на аппаратном обеспечении: от настольной мини-ЭВМ MV / 1000DC до большой 6-процессорной ЭВМ MV/ 6000HA. AOS/VS II является ОС общего назначения с разделением времени. AOS/VS II обеспечивает виртуальную память, иерархическую файловую систему и базирующуюся на кольцах защиту ОС и доверенных приложений. Сертифицированная версия оценивалась в несетевой конфигурации и при отсутствии возможностей распределенной обработки. Также прошла оценку версия 3.10 данной операционной системы.

2. Особенности защиты.

Произвольное управление доступом между пользователями и объектами системы реализовано на основе ACL. ACL содержит список пользователей и групп с их правами доступа к объекту. В ACL возможна любая комбинация прав доступа, включая нулевой доступ. Информация о пользователях, такая как привилегии, пароли сохраняются в профиле пользователя. Все профили пользователя имеют нулевой ACL, что означает, что только процесс супервизора может читать

и модифицировать данные объекты. Перед сохранением в профиле пользователя пароль (длиной от 6 до 15 символов) может быть зашифрован. Доступ к данным аудита возможен только пользователям с привилегиями не меньше, чем менеджер системы или со специальной консоли. Аудит событий в системе прекращается только с выключением питания. AOS/VS II осуществляет принудительную очистку следующих объектов: кэш файловой системы, страницы памяти, разделяемые страницы, объекты файловой системы и дисковые устройства.

OpenVMS VAX 6.0 с VAXSMUP03_060

1. Назначение и Аппаратное обеспечение.

OpenVMS VAX Version 6.0 с VAXSMUP03_060 является многопользовательской ОС общего назначения, выполняющейся на процессоре VAX фирмы DEC. Процессор VAX поддерживает ОС путем обеспечения четырех иерархических аппаратных мод доступа и базирующейся на них страничной защите памяти. Кроме того, аппаратное обеспечение поддерживает изоляцию процессов через механизм виртуальной памяти и переключение контекстов. Open VMS обеспечивает среду для распределенных вычислений с единой политикой безопасности. Также прошли оценку версии 6.1 и OpenVMS VAX Version 6.1 Alpha Version 6.1 с ALPRAMP01_061 данной операционной системы.

2. Особенности защиты.

Базовое произвольное управление доступом является стандартным UNIX-механизмом защиты с битами системы, собственника, группы и остального мира. Кроме того поддерживается ACL, содержащий идентификаторы и разрешенные для данного идентификатора виды доступа. Механизм очистки объектов применяется к инициализации памяти, дискам и лентам.

AS/400 OS/400.

1. Назначение и аппаратное обеспечение.

IBM AS/400 является многопользовательской полностью интегрированной вычислительной системой, включающей: реляционную базу данных с SQL интерфейсом, редактор и полный набор обычных функций ОС. IBM AS/400 является системой, основанной на объектах со строгой типизацией и динамическим связыванием. Адресация основана на едином 48-битном адресном пространстве (256 Тбайт), разделяемом всеми пользователями. Все данные инкапсулированы в объекты, управляемые ОС. Другой особенностью IBM AS/400 является концепция абстрактного интерфейса, известного как машинного интерфейса (MI) . Доверенный транслятор преобразует MI код в форму, исполняемую аппаратным обеспечением. Также прошли оценку версии AS/400 OS/400 V2R3M0 и AS/400 с OS/400 V3R2M0 с Feature Code 1920 версии2 для Advanced Series IMPI Hardware данной операционной системы.

2. Особенности защиты.

Так как все разделяемые данные хранятся инкапсулированными в объекты, произвольное управление доступом поддерживается для каждого объекта в системе с использованием общих функций, верифицирующих авторизацию доступа каждого пользователя к каждому экземпляру объекта. Механизм произвольного управления доступом обеспечивает авторизацию как для групп, так и для индивидуальных объектов. Повторное использование объектов в IBM AS/400 контроли-

руется механизмом хранения объектов в момент размещения в памяти хранимых объектов. Все пользователи после регистрации ассоциируются с объектом профиля пользователя, создаваемого администратором системы. Аудит событий в системе можно подразделить на три категории: действия, объекты и пользователи. За счет выбора подмножества событий аудита возможно выбрать уровень контролируемости событий в системе незначительно влияющий на производительность системы. Кроме описанных выше особенностей IBM AS/400 обеспечивает целостность данных. Изоляция ОС осуществляется верификацией всех указателей, передаваемых в TCB через прикладной интерфейс программирования, команды ОС и инструкции MI. Концепция состояния пользовательской программы и системного состояния реализована программно с минимальной аппаратной поддержкой. Инкапсуляция программ и строгая типизация запрещают исполнение данных как программ. Так как только программные объекты исполнимы и только ОС управляет процессами, все межпроцессные коммуникации контролируются системой и могут быть записаны в данные аудита.

Windows NT Workstation и Windows NT Server Version 3.5 с Service Pack 3

1. Назначение и аппаратное обеспечение.

Microsoft Windows NT Workstation и Windows NT Server Version 3.5 являются 32-битными, графическими ОС, поддерживающими популярные Windows приложения, примитивную мультизадачность и симметричную мультипроцессорность (SMP). В оцененную конфигурацию был включен Microsoft Windows NT Service Pack 3 for Windows NT Workstation and Windows NT. ОС оценивалась на аппаратном обеспечении на основе архитектур *Intel* и *Digital Alpha*. Windows NT Server является оптимизированным для работы в сети, так что он обеспечивает не только файловый сервис и сервис принтера, но и приемлимую платформу для вычислений клиент-сервер. Windows NT Server обеспечивает большое количество графических средств администрирования и средств защиты от сбоя. В оцениваемой конфигурации Windows NT Server поддерживал до 4 процессоров. Windows NT Workstation является 32 битовой ОС и поддерживает приложения, разработанные для MS-DOS, Windows 3.x, and Windows NT. Особенностью является полная защита от сбоев приложений и ОС. В оцениваемой конфигурации Windows NT Workstation поддерживал до 4 процессоров. Так как оцениваемая конфигурация не включала сетевого окружения, оба продукта рассматривались как отдельные рабочие станции. При этом различия в безопасности между Windows NT Workstation и Windows NT Server в оцениваемой конфигурации минимальны, что позволяет рассматривать продукт как платформу Windows NT.

2. Особенности защиты.

Платформа Windows NT реализует защиту данных на основе произвольного управления доступом. Модель безопасности Windows NT позволяет контролировать доступ ко всем объектам системы и всем файлам, использующим Windows NT файловую систему (NTFS). До того, как приложение или процесс может получить указатель на объект, включая файл и объект данных, система безопасности Windows NT верифицирует то, что процесс имеет подходящую авторизацию для этого. Windows NT не позволяет получить процессу доступ к файлу до тех пор, пока собственник файла или системный администратор не позволят этого.

Windows NT включает безопасную последовательность входа в систему (logon) с использование комбинации клавиш Ctrl-Alt-Del.

Tandem Guardian 90 с Safeguard

1. Назначение и аппаратное обеспечение.

Tandem NonSTop - многопроцессорная система, предназначенная для непрерывной обработки транзакций. Комбинация аппаратной архитектуры NonSTop с программным обеспечением ОС Guardian 90 обеспечивает защищенную от сбоев и обеспечивающую целостность данных систему на основании следующих особенностей: процессоры связаны высокоскоростными внутренними шинами, дублирующие диски и контролеры устройств связаны с различными процессорами, несколько источников питания, отказоустойчивое программное обеспечение выполняется параллельно на нескольких процессорах, мониторинг транзакций, а также подистсема детектирования программных и аппаратных ошибок. Аппаратное обеспечение NonStop обеспечивает два режима выполнения программ: пользовательский и привилегированный для выполнения программного обеспечения Guardian 90. Аппаратное обеспечение обеспечивает также механизмы виртуальной памяти, разделения адресного пространства программ и данных, а также разделение индивидуальных пространств процессов.

2. Особенности защиты.

ОС Guardian 90 совместно с подсистемой Safeguard обеспечивает произвольное управление доступом к объектам следующих типов: тома дисков, подтома, файлы, устройства и процессы. ACL могут разрешать и запрещать доступ к объектам или группам объектов для пользователей и групп. Только системный администратор может устанавливать значения ACL. Записи о данных аутентификации пользователя хранятся в едином защищенном файле. В системе осуществляется аудит следующих событий: logon и logoff пользователей, открытие и закрытие файлов, создание нового процесса и использование команд оператора. Поддерживается три административных типа ролей. Первые два типа определяют пользователей, которым разрешено обрабатывать команды аудита, терминальные команды, изменять настройки системы и прерывать функционирование системы. Третий тип является владельцем и отвечает за поддержку контроля доступа субъектов системы к объектам.

Amdahl UTS/MLS Release 2.1.5+

1. Назначение и аппаратное обеспечение.

Amdahl UTS/MLS 2.1.5+ является расширением, поддерживающим мандатный доступ, Amdahl UTS 2.1.5+, UNIX System V совместимого продукта. UTS/MLS – многопользовательская, многозадачная ОС, поддерживающая до 65000 пользователей. UTS/MLS поддерживает System V совместимость приложений и совместима с System V Interface Definition (SVID). UTS/MLS оценивалась на аппаратном обеспечении Amdahl 5990 и 5995 мэйнфреймах, реализующих архитектуру System 370/XA. Также прошла оценку версия ULTRIX MLS+ версии2.1 на VAX Station 3100 данной операционной системы.

2. Особенности защиты.

В добавление к традиционным механизмам защиты ОС UNIX, UTS/MLS обеспечивает мандатный контроль доступа для ограничения потока информации на основе классификации информации и степени доверия к пользователю, пытающемуся получить доступ к информации. Мандатная политика безопасности построена на основании модели Белла и Лападула и соответствует политике DoD. UTS/MLS обеспечивает 255 иерархических классификационных уровня и 1024 неиерархических категории. Произвольное управление доступом применимо к файлам, директориям, именованным каналам, устройствам, символическим связям, разделяемой памяти, очередям сообщений и множеству семафоров. Мандатный контроль применим к тому же множеству объектов также как и к процессам, заданиям принтера и почтовым сообщениям. Администратор имеет возможность ограничить пользователей и порты login в соответствии с выбранными им областями классификации. UTS/MLS реализует политику безопасности, предотвращающую неавторизованную деклассификацию информации и неавторизованное изменение кода программ. Механизм мандатного контроля доступа реализован на основе расширения стандартных полей групп UNIX. Произвольное управление доступом использует стандартную концепцию битов защиты ОС UNIX Owner/Group/Other. Доверенный путь реализуется во время фазы login для проверки того, что пользователи работают действительно с TCB системы. Безопасная коммуникация между процессами позволяет доверенному серверу надежно идентифицировать и получить степень доверия процесса-клиента, посылающего сообщение серверу.

SEVMS VAX версии 6.0

1. Назначение и аппаратное обеспечение.

SEVMS VAX Version 6.0 с SEVMS_VAXSMUP03_060 является многопользовательской ОС общего назначения, выполняющейся на Digital's Virtual Address Extension (VAX) процессорах. Аппаратное обеспечение VAX поддерживает механизм защиты на основе четырех иерархических режимов доступа и защиты страниц памяти на основе данных мод. Кроме того, аппаратно поддерживается изоляция процессов на основе управления виртуальной памятью и переключения контекста процесса. Оценивалась локальная и кластерная конфигурация VAX систем (при которой единая политика безопасности применена ко всем узлам кластера). Также прошли оценку версии SEVMS VAX Version 6.1 и SEVMS VAX Version 6.1 Alpha Version 6.1 with ALPRAMP01_061 данной операционной системы.

2. Особенности защиты.

Базовое произвольное управление доступом обеспечивается категориями пользователей: system, owner, group, и world. Кроме того, поддерживаются списки контроля доступа, содержащие идентификаторы и авторизованные списки доступа для идентификатора. Пользователь может быть ассоциирован с различным числом идентификаторов, что обеспечивает удобный механизм разрешений для группирований разрешений доступа на основе идентификаторов. Мандатный контроль доступа поддерживает разрешения о доступе на основе меток безопасности данных и пользователя. Доступ ко всем объектам SEVMS подчиняется мандатному контролю доступа. Повторное использование объектов включает контроль за по-

вторным использованием памяти, областей дисков и лент, а также данных принтера.

CX/SX версии 6.1.1

1. Назначение и аппаратное обеспечение.

CX/SX версии 6.1.1. есть расширение Harris' UNIX-основанной CX/UX ОС, которая выполняется на серии 4000 Night Hawk. Night Hawk 4000 серия – многопроцессорная (до 8 процессоров) система на основе Motorola 88100, процессоре с уменьшенным множеством инструкций. Архитектура двойной шины используется для интерфейса с памятью, контроллерами и периферией для достижения высокой производительности системы. CX/SX – многозадачная система, использующая примитивы, что позволяет полностью использовать преимущества конфигурации Night Hawk's для достижения высокой производительности и безопасности приложений. CX/SX была создана добавлением AT&T's System V/MLS особенностей в CX/UX в дополнение к правилам безопасности, разработанных Harris. Также прошла оценку версия 6.2.1 данной операционной системы.

2. Особенности защиты.

CX/SX является программно совместимой с CX/US системой, несмотря на добавление особенностей, относящихся к безопасности системы.. CX/SX обеспечивает традиционные для контроля доступа в UNIX биты защиты для произвольного управления доступом. В системе также добавлены команды, связанные с возможностью пользователя динамически создавать группы в системе. Мандатный контроль доступа обеспечивает ограничение распространения информации только авторизованным пользователям. CX/SX обеспечивает 255 иерархических уровня и 1024 неиерархических категории. Мандатная политика безопасности построена на основании модели Белла и Лападула. CX/SX мандатная политика безопасности также предотвращает неавторизованную деклассификацию информации и неавторизованное изменение кода программ.

HP-UX BLS Release 8.0.4

1. Назначение и аппаратное обеспечение.

Hewlett Packard's HP-UX BLS (Уровень B безопасности) является HP версией Unix, разработанной для того, чтобы удовлетворить DoD Trusted Computer System Evaluation Criteria для систем класса B1. HP-UX BLS совместима с System V Interface Definition 2 и стандартами IEEE POSIX 1003.1 и1003.2. HP-UX BLS включает важные особенности из Berkeley Software Distribution 4.3. HP-UX BLS оценивалась на компьютерах семейства HP9000/S800.

2. Особенности защиты.

HP включила стандартную технологию разработки защищенных систем, разработанную SecureWare, независимой фирмой, в HP-UX, стандартную многопользовательскую многозадачную UNIX ОС, разработанную HP. Основой HP-UX BLS является многоуровневая безопасность, характеризуемая концепцией меток субъектов / объектов в совокупности с политикой мандатного контроля доступа. Политика мандатного контроля доступа позволяет присвоить начальные уровни безопасности для пользователей, файлов, устройств и коммуникациям между программами. Мандатная политика безопасности построена на основании модели Белла и Лападула. Произвольное управление доступом реализовано посредством объединения стандартных для ОС битов защиты с ACL. Система поддерживает

принцип наименьших привилегий для программ и пользователей, контролируемый экспорт и импорт данных в многоуровневое безопасное окружение, контроль доступа к общим директориям, а также расширенный аудит.

Trusted IRIX/B Release 4.0.5

1. Назначение и аппаратное обеспечение.

Trusted IRIX/B является распределенной системой из IRIS Indigo/Entry Graphics рабочих станций, связанных через физически защищенный Ethernet. Система разработана Silicon Graphics Computer Systems, Inc. Trusted IRIX/B – многопользовательская многозадачная ОС общего назначения на основе UNIX, с поддержкой 3-D графики, 2-D графики, видео и аудио. Trusted IRIX/B оценивалась на рабочей станции с процессором MIPS R4000SC. Система поддерживает TCP/IP протокол и сервисы для коммуникации между станциями.

2. Особенности защиты.

Trusted IRIX/B – безопасная версия IRIX 4.0, IRIS Indigo многопользовательской многозадачной ОС. Система использует стандартный для UNIX механизм. Кроме того, Trusted IRIX/B обеспечивает мандатного контроля доступа для ограничения потока информации. Решения Trusted мандатного контроля доступа IRIX/B базируются на секретности и целостности информации и степени доверия пользователя, пытающегося получить доступ к информации. Мандатная политика безопасности построена на основании модели Белла и Лападула и модель Биба для целостности. Система обеспечивает 256 иерархических уровня и 65536 неиерархических категории. Мандатная политика безопасности также предотвращает неавторизованную деклассификацию информации и неавторизованное изменение кода программ. Мандатный контроль доступа используется как механизм изоляции TCB. Trusted IRIX/B реализует мандатное и произвольное управление доступом по отношению к сокетам. Произвольное управление доступом разрешает посылать процессу данные в сокет только если процесс включен в ACL сокета. Все рабочие станции в системе поддерживают одну базу данных идентификации и аутентификации с использованием Network File System (NFS).

OS 1100/2200 версия SB4R7

1. Назначение и аппаратное обеспечение.

OS 1100/2200 версии SB4R7 – многозадачная ОС общего назначения, исполняемая на Unisys 1100/90, 2200/100, 2200/200, 2200/400, 2200/600, 2200/600ES и 2200/900 аппаратном обеспечении. Эти модели имеют сходную архитектуру, реализующую аппаратную защиту памяти вместе с несколькими режимами выполнения. ОС поддерживает многозадачность, причем каждая задача имеет свое виртуальное адресное пространство и возможность разделения защищенных подсистем (общие банки памяти). OS 1100/2200 версииSB4R7 поддерживает пакетную обработку, режим разделения времени, и режим обработки транзакций. TCB OS 1100/2200 версии SB4R7 состоит из особых реализаций следующих компонентов: Executive, CMS1100, TELCON, SOLAR, PC Console, COMUS, FAS, IRU, LA, UDSC, MCB, PERCON, SIMAN, SSP/SCF, TLABEL, и DPREP1100. Коммуникации обеспечиваются Distributed Communications Processors (DCPs), работающим как оконечный процессор под управлением CMS 1100 и TELCON. OS 1100/2200

Executive является основным компонентом системы. TCB реализует произвольное *управление доступом и мандатную политику безопасности и базис для построения* безопасных приложений. Также прошли оценку версии SB3R6, SB3R8 и версия SB4R2 данной операционной системы.

2. Особенности защиты.

OS 1100/2200 версии SB4R7 обеспечивает изоляцию OS 1100/2200 Executive с помощью аппаратного механизма защиты. Доступ к подсистемам OS 1100 (разделяемым банкам памяти) защищен контролем доступа в точках входа. Если доступ во входной точке разрешен, система переключает атрибут безопасности процесса в значение, определенное подсистемой. Изоляция процессов достигается виртуальным адресным пространством процесса, отдельными для процессов стеками и определенным изменением состояний. OS 1100/2200 версииSB4R7 обеспечивает мандатный контроль доступа на основе модели Белла и Лападула. При этом система поддерживает 64 иерархических уровня и 30 категорий. Администратор может присвоить символическое имя каждому уровню безопасности и каждой категории. Произвольное управление доступом, обеспечиваемое OS 1100/2200 версии SB4R7, основан на ACL. При произвольном управлении доступом возможно специфицировать: кто, когда и как может иметь доступ к объекту. OS 1100/2200 *обеспечивает возможность ограничения привилегий пользователя*, которые он требует для выполнения полномочий. Все пользователи (включая операторов и администраторов) являются субъектами данного механизма. Система поддерживает роль администратора безопасности и другие роли. Администратор отвечает за безопасность всей системы. Активность всех специфических ролей автоматически подвергается аудиту.

Trusted XENIX версии 4.0

1. Назначение и аппаратное обеспечение.

Trusted XENIX версии 4.0 – многопользовательская многозадачная многоуровневая в смысле безопасности UNIX-подобная ОС, являющаяся расширением Trusted XENIX версии 3.0. Trusted XENIX содержит много функциональных расширений и расширений безопасности при сохранении двоичной совместимости с программами, разработанными для IBM Personal Computer XENIX версий1.0 и 2.0. Также прошла оценку версия 3.0 данной операционной системы.

2. Особенности защиты.

Trusted XENIX разработан для обеспечения высокого уровня безопасности для окружения, требующего обработки секретных данных на персональных компьютерах или интерфейса с многими уровнями безопасности. Trusted XENIX *обеспечивает мандатный контроль доступа на основе модели Белла и Лападула.* Произвольное управление доступом включает как стандартные для UNIX биты защиты, так и ACL. Система поддерживает принцип наименьших привилегий для каждой из четырех предопределенных ролей пользователя, доступных в многопользовательской моде. Существуют следующие привилегированные роли: администратор безопасности системы, оператор безопасности, администратор пользователей и аудитор. Это разграничение обеспечивается строгим ограничением привилегий для предопределенных операций. Кроме того, все действия привилегированных пользователей контролируются аудитом и записи аудита не могут быть изменены непривилегированными пользователями, а также администратором

безопасности системы, оператором безопасности или администратором пользователей. Системный программист отвечает за начальную конфигурацию и настройку системы. Данная роль существует только в однопользовательском режиме. Возможность удаленной связи с Trusted XENIX реализована в конфигурации клиент / сервер.

XTS-300 версии 4.1

1. Назначение и аппаратное обеспечение.

XTS-300 - объединение многоуровневая в смысле безопасности ОС STOP 4.1 и аппаратного обеспечения Intel 80486 PC/AT, использующего шину EISA. STOP - многопользовательская система, поддерживающая до 19 терминалов. В системе могут выполняться до 200 процессов одновременно, причем каждый из них имеет до 4 Гб виртуальной памяти. STOP поддерживает интерфейс UNIX System V interface и обеспечивает совместимость в объектном коде со стандартом Intel386 Family Binary Compatibility Specification 2. Сетевая поддержка осуществляется на основе Secure Communications Subsystem, которая выносит обработку низких уровней сетевых протоколов из TCB. Также прошли оценку версии 3.1E, 3.2E и 4.1a данной операционной системы.

2. Особенности защиты.

STOP состоит из четырех компонентов: ядра безопасности, которое выполняется в самом привилегированном кольце и выполняет мандатный и часть произвольного управления доступом; TCB системный сервис, которое выполняется в следующем кольце и реализует иерархическую файловую систему, поддерживает пользовательский ввод / вывод, и реализует оставшуюся часть произвольного управления доступом; доверенное программное обеспечение, обеспечивающее оставшиеся сервисы безопасности и команды пользователя; Общий системный сервис приложений, невключенный в TCB, которой выполняется в наименее привилегированном кольце и обеспечивает самом UNIX-подобный интерфейс. Система обеспечивает мандатное и произвольное управление доступом для целостности и конфиденциальности данных. XTS-300 Trusted XENIX обеспечивает мандатный контроль доступа на основе модели Белла и Лападула; целостность поддерживается на основе модели Биба. Механизм доверенного пути обеспечивается реализацией Secure Attention Key (SAK). Разделение ролей администратора и оператора реализовано с использованием политики целостности. Система реализует принцип наименьших привилегий для каждой из ролей, причем вес действия привилегированных пользователей контролируются с помощью аудита. STOP поддерживает механизмы поиска нарушителя.

TCB имеет следующие архитектурные характеристики: минимизация, иерархичность, абстракция и скрытие данных. Система спроектирована на основе модели безопасности и спецификаций высокого уровня.

Таблица П.5.1

N	Продукт	Тип системы	Производитель	Дата Сертификации	Класс защиты	Дополнительные требования и классы, которым удовлетворяют отдельные компоненты и функции защиты	Платформа
1	RACF с MVS 1.0	ПС	IBM	23.07.84	C1		IBM MVS
2	ACF2 с MVS/SP	ПС	Computer Associates International	03.08.84	C2		IBM MVS
3	TOP SECRET с MVS 3.0	ПС	Computer Associates International	02.04.85	C2		IBM MVS
4	UTX/32S 1.0	ОС	Gould, Inc., Computer Systems Division	31.12.86	C2		Gould PowerNode 6000 и 9000
5	MCP/AS w/Infoguard 3.7	ПС	Unisys Corporation	05.09.87	C2		Unisys
6	ACF2/VM с VM/SP 3.1	ПС	Computer Associates International	11.09.87	C2	1.Управления конфигурацией (B2). 2.Произвольный доступ (B3).	IBM 370
7	MVS/XA с RACF	ОС	International Business Machines	15.06.88	C2		IBM370/XA
8	Primos 21.0.1.DODC2A	ОС	Prime Computer, Inc.	24.06.88	C2		Prime's серия 50
9	VM/SP с RACF	ОС	International Business Machines	28.09.89	C2		IBM 370
10	SVS/OS CAP 1.00	ОС	Wang Laboratories, Inc.	28.09.90	C2		Wang семейство продуктов VS
11	ConvexOS/Secure 10.0	ОС	CONVEX Computer Corporation	29.05.92	C2		CONVEX серий C2 и C3
12	Trusted OS/32 08-03.3S	ОС	Concurrent Computer Corporation	01.10.92	C2	1. Архитектура системы (B1).	Concurrent серии 3200 и 8/32
13	AOS/VS II 3.01	ОС	Data General Corporation	01.06.94	C2	1. Архитектура системы (B1) 2. Произвольный доступ (B3)	Data General's ECLIPSE MV
14	AOS/VS II 3.10	ОС	Data General Corporation	05.12.94	C2	1. Архитектура системы (B1) 2. Произвольный доступ (B3)	Data General's ECLIPSE MV

15	Open VMS VAX 6.0	ОС	Digital Equipment Corporation	30.06.94	C2	1. Подсистема обнаружения нарушителя	VAX
16	Open VMS VAX 6.1	ОС	Digital Equipment Corporation	14.07.95	C2	1. Подсистема обнаружения нарушителя	VAX
17	AS/400 с OS/400 V2R3M0	ОС	IBM	05.10.95	C2	1. Произвольный доступ (B3). 2. Управление безопасностью (B2).	IBM AS/400
18	Windows NT 3.5	ОС	Microsoft	31.07.95	C2	1. Управление безопасностью (B2) 2. Прямое взаимодействие (B2)	Intell, Alpha
19	Guardian-90 с Safeguard S00.01	ОС	Tandem Computers Inc.	14.06.93	C2	1. Архитектуре системы (B1). 2. Произвольный доступ (B3)	Tandem NonSTop
20	Netware 4.11 с IntranetWare Support Pack 3A And Directory Services Update DS.NLM v5.90, DSREPAIR.NLM v4.48, ROLLCALL .NLM v4.10	СК	Novell Incorporated	07.10.97	C2		Intel
21	INFORMIX-OnLine/Secure 4.1	СУБД	Informix Software, Inc.	15.11.94	C2		UNIX
22	INFORMIX-OnLine/Secure 5.0	СУБД	Informix Software, Inc.	15.11.94	C2		UNIX
23	Oracle7 7.0.13.1 с Procedural Option	СУБД	Oracle Corporation	05.04.94	C2		UNIX
24	SQL Server 11.0.6	СУБД	Sybase, Inc	07.01.97	C2		UNIX
25	System V/MLS 1.1.2	ОС	AT&T	07.09.89	B1	1. Прямое взаимодействие (B2). 2. Метки чувствительности субъектов (B2) 3. Метки устройств (B2)	AT&T 3B2/500 и AT&T 3B2/600
26	MVS/ESA	ОС	International Business Machines	17.09.90	B1	1. Произвольный доступ (B3). 2. Архитектура класса B1	IBM ESA/370

27	CMW + for A/UX 1.0	ПС	SecureWare, Inc.	30.01.91	B1		
28	OS 1100/2200 SB3R6	ОС	Unisys Corporation	05.04.91	B1		Unisys 1100/90, 2200/100, 2200/200, 2200/400, 2200/600, 2200/600ES и 2200/900
29	OS 1100/2200 SB3R8	ОС	Unisys Corporation	05.11.91	B1	1. Произвольный доступ (B3). 2. Управление безопасностью (B2).	Unisys 1100/90, 2200/100, 2200/200, 2200/400, 2200/600, 2200/600ES и 2200/900
30	UTS/MLS 2.1.5+	ОС	Admahl Corporation	07.01.94	B1	1. Метки чувствительности субъектов (B2) 2. Метки устройств (B2) 3. Прямое взаимодействие (B2)	Amdahl 5990 и 5995
31	SEVMS VAX 6.0	ОС	Digital Equipment Corporation	30.06.94	B1	1. Метки чувствительности субъектов (B2) 2. Метки устройств (B2) 3. Прямое взаимодействие (B2)	VAX
32	SEVMS VAX 6.1	ОС	Digital Equipment Corporation	14.07.95	B1	1. Метки чувствительности субъектов (B2) 2. Метки устройств (B2) 3. Прямое взаимодействие (B2)	VAX
33	Ultrix MLS+ 2.1	ОС	Digital Equipment Corporation	18.04.96	B1	1. Управления конфигурацией (B2) 2. Произвольный доступ (B3) 3. Дистрибуция (A1) 4. Управление безопасностью (B3) 5. Прямое взаимодействие (B2)	VAX
34	CX/SX 6.1.1	СК	Harris Computer Systems Corporation	15.09.93	B1	1. Метки чувствительности субъектов (B2) 2. Метки устройств (B2)	4000 Night Hawk

35	CX/SX 6.2.1	СК	Harris Computer Systems Corporation	18.09.95	B1	1. Метки чувствительности субъектов (B2) 2. Метки устройств (B2)	4000 Night Hawk
36	HP-UX BLS 8.04	ОС	Hewlett Packard Corporation	21.09.93	B1	1. Произвольный доступ (B3). 2. Управление безопасностью (B2).	HP9000/S800
37	HP-UX BLS 9.0.9+	ОС	Hewlett Packard Corporation	01.12.94	B1	1. Произвольный доступ (B3). 2. Управление безопасностью (B2).	HP9000/S800
38	Trusted Irix/B 4.0.5	ОС	Silicon Graphics Inc	06.02.95	B1		Indigo/Entry Graphics
39	OS 1100/2200 SB4R7	ОС	Unisys Corporation	20.04.95	B1	1. Произвольный доступ (B3). 2. Управление безопасностью (B2) 3. Прямое взаимодействие (B2).	Unisys 1100/90, 2200/100, 2200/200, 2200/400, 2200/600, 2200/600ES и 2200/900
40	Trusted UNICOS 8.0	СК	Cray Research, Inc.	09.03.95	B1	1. Произвольный доступ (B3). 2. Управление безопасностью (B2) 3. Дистрибуция (A1)	CRAY Y-MP C90, M90, E, и EL серий
41	INFORMIX-OnLine/Secure Release 4.1	СУБД	Informix Software, Inc	21.03.94	B1		UNIX
42	INFORMIX-OnLine/Secure Release 5.0	СУБД	Informix Software, Inc	15.11.94	B1		UNIX
43	Trusted Oracle7 7.0.13.1 c Procedural Option	СУБД	Oracle Corporation	05.05.94	B1		UNIX
44	Secure SQL Server Version 11.0.6	СУБД	Sybase, Inc.	07.01.94	B1		UNIX
45	Multics MR 11.0	ОС	Honeywell Information Systems	01.09.85	B2		Honeywell Level 68 and DPS-8M
46	Trusted Xenix 3.0	ОС	Trusted Information Systems, Inc.	08.04.92	B2		Intel

47	OS 1100/2200 SB4R2	ОС	Unisys Corporation	07.10.92	B2		Unisys 1100/90, 2200/100, 2200/200, 2200/400, 2200/600, 2200/600ES и 2200/900
48	Trusted Xenix 4.0	ОС	Trusted Information Systems, Inc	17.09.93	B2		Intel
49	VSLAN 5.0 Network Component MDIA	СК	Verdix Corporation	22.09.90	B2		
50	VSLAN 5.1 VSLANE 5.1	СК	Verdix Corporation	11.01.94	B2		
51	VSLAN 6.0 VSLANE 6.0	СК	General Kinetics Inc	20.07.95	B2		
52	XTS-200 STOP 3.1.E	ОС	WANG Federal, Inc.	27.05.92	B3		Bull HN Information Systems Inc. DPS 6 PLUS и DPS 6000
53	XTS-300 STOP 4.1	ОС	Wang Federal, Inc.	30.05.95	B3		Intel
54	XTS-300 STOP 4.1.a	ОС	Wang Federal, Inc.	31.10.95	B3		Intel
55	SCOMP STOP 2.1	СК	HFS, Incorporated	24.12.84	A1		HONEYWELL уровень 6/DPS 6 16-bit миникомпьютер с Модулем защиты (Security Protection Module), добавленным в центральный процессор
56	MLS LAN Secure Network Server System	СК	Boeing Company	28.08.91	A1		

Приложение 6. Формальная модель безопасности Trusted Mach

Модель Trusted Mach основана на традиционной модели Bell-LaPadula, представляющей собой модель состояний и событий. Состояния — абстрактной машины характеризуются субъекты, объекты, уровни безопасности, списков прав доступа, текущее множество доступов и нерархии объектов. События — это операции, вызывающие переход системы из одного пространства состояний из одного состояния в другое. События определяют, как переменные состояния могут менять свои значения. Модель использует оператор следующего значения, определяющий новые значения, присваиваемые переменным новые состояния после произошедших событий.

1. Условные обозначения

Для описания модели используется логика предикатов первого порядка, дополненная оператором временного значения, называемым оператором следующего значения. Ниже приведены математические условные обозначения, используемые при дальнейшем изложении:

$\in$	принадлежность, $x \in S$ означает, что элемент x принадлежит множеству S
$\neg$	оператор отрицания ($\neg p$ эквивалентно утверждению "p ложно")
$\exists$	квантор существования
$\forall$	квантор всеобщности
если p, то q	логическое выражение, эквивалентное (p и q) или $\neg p$
если p, то $q1$, иначе $q2$	логическое выражение, эквивалентное (p и $q1$) или ($\neg p$ и $q2$)
$\cap$	пересечение множеств
$\cup$	объединение множеств
$\subseteq$	подмножество
$\setminus$	разность множеств
$'$	оператор следующего значения (например, $b' = b \cup \{(t, o, r)\}$ утверждает, что новое значение переменной состояния, называемой текущим множеством доступов b, представляет собой ее текущее множество значений, дополненное доступом r к объекту o для субъекта t.
Пусть	привязка идентификатора к выражению (например, пусть $x = f(a, b)$ $E(x)$ означает, что в выражении E каждое вхождение x должно быть заменено на $x = f(a, b)$)

И	логический оператор (p и q утверждает, что и p, и q истинны)
Или	логический оператор (p или q утверждает, что p истинно, q истинно, или что и p, и q истинны))
≤	меньше или равно ($l_1 \le l_2$ означает, что l_2 доминирует над l_1, где l_1 и l_2 - уровни безопасности)
≥	больше или равно ($l_1 \ge l_2$ означает, что l_1 доминирует над l_2, где l_1 и l_2 - уровни безопасности)

2. Математическая модель безопасности Trusted Mach

Система

Система Σ представляет собой кортеж (V, E, ν, ν_0), где

V - пространство состояний,

E - множество событий,

ν - отношение следующего состояния вида ν: $E \times V \to V$, и

$\nu_0 \in V$ - начальное состояние.

Пространство состояний

Пространство состояний V состоит из элементов $\nu = (b, \alpha, f, H, M)$, где b - *текущее множество доступов*, α - *предикат активации*, f - *функция уровня безопасности*, H - *граф иерархии объектов*, а M - *список разрешения доступов для объектов*.

Текущее множество доступов b является подмножеством множества ($S \times O \times A$), где S - множество субъектов, O - множество объектов, A - множество режимов доступа, определенных в системе. Текущее множество доступов представляет типы доступа к объектам, разрешенные в настоящий момент субъектам системы. Например, (s, o, *read*) $\in b$ утверждает, что субъект s в настоящий момент имеет доступ чтения к объекту o, что позволяет субъекту s получать содержимое объекта o без посредничества ТСВ.

Для любого объекта постоянный предикат *named* определяет, является ли объект именованным или нет.

Предикат активации α лучше всего описать в терминах двух вторичных предикатов, α_s и α_o, которые определяют, является ли субъект или, соответственно, объект активным. Данная модель не предусматривает непосредственное моделирование создания и уничтожения субъектов и объектов. Вместо этого предполагается, что все субъекты и объекты уже были созданы в процессе инициализации системы, и операции их создания и удаления рассматриваются, соответственно как их активация и деактивация. Далее будем считать, что субъекты и объекты удовлетворяют принципу постоянства (т.е. мандатные и дискреционные атрибуты каждого субъекта, а также мандатные атрибуты каждого объекта остаются неизменными в течение всего времени их существования). Изменение дискреционных атрибутов объектов допускается.

Функция уровня безопасности f - это постоянная функция, образованная тремя составляющими функциями, f_s, f_o и f_{attr}, которые определяются следующим образом:

f_s - функция, определенная на множестве субъектов S. Для каждого субъекта s, $f_s(s)$ определяет его уровень безопасности.

f_o - функция, определенная на множестве объектов O. Для каждого объекта o, $f_o(o)$ определяет его уровень безопасности.

f_{attr} - функция, определенная на множестве всех именованных объектов.

Для каждого именованного объекта o, $f_{attr}(o)$ определяет уровень безопасности его атрибутов, не связанных с содержанием этого объекта.

Областью значений каждой из вышеназванных функций является множество уровней безопасности L.

Граф иерархии объектов H - это множество упорядоченных пар $O \times O$, объединенных отношением *Edges*, где O - множество объектов, а *Edges* - множество упорядоченных пар $O \times O$. В данной модели предполагается, что все объекты уже существуют, а создание объекта интерпретируется как его активация. Следовательно, события, связанные с созданием и удалением объектов, могут изменять только ребра графа H, обозначаемые как *Edges*(H).

Для простоты будем полагать, что все безымянные объекты содержатся в системном каталоге (*IPC_root*), который всегда является активным и имеет уровень безопасности ниже, чем уровень безопасности любого объекта.

Для того, чтобы определить функцию разрешения доступа, определим сначала ACL_{model} как подмножество множества $S \times A$, где S - это множество субъектов, а A - множество режимов доступа. Тогда функция разрешения доступа M определяется как:

$$M: O \rightarrow ACL_{model},$$

где для каждого объекта $o \in O$, $M(o)$ определяет множество режимов доступа, разрешенных субъектам.

Событие

Событие e определяет недетерминистские переходы пространства состояний V в себя.

Действие события e описывается следующим выражением:

$$\forall\, v \in V,\ v \cdot e = \{v' \in V \mid ((e, v), v') \in v\}$$

Обобщение этого выражения на множества описывается следующим выражением:

$$V_0 \cdot e = \{v' \in V \mid v \in V_0 \text{ и } ((e, v), v') \in v\},$$

где V_0 является подмножеством V.

Функция *invoker_of* определена на множестве событий следующим образом:

$$invoker\_of: E \rightarrow S \cup TCB,$$

где *invoker_of*(e) - это субъект, вызывающий событие.

invoker представляет собой объединение двух множеств: множества субъектов и TCB.

Замечание: Для данного состояния v будем говорить, что v изменяется на v′ в результате события e так, что $((e, v), v') \in v$.

Последовательность состояний

Для данной системы $\Sigma = (V, E, \nu, \nu_0)$ последовательность состояний описывается как

$$(\nu_0, e_1, \nu_1, ..., e_n, \nu_n),$$

где $((e_i, \nu_{i-1}), \nu_i) \in \nu$.

Элемент состояния $\nu \in V$ называется достижимым, если существует вычисление

$$(\nu_0, e_1, \nu_1, ..., e_n, \nu_n = \nu),$$

где $e_i \in E$ для $i = 1, ..., n$.

Режимы доступа

Пусть $A = \{read, append, write, create, control\}$ - множество режимов доступа. Режимы доступа имеют следующую семантику:

read - субъект, который имеет к объекту доступ чтения, может просматривать содержимое объекта.

write - субъект, который имеет к объекту доступ записи, может изменять содержимое объекта; в этой модели доступ записи включает в себя и доступ чтения.

append - субъект, который имеет к объекту доступ добавления данных, может изменять содержимое объекта "вслепую" (например, добавлять данные в конец объекта, не просматривая его).

create - субъект, который имеет к объекту данного типа доступ создания, может создавать объекты этого типа.

control - субъект, который имеет к объекту доступ контроля, может изменять ACL объекта, а также уничтожать объект.

Реализация Trusted Mach поддерживает также и доступ управления *manage* для типов. Если субъект имеет право доступа *manage* к объекту типа T, то этот субъект может стать сервером, обслуживающим все объекты, которые являются экземплярами T. В этой модели право доступа *manage* совмещено с доступом записи *write* для всех объектов, которые являются экземплярами T.

Критерий безопасности системы

Пусть V_Σ - пространство состояний,

C – подмножество безопасных состояний V_Σ,

CT - подмножество пространства переходов $V_\Sigma \times V_\Sigma$,

$\Sigma = (V_\Sigma, E_\Sigma, \nu_\Sigma, \sigma_0)$ - система.

Тогда будем говорить, что состояние $\nu \in V_\Sigma$ *безопасно* по отношению к C, если $\nu \in C$. Будем говорить, что событие $e \in E_\Sigma$ *безопасно* по отношению к C и CT, если для всех $\nu, \nu' \in V_\Sigma$ таких, что $((e, \nu), \nu') \in \nu$, имеет место следующее:

если $\nu \in C$, то $\nu' \in C$, и если $(\nu, \nu') \in CT$.

Будем называть Σ безопасной системой по отношению к C и CT, если имеет место следующее:

- σ_0 безопасно по отношению к C
- каждое событие $e \in E_\Sigma$ безопасно по отношению к CT.

В частности, если система Σ безопасна, то любое состояние v, достижимое из начального состояния σ_0, также принадлежит множеству C.

Политика безопасности системы определяет состав множеств C и CT.

3. Правила политики безопасности

В этом параграфе представлена формальная политика безопасности, основанная на предикатах двух типов[1]: критерий и ограничение. Критерии используются для определения отношений, которые должны быть истинны для переменных в *любом* состоянии системы. Именно поэтому критерии используются для формулировки всех требований нормативного управления доступом. Например, простая политика безопасности рассматривается как критерий, поскольку *всякий раз*, когда субъект получает доступ чтения к объекту, уровень безопасности субъекта должен доминировать над уровнем безопасности объекта. Предикаты, относящиеся к ограничениям, показывают, каким образом могут изменяться переменные состояния при переходе от одного состояния к другому. Предикаты-ограничения используются для формализации требований дискреционного управления доступом и условий, при которых может осуществляться активация и деактивация субъектов и объектов. Например, одно из ограничений состоит в том, что всякий раз, когда субъект активируется (т.е. из неактивного становится активным), за этот процесс отвечает TCB.

Критерий безопасности C системы Trusted Mach определяется следующим образом:

$C = \{v = (b, \alpha, f, H, M) \in V \mid C_1(b, \alpha, f, H, M)$ и
$C_2(b, \alpha, f, H, M)$ и
$C_3(b, \alpha, f, H, M)$ и
.
.
.
$C_n(b, \alpha, f, H, M)\}$,

где каждое C_i - корректно сформулированное утверждение политики (т.е. требование, определяющее отношения между переменными состояния).

Соответственно, ограничение CT определяется следующим образом:

$CT = \{(v, v', e) \mid ((e, v), v') \in v$ и
$CT_1(b, \alpha, f, H, M, b', \alpha', f', H', M', e)$ и
$CT_2(b, \alpha, f, H, M, b', \alpha', f', H', M', e)$ и
$CT_3(b, \alpha, f, H, M, b', \alpha', f', H', M', e)$ и
.
.

[1] Каждое подмножество определяет предикат следующим образом. Пусть S является подмножеством множества V. Тогда для любого $v \in V$, $S(v)$ истинно тогда и только тогда, когда $v \in S$. В настоящей статье понятия предикатов и подмножеств взаимозаменяемы.

$C_m(b, \alpha, f, H, M, b', \alpha', f', H', M', e)\}$,

где $v = (b, \alpha, f, H, M)$, $v' = (b', \alpha', f', H', M')$, и каждое CT_i - корректно определенная формула, ограничивающая условия перехода системы из v в v' в результате события e.

Рассмотрим составные части критерия безопасности ограничения - C_i и CT_i.

Критерий 1. *Уровень безопасности атрибутов объектов*

Система Trusted Mach связывает с каждым объектом уровень безопасности, который определяет его положение в мандатной модели управления доступом. Кроме того, с именованными объектами система связывает еще один уровень, который характеризует атрибуты объекта не связанные с информацией, содержащейся в объекте. Согласно этому критерию, уровень безопасности каждого активного именованного объекта доминирует над уровнем безопасности атрибутов объекта, не связанных с содержанием.

$$\forall o \in O, \text{ если } named(o) \text{ и } \alpha(o), \text{ то } f_{attr}(o) \le f_o(o)$$

Критерий 2. *Свойство активного состояния*

Согласно этому требованию, всякий раз, когда субъект имеет доступ к объекту, и субъект, и объект должны быть активными.

Для любого субъекта s, объекта o и режима доступа x, если $(s, o, x) \in b$, то $\alpha_s(s)$ и $\alpha_o(o)$.

Критерий 3. *Простая безопасность*

Согласно этому требованию, всякий раз, когда субъект имеет доступ чтения к объекту, уровень безопасности субъекта должен доминировать над уровнем безопасности объекта.

Для любого субъекта s, объекта o, если $(s, o, read) \in b$, то $f_s(s) \ge f_o(o)$.

Критерий 4. **-свойство*

Согласно этому требованию, всякий раз, когда субъект имеет к объекту доступ добавления данных, записи или контроля, уровень безопасности объекта должен доминировать над уровнем безопасности субъекта. Кроме того, при доступе записи субъект и объект должны принадлежать одному и тому же уровню безопасности: для доступа контроля уровень безопасности субъекта должен совпадать с уровнем безопасности атрибутов объекта, не связанных с содержанием.

Пусть s - субъект и $(s, o, x) \in b$ для некоторого объекта o, где $x \in \{append, write, control\}$

Если $x = append$, то $f_o(o) \ge f_s(s)$

Если $x = write$, то $f_s(s) = f_o(o)$

Если $x = control$, то $f_s(s) = f_{attr}(o)$

Критерий 5. *Свойство совместимости*

Свойство совместимости утверждает, что уровень безопасности объекта должен быть не выше чем уровни безопасности всех его потомков в

иерархическом графе. Более того, неактивные каталоги не могут содержать объектов. Формально этот критерий формулируется так:

$\forall\ d,\ o \in O$, если $(d,\ o) \in Edges(H)$, то $\alpha_o(d)$ и $\alpha_o(o)$. Кроме того, если $named(o)$, то $f_o(d) = f_{attr}(o)$.

Заметим, что $f_o(d) \le f_o(o)$ следует из $f_{attr}(o) \le f_o(o)$ (критерий 5.1).

Из этого критерия следует, что все объекты в цикле должны принадлежать одному и тому же уровню безопасности.

Подытожим критерии 1-5:

Если $named(o)$, то $f_{attr}(o) \le f_o(o)$

Если $(s,\ o,\ x) \in b$, то $\alpha(o)$ и $\alpha(s)$

Если $(s,\ o,\ read) \in b$, то $f_o(o) \le f_s(s)$

Если $(s,\ o,\ write) \in b$, то $f_s(s) = f_o(o)$

Если $(s,\ o,\ append) \in b$, то $f_s(s) \le f_o(o)$

Если $(s,\ o,\ control) \in b$, то $f_s(s) = f_{attr}(o)$

Если $(d,\ o) \in Edges(H)$, то $\alpha_o(o)$ и $\alpha_o(d)$, и (если $named(o)$, то $f_o(d) = f_{attr}(o)$).

Ограничение 1. *Дискреционное управление доступом*

Согласно этому требованию всякий раз, когда субъекту разрешается доступ к объекту, субъект должен иметь необходимые дискреционные полномочия на доступ к этому объекту.

$\forall\ s \in S,\ o \in O,\ x \in A,$

если $(s,\ o,\ x) \in b' \backslash b$ и $named(o)$, то $((s,\ x) \in M(o)$ или $(s,\ control) \in M(o))$.

Строго говоря, дискреционные полномочия выдаются пользователям, но здесь мы предполагаем, что субъекты наследуют дискреционные полномочия пользователей.

Ограничение 2. *Активация и деактивация субъектов*

Субъекты могут быть активированы только средствами, входящими в состав TCB, и представлять легальных пользователей системы. Субъекты могут быть деактивированы другими субъектами, при условии, что их уровни безопасности совпадают. Согласно ограничению деактивации, если субъект деактивируется, то либо субъектом-инициатором должна быть TCB, либо этот субъект и субъект-инициатор должны принадлежать одному и тому же уровню безопасности.

$\forall\ s \in S$, если $\neg\alpha_s(s)$ и $\alpha'_s(s)$, то $invoker\_of(e) = TCB$

и

$\forall\ s \in S$, если $\alpha_s(s)$ и $\neg\alpha'_s(s)$, то $(invoker\_of(e) = TCB$ или $f_s(invoker\_of(e)) = f_s(s)$

Ограничение 3. *Принцип постоянства*

Система Trusted Mach обеспечивает неизменяемость уровня безопасности в соответствии с мандатной моделью управления доступом. Это означает, что после того, как объект или субъект созданы, их уровень безопасности должен оставаться неизменным в течение всего времени их существования:

$\forall\ s \in S,\ o \in O,$

$f_{attr}(o) = f_{attr}(o)$ и

$f'_o(o) = f_o(o)$ и
$f'_s(s) = f_s(s)$

Ограничение 4. *Активация и деактивация объектов*

Всякий раз, когда объекты активируются или деактивируются, их содержимое должно быть уничтожено. Моделирование этого аспекта составляет значительную сложность. Во-первых в предлагаемой модели, объекты никогда повторно не используются: повторно используются лишь связанные с ними ресурсы. Кроме того, для модели управления доступом трудно сформулировать свойства этих ресурсов. Поэтому в данной модели не рассматривается вопрос повторного использования объектов, а только показывается, что система безопасно инициирует любой новый объект, не оставляя никаких следов от его прежнего содержания.

Согласно ограничению активации, всякий раз, когда создается объект, инициатор активации объекта должен удовлетворять следующим условиям:

- Если объект именованный, то инициатором запроса должен быть субъект. Кроме того, субъект должен иметь необходимые полномочия для активации объекта.
- Если объект безымянный, то инициатором запроса должно быть TCB.

Согласно ограничению деактивации, всякий раз, когда объект деактивируется, инициатор деактивации объекта должен удовлетворять следующим условиям:

- Если объект именованный, то инициатором запроса должен быть субъект. Кроме того, субъект должен иметь доступ записи к каталогу, содержащему объект.[2]
- Если объект безымянный, то инициатором может быть либо TCB, либо субъект. Однако, если инициатором является субъект, то и субъект, и объект должны принадлежать одному и тому же уровню безопасности.

$\forall\ o \in O,$
если $\neg\alpha_o(o)$ и $\alpha'_o(o)$ и $named(o)$,
то
$\exists\ d$ такое, что
$(d, o) \in Edges(H')$ и
$(invoker\_of(e), d, write) \in b$ и
$(invoker\_of(e), create) \in M(o)$
и
$\forall\ o \in O,$
если $\neg\alpha_o(o)$ и $\alpha'_o(o)$ и $\neg named(o)$,
то
$invoker\_of(e) = TCB$
и
$\forall\ o \in O,$
если $\alpha_o(o)$ и $\neg\alpha'_o(o)$ и $named(o)$,
то

[2] Объект может содержаться в нескольких каталогах. В этом случае субъекту достаточно иметь доступ записи только к одному из этих каталогов.

$\exists\ d$ такое, что
$(d, o) \in Edges(H)$ и
$(invoker\_of(e), d, write) \in b$
и
$\forall\ o \in O$,
если $\alpha_a(o)$ и $\neg\alpha'_a(o)$ и $\neg named(o)$,
то
$f_s(invoker\_of(e)) = f_o(o)$ или $invoker\_of(e) = TCB$

Ограничение 5. *Ограничение на модификацию ACL*

Это ограничение описывает условия, при которых субъекты могут изменять ACL объектов. Субъект может изменять ACL объекта, если субъект имеет доступ контроля к этому объекту. Кроме того, любой субъект может разрешить или запретить другому субъекту создавать объект, при условии, что уровень безопасности не связанных с содержанием атрибутов объекта доминирует над уровнем безопасности субъекта.[3]

$\forall\ o \in O$
если $M'(o) \neq M(o)$ и $\alpha_a(o)$,
то
$(invoker\_of(e), o, control) \in b$
или
$(((M'(o) \setminus M(o)) \cup (M(o) \setminus M'(o)) \subseteq S \times \{create\}$ и
$f_s(invoker\_of(e)) \leq f_{cmr}(o)$).

4. Начальное состояние

Безопасное начальное состояние - это любое состояние, удовлетворяющее критериям 1 - 5. В частности, безопасным является следующее состояние:

- Каждый объект является неактивным.
- Каждый субъект является неактивным.
- Текущее множество доступов пусто.
- Множество ребер иерархического графа пусто.

Однако на практике активация многих объектов пространства имен происходит как часть процесса инициализации. В частности, так активируются все элементы данных, управляемые аттестованными серверами. Но изначально в системе нет никаких активных субъектов, и, следовательно, текущее множество доступов пусто.

5. Спецификации операций

Ограничения, накладываемые на операции обеспечивают реализацию представленной политики безопасности. В системе Trusted Mach реализовано шесть классов операций. К первому классу относятся операции, запрашивающие доступ к объекту. Ко второму классу относятся операции, осуществляющие прекращение доступа. К третьему классу относятся операции, активирующие и деактивирующие субъектов. К четвертому классу относятся операции,

[3] Эта реализация осуществляет более строгую политику, основанную на ACL типа объекта. Документацию реализуемой политики можно найти в [6].

активирующие и деактивирующие объекты. К пятому классу относятся операции, модифицирующие дискреционные полномочия доступа к именованным объектам. К последнему классу относятся операции, помещающие ссылки на объекты в каталоги.

Для описания этих операций приняты некоторые соглашения. Первый параметр каждого события - это активный элемент, который является инициатором события. В большинстве случаев выполняющим элементом является субъект. Однако при активации субъектов инициатором должно быть TCB. Оператор (′) используется для обозначения новых значений переменных состояния, измененных операцией. Например, b' обозначает новое значение текущего множества доступов. Правила описываются недетерминистски, поскольку TCB может быть не в состоянии обслужить запрос субъекта, даже если этот субъект удовлетворяет требованиям TCB, например, из-за исчерпания ресурсов и других ошибок. Поэтому предполагается, что любой запрос к TCB удовлетворяет всем необходимым требованиям.

И наконец, в спецификациях операций описано лишь то, как каждая из этих операций изменяет переменные состояния. Следовательно, каждая спецификация неявно подразумевает, что любую явно не указанную переменную модели эта операция оставляет неизменной. Например, спецификация операции *get_read_access* описывает лишь то, как эта операция изменяет значение текущего множества доступов (b). Тот факт, что остальные компоненты пространства состояний, α, f, H и M, в этой спецификации не упомянуты, означает, что эта операция оставляет их неизменными.

5.1 Операции получения доступа

Операция **get_read_access(*s*: subject, *o*: object)** представляет собой запрос доступа чтения к объекту *o* от субъекта *s*.

$\alpha_s(s)$ и
$\alpha_o(o)$ и
$f_s(s) \geq f_o(o)$ и
($\neg named(o)$ или $(s, read) \in M(o)$ или $(s, control) \in M(o)$) и
$b' = b \cup \{(s, o, read)\}$
или
пространство состояний остается неизменным
$b' = b$

Операция **get_append_access(*s*: subject, *o*: object)** представляет собой запрос доступа добавления данных к объекту *o* от субъекта *s*.

$\alpha_s(s)$ и
$\alpha_o(o)$ и
$f_o(o) \geq f_s(s)$ и
($\neg named(o)$ или $(s, append) \in M(o)$ или $(s, control) \in M(o)$) и
$b' = b \cup \{(s, o, append)\}$
или
пространство состояний остается неизменным
$b' = b$

Операция **get_write_access(*s*: subject, *o*: object)** представляет собой запрос доступа записи к объекту *o* от субъекта *s*.

$\alpha_s(s)$ и
$\alpha_o(o)$ и
$f_s(s) = f_o(o)$ и
($\neg named(o)$ или $(s, write) \in M(o)$ или $(s, control) \in M(o)$) и
$b' = b \cup \{(s, o, write)\}$
или
пространство состояний остается неизменным
$b' = b$

Операция **get_control_access(*s*: subject, *o*: object)** представляет собой запрос доступа контроля к объекту *o* от субъекта *s*.

$\alpha_s(s)$ и
$\alpha_o(o)$ и
$f_s(s) = f_{attr}(o)$ и
($\neg named(o)$ или $(s, control) \in M(o)$) и
$b' = b \cup \{(s, o, control)\}$
или
пространство состояний остается неизменным
$b' = b$

5.2 Операции прекращения доступа

Операция **release_access(*s*: subject, *o*: object, *x*: access)** представляет собой запрос прекращения доступа в режиме *x* к объекту *o* из текущего множества доступов субъекта *s*.

$b' = b \setminus \{(s, o, x)\}$
или
пространство состояний остается неизменным
$b' = b$

5.3 Активация и деактивация субъектов

В отличие от всех других операций, предполагается, что TCB инициатором двух следующих операций, поскольку TCB модифицирует α_s, вызывая либо *activate_subject*, либо *deactivate_subject*.

Операция **activate_subject(*s*: invoker, *t*: subject)** представляет собой запрос сделать субъект *t* активным. Инициатором *s* должно быть TCB.

$\neg\alpha_s(t)$ и
$\alpha'_s(t)$ и
$s = TCB$
или
пространство состояний остается неизменным
$\alpha'_s(t) = \alpha_s(t)$

Операция **deactivate_subject(*s*: invoker, *t*: subject)** представляет собой запрос сделать субъект *t* неактивным. Субъект *t* можно сделать неактивным при

условии, что он прекратил все режимы доступа к объектам системы. Инициатором деактивации субъектам s может быть либо TCB, либо субъек, находящийся на одном уровне безопасности с t.

$(f_s(s) = f_s(t)$ или $s = TCB)$ и
$\alpha_s(t)$ и
$\neg\alpha'_s(t)$ и
$\forall o, x\ (t, o, x) \notin b$
или
пространство состояний остается неизменным
$\alpha'_s(t) = \alpha_s(t)$

5.4 Активация и деактивация объектов

Операция **activate_object(***s*: **invoker,** *o*, *d*: **object, acl:** ACL_{model}**)** представляет собой запрос сделать активным объект o из объекта-каталога d. Объект должен быть неактивным, а уровень безопасности объекта o должен доминировать над уровнем безопасности s.

$\neg\alpha_o(o)$ и
$\alpha_o(d)$ и
$\neg named(o)$ и
$s = TCB$ и $d = IPC\_root$ и
$Edges(H') = Edges(H) \cup \{(d, o)\}$ и
$\alpha'_o(o)$
или
$\neg\alpha_o(o)$ и
$\alpha_o(d)$ и
$named(o)$ и
$(s, d, write) \in b$ и
$f_o(d) = f_{attr}(o)$ и
$f_{attr}(o) \leq f_o(o)$ и
$(s, create) \in M(o)$ и
$Edges(H') = Edges(H) \cup \{(d, o)\}$ и
$\alpha'_o(o)$ и
$M'(o) = acl$
или
пространство состояний остается неизменным
$\alpha'_o(o) = \alpha_s(o)$ и
$M'(o) = M(o)$

Операция **deactivate_named_object(***s*: **subject,** *o*, *d*: **object)** представляет собой запрос сделать именованный объект o, содержащийся в каталоге d, неактивным. Прежде чем объект может быть сделан неактивным, все доступы, разрешенные к этому объекту, должны быть прекращены.

Возможны три случая:

- объект не является активным;
- имеются другие ссылки на этот объект;
- деактивация не состоялась, и пространство состояний остается неизменным.

$\alpha_o(o)$ и
$(s, o, control) \in b$ и
$b' = b \setminus \{(t, o, x) \mid t \in S$ и $x \in A\}$
или
пространство состояний остается неизменным
$b' = b$

5.6 Операция вставки в каталог ссылки на объект

Объект в системе Trusted Mach может одновременно принадлежать нескольким каталогам. Операция *directory_insert* позволяет субъектам вставлять существующий объект в каталог, при условии, что уровень безопасности этого каталога совпадает с уровнем безопасности атрибутов объекта, не связанных с его содержанием. Более того, инициатор должен иметь право доступа к указанному каталогу.

Операция **directory_insert(*s*: subject, *o*, *d*: object)** представляет собой запрос вставить ссылку на объект *o* в каталог *d*. И *o*, и *d* должны уже быть активными. Кроме того, каталог *d* и каталог, содержащий *o*, должны принадлежать одному и тому же уровню безопасности.

$\alpha_o(o)$ и
$\alpha_o(d)$ и
$(s, d, write) \in b$ и
$f_d(d) = f_{attr}(o)$ и
$Edges(H') = Edges(H) \cup \{(d, o)\}$
или
$Edges(H') = Edges(H)$.

$\alpha_o(o)$ и *named*(o) и
$(d, o) \in Edges(H)$ и
$(s, d, write) \in b$ и
$\neg\alpha'_o(o)$ и
$b' = b\backslash\{(t, o, x) \mid t \in S$ и $x \in A\}$ и
$Edges(H') = Edges(H) \setminus \{(d', o) \mid d' \in O\}$
или
$\alpha'_o(o) = \alpha_o(o)$ и
named(o) и
$(s, d, write) \in b$ и
$\exists\, d'$ такое, что $d' \neq d$ и $(d', o) \in Edges(H)$ и
$b' = b$ и
$Edges(H') = Edges(H) \setminus \{(d, o)\}$
или
пространство состояний остается неизменным
$\alpha'_o(o) = \alpha_o(o)$ и
$b' = b$ и
$Edges(H') = Edges(H)$

Операция **deactivate_unnamed_object(*s*: subject, *o*, *d*: object)** представляет собой запрос сделать безымянный объект *o* неактивным. Прежде чем объект может быть сделан неактивным, все доступы, разрешенные к этому объекту, должны быть прекращены.

$\alpha_o(o)$ и $\neg$*named*(o) и
$(IPC\_root, o) \in Edges(H)$ и
$(f_s(s) = f_o(o)$ или $s = TCB)$ и
$\neg\alpha'_o(o)$ и
$b' = b\backslash\{(t, o, x) \mid t \in S$ и $x \in A\}$ и
$Edges(H') = Edges(H) \setminus \{(IPC\_root, o)\}$

5.5 Операции модификации ACL

Система Trusted Mach позволяет субъектам изменять ACL объектов. Кроме того, субъекты могут запросить, чтобы изменение произошло немедленно, с модификацией всех текущих доступов.

Операция **change_ACL(*s*: subject, *o*: object, *y*:ACL_{model})** представляет собой запрос изменить ACL доступа субъекта *s* к объекту *o* с *x* на *y*.

$(s, o, control) \in b$ и
$M'(o) = y$
или
пространство состояний остается неизменным
$M' = M$

Операция **revoke_access(*s*: subject, *o*: object)** представляет собой запрос прекратить все текущие режимы доступа, полученные субъектами.

ОГЛАВЛЕНИЕ