Грушо А.А. Тимонина Е.Е.
Теоретические основы защиты информации
1996 г. Издательство Агентства "Яхтсмен"

В настоящее время и у нас в стране, и за рубежом достаточно много публикаций по современным стандартам защиты, средствам и методам защиты. Однако работы, которая являлась бы творческим обобщением существующих формальных моделей, нормативных документов, зарубежных стандартов в области защиты информации на нашем книжном рынке пока еще не было. Авторам удалось "оживить" разрозненные, порой противоречивые фрагменты, придав им новое качество, в виде целостных теоретических основ защиты информации.

Показаны пути использования формальных математических методов для защиты информации. Изложены основы теории защиты информационных систем. Приведены примеры доказательства защищенности автоматизированных информационных систем.

Для специалистов, работающих в области искусственного интеллекта, программирования для ЭВМ, защиты информационных и банковских систем.

Книга доктора физико-математических наук, член-корреспондента Академии Криптографии Российской Федерации, действительного члена Международной Академии Информации, профессора Александра Александровича Грушо и член-корреспондента Международной Академии Информатизации, кандидата физико-математических наук Елены Евгеньевны Тимониной посвящена введению в формальную теорию защиты информации. Выполненная в академическом стиле, работа наряду со строгими определениями и доказательствами содержит большое количество примеров из практической жизни, доходчиво иллюстрирующих основные теоретические положения.

Особо хотелось бы выделить разделы, посвященные **гарантированно защищенным распределенным системам, доказательному подходу к построению систем защиты и определению политики безопасности.** Что или кто является гарантом защищенности информации в автоматизированных системах в условиях, когда в общем виде проблема обеспечения безопасности в АИС относится к алгоритмически неразрешимым проблемам? Авторы дают ответ на этот вопрос, сведя его к вопросу о том, поддерживает или нет система защиты сформулированную политику безопасности.

На сегодняшний день и у нас в стране, и за рубежом достаточно много публикаций по современным стандартам защиты, средствам и методам защиты. Однако работы, которая являлась бы по своей сути творческим обобщением существующих формальных моделей, нормативных документов, зарубежных стандартов в области защиты информации на нашем книжном рынке не было. А.А. Грушо и Е.Е. Тимониной удалось "оживить" разрозненные, порой противоречивые фрагменты, придав им новое качество, в виде целостных теоретических основ защиты информации.

Хотелось бы отметить ориентацию данной книги на ведение учебной работы, что позволяет ее рекомендовать в качестве учебника для студентов и специалистов, работающих в области обеспечения безопасности самого широкого плана - от разработки политики безопасности для всей организации, до разработки конкретных систем защиты информации в АИС.

Доктор технических наук С. П. Расторгуев

Введение

Предлагаемая работа написана в стиле Lecture Notes и представляет введение в формальную теорию защиты информации в электронных системах обработки данных. В отличие от других руководств и пособий по защите информации здесь не ставилась задача перечислить как можно полнее все угрозы, механизмы защиты и другие детали проблемы защиты информации. Основная цель работы - доступно изложить методы анализа систем защиты информации в электронных системах обработки данных. Как большинство современных методов изучения сложных систем, анализ систем защиты

предполагает иерархическую декомпозицию. Верхний уровень иерархии составляет политика безопасности со своими специфическими методами ее анализа. Следующий уровень - основные системы поддержки политики безопасности (мандатный контроль, аудит и т.д.). Затем следует уровень механизмов защиты (криптографические протоколы, криптографические алгоритмы, системы создания защищенной среды, системы обновления ресурсов и т.д.), которые позволяют реализовать системы поддержки политики безопасности. Наконец самый низкий уровень - реализация механизмов защиты (виртуальная память, теговая архитектура, защищенные режимы процессора и т.д.). В работе рассматриваются только верхние уровни этой иерархии, так как механизмы защиты достаточно полно описаны в литературе. Ограничения изложения рамками верхних уровней иерархии позволило с высокой степенью формализации ввести основные понятия и модели защиты информации, определения, что такое "хорошая" или "плохая" системы защиты, описать доказательный подход в построении систем защиты, позволяющий говорить о гарантированно защищенных системах обработки информации. Описанный подход лежит в основе многих стандартов для систем защиты и позволяет проводить анализ и аттестование защищенных систем.

Следует обратить внимание на то, что в большинстве руководств по защите верхние уровни иерархии не отражены, или не объяснено их значение. В результате создается иллюзия, что качественная защита определяется только надежностью и количеством механизмов защиты. Наводят на определенные размышления неточные переводы требований стандартов по защите, которые привозят к нам иностранные специалисты. В связи с этим, базисом для работы явился анализ исходных текстов американских стандартов по защите и статей, в которых излагались результаты анализа моделей, лежащих в основе этих стандартов. Поэтому работу можно рассматривать как изложение определенной "платформы" для разработки систем защиты информации, которая взята за основу в США, Канаде и Западной Европе.

Глава 1 ВСПОМОГАТЕЛЬНЫЕ СТРУКТУРЫ (МОДЕЛИ), ИСПОЛЬЗУЕМЫЕ В ЗАЩИТЕ ИНФОРМАЦИИ

Для того, чтобы провести исследование и получить практический результат часто применяют следующий прием. В объект исследования вносят некоторую структуру, которая имеет искусственный характер, но облегчает исследование. Например, для анализа взаимодействия тел физики внесли в описание окружающего мира структуры силовых взаимодействий, электрических взаимодействий. Математики для анализа реальных процессов вносят структуры вероятностного пространства и вероятностей событий. Иногда говорят, что строится модель объекта. Оба названия приемлемы, однако мы предпочитаем первое, так как в ходе анализа потребуется вносить в информацию несколько структур и анализировать их взаимодействие. В терминах моделей такие нагромождения представляются более трудными для объяснения. Далее мы рассмотрим следующие структуры, которые вносятся в неопределяемый объект под названием информация, теории защиты которого будет посвящена вся работа:

- структура языка, позволяющая говорить об информации как о дискретной системе объектов;
- иерархическая модель вычислительных систем и модель OSI/ISO, позволяющие аппаратную, программную, прикладную компоненты вычислительных систем и сетей связи представлять в виде объектов некоторых языков, а также представляют примеры неиерархической декомпозиции и анализа сложных систем;
- структура информационного потока, позволяющая описывать и анализировать угрозы информации;
- структура ценности информации, позволяющая понять, что надо и что не надо защищать;

- структуры реляционных баз данных, которые рассматриваются как примеры организации информации в вычислительных системах, и используются для пояснения наиболее трудных проблем защиты.

В заключение главы в качестве примера проблемы согласования различных структур между собой рассматривается задача совмещения одной из структур ценности информации и структуры реляционной базы данных.

1.1. ЯЗЫК, ОБЪЕКТЫ, СУБЪЕКТЫ.

Используем некоторые понятия математической логики. Пусть А конечный алфавит, *А* - множество слов конечной длины в алфавите А.

Из А при помощи некоторых правил выделено подмножество *Я* правильных слов, которое называется языком. Если $Я_1$ - язык описания одной информации, $Я_2$ - другой, то можно говорить о языке Я, объединяющем $Я_1$ и $Я_2$ описывающем ту и другую информацию. Тогда $Я_1$ и $Я_2$ подъязыки *Я.*

Будем считать, что любая информация представлена в виде слова в некотором языке *Я.* Кроме того, можно полагать, что состояние любого устройства в вычислительной системе достаточно полно описано словом в некотором языке. Тогда можно отождествлять слова и состояния устройств и механизмов вычислительной системы или произвольной электронной системы обработки данных (ЭСОД). Эти предположения позволяют весь анализ вести в терминах некоторого языка.

Определение Объектом относительно языка Я (или просто объектом, когда из контекста однозначно определен язык) называется произвольное конечное множество языка *Я.*

Пример 1. Произвольный файл в компьютере есть объект. В любой момент в файл может быть записано одно из конечного множества слов языка *Я,* в некотором алфавите А, которое отражает содержимое информации, хранящейся в файле. Значит, файл можно рассматривать как конечное множество слов, которые в нем могут быть записаны (возможные содержания файла). То, что в данный момент в файле содержится только одно слово, не исключает потенциально существования других записей в данном файле, а в понятие файла как объекта входят все допустимое множество таких записей. Естественно выделяется слово, записанное в файле в данный момент, - это состояние объекта в данный момент.

Пример 2. Пусть текст в файле разбит на параграфы так, что любой параграф также является словом языка *Я* и, следовательно, тоже является объектом. Таким образом, один объект может являться частью другого.

Пример 3. Принтер компьютера - объект. Существует некоторый (достаточно сложный) язык, описывающий принтер и его состояния в произвольный момент времени. Множество допустимых описаний состояний принтера является конечным подмножеством слов в этом языке. Именно это конечное множество и определяет принтер как объект.

В информации выделим особо описания преобразований данных.

Преобразование информации отображает слово, описывающее исходные данные, в другое слово. Описание преобразования данных также является словом. Примерами объектов, описывающих преобразования данных, являются программы для ЭВМ

Каждое преобразование информации может:

а) храниться;

б) действовать.

В случае а) речь идет о хранении описания преобразования в некотором объекте (файле). В этом случае преобразование ничем не отличается от других данных. В случае б) описание программы взаимодействует с другими ресурсами вычислительной системы - памятью, процессором, коммуникациями и др.

Определение. Ресурсы системы, выделяемые для действия преобразования, называются доменом.

Однако для осуществления преобразования одних данных в другие кроме домена необходимо передать этому преобразованию особый статус в системе, при котором ресурсы системы осуществляют преобразование. Этот статус будем называть "управление".

Определение. Преобразование, которому передано управление, называется процессом.

При этом подразумевается, что преобразование осуществляется в некоторой системе, в которой ясно, что значит передать управление.

Определение. Объект, описывающий преобразование, которому выделен домен и передано управление, называется субъектом.

То есть субъект - это пара (домен, процесс). Субъект для реализации преобразования использует информацию, содержащуюся в объекте O, то есть осуществляет доступ к объекту O.

Рассмотрим некоторые основные примеры доступов.

1. Доступ субъекта S к объекту O на чтение (r) данных в объекте O.

При этом доступе данные считываются в объекте O и используются в качестве параметра в субъекте S.

2. Доступ субъекта S к объекту O на запись (w) данных в объекте O.

При этом доступе некоторые данные процесса *S* записываются в объект O. Здесь возможно стирание предыдущей информации.

3. Доступ субъекта S к объекту O на активизацию процесса, записанного в O как данные (exe). При этом доступе формируется некоторый домен для преобразования, описанного в O, и передается управление соответствующей программе.

Существует множество других доступов, некоторые из них будут определены далее. Множество возможных доступов в системе будем обозначать **R**.

Будем обозначать множество объектов в системе обработки данных через **O**, а множество субъектов в этой системе **S**. Ясно, что каждый субъект является объектом относительно некоторого языка (который может в активной фазе сам менять свое состояние). Поэтому $\mathbf{S} \subseteq \mathbf{O}$. Иногда, чтобы не было различных обозначений, связанных с одним преобразованием, описание преобразования, хранящееся в памяти, тоже называют субъектом, но не активизированным. Тогда активизация такого субъекта означает пару (домен, процесс).

В различных ситуациях мы будем уточнять описание вычислительной системы. Однако мы всегда будем подразумевать нечто общее во всех системах. Это общее состоит в том, что состояние системы характеризуется некоторым множеством объектов, которое будем предполагать конечным.

В любой момент времени на множестве субъектов введем бинарное отношение **a** активизации. $\mathbf{S_1 a S_2}$, если субъект $\mathbf{S}_1$, обладая управлением и ресурсами, может передать $\mathbf{S}_2$ часть ресурсов и управление (активизация). Тогда в графах, которые определяются введенным бинарным отношением на множестве объектов, для которых определено понятие активизации, возможны вершины, в которые никогда не входит ни одной дуги. Таких субъектов будем называть пользователями. Субъекты, в которые никогда не входят дуги и из которых никогда не выходят дуги, исключаются.

В рассмотрении вопросов защиты информации мы примем аксиому, которая положена в основу американского стандарта по защите ("Оранжевая книга") и будет обсуждаться далее. Здесь мы сформулируем ее в следующей форме.

Аксиома. *Все вопросы безопасности информации описываются доступами субъектов к объектам.*

Если включить в рассмотрение гипотетически такие процессы как пожар, наводнение, физическое уничтожение и изъятие, то эта аксиома охватывает практически все известные способы нарушения безопасности в самых различных вариантах понимания безопасности. Тогда для дальнейшего рассмотрения вопросов безопасности и защиты информации достаточно рассматривать множество объектов и последовательности доступов.

Пусть время дискретно, $\mathbf{O}_t$ - множество объектов момент t, $\mathbf{S}_t$ - множество субъектов в момент *t*. На множестве объектов $\mathbf{O}_t$ как на вершинах определим ориентированный граф доступов $\mathbf{G}_t$ следующим образом: дуга $S \xrightarrow{\rho} O$ с меткой p⊆R принадлежит $\mathbf{G}_t$ тогда и только тогда, когда в момент t субъект S имеет множество доступов p к объекту O.

Согласно аксиоме, с точки зрения защиты информации, в процессе функционирования системы нас интересует только множество графов доступов $\{\mathbf{G}_t\}_{t=1}^{T}$. Обозначим через $\Psi=\{\mathbf{G}\}$ множество возможных графов доступов. Тогда Ψ можно рассматривать как фазовое пространство системы, а траектория в фазовом пространстве Ψ соответствует функционированию вычислительной системы. В этих терминах удобно представлять себе задачу защиты информации в следующем общем виде. В фазовом пространстве Ψ определены возможные траектории Ф, в Φ выделено некоторое подмножество N неблагоприятных траекторий или участков таких траекторий, которых мы хотели бы избежать. Задача защиты информации состоит в том, чтобы любая реальная траектория вычислительного процесса в фазовом пространстве Ψ не попала во множество N. Как правило, в любой конкретной вычислительной системе можно наделить реальным смыслом компоненты модели Ψ,Φ и N. Например, неблагоприятными могут быть траектории, проходящие через данное множество состояний $\Psi' \subseteq \Psi$.

Чем может управлять служба защиты информации, чтобы траектории вычислительного процесса не вышли в N? Практически такое управление возможно только ограничением на доступ в каждый момент времени. Разумеется, эти ограничения могут зависеть от всей предыстории процесса. Однако, в любом случае, службе защиты доступно только локальное воздействие. Основная сложность защиты информации состоит в том, что имея возможность использовать набор локальных ограничений на доступ в каждый момент времени, мы должны решить глобальную проблему недопущения выхода любой возможной траектории в неблагоприятное множество N. При этом траектории множества N не обязательно определяются ограничениями на доступы конкретных субъектов к конкретным объектам. Возможно, что если в различные моменты вычислительного процесса субъект S получил доступ к объектам O_1 и O_2, то запрещенный доступ к объекту O_3 реально произошел, так как из знания содержания объектов O_1 и O_2 можно вывести запрещенную информацию, содержащуюся в объекте O_3.

Пример 4. Пусть в системе имеются два пользователя U_1, и U_2, один процесс S чтения на экран файла и набор файлов $O_1 \ldots O_m$. В каждый момент работает один пользователь, потом система выключается и другой пользователь включает ее заново. Возможны следующие графы доступов

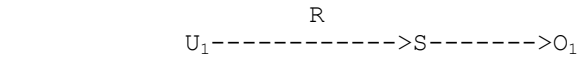
$$Uj \xrightarrow{R} S \xrightarrow{r} Oi \quad , \ i=1,\ldots,m,\ j=1,2. \quad (1)$$

Множество таких графов - ψ. Траектории - последовательности графов вида (1). Неблагоприятными считаются траектории, содержащие для некоторого i= 1...m состояния

$$U_1 \xrightarrow{R} S \xrightarrow{r} Oi$$
$$U_2 \xrightarrow{R} S \xrightarrow{r} Oi$$

То есть неблагоприятная ситуация, когда оба пользователя могут прочитать один объект. Ясно, что механизм защиты должен строить ограничения на очередной доступ, исходя из множества объектов, с которыми уже ознакомился другой пользователь. В этом случае, очевидно, можно доказать, что обеспечивается безопасность информации в указанном смысле.

Пример 5. В системе, описанной в примере 4, пусть неблагоприятной является любая траектория, содержащая граф вида



В этом случае очевидно доказывается, что система будет защищена ограничением доступа на чтение пользователя U_1 к объекту O_1

1.2. ИЕРАРХИЧЕСКИЕ МОДЕЛИ И МОДЕЛЬ ВЗАИМОДЕЙСТВИЯ ОТКРЫТЫХСИСТЕМ (OSI/ISO).

Ясно, что реализация автоматизированных информационных систем требует большого программно-аппаратного комплекса, который надо спроектировать, создать, поддерживать в работоспособном состоянии. Сложность этих систем такова, что требуется разработка специальной технологии проектирования и создания таких систем. В настоящее время основным инструментом решения задач анализа, проектирования, создания и поддержки в рабочем состоянии сложных систем является иерархический метод.

В основе метода лежит разбиение системы на ряд уровней, которые связаны однонаправленной функциональной зависимостью. В литературе предлагалось несколько вариантов формального и полуформального описания такой зависимости. Например, Парнас (D.L.Parnas, 1974) описывал такую зависимость следующим образом. Уровень А зависит в правильности своего функционирования от уровня В, или уровень А обращается к уровню В, или уровень А использует уровень В, или А требует присутствия правильной версии В. Однако это неформальные описания, а формализация здесь не удается из-за широкой общности понятия "сложная система" и неоднозначности разбиения на уровни. В целях понимания одного и того же в декомпозициях различной природы сложных систем можно договориться об универсальных принципах описания иерархического метода. Предположим, что интересующая нас сложная система А адекватно описана на языке ***Я***. Предположим, что мы провели декомпозицию (разложение) языка ***Я*** на семейство языков $\mathbf{D_1,D_2,...,D_n}$. Если язык **D**i, **i=2,.., n**, синтаксически зависит только от словоформ языка $\mathbf{D}_{i-1}$, то будем говорить, что они образуют два соседних уровня. Тогда система А может быть описана наборами слов $\mathbf{B_1,...,B_n}$ в языках $\mathbf{D_1,D_2,...,D_n}$ причем так, что описание $\mathbf{B}_i$ синтаксически может зависеть только от набора $\mathbf{B}_{i-1}$. В этом случае будем говорить об иерархической декомпозиции системы А и уровнях декомпозиции $\mathbf{B_1,...,B_n}$, где уровень $\mathbf{B}_i$ непосредственно зависит от $\mathbf{B}_{i-1}$. Рассмотрим ряд простейших примеров иерархического построения сложных систем.

Пример 1. Пусть вся информация в системе разбита на два класса Secret и Top Secret, которые в цифровой форме будем обозначать 0 и 1. Пусть все пользователи разбиты в своих возможностях допуска к информации на два класса, которые также будем обозначать 0 и 1. Правило допуска к информации X при запросе пользователя Y определяется условием, если класс $\underline{x}$ запрашиваемой информации X, а класс $\underline{y}$ пользователя Y, то допуск к информации разрешен тогда и только тогда, когда $\underline{x}=\underline{y}$. Это условие можно описать формально формулой некоторого языка $\mathbf{D}_2$

if $\underline{x}=\underline{y}$ then "Допуск Y к X"

Для вычисления этого выражения необходимо осуществить следующие операции, которые описываются в терминах языка $\mathbf{D}_1$:

$\underline{x}:=U_1(X)$,

$\underline{y}:=U_2(Y)$,

$z=\underline{x}\oplus \underline{y}$

$$U(X,Y,z),$$

где $U_1(X)$ - оператор определения по имени объекта X номера класса доступа x; $U_2(Y)$ - оператор определения по имени пользователя Y номера класса допуска y; $\oplus$ - сложение по mod 2; U(X, Y, z) - оператор, реализующий доступ Y к X, если z=0 , и блокирующий систему, если z=1.

По построению уровень $\mathbf{B}_2$ зависит от $\mathbf{B}_1$, а вся система представлена иерархической двухуровневой декомпозицией с языками $\mathbf{D}_1$ и $\mathbf{D}_2$. Причем $\boldsymbol{Я}=(\mathbf{D}_1, \mathbf{D}_2)$.

Пример 2. Гораздо чаще используется неформальное иерархическое описание систем. Например, часто используется иерархическая декомпозиция вычислительной системы в виде трех уровней.

Аппаратная часть - Операционная система -Пользовательские программы.

МОДЕЛЬ OSI/ISO.

1. Одним из распространенных примеров иерархической структуры языков для описания сложных систем является разработанная организацией международных стандартов (ISO) эталонная модель взаимодействия открытых систем (OSI), которая принята ISO в 1983 г.

ISO создана, чтобы решить две задачи:

- своевременно и правильно передать данные через сеть связи (т.е. пользователями должны быть оговорены виды сигналов, правила приема и перезапуска, маршруты и т.д.);
- доставить данные пользователю в приемлемой для него распознаваемой форме.

2. Модель состоит из семи уровней. Выбор числа уровней и их функций определяется инженерными соображениями. Сначала опишем модель.

Пользователь 1			Пользователь 2
прикладной уровень	7	7	прикладной уровень
представительский уровень	6	6	представительский уровень
сеансовый уровень	5	5	сеансовый уровень
транспортный уровень	4	4	транспортный уровень
сетевой уровень	3	3	сетевой уровень
канальный уровень	2	2	канальный уровень
физический уровень	1	1	физический уровень

Физическая среда

Верхние уровни решают задачу представления данных пользователю в такой форме, которую он может распознать и использовать. Нижние уровни служат для организации передачи данных. Иерархия состоит в следующем. Всю информацию в процессе передачи сообщений от одного пользователя к другому можно разбить на уровни; каждый уровень является выражением некоторого языка, который описывает информацию своего уровня. В терминах языка данного уровня выражается преобразование информации и "услуги", которые на этом уровне предоставляются следующему уровню. При этом сам язык опирается на основные элементы, которые являются "услугами" языка более низкого уровня. В модели OSI язык каждого уровня вместе с порядком его использования называется протоколом этого уровня.

Каково предназначение каждого уровня, что из себя представляет язык каждого уровня?

Прикладной уровень (ПУ). ПУ имеет отношение к семантике обмениваемой информации (т.е. смыслу). Язык ПУ обеспечивает взаимопонимание двух прикладных процессов в разных точках, способствующих осуществлению желанной обработки информации. Язык ПУ описывает два вида ситуаций:

- общие элементы для всех прикладных процессов, взаимодействующих на прикладном уровне;
- специфические, нестандартизируемые элементы приложений.

Язык ПУ состоит и постоянно расширяется за счет функциональных подъязыков. Например, разработаны протоколы (языки ПУ):

- виртуальный терминал (предоставление доступа к терминалу процесса пользователя в удаленной системе);
- файл (дистанционный доступ, управление и передачу информации, накопленной в форме файла);
- протоколы передачи заданий и манипуляции (распределенная обработка информации);
- менеджерские протоколы;

•одним из важных протоколов прикладного уровня является "электронная почта" (протокол X.400), т.е. транспортировка сообщений между независимыми системами с различными технологиями передачи и доставки сообщений.

Уровень представления (УП). УП - решает те проблемы взаимодействия прикладных процессов, которые связаны с разнообразием представлений этих процессов. УП предоставляет услуги для двух пользователей, желающих связаться на прикладном уровне, обеспечивая обмен информацией относительно синтаксиса данных, передаваемых между ними. Это можно сделать либо в форме имен, если обеим связывающимся системам известен синтаксис, который будет использоваться, либо в форме описания синтаксиса, который будет использоваться, если он не известен. Когда синтаксис передаваемой информации отличается от синтаксиса, используемого принимающей системой, то УП должен обеспечить соответствующее преобразование.

Кроме того, УП обеспечивает открытие и закрытие связи, управление состояниями УП и контролем ошибок.

Уровень сеанса (УС). УС обеспечивает управление диалогом между обслуживаемыми процессами на уровне представления. Сеансовое соединение сначала должно быть установлено, а параметры соединения оговорены путем обмена управляющей информацией. УС предоставляет услугу синхронизации для преодоления любых обнаруженных ошибок. Это делается следующим образом: метки синхронизации вставляются в поток данных пользователями услуги сеанса. Если обнаружена ошибка, сеансовое соединение должно быть возвращено в определенное состояние, пользователи услуги должны вернуться в установленную точку диалогового потока информации, сбросить часть переданных данных и затем восстановить передачу, начиная с этой точки.

Транспортный уровень (ТУ). ТУ представляет сеансовому уровню услугу в виде надежного и прозрачного механизма передачи данных (вне зависимости от вида реальной сети) между вершинами сети.

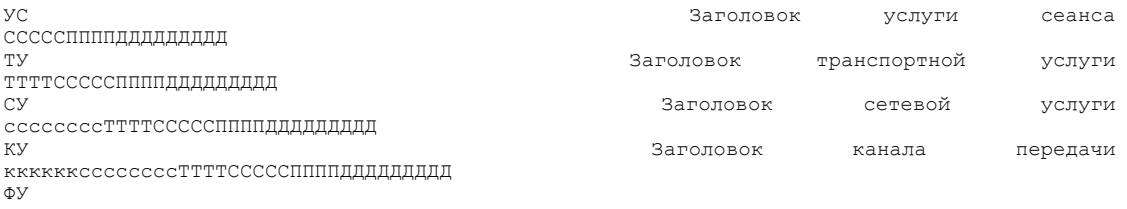
Сетевой уровень (СУ). СУ представляет ТУ услуги связи. СУ определяет маршрут в сети. Организует сетевой обмен (протокол IP). Управляет потоками в сети.

Канальный уровень. Представляет СУ услуги канала. Эта услуга состоит в безошибочности последовательной передачи блоков данных по каналу в сети. На этом уровне реализуется синхронизация, порядок блоков, обнаружение и исправление ошибок, линейное шифрование.

Физический уровень. Физический уровень обеспечивает то, что символы, поступающие в физическую среду передачи на одном конце, достигали другого конца.

Каким образом реализуются функции обмена информацией на каждом уровне? Для этого на каждом уровне к исходному блоку данных добавляется информация в виде выражения языка соответствующего уровня (как правило в виде дополнительного заголовка).

Уровни функционально взаимодействуют, во-первых, как одинаковые уровни у различных абонентов, во-вторых, как смежные уровни в одной иерархии.



Связь любых одинаковых уровней у различных абонентов при передаче, использующей соединение, состоит из трех фаз:

а) - установление соединения;

б) - передача данных;

в) - разъединение.

В фазе а) происходит переговор о наборе параметров передачи. В фазе б) - выявление и исправление ошибок, поддержание соединения.

На каждом уровне свои способы выявления и исправления ошибок:

канал - исправление битов;

сеть - разъединение и потеря пакетов; соответственно исправление этих ошибок;

сеанс, транспортирование - исправление адресации.

Важные информационные элементы каждого уровня будем называть объектами, таким образом, связь (N-1)-го, N-го и (N+l)-го уровней выглядит следующим образом; каждый уровень содержит набор объектов, которые взаимодействуют между собой в разных системах на одном уровне. На разных уровнях объекты взаимодействуют через ключи доступа услуг.

В модели OSI стандартизированы виды взаимодействия поставщика услуги на уровне N и потребителя на уровне (N+l). Эти виды называются примитивами услуг. Их четыре:

- запрос;
- признак;
- ответ;
- подтверждение.

Запрос подается пользователем услуги на (N+l)-м уровне системы А для того, чтобы обратиться к услуге протокола-поставщика услуги на N-уровне. Это приводит к посылке сообщения N-го уровня в систему В. В системе В запрос на уровне N вызывает появление примитива "признак", выпускаемого поставщиком услуги на уровне N в В на (N+l)-ый уровень.

Примитив "ответ" выпускается пользователем услуги на уровне N+l в В в ответ на признак, явившийся в точке доступа к услуге между уровнями N и N+l системы В. Примитив "ответ" является директивой протоколу уровня N завершить процедуру обращения примитива "признак". Протокол на уровне N в системе В генерирует сообщение, которое передается в сети и повторяется на уровне N системы А. Это вызывает посылку примитива "подтверждение", который выпускается поставщиком услуги в системе А на уровне N в точку доступа к услуге между уровнями N и N+l. На этом процедура, начатая запросом в точке доступа к услуге между уровнями N и N+l в системе А завершается.

1.3. ИНФОРМАЦИОННЫЙ ПОТОК.

Структуры информационных потоков являются основой анализа каналов утечки и обеспечения секретности информации. Эти структуры опираются на теорию информации и математическую теорию связи. Рассмотрим простейшие потоки.

1. Пусть субъект S осуществляет доступ на чтение *(r)* к объекту O. В этом случае говорят об информационном потоке от O к S. Здесь объект O является источником, а S - получателем информации.

2. Пусть субъект S осуществляет доступ на запись (w) к объекту O. В этом случае говорят об информационном потоке от S к O. Здесь объект O является получателем, а S - источником информации.

Из простейших потоков можно построить сложные. Например, информационный поток от субъекта S2 к субъекту S1 по следующей схеме:

r w

S_1 ----------→ O ←---------- S_2 (1)

Субъект S_2 записывает данные в объект О, а затем S_1 считывает их. Здесь S_2 - источник, а S_1 - получатель информации. Можно говорить о передаче информации, позволяющей реализовать поток. Каналы типа (1), которые используют общие ресурсы памяти, называются каналами по памяти.

С точки зрения защиты информации, каналы и информационные потоки бывают законными или незаконными. Незаконные информационные потоки создают утечку информации и, тем самым, могут нарушать секретность данных.

Рассматривая каналы передачи информационных потоков, можно привлечь теорию информации для вычисления количества информации в потоке и пропускной способности канала. Если незаконный канал нельзя полностью перекрыть, то доля количества информации в объекте, утекающая по этому каналу, служит мерой опасности этого канала. В оценках качества защиты информации американцы используют пороговое значение для допустимой пропускной способности незаконных каналов.

С помощью теоретико-информационных понятий информационные потоки определяются следующим образом.

Будем считать, что всю информацию о вычислительной системе можно описать конечным множеством объектов (каждый объект - это конечное множество слов в некотором языке ***Я***). В каждом объекте выделено состояние, а совокупность состояний объектов назовем состоянием системы. Функция системы - это последовательное преобразование информации в системе под действием команд. В результате, из состояния s мы под действием команды α перейдем в состояние s', обозначается: s|--s'(α). Если α последовательность команд, то композиция преобразований информации обозначается также, т.е. s|---s'(α) означает переход из состояния s в s' под действием последовательности команд α (автоматная модель вычислительной системы).

В общем виде для объектов X в s и Y в s' определим информационный поток, позволяющий по наблюдению Y узнать содержание X.

Предположим, что состояние X и состояние Y - случайные величины с совместным распределением P(x, y)=P(X=x, Y=y), где под {X=x} понимается событие, что состояние объекта X равно значению x (аналогично в других случаях). Тогда можно определить: P(x), P(y/x), P(x/y), энтропию H(X), условную энтропию H(X/Y) и среднюю взаимную информацию

$$I(X, Y) = H(X) - H(X/Y).$$

Определение. Выполнение команды α в состоянии s, переводящей состояние s в s', вызывает информационный поток от X к Y (обозначение X-->αY),если I(X, Y)>0. Величина I(X, Y) называется величиной потока информации от X к Y.

Определение. Для объектов X и Y существует информационный поток величины C (бит), если существуют состояния s и s' и последовательность команд α такие, что s|-- s'(α), X-->αY.

Оценка максимального информационного потока определяется пропускной способностью канала связиX--->α Y и равна по величине

$$C(\alpha, X, Y)=\max_{P(x)} I(X, Y).$$

Рассмотрим дальнейшие примеры информационных потоков в вычислительных системах.

1. Рассмотрим операцию присвоения значения переменных

Y:=X.

Пусть X - целочисленная случайная величина со значениями [0,15] и P(x) - равновероятная мера на значениях X. Тогда H(X)=4 бит. После выполнения операции присвоения по полученной в состоянии s‘ величине Y однозначно восстанавливается X, следовательно H(X/Y)=0 ⇒I(X, Y)=4⇒C(α, X, Y)=4, т.к. рассмотренный канал - симметричный.

2. Y:=X
Z:=Y.

Выполнение этих команд вызывает непрямой (косвенный) поток информации X-->Z, такой же величины как прямой поток X-->Y.

3. Z:=X + Y.

Предполагаем, что X, Y ∈[0,15] и равновероятны. Тогда H(X)=4, H(Y)=4.

$$0 < H(X/Z) = \sum_{(xz)} P(x, z)\ \log P(x/z) < 4,$$

следовательно, $0 < I(X, Z) < 4$ бит.

4. Пусть $X_1, X_2, \ldots, X_n$ - независимые одинаково распределенные равновероятные случайные величины со значениями 0 и 1.

$$Z = \sum_{i=1}^{n} Xi\ , \qquad H(X_1) \equiv 1,$$

$$H(X1/Z) = -\sum_{k=o}^{n-1} P(X1=0,\ Z=k)\ \log P(X=0/Z=k) - \sum_{k=1} P(X1=1,\ Z=k)\ \log P(X=l\ /Z=k)$$

Если $n \to \infty$, то $H(X_1/Z) = H(X_1)(1 + O(1))$, откуда следует, что $I(X_1/Z)=O(1)$.

Отсюда возникает возможность прятать конфиденциальные данные в статистические данные.

5. $Z:=X \oplus Y$, X и Y - равновероятные булевы случайные величины, $\oplus$ - сложение по mod 2, тогда Z не несет информации о X или Y.

6. If X=l then Y=l. $X \in \{0,1\}$, где величина X принимает свои значения с вероятностями P(X=0)=P(X==1)=1/2, начальное значение Y=0, H(X)=1.

$$H(X/Y) = \sum_{(x,y)} P(x, y)\ \log P(x/y) = 0.$$

Следовательно, I(X, Y) = 1. Поток называется неявным, в отличие от явного при операции присвоения.

7. If(X=1) и (Y=l) then Z:=l.

$$H(X)=H(Y)=l,\quad Z=l \Rightarrow X=l=Y$$

$$Z = 0 \Rightarrow \begin{cases} X=0 \text{ с } P=2/3 \\ X=1 \text{ с } P=1/3 \end{cases} \text{ апостериорные вероятности}$$

Отсюда $H_z(X) \cdot \approx O,7$. Поэтому количество информации о X в Z равно

$$I(Z, X) \approx 0,3.$$

Если $X_1, X_2, \ldots, X_n$ - исходные (ценные) переменные системы (программы), а $Y=(Y_1, \ldots, Y_m)$ - выходные, то $I(X_i,Y)$ - количество информации о X_i, в потоке, который индуцируется системой. Тогда отношение $I(X_i,Y)/H(X_1)$ - показатель "утечки" информации о X_1. Если установить порог $\lambda > O$ для "утечки", то из условия при каждом i=l.....n,

$$I(X_i,Y)/H(X_i) < \lambda,$$

следуют требования к защите Y.

1.4. ЦЕННОСТЬ ИНФОРМАЦИИ.

Чтобы защитить информацию, надо затратить силы и средства, а для этого надо знать какие потери мы могли бы понести. Ясно, что в денежном выражении затраты на защиту не должны превышать возможные потери. Для решения этих задач в информацию вводятся вспомогательные структуры - ценность информации. Рассмотрим примеры.

1 . Аддитивная модель. Пусть информация представлена в виде конечного множества элементов и необходимо оценить суммарную стоимость в денежных единицах из оценок компонент. Оценка строится на основе экспертных

оценок компонент, и, если денежные оценки объективны, то сумма дает искомую величину. Однако, количественная оценка компонент невсегда объективна даже при квалифицированной экспертизе. Это связано с неоднородностью компонентв целом. Поэтому делают единую иерархическую относительную шкалу (линейный порядок, который позволяет сравнивать отдельные компоненты по ценности относительно друг друга). Единая шкала означает равенство цены всех компонент, имеющих одну и туже порядковую оценку.

Пример 1 $0_1,...,0_n$ - объекты, шкала 1<...<5. Эксперты оценили (2, 1, 3,...., 4) - вектор относительных
ценностей объектов. Если есть цена хотя бы одного объекта, например, C_1=100 руб., то вычисляется оценка одного балла C_1/λ. = 50 руб.,

где λ - число баллов оценки первого объекта, и вычисляется цена каждого следующего объекта: C_2=50руб., C_3=150 руб. и т.д. Сумма дает стоимость всей информации. Если априорно известна цена информации, то относительные оценки в порядковой шкале позволяют вычислить цены компонент.

2. Анализ риска. Пусть в рамках аддитивной модели проведен учет стоимости информации в системе. Оценка возможных потерь строится на основе полученных стоимостей компонент, исходя из прогноза возможных угроз этим компонентам. Возможности угроз оцениваются вероятностями соответствующих событий, а потери подсчитываются как сумма математических ожиданий потерь для компонент по распределению возможных угроз.

Пример 2. Пусть $O_1,...,O_n$ - объекты, ценности которых $C_1,...,C_n$. Предположим, что ущерб одному объекту не снижает цены других, и пусть вероятность нанесения ущерба объекту O_i равна p_i, функция потерь ущерба для объекта O_i равна

$$W_i = \begin{cases} C_i, & \text{если объету } i \text{ нанесен ущерб,} \\ 0, & \text{в противном случае.} \end{cases}$$

Оценка потерь от реализации угроз объекту i равна $EW_i = p_iC_i$.

Исходя из сделанных предположений, потери в системе равны $W=W_1+...+W_n$. Тогда ожидаемые потери(средний риск) равны:

$$EW=\sum_{i=1}^{n} p_iC_i$$

Существуют ППП, позволяющие автоматизировать оценку риска, например, RASYS.

3. Порядковая шкала ценностей. Далеко не всегда возможно и нужно давать денежную оценку информации. Например, оценка личной информации, политической информации или военной информации не всегда разумна в денежном исчислении. Однако подход, связанный со сравнением ценности отдельных информационных элементов между собой, по-прежнему имеет смысл.

Пример 3. При оценке информации в государственных структурах используется порядковая шкала ценностей. Все объекты (документы) государственного учреждения разбиваются по грифам секретности. Сами грифы секретности образуют порядковую шкалу: несекретно < для служебного пользования <секретно < совершенно секретно (НС<ДСП<С<СС) или у американцев : unclassified<confidential<secret<top secret (U<Conf<S<TS). Более высокий класс имеет более высокую ценность и поэтому требования по его защите от несанкционированного доступа более высокие.

4. Модель решетки ценностей. Обобщением порядковой шкалы является модель решетки. Пусть дано SC - конечное частично упорядоченное множество относительно бинарного отношения <, т.е. для каждых A, B, C выполняется

1) рефлексивность: A<A,
2) транзитивность: A<B, B<C==>A<C,
3) антисимметричность: A<B, B<A => A=B.

Определение. Для A, B∈SC элемент C=A⊕B∈SCназывается наименьшей верхней границей (верхней гранью), если
1) A<C, B<C;
2) A<D, B<D⇒C<D для всех D∈SC.

Элемент A⊕B, вообще говоря, может не существовать. Если наименьшая верхняя граница существует, то из антисимметричности следует единственность.

Упражнение. Доказать это.

Определение. Для A, B∈C элемент E=A⊗B∈SCназывается наибольшей нижней границей (нижней гранью), если
1) E<A, E<B;
2) D<A, D<B⇒D<E.

Эта граница также может не существовать. Если она существует, то из антисимметричности следует единственность.

Упражнение. Доказать этот факт.

Определение. (SC, <) называется решеткой, если для любых A, B∈SC существует A⊕B∈SC и A⊗B∈SC.

Лемма. *Для любого набора $S=\{A_1,...,A_n\}$ элементов из решетки SC существуют единственные элементы,:*
$\oplus S=A_1\oplus...\oplus A_n$ - наименьшая верхняя граница S;
$\otimes S=A_1\otimes...\otimes A_n$ - наибольшая нижняя граница S.

Доказательство. Докажем ассоциативность операции ⊕.

$$C_1=(A_1\oplus A_2)\ \oplus A3=A1\oplus(A_2\oplus A_3)=C_2.$$

По определению $C_1>A_3$, $C_1>A_1\oplus A_2$. Отсюда следует $C_1>A_3$, $C_1>A_2$, $C_1>A_1$. Тогда $C_1>A_2\oplus A_3$, $C_1>A_1$, следовательно, $C_1>C_2$. Аналогично $C_2>C_1$. Из антисимметричности $C_1=C_2$.

Отсюда следует существование и единственность ⊕S. Такими же рассуждениями доказываем, что существует ⊗S и она единственна. Лемма доказана.

Для всех элементов SC в конечных решетках существует верхний элемент High = ⊕SC, аналогично существует нижний элемент Low = ⊗SC.

Определение. Конечная линейная решетка - это линейно упорядоченное множество, можно всегда считать {0, 1 ,..., n}=SC .

Для большинства встречающихся в теории защиты информации решеток существует представление решетки в виде графа. Рассмотрим корневое дерево на вершинах из конечного множества $X=\{X_1, X_2...X_n\}$с корнем в X_i. Пусть на единственном пути, соединяющем вершину X_1 с корнем, есть вершина X_j. Положим по определению, что $X_i<X_j$. Очевидно, что таким образом на дереве определен частичный порядок. Кроме того, для любой пары вершин X_i и X_j существует элемент $X_i\oplus X_j$, который определяется точкой слияния путей из X_i и X_j в корень. Однако такая структура не является решеткой, т.к. здесь нет нижней грани. Оказывается, что от условия единственности пути в корень можно отказаться, сохраняя при этом свойства частичного порядка и существование верхней грани. Например, добавим к построенному дереву вершину L, соединив с ней все концевые вершины. Положим i=1,...,n, $L<X_j$. Для остальных вершин порядок определяется как раньше. Построенная структура является решеткой.

Упражнение Доказать этот факт.

Приведенный пример не исчерпывает множество решеток, представимых в виде графов, однако поясняет как связаны графы и решетки.

Упражнение. Покажите, что следующие графы определяют решетки.

Не всякий граф определяет решетку. Например,

Упражнение. Доказать, что это так.

РЕШЕТКА ПОДМНОЖЕСТВ X.

Для ∀A, B∈X. Определим A<B⇒A⊆B. Все условия частичного порядка 1), 2), 3) выполняются. Кроме того, A⊕B - это A∪B, A⊗B=A∩B. Следовательно, это решетка.

Пример 4. X={1, 2, 3}.

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 2 & 3 \end{pmatrix} \quad \begin{matrix} & \\ & \end{matrix} \quad \begin{matrix} 1 & 3 \end{matrix}$$

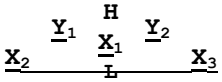
1 2 3

∅

Пусть программа имеет $X=\{X_1,...,X_m\}$ - входные, $Y_1...Y_n$ - выходные элементы. Каждый выходной элемент зависит от некоторых входных элементов. Отношение вход-выход описывается решеткой рассматриваемого типа. Решетка подмножеств строится по подмножествам X следующим образом. Для каждой X_i $\underline{X_i}=\{X_i\}$. Для каждой Y_j $\underline{Y_j}=\{X_i | X_j \rightarrow Y_j\}$.

Пример 5. X_1, X_2, X_3, Y_1, Y_2. Y_1 зависит только от X_1, X_2; Y_2 зависит от X_1 и X_3.

$Y_1=\{X_1,X_2\}$ $\underline{Y_2}=\{X_1,X_3\}$



MLS РЕШЕТКА

Название происходит от аббревиатуры Multilevel Security и лежит в основе государственных стандартов оценки информации. Решетка строится какпрямое произведение линейной решетки L и решетки SC подмножеств множества X, т.е. (α,β), (α',β') -элементы произведения, $\beta,\beta' \in L$ - линейная решетка, $\alpha,\alpha' \in SC$ - решетка подмножеств некоторого множества X. Тогда

$$(\alpha,\beta) < (\alpha',\beta') \Leftrightarrow \alpha \subseteq \alpha', \beta < \beta'$$

Верхняя и нижняя границы определяются следующим образом:

$$(\alpha,\beta) \oplus (\alpha',\beta') \Leftrightarrow (\alpha \cup \alpha', \max\{\beta,\beta'\}),$$

$$(\alpha,\beta) \otimes (\alpha',\beta') \Leftrightarrow (\alpha \cap \alpha', \min\{\beta,\beta'\}).$$

Вся информация {объекты системы} отображается в точки решетки $\{(a,\beta)\}$. Линейный порядок, как правило, указывает гриф секретности. Точки множества X обычно называются категориями.

Свойства решетки в оценке информации существенно используются при классификации новых объектов, полученных в результате вычислений. Пусть дана решетка ценностей SC, множество текущих объектов **O**, отображение $C: 0 \rightarrow S$, программа использует информацию объектов $0_1,..,0_n$, которые классифицированы точками решетки $C(0_1),...,C(0_n)$. В результате работы программы появился объект O, который необходимо классифицировать. Это можно сделать, положив $C(0) = C(0_1) \oplus ... \oplus C(0_n)$. Такой подход к классификации наиболее распространен в государственных структурах. Например, если в сборник включаются две статьи с грифом секретно и совершенно секретно соответственно, и по тематикам: первая - кадры, вторая - криптография, то сборник приобретает гриф совершенно секретно, а его тематика определяется совокупностью тематик статей (кадры,криптография).

1.5. РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Предположим, что мы хотим внести решетку ценностей (например, MLS) в конкретную информацию,которая хранится и обрабатывается на ЭВМ. Рассмотрим примеры механизмов такого внесения и проблемы, которые здесь возникают. Для определенности рассмотрим информацию, структурированную и обрабатываемую при помощи реляционной базы данных.Такая модель информации называется реляционноймоделью данных (РМ). Материалы следующих параграфов 1.5 и 1.6 базируются на работах D.Denning, T.Lunt и их коллег.

РМ состоит из отношений, которые представляют собой таблицы со многими входами, и алгебры отношений, которая позволяет строить новые отношения в терминах других отношений (в РМ входят также правила целостности хранимой информации и производной информации).

Каждое отношение R определяется схемой R (A_1,...,A_n), которая характеризуется множеством атрибутовA_1,..., A_n, т.е. переменных, описывающих входы таблицы.Отношение состоит из множества записей (векторов или строк), которые представляют собой значения данных в области определения атрибутов (то есть элементы таблицы - значения атрибутов).

Реляционная алгебра состоит из операторов для выбора всех или части записей, имеющих определенные значения из отношения (таблицы), и для добавления данных в различные отношения.

Для иллюстрации рассмотрим базу данных "Flight"("Полеты''). Эта база определена следующими схемами. Отношение ITEM с атрибутами: номера мест, имена, веса. Отношение Flight с атрибутами: номер рейса, дата вылета, назначение, общий вес груза. ОтношениеPAYLOAD, дающее количество использованных местна борту во время полета.

ITEM (ITEM #, ITEMNAME, Weight)

Flight (Flight #, Date, Dest, Weight)

PAYLOAD (Flight #, ITEM #, QTY, Weight)

Множество схем, определяющих отношения в базе данных, само представляется как отношение

Relation (Relname, Attmame),

которое содержит атрибуты всех отношений (иногда используется два отношения: одно для названий отношений, другое - для названий атрибутов).Например, в этой таблице запись

< Flight, Weight >

определяет, что отношение Flight содержит атрибутWeight.

ФУНКЦИОНАЛЬНАЯ ЗАВИСИМОСТЬ (FD)

Пусть X и Y два множества атрибутов в схеме R.

Определение. X функционально определяет Y(обозначается X->Y) тогда и только тогда, когда не
существует двух различных строк (векторов) в R с одноименными значениями из X и различными из Y.

Например, в классном журнале с атрибутами имя, присутствие и отметка за данное число. Атрибут имя функционально определяет все остальные атрибуты.

Функциональная зависимость позволяет определить базовое для РМ понятие первичного ключа отношения.

Определение. Множество атрибутов называется кандидатом в ключи для отношения, если оно функционально определяет все другие атрибуты и является минимальным множеством (т.е. ни один атрибут из множества не может быть исключен так, чтобы оно по-прежнему функционально определяло остальные).

Определение. Для любого отношения один из кандидатов в ключи отношения выделяется и называется первичным ключом.

Замечание. Первичный ключ не обновляется. Первичный ключ может быть использован для выбора спецификации строк в отношении и для того,чтобы связать отношение в нечто целое.

Определение. Первичный ключ для R_1, помещенныйв R_2, называется вторичным ключом в R_2.

Одно из основных правил целостности данных связано с первичным ключом. Целостность для элементов информации реляционной базы данных: ни одна строка таблицы не может принимать нулевое значение в каком-либо атрибуте из первичного ключа.Это свойство означает, что любые строки однозначно идентифицируемы.

Понятие функциональной зависимости позволяет определить и исследовать некоторые каналы утечки в базах данных, которые появляются из возможности вывода из одних объектов других (или еще один случай потока информации). Пусть множество F - это класс пар X-->Y, X, $Y \subseteq U$=Attr(R) - множество атрибутов R.

Теорема 1. *1. Если $Y \subseteq X \subseteq U$, то X->Y.*
2. Если X->Y, Y->Z, $X,Y,Z \subseteq U$, то X->Z.
3. Если X->Y, X->Z, $X,Y \subseteq U$, то $X->Y \cap Z$.
4.Если X->Y, X->Z, $X,Y,Z \subseteq U$, то $X->Y \cup Z$.

Доказательство.

1. Пусть есть два набора значений атрибутов Y, что $y \neq y'$. Y соответствует компонентам вектора X, для которых значения атрибутов в X не совпадают на y и y'.

2. Для строк с различными значениями атрибутовY значения векторов для атрибутов X различны. Для строк с различными значениями Z значения Y различны. Тогда значения X различны. Следовательно,X->Z.

3. Очевидно.

4. $a \neq a'$ из $Y \cup Z$, тогда отличаются значения атрибутов хотя бы Y или Z, например, Z. Тогда есть отличия на X. Теорема доказана.

Определение. Множество пар функциональных зависимостей F^+ называется замыканием множества пар функциональных зависимостей F, если F^+ - множество всех функциональных зависимостей, которые порождаются множеством F при помощи пп.1, 2, 3, 4 теоремы 1, то есть конструктивно построены, исходя из F, используя правила теоремы.

Определение. Два множества функциональных зависимостей (FD) F и G называются эквивалентными,если $F^+=G^+$.

Доказанная ниже теорема позволяет значительно сократить класс множеств FD, которые необходимо изучать для одного и того же F^+. В частности, достаточно ограничиться редуцированными FD, эквивалентными F.

Определение. Множество F функциональныхзависимостей называется редуцированным, если:

1) в F нет двух пар X->Y и X'->Y' таких, чтоX=X' и Y≠Y';

2) для всех X->Y в F $X \cap Y=0$.

Теорема 2. *Для произвольного F существует редуцированный G такой, что $F^+=G^+$.*

Доказательство. Для каждого X->Y из F построим эквивалентные функциональные зависимости в G, удовлетворяющие условиям редуцированного класса. Пусть X->Y, X'->Y' из F такие, чтоX=X', Y≠Y'.Если Y≠Y', то возьмем в G $X->Y \cup Y'$ по п. 4.

Наоборот: если в G есть $X->Y \cup Y'$, то в G^+ по п.1есть $Y \cup Y'->Y$ и $Y \cup Y'->Y'$.

Тогда по п. 2 теоремы 1 в G^+ есть X->Y, X->Y'.Пункт 1 доказан.

Пусть $X->Y$, $X \cap Y \neq \varnothing$.

Отсюда по п.1 и п.2 теоремы 1: $Y->Y \setminus (Y \cap X)=>Y->X->Y \setminus (Y \cap X)$.

Обратно. Если $X->Y \setminus (Y \cap X)$ есть в G, то X->Y есть в G^+, т.к. по п.1 теоремы 1: $X->X \cap Y$ и $X->(Y \setminus (Y \cap X)) \cup (X \cap Y)=Y$ по п.4. Теорема доказана.

Использование функциональной зависимости для компрометации базы данных иллюстрируется следующим простым примером.

Пример 1. Пусть R - отношение в реляционной базе данных некоторой компании, содержащее атрибуты имя - ранг - зарплата. Предположим, что зарплата - совершенно секретные сведения, а имя и ранг - секретные. Предположим также, что в R выполняется следующая функциональная зависимость РАНГ -> зарплата, что означает, что все служащие одного ранга получают одинаковую зарплату. Тогда пользователь, имеющий допуск к данным не выше секретно, может получить допуск к совершенно секретной информацию о конкретном лице, если он знает соответствие ранг-зарплата

хотя бы для некоторых лиц.

Для того, чтобы проанализировать возможность возникновения подобных зависимостей, надо изучить все множество F^+ для набора исходных зависимостей F. Это удобнее сделать, если F - редуцированное множество, что не ограничивает общности, благодаря доказанной теореме.

ЦЕЛОСТНОСТЬ В РМ.

Если описывать условия целостности данных в РМ формально, это можно сделать, определив конъюнкцию одного или нескольких условий, которым должны удовлетворять набор атрибутов, выраженных формулами какого-либо языка.

Пример 2. Ограничения целостности данных могут иметь следующий вид: I=(R.1A>0)∧(R1.A=R2.A)∧(0<R1.B+R2.B<100),где R1.A - атрибут A в отношении R1, R2.A -соответственно в R2, R1.B - атрибут B в R1, R2.B -в R2.

Если даны $I_1,...I_m$, то область состояний D базы данных определяется формулой

$$D=\bigcap_{k=1}^{m} I_k$$

где под I_k также подразумевается множество значений, которые могут принимать атрибуты базы данных, удовлетворяющие логической формуле I_k. Будем предполагать $D\neq\varnothing$.

РЕЛЯЦИОННЫЕ ОПЕРАТОРЫ (РО).

В реляционной модели выделяют реальные (или базовые) отношения, которые соответствуют хранящимся данным, и производные отношения, которые образуются с помощью реляционных операторов.

РО строит новые отношения из одного или нескольких существующих. Можно определить пять основных операторов.

1 . Select (R, F).

Строит новое отношение, состоящее из всех векторов (строк) R, удовлетворяющих F, где F -формула вида "$A_i\Theta V$'' или "$A_i\theta A_j$" где θ - отношение сравнения (=, < и т.д.) и V - значение из области D_i атрибута A_i. (Этот оператор также называют "θ -выбор").

2. Project (R ($Ai_1,...,Ai_k$)).

Строит новое отношение следующим образом: берутся по очереди все строки R и из каждой из них выбрасываются все элементы в координатах, не являющиеся атрибутами $Ai_1,...,Ai_k$, затем удаляются дубликаты в множестве получившихся строк нового отношения.

3. Union (R_1, R_2).

Строит новое отношение, состоящее из строк, которые есть хотя бы в одном отношении R_1 или R_2.Схемы для R_1 и R_2 должны быть совместимы, то есть иметь одинаковое число атрибутов, согласование по области определения атрибутов.

4. Diff (R_1, R_2).

Строит новое отношение, состоящее из тех и только тех строк R_1, которые не входят в R_2. Схемы R_1 и R_2 должны быть совместимы.

5. Product (R_1, R_2).

Образует новое отношение из прямого произведения таблиц R_1 и R_2, т.е. каждая строка R_1 приписывается каждой строке R_2.

Кроме перечисленных основных РО полезно определить следующие два производных оператора.

6. Natural-join (R_1, R_2, ($Ai_1,...,Ai_k$)).

Этот оператор выбирает и оставляет в новой таблице вектора из прямого произведения R_1 и R_2 такие, в которых атрибуты $Ai_1,...,Ai_k$ принимают одинаковое значение и затем выбрасываются лишние атрибуты (встречающиеся дважды).

7. Outer-join (R_1, R_2, ($Ai_1,...,Ai_k$)).

Строит Natural-join из R_1 и R_2 и к нему добавляет каждую строку в R_1, которая не имеет подходящей строки в R_2, а также добавляет каждую строку в R_2, которая не имеет подходящей строки в R_1.

1.6. МНОГОУРОВНЕВЫЕРЕЛЯЦИОННЫЕБАЗЫ ДАННЫХ.

Следующий шаг - внести решетку ценностей в информацию, наделенную структурой реляционной базы данных. Такое внесение возможно не всегда. Функционирование базы данных может привести к противоречию с размещением информации в том или ином классе решетки и затем, к компрометации этой информации.Чтобы этого не случилось надо согласовывать все элементы БД (т.е. отношения и реляционную алгебру) и решетку ценностей. А именно, при внесении решетки в БД (короче, при классификации информации) необходимо решить следующие задачи:

1. Уметь классифицировать отдельные (неделимые) факты и объекты. В реляционных БД требуется поддерживать MLS на уровне элементов потому, что каждая строка отношения может содержать много различных фактов, имеющих разные классификации (например, время вылета - секретно, время прибытия - секретно, назначение - совершенно секретно). Хотя в литературе использовались и другие подходы.
2. Уметь создавать для просмотра пользователями виртуальные отношения. Будем называть их *обзорами,* в которых не все данные имеют одну классификацию. Поскольку обзоры это производные данные, то создание обзора требует проведения классификации производных данных.
3. Уметь вносить новые данные и обновлять старые, причем элементы вносимых данных могут иметь разную классификацию.
4. Необходима состоятельность классов информации, т.е. для каждого факта не должны при различных видах обработки получаться различные классы ценности. При этом пользователь может не иметь доступа к некоторым классам, но не должен из-за этого терять возможность работать с БД.
5. Уметь определять ограничения целостности на данных, имеющих различную классификацию. В частности, это надо делать автоматически, чтобы не заставлять пользователя запоминать все правила классификации информации.
6. Уметь восстанавливать данные с учетом их классификации.

Основная идея внесения MLS в БД (D. Denning, T.Lunt и др.) состоит в создании нового отношения, где классы MLS входят как атрибуты отношения. То есть, любая данная схема расширяется включениями классификационного атрибута C_i для каждого атрибута данных A_i. Область значений C_i определяется парой классов (L_i, H_i), которые определяют подрешетку отнизшего класса L_i до высшего для данного атрибута класса H_i. Класс элемента a_i в данном векторе определяется $C(a_i)=C_i$ в этой подрешетке (функцияC(a)=class(a) - обозначает отображение элементовPM в решетку).

Определение. Многоуровневое базовое отношение (MLS R) определяется как произвольное отношение, у которого существуют классификационные атрибуты C_i для всех атрибутов данных A_i. Такое отношениепредставляется схемой:

$R(A_1, C_1, ..., A_n, C_n)$

Пример I. Решетка {S, TS}; A 1 - первичный ключ.

A1	C1	A2	C2	A3	C3
mad	S	17	S	X	S
foo	S	34	S	W	TS
ark	TS	S	TS	y	TS

Схема, в целом, классифицирована как S.

Определение. Атрибут A_i и соответственно атрибут C_i в MLS R называется одноуровневым, если область, определяемая C_i, - одна точка в решетке, иначе A_i называется многоуровневым.

Определение. MLS R называется одноуровневым, если все атрибуты одноуровневые и соответствуют одному классу.

Схема для MLS R также классифицируется. Этот класс обозначается class(R) и этот класс относится к имени отношения, имени всего набора атрибутов R и схеме. Class(R) должен доминироваться нижней гранью $L_1,...,L_n$ классификационных атрибутов $C_1,...,C_n$ в схеме. Это свойство позволяет доминировать class(R) всеми элементами таблицы.

В стандартной реляционной базе любой отсутствующий элемент представляется каким-нибудь аналогом нулевого значения. Положим, что

это выполняется и для многоуровневой реляционной модели. Кроме того, нулевое значение атрибута A_i будет определять наименьшее значение L_i атрибута C_i.

Тогда, если новое отношение V, например, обзор, не должен содержать TS данных, то должна осуществляться фильтрация данных, помещаемых в V. Тогда S - отфильтрованный обзор предыдущего примера,примет вид.

Пример 2.

A1	C1	A2	C2	A3	C3
mad	S	17	S	X	S
foo	S	34	S	null	S

КЛАССИФИКАЦИОННЫЕ ОГРАНИЧЕНИЯ.

Классификация информации в ходе функционирования системы происходит автоматически на основе классификационных ограничений (КО).

Определение. КО S - это правило, которое определяет значения для одного или более классификационных атрибутов C_i. Формально S - это четверка вида:

$$C=(R, A, E, L),$$

где R - имя отношения; A - набор из одного или нескольких атрибутов в R; E - опциональное выражение (например, формула логики предикатов); L - класс (точка в решетке ценностей).

Правило S интерпретируется следующим образом:

$$\text{if } E \text{ then class } (R.A)=L,$$

где R.A - означают атрибуты A в отношении R. Без ограничения общности, далее полагаем, что A_i определены на множестве действительных чисел (целые числа - частный случай, а значения не целочисленных атрибутов могут быть перенумерованы).

Замечание. Класс L - это выражение, которое может быть или константой, например, Secret, Top Secret, или содержать одну или несколько переменных. Например, "Classt(B)" или "Class(B) , Class(C)".

Определение. Два класса L_1 и L_2 называются равными, если они представлены символьно идентичными записями.

Замечание. Это позволяет конструктивно проверятьсостоятельность.

Выражение E есть конъюнкция одного или более условий, которым удовлетворяют наборы атрибутов в БД (дизъюнкции сводятся к конъюнкциям).

Пример 3.

$E=(R1.A>0)\wedge((R1.A)=(R2.A))\wedge(0<R1.B\oplus R2.B<100)$.

Существует 4 типа классификационных ограничений.

1. Зависящие от типа. Выражение E отсутствует и класс L - постоянный, так что все элементы, связанные с атрибутом, имеют один класс. Этот тип ограничений определяет одноуровневый атрибут. Например,

$$(R, A, secret)$$
$$(R, (A, B), , secret)$$

2. Зависящие от значения. Здесь или E присутствует, или L описывается выражением от переменной (или оба случая), так что класс элемента зависит от значения элемента, или от значения или класса других данныхБД. Например,

$$(R, A, A=1, conf)$$
$$(R, A, A=2, secret)$$
$$(R, A, , class(R.B))$$
$$(R, A, A=1, class(R.B))$$

3. Зависящие от уровня источника. Класс L есть выражение "класс пользователя". Например,

$$(R, A, , class\ (user))$$
$$(R, B, B>0, class\ (user))$$

4. Зависящие от грифа источника. Класс L есть выражение *, которое означает, что субъект, вводящий элемент, определяет также и класс элемента. Например,

$$(R.A, , *)$$
$$(R, B, B > O, *)$$

СОСТОЯТЕЛЬНОСТЬ.

Напомним :

Определение. Множество классификационных ограничений называется состоятельным, если для каждой возможной строки реляционной БД из D,

никакие два ограничения не определяют конфликтные классы для одного и того же элемента.

Пример 4. Рассмотрим БД Flight и классификационные ограничения на атрибуты Flight #, Dest, Date в отношении Flight.

(Flight, (Flight #, Dest, Date), (Date < 500), Secret)

(Flight, (Flight #, Dest, Date), (1< Dest< 2), T.S)

(Flight , (Flight #, Dest, Date), (Dest>2)v(Date > 500),class(user)).

Предположим, что ограничения целостности разрешают строку со следующими значениями:

(Flight #=1750, Dest=l, Date=450).

Тогда ограничения несостоятельны, так как строка удовлетворяет обоим первым двум ограничениям, но эти ограничения не определяют одинаковые классы. Однако классификационные ограничения не будут несостоятельными, если выполняются еще следующие ограничения целостности:

(1 <Dest<2)∧(Date>500),

так как тогда нижние два из классификационных ограничений не выполняются одновременно.

Для того, чтобы определить, является ли множество классификационных ограничений состоятельным при данных ограничениях целостности, достаточно определить, является ли каждая пара ограничений S_i иS_j состоятельной. (По определению проверить состоятельность требуется для каждой пары. Так как надо каждый элемент покрыть хотя бы одним ограничением, если покрывают 3 и более, и они конфликтны, то, следовательно, существуют два конфликтные.)

Для проверки пары используется следующая теорема.

Теорема. 1. *Данная пара S_i и S_j классификационных ограничений состоятельная, если выполняется хотя бы одно из следующих условий'.*

1. $L_i=L_j$ - оба ограничения определяют один класс (напоминаем, равенство символьное);

2. .$A^{(i)}\cap A^{(j)}=\varnothing$ - S_i и S_j накладывают ограничения на непересекающиеся множества атрибутов;

3. $E_i\cap E_j=\varnothing$ - оба ограничения не могут выполняться одновременно;

0

1

2

3

4

5

6

7

8

4. $E_i\cap E_j\cap D=\varnothing$ - оба ограничения не совместимы с условиями целостности.

Доказательство.

1. Очевидно, так как требуется символьное равенство выражений, откуда следует, что конфликтное присвоение классов невозможно.

2. Происходит присвоение классов разным атрибутам.

3. Невозможно присвоить хоть один класс.

4. Невозможно присвоить хоть один класс. Теорема доказана.

Простые достаточные условия теоремы позволяют реализовать алгоритм, их проверяющий, и эффективно проверить состоятельность классификационных ограничений на практике.

ПОЛНОТА КЛАССИФИКАЦИОННЫХ ОГРАНИЧЕНИЙ.

Пусть доказано, что система состоятельна. Теперь надо проверить, что она полна.

Определение. Множество классификационных ограничений называется полным, если для любого набора значений атрибутов из области определения базы данных каждому элементу приписывается класс хотя бы одним классификационным ограничением.

Рассмотрим процедуру проверки полноты.

1. Для каждого атрибута A рассмотрим $\{S_i\}$ ивыберем все ограничения, содержащие A; если это множество пусто, то система неполная.

2. Если классификационные ограничения не покрывают область возможных значений какого-либо атрибута, то система неполная. В противном случае система ограничений полная.

Пример 5. Пусть есть два атрибута A и B

D(A) = {(AB), 10<A<20},

D(B) = {(AB), -10<B<30},

S_1= (R, (A, B), A + 2B<30, Sec),

S_2= (R, (A, B), 5A - 2B<60, Sec),

S_3= (R, (A, B), 3B - 2A<30, Sec).

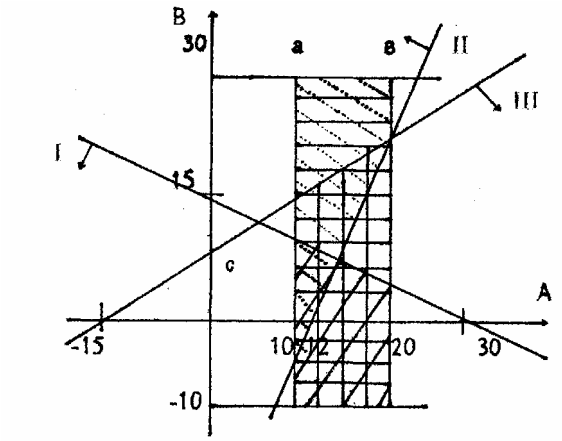


рис. 1.

Таким образом, все точки покрыты, отсюда следует полнота. Из того, что классы одинаковы, следует состоятельность.

ПРОБЛЕМА ПОЛИИНСТАНТИНАЦИИ.

Суть проблемы поясним на простом примере.

Пример 6. Пусть пользователь получил обзор, как это было сделано в примере 2 и решил дополнить его имеющимся в его распоряжении данными. Это нельзя запрещать, т.к. легко строится канал утечки. Предположим, что он достроил отношение примера 2.

A1	C1	A2	C2	A3	C3
mad	S	17	S	X	S
foo	S	34	S	U	S
ark	S	22	S	Z	S

Тогда в базе данных появились две таблицы с одинаковым набором значений ключевого атрибута, в том виде, как он виден пользователю. Это явление получило название полиинстантинация. Одно из решений проблемы - ввести классификационные атрибуты в состав ключевых атрибутов.

ДЕКОМПОЗИЦИЯ MLS R В СТАНДАРТНЫЕ БАЗОВЫЕ ОТНОШЕНИЯ РЕЛЯЦИОННОЙМОДЕЛИ.

Мы построим декомпозицию произвольной MLS R в семейство стандартных базовых отношений БД. Тогда реальные отношения получаются как обзор этих базовых. При этом, стандартные базовые отношения будут одноуровневыми. Отсюда следует, что механизм хранения их идентичен механизму хранения в обычной БД (не MLS), т.е. имеется ввиду отображение при помощи программного обеспечения в файлы, сегменты и т.д. Механизм защиты здесь - внешний и не связан с конкретными элементами информации. Этот механизм определяет защиту и доступ к информации любого заданного класса.Сначала несколько замечаний.

Определение. Пусть множество атрибутов A одинаково классифицируются на всех строках отношения, тогда говорят, что они равномерно классифицированы.

Теорема 3. *При выполнении основных ограничений целостности множество атрибутов первичного ключа равномерно классифицировано и этот класс доминируется классами всех остальных элементов строки.*

Доказательство. Если первичный ключ не равномерно классифицирован, то возможно существует пользователь, который не имеет права на высший класс, но имеет право на низший. В этом случае, в обзоре этого отношения в атрибуте высшего класса первичного ключа должны стоять null, что противоречит требованиям целостности хранимой информации (в первичном ключе нельзя иметь null). Аналогично, если какой-то элемент имеет класс меньший, чем элементы первичного ключа, то в обзоре эта строка появится, а первичный ключ будет null. Теорема доказана.

Пусть дано $R(A_1,C_1,\ldots, A_n,C_n)$, где A_1 - первичный ключ. Первый шаг декомпозиции - создать базовоеотношение $R_{1,X}$ для каждого значения x атрибута C_1 первичного ключа A_1, т.е. для каждого $x\subseteq(L_1, H_1)$создаем $R_{1.X}(A_1)$ с классом x. Назовем их "отношениями, связанными с первичным ключом". Если первичный ключ состоит из нескольких атрибутов, то, в силу равномерности классификации, можно считатьего одним атрибутом A_1 с одним классификационным атрибутом C_1 и для него создать $R_{1.X}(A_1)$.

Второй шаг. Для произвольного A.(которое может также быть множеством атрибутов, но равномерно классифицированным с атрибутом C_i) образуем базовые отношения (называемые "отношения, связанные с атрибутом A_i").

Для $i = 2,\ldots,n$ каждой паре x,y, $x\in(L_1, H_1)$, $y\in(L_i, H_i)$, $y>x$ (из теоремы 3) строим $R_{i,X,Y}(A_1, A_i)$ с классом y.

В этих отношениях, очевидно, A_1 - первичный ключ. Однако таким отношениям присваивается класс y (поднимая, возможно, уровень значения первичного ключа, что при восстановлении исправляется исходя из информации в $R_{1,X}$), таким образом, эти отношения одноуровневые по построению. Алгоритм декомпозиции следующий (заменим R на $\{R_{1,X}(A_1), R_{i,X,Y}(A_1, A_i)\}$): для каждой строки $(a_1,c_1,\ldots, a_n,c_n)$ в R поместим a_1 в $R_{1,C1}$ и для i=1,..., n поместим (a_1, a_n) в $R_{i.C1.Ci}$. Любая новая строка при коррекции БД разлагается также.

Пример 7. Рассмотрим отношение из примера 1.

R_{1S}

A1
mad
foo

sec

R_{2SS}

A1	A2
mad	17
foo	34

sec

R_{1TS}

A1
arc

TS

R_{2TSTS}

A1	A2
arc	5

TS

R_{3SS}

A1	A3
mad	X

sec R_{3TSTS}

A1	A3
arc	Y

TS

R_{3STS}

A1	A3
foo	W

TS

Очевидно, что, если исходное отношение - одноуровневое, то процедура декомпозиции тривиальна.

Рассмотрим процедуру восстановления MLS из базовых одноуровневых отношений. Определим оператор "ml", который преобразует строку стандартных отношений $R_{1,X}$ или $R_{i,X,Y}$, в строку MLS отношения.

а) $m1(R_{i,X})$ - многоуровневое отношение R' со схемой $R'(A_1,C_1)$. Для каждой строки a_1 в $R_{1,X}$ строка (a_1, x) строится в R'.

б) $m1(R_{i,X,Y})$ - многоуровневое отношение R' со схемой $R'(A_1, C_1, A_i, C_i)$. Для каждой строки a_1, a_i в $R_{i,X,Y}$ строка (a_1, x, a_i, y) строится в R'.

Обозначим через $У_{AB}$ оператор Outer-join относительно атрибутов AB и через **U** оператор Union. Тогда формула для восстановления R имеет вид:

$$Recover(R) = \cup\ (R'_{1,X} У_{A1C1} R'_{2,X}\ У_{A1C1} \ldots У_{A1C1} R'_{n,X}),$$

где $R'_{i,X} = \bigcup_{y \in \{max(L_i,x),H_i\}} R'_{i,X,Y}$, $i = 2,\ldots, n$,

$R'_{1,X} = m1(R_{1,X})$,

$R'_{i,X,Y} = m1(R'_{i,X,Y})$.

Можно доказать следующую теорему.

Теорема 4. Recover (Decompose R) = R.

Вместо доказательства восстановим пример 1.

Пример 8. $m1(R_{i,X}) = R'_{i,X}$

$m1(R_{i,X,Y}) = R'_{i,X,Y}$

$R'_{1,S}$

A1	C1
mad	S
foo	S

$R'_{1,TS}$

A1	C1
arc	TS

$R'_{2,S,S}$

A1	C1	A2	C2
mad	S	17	S
foo	S	34	S

$R'_{2,S,TS}$

A1	C1	A2	C2
arc	TS	5	TS

$R'_{3,S,TS}$

A1	C1	A3	C3
foo	S	W	TS

$R'_{3,TS,TS}$

A1	C1	A3	C3
arc	TS	y	TS

$R'_{3,S,S}$

A1	C1	A3	C3
mad	S	X	S

$$Recover(R) = ((R'_{1S} У_{A1C1} R'_{2SS}) У_{A1C1}, (R'_{3SS} \mathbf{U} R'_{3STS})) \mathbf{U} (R'_{1TS} У_{A1C1} R'_{2TSTS} У_{A1C1} R'_{3TSTS}).$$

Отсюда получаем отношение примера 1.

Глава 2

Угрозы информации

Если информация представляет ценность, то необходимо понять, в каком смысле эту ценность необходимо оберегать. Если ценность информации теряется при ее раскрытии, то говорят, что имеется опасность нарушения секретности информации. Если ценность информации теряется при изменении или уничтожении информации, то говорят, что имеется опасность для целостности информации. Если ценность информации в ее оперативном использовании, то говорят,что имеется опасность нарушения доступности информации. Если ценность информации теряется при сбоях в системе, то говорят, что есть опасность потери устойчивости к ошибкам. Как правило, рассматривают три опасности, которые надо предотвратить путем защиты: секретность, целостность, доступность. Хотя, как показывают примеры действий в боевых условиях, развитие сложных систем Hewlett-Packard,

Tandem, практически добавляется четвертое направление: устойчивость к ошибкам.

Под угрозами подразумеваются пути реализации воздействий, которые считаются опасными. Например, угроза съема информации и перехвата излучения с дисплея ведет к потере секретности, угроза пожара ведет к нарушению целостности информации, угроза разрыва канала может реализовать опасность потерять доступность. Угроза сбоя электроэнергии может реализовать опасность неправильной оценки ситуации в системе управления и т.д.

В главе рассматриваются вопросы анализа опасностей, выявления угроз. Далее рассматриваются основные угрозы нарушения секретности в ЭСОД и механизмы их предотвращения, угрозы нарушения целостности и механизмы защиты от них. Связь между видом опасности и возможной угрозой состоит в месте, времени и типе атаки, реализующей угрозу. Анализ опасности должен показать, где и когда появляется ценная информация, в каком месте системы эта информация может потерять ценность. Угроза характеризует способ нападения в. определенном месте и в определенный момент. Угроза реализуется через атаку в определенном месте и в определенное время.

2.1. УГРОЗЫ СЕКРЕТНОСТИ

В руководстве по использованию стандарта защиты информации американцы говорят, что существуеттолько два пути нарушения секретности:

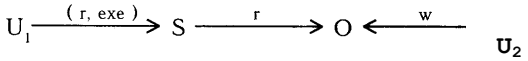
- утрата контроля над системой защиты;
- каналы утечки информации.

Если система обеспечения защиты перестает адекватно функционировать, то, естественно, траектории вычислительного процесса могут пройти через состояние, когда осуществляется запрещенный доступ. Каналы утечки характеризуют ту ситуацию, когда либо проектировщики не смогли предупредить, либо система не в состоянии рассматривать такой доступ как запрещенный. Утрата управления системой защиты может быть реализована оперативными мерами и здесь играют существенную роль административные и кадровые методы защиты. Утрата контроля за защитой может возникнуть в критической ситуации, которая может быть создана стихийно или искусственно. Поэтому одной из главных опасностей для системы защиты является отсутствие устойчивости к ошибкам.

Утрата контроля может возникнуть за счет взламывания защиты самой системы защиты. Противопоставить этому можно только создание защищенного домена для системы защиты.

Разумеется, в реальной жизни используются комбинации этих атак.

Большой спектр возможностей дают каналы утечки. Основной класс каналов утечки в ЭСОД - каналы по памяти (т.е. каналы, которые образуются за счет использования доступа к общим объектам системы). Графически канал по памяти можно изобразить следующим образом:



Пользователь U_1 активизирует процесс, который может получить доступ на чтение к общему с пользователем U_2 ресурсу O, при этом U_2 может писать в O, а U_1 может читать от S. Приведем примеры таких каналов.

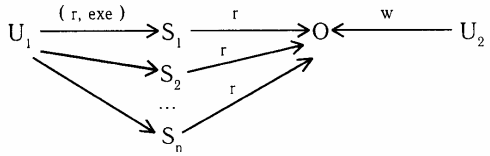
Пример 1. В директорию O внесены имена файлов. Хотя доступ к самим файлам для субъекта S_1 закрыт, доступ к директории возможен. Если субъект S_2 создал закрытые файлы, то информация о файловой структуре стала доступной S_1. Произошла утечка части информации. В частности, существование или нет одного конкретного файла - 1 бит.

Значит, в этом случае создан канал утечки одного бита из той информации, которая принадлежит S_2.

Пример *2.* Вирус-архиватор, созданный пользователем U_1, заражает командные файлы пользователя U_2 за счет использования совместных ресурсов объекта в виде компьютерной игры. Съем информации осуществляется при помощи записи архива сделанных U_2 файлов на каждую принесенную дискету. Это гарантирует анонимность истинного получателя информации в случае выявления вируса.

Защитные механизмы основаны на правильном выборе политики безопасности.

Пример 3. Очень важным примером канала утечки по памяти является возможность статистического вывода в базах данных. Обычно в базах данных с ограниченным доступом функции вычисления статистик по закрытым данным являются общедоступными. Это создает ситуацию совместного использования закрытых ресурсов допущенными и незаконными пользователями. Как было показано в разделе "информационные потоки", канал связи от закрытой информации к незаконному пользователю может быть сильно зашумлен. Однако использование различных статистик и модификация запросов могут позволить отфильтровать информацию.



где U_1 - незаконный пользователь; U_2 - закон-ный пользователь ценной информации в объекте O; S_1, S_2,...,S_n - процессы вычисления ответов на различные запросы пользователя U_1. Доступ S_i к O разрешен, так как в каждом случае по O вычисляетсястатистическая характеристика, не дающая достаточно полной информации об объекте O. Защитные механизмы основаны на контроле возможностей вывода и контроле информационных потоков.

Следующий основной класс каналов утечки американцы называют каналами по времени. Канал по времени является каналом, передающим противнику информацию о процессе, промодулированном ценной закрытой информацией. Графически канал по времени можно изобразить следующей схемой

$$U_1 \xrightarrow{(r,exe)} S \xrightarrow{r} S_m \xleftarrow{w} S_ц \xleftarrow{exe} U_2$$

где U_1 - злоумышленник; U_2 - пользователь, оперирующий ценной информацией; $S_ц$ - субъект, информация о котором представляет интерес; S_m -субъект, процесс которого модулируется информацией процесса $S_ц$; S - процесс от имени пользователя U_1, позволяющий наблюдать процесс S_m.

Функционирование канала утечки определяется той долей ценной информации о процессе $S_ц$, которая передается путем модуляции процессу S_m.

Пример 4. Пусть процесс $S_ц$ использует принтер для печатания результатов очередного цикла обработки информации. Процесс $S_м$ определяется работой принтера, который является общим ресурсом U_1 и U_2 с приоритетом у U_2. Тогда процесс S регулярно с заданной частотой посылает

запрос на использование принтера и получает отказ, когда $S_ц$ распечатывает очередную порцию информации. Тогда в единицах частоты запроса пользователь U_1 получает информацию о периодах обработки процессом $S_ц$ ценной информации, то есть получаем канал утечки. Защитные механизмы от таких каналов основаны на контроле информационных потоков в системе.

Пример 5 Перехват информации в канале связи является примером канала утечки по времени. Здесь реализуется непосредственный доступ к процессу обработки (передачи) ценной информации. Съем информации об этом процессе и накопление ее во времени восстанавливают переданную ценную информацию. Защита от этих каналов основана на криптографии.

Пример 6. Побочные каналы утечки по излучению, питанию или акустике являются типичными каналами утечки по времени. Защитные механизмы основаны на экранировании, фильтрах и зашумлении.

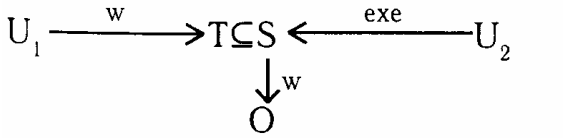
2.2. УГРОЗЫ ЦЕЛОСТНОСТИ

Нарушения целостности информации - это незаконные уничтожение или модификация информации.

Традиционно защита целостности относится к категории организационных мер. Основным источником угроз целостности являются пожары и стихийные бедствия. К уничтожению и модификации могут привести также случайные и преднамеренные критические ситуации в системе, вирусы, "троянские кони" и т.д.

Язык описания угроз целостности в целом аналогичен языку угроз секретности. Однако в данном случаев место каналов утечки удобнее говорить о каналах воздействия на целостность (или о каналах разрушающего воздействия). По сути они аналогичны каналам утечки, если заменить доступ (r) доступом (w).

Пример 1. Канал несанкционированной модификации, использующий "троянского коня", изображен на следующей схеме:



где U_1 - злоумышленник; U_2 - пользователь; O -объект с ценной информацией; S - процесс (программа), являющаяся общим ресурсом U_1 и U_2.

Пользователь U_1, пользуясь правом w, модифицировал общий ресурс S, встроив в него скрытую программу T, модифицирующую информацию в O при запуске ее пользователем U_2.

Исследованием схем примера 1 занимается теория распространения вирусов.

Основой защиты целостности является своевременное регулярное копирование ценной информации.

Другой класс механизмов защиты целостности основан на идее помехозащищенного кодирования информации (введение избыточности в информацию) и составляет основу контроля целостности. Он основан на аутентификации, т.е. подтверждении подлинности, целостности информации. Подтверждение подлинности охраняет целостность интерфейса, а использование кодов аутентификации позволяют контролировать целостность файлов и сообщений. Введение избыточности в языки и формальное задание спецификации позволяет контролировать целостность программ.

Наконец, к механизмам контроля и защиты целостности информации следует отнести создание системной избыточности. В военной практике такие меры называются: повышение "живучести" системы. Использование таких механизмов позволяет также решать задачи устойчивости к ошибкам и задачи защиты от нарушений доступности.

Глава 3

Политика безопасности

Иногда удается достичь общепринятого понимания оптимальности принимаемого решения и доказать его существование. Например, в математической статистике для проверки простой гипотезы против простой альтернативы всеми признано понятие оптимального решения, которое минимизирует ошибку второго рода, а также доказано существование такого критерия (лемма Неймана-Пирсона). Однако, когда решение многоальтернативное, то общепринятого понимания оптимальности не получается, а в тех случаях, когда рассматривается вопрос об оптимальном в каком-то смысле решении, то его существование, чаще всего, удается доказать лишь в частных задачах.

Подобная ситуация существует в задачах защиты информации, поскольку неоднозначно решение о том, что информация защищена. Кроме того, система защиты - не самоцель и должна нести подчиненную функцию по сравнению с главной целью вычислительного процесса. Приведем примеры, поясняющие эти утверждения.

Пример 1. Пусть два инженера ведут разработки двух приборов, которые требуют решения задач $\xi_1, \ldots, \xi_{n1}$ - первым и задач $\xi'_1, \ldots, \xi'_{n2}$ - вторым инженером. Предположим, что информация о решении каждой задачи собирается в отдельном файле $O_1, \ldots, O_{n1}$ и $O'_1, \ldots, O'_{n2}$ соответственно. Предположим, что среди множеств задач первого и второго инженеров есть одинаковые. К сожалению, обычный офицер службы безопасности, разрешающий или запрещающий доступ к файлам, не в состоянии решить, что в двух файлах накапливается информация по решению одной задачи. Рассмотрим различные решения офицера по обеспечению безопасности информации.

1. Если он разрешит доступ инженеров к файлам друг друга, то один из них, взяв информацию другого или свою, анонимно и поэтому безнаказанно, продаст эту информацию, так как нет персональной ответственности (невозможно установить, кто продал информацию из данного файла). При этом безнаказанность может стимулировать преступление.

2. Если он не разрешит доступ инженеров к файлам друг друга, то возникает опасность ущерба из-за недоступности информации (один нашел, а второй не нашел решение одной задачи; тогда вся задача второго инженера оказалась нерешенной, из-за чего возможен большой ущерб для фирмы, т.к. соответствующий прибор сделали конкуренты).

Очевидно, что в обоих случаях достигается снижение одной опасности за счет возрастания другой.

Пример 2. Пример посвящен проблеме компромисса задачи защиты и других задач вычислительной системы. Пусть в базе данных собирается информация о здоровье частных лиц, которая в большинстве стран считается конфиденциальной. База данных нужна, т.к.эта информация позволяет эффективно производить диагностику. Если доступ к этой базе из соображений защиты информации сильно ограничен, то в такой базе не будет пользы для врачей, ставящих диагнозы, и не будет пользы от самой базы. Если доступ открыть, то возможна утечка конфиденциальной информации, за которую по суду может быть предъявлен большой иск. Каким должно быть оптимальное решение?

Результатом решения в приведенных примерах и других аналогичных задачах является выбор правил распределения и хранения информации, а также обращения с информацией, что и называется политикой безопасности. Соблюдение политики безопасности должно обеспечить выполнение того компромисса между альтернативами, который выбрали владельцы ценной информации для ее защиты. Ясно, что, являясь результатом компромисса, политика безопасности никогда не удовлетворит все стороны, участвующие во взаимодействии с защищаемой информацией. В тоже время выбор политики безопасности - это окончательное решение проблемы: что - хорошо и что - плохо в обращении с ценной информацией. После принятия такого решения можно строить защиту, то есть систему поддержки выполнения правил политики безопасности. Таким образом, построенная система защиты информации хорошая, если она надежно поддерживает выполнение правил политики безопасности. Наоборот, система защиты информации - плохая, если она ненадежно поддерживает политику безопасности.

Такое решение проблемы защищенности информации и проблемы построения системы защиты позволяет привлечь в теорию защиты точные математические методы. То есть доказывать, что данная система в заданных условиях поддерживает политику безопасности. В этом суть доказательного подхода к защите информации, позволяющего говорить о "гарантированно защищенной системе". Смысл "гарантированной защиты" в том, что при соблюдении исходных условий заведомо выполняются все правила политики безопасности. Термин "гарантированная защита" впервые встречается в стандарте министерства обороны США на требования к защищенным системам ("Оранжевая книга").

В данной главе приводятся определения и примеры политик безопасности, показаны последствия плохо выбранных политик. Определены такие политики как дискреционная политика, политика MLS, политика защиты целостности Biba и проведен их анализ. На примере РМ рассмотрены математические проблемы корректного определения политики в данной вычислительной системе.

3.1. ОПРЕДЕЛЕНИЕ ПОЛИТИКИ БЕЗОПАСНОСТИ

Будем следовать общепринятому определению политики безопасности (ПБ), приведенному в стандарте "Оранжевая книга" (1985 г.).

Определение. Политика безопасности это набор норм, правил и практических приемов, которые регулируют управление, защиту и распределение ценной информации .

Полное описание ПБ достаточно объемно даже в простых случаях, поэтому далее будем пользоваться сокращенными описаниями.

Если вспомнить модель защиты, построенную в параграфе 1.1, то смысл политики безопасности очень прост - это набор правил управления доступом. Заметим отличие ПБ от употребляемого понятия несанкционированный доступ (НСД). Первое отличие состоит в том, что политика определяет как разрешенные, так и неразрешенные доступы. Второе отличие - ПБ по своему определению конструктивна, может быть основой определения некоторого автомата или аппарата для своей реализации.

Пример 1. Сформулируем простую политику безопасности в некотором учреждении. Цель, стоящая перед защитой, - обеспечение секретности информации. ПБ состоит в следующем: каждый пользователь пользуется своими и только своими данными, не обмениваясь с другими пользователями. Легко построить систему, поддерживающую эту политику. Каждый пользователь имеет свой персональный компьютер в персональной охраняемой комнате, куда не допускаются кроме него посторонние лица. Легко видеть, что сформулированная выше политика реализуется в этой системе. Будем называть эту политику тривиальной разграничительной (дискреционной) политикой.

ПБ определяется неоднозначно и, естественно, всегда связана с практической реализацией системы и механизмов защиты. Например, ПБ в примере 1 может полностью измениться, если в организации нет достаточного числа компьютеров и помещений для поддержки этой политики.

Выбор ПБ определяется фазовым пространством, допустимыми природой вычислительных процессов, траекториями в нем и заданием неблагоприятного множества ***N***. Корректность ПБ в данных конкретных условиях должна быть, вообще говоря, доказана.

Построение политики безопасности обычно соответствует следующим шагам:

1 шаг. В информацию вносится структура ценностей и проводится анализ риска.

2 шаг. Определяются правила для любого процесса пользования данным видом доступа к элементам информации, имеющим данную оценку ценностей.

Однако реализация этих шагов является сложной задачей. Результатом ошибочного или бездумного определения правил политики безопасности, как правило, является разрушение ценности информации без нарушения политики. Таким образом, даже хорошая система защиты может быть "прозрачной" для злоумышленника при плохой ПБ.

Рассмотрим следующие примеры.

Пример 2. Пусть банковские счета хранятся в зашифрованном виде в файлах ЭВМ. Для зашифрования, естественно, используется блочная система шифра, которая для надежности реализована вне компьютера и оперируется с помощью доверенного лица. Прочитав в книгах о хороших механизмах защиты, служба безопасности банка убеждена, что если шифр стойкий, то указанным способом информация хорошо защищена. Действительно, прочитать ее при хорошем шифре невозможно, но служащий банка, знающий стандарты заполнения счетов и имеющий доступ к компьютеру, может заменить часть шифртекста в своем счете на шифртекст в счете богатого клиента. Если форматы совпали, то счет такого служащего с большой вероятностью возрастет. В этом примере игра идет на том, что в данной задаче опасность для целостности информации значительно выше опасности для нарушения секретности, а выбранная политика безопасности хорошо защищает от нарушений секретности, но не ориентирована на опасность для целостности.

Пример 3. Как было описано в примере в конце параграфа 1.4 для государственных структур традиционно принято определять гриф результирующего документа как верхнюю грань грифов и категорий составляющих этот документ частей. Формальное перенесение этого правила традиционной ПБ в ПБ для электронных документов может привести к возникновению канала утечки информации. В самом деле, рассмотрим многоуровневую реляционную базу данных как в параграфе 1.7 с решеткой ценностей {несекретно(Н), секретно (С)}. Пусть R - отношение, A_1,..., A_m - атрибуты, причем A_1- первичный ключ. По запросу строится "обзор" R', который состоит из элементов различной классификации. ПБ может включать одно из двух правил формализованного грифа отношения R':

1. (Привычный для госструктур) R' имеет гриф, равный наибольшему значению из грифов элементов, которые принимают входящие в него атрибуты.

2. (Как это было сделано в параграфе 1.7) R' имеет гриф, равный наименьшему значению из грифов элементов, которые принимают входящие в него атрибуты.

Покажем, что в случае 1 возможна утечка информации с грифом С пользователю, которому разрешен доступ только к информации с грифом Н. Для этого достаточно построить пример базы данных, где такая утечка очевидна.

Пусть реляционная база данных реализует геоинформационную систему. Например, она содержит географическую карту некоторого района пустыни. Базовое отношение РМ имеет три атрибута:

A_1 - ключевой атрибут, содержащий координаты и размеры прямоугольного сектора в некоторой сетке координат;

A_2 - изображение (карта) местности в секторе с координатами, задаваемыми атрибутом A_1;

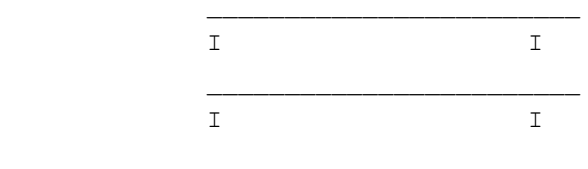
Аз - координаты колодцев с водой в рассматриваемом секторе.

Для простоты будем считать, что на запрос в базу данных мы получаем на экране изображение карты сектора, определяемого значением атрибута A_1. Пустьзначения атрибутов A_1 и A_2 имеют гриф Н, а значения атрибута A_3 - С.

Если выбрать политику безопасности пункта 1, то мы покажем, как пользователь, не имеющий доступа к секретной информации, реализует канал утечки секретных данных о том, где в пустыне находится колодец с водой (пусть, для простоты, в рассматриваемой местности есть только один колодец). Для получения секретной информации пользователь делает последовательность запросов в базу данных, причем каждый следующий запрос (можно говорить о шагах алгоритма пользователя) определяется ответом на предыдущий.

1 шаг. Разбиваем район (для удобства - квадрат) на полосы и делаем запрос на эти участки в базу данных. Ответ возможен в двух формах:

- отказ от показа карты, если она секретная, таккак пользователю, не имеющему допуска к секретнойинформации, база данных, естественно, не должна еепоказывать;
- представление карты на экране, если она имеетгриф Н.



Если есть отказ в доступе, то в этом случае в прямоугольнике есть колодец.

2 шаг. Разбиваем полосу, где есть колодец (т.е. где есть отказ в доступе) пополам на две полосы и делаем два запроса в базу данных. Отказ означает, что в данной полосе есть колодец.

И так далее.

В результате вычисляется первая координата колодца с любой заданной точностью. Затем, в оставшейся полосе аналогично вычисляем вторую координату.

Таким образом, ПБ соблюдена, однако, произошла утечка секретной информации.

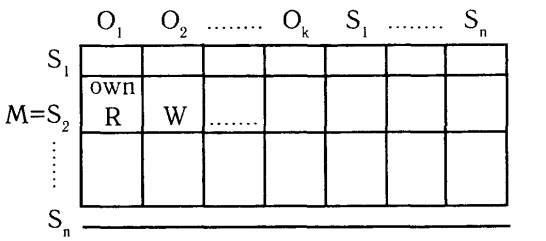
Если использовать ПБ пункта 2, то любой пользователь получает карту, но пользователь с допуском к секретной информации получает карту с нанесенным колодцем, а пользователь без такого доступа - без колодца. В этом случае канал, построенный выше, не работает и ПБ надежно защищает информацию.

3.2. ДИСКРЕЦИОННАЯ ПОЛИТИКА.

Заглавие параграфа является дословным переводом Discretionary policy, еще одним вариантом перевода является следующий - разграничительная политика. Рассматриваемая политика - одна из самых распространенных в мире, в системах по умолчанию имеется ввиду именно эта политика.

Пусть **O** - множество объектов, **S** - множество субъектов, $S \subseteq O$. Пусть $U=\{U_1,...,U_m\}$ - множество пользователей. Определим отображение: own: **O→U.**

В соответствии с этим отображением каждый объект объявляется собственностью соответствующего пользователя. Пользователь, являющийся собственником объекта, имеет все права доступа к нему, а иногда и право передавать часть или все права другим пользователям. Кроме того, собственник объекта определяет права доступа других субъектов к этому объекту, то есть политику безопасности в отношении этого объекта. Указанные права доступа записываются в виде матрицы доступа, элементы которой - суть подмножества множества **R**, определяющие доступы субъекта S, к объекту O_i ($i = 1, 2,...,; j = 1, 2,...$).



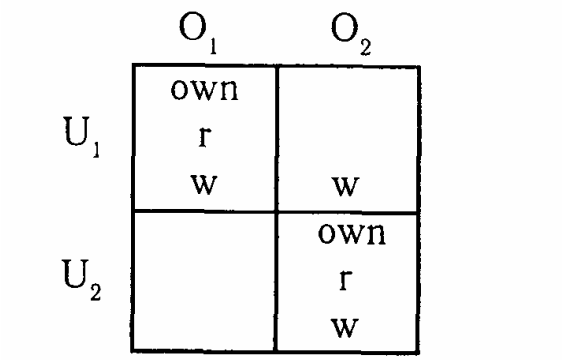
Существует несколько вариантов задания матрицы доступа.

1. Листы возможностей: Для каждого субъекта S_i создается лист (файл) всех объектов, к которому имеет доступ данный объект.

2. Листы контроля доступа: для каждого объекта создается список всех субъектов, имеющих право доступа к этому объекту.

Дискреционная политика связана с исходной моделью таким образом, что траектории процессов в вычислительной системе ограничиваются в каждом доступе. Причем вершины каждого графа разбиваются на классы и доступ в каждом классе определяется своими правилами каждым собственником. Множество неблагоприятных траекторий ***N*** для рассматриваемого класса политик определяется наличием неблагоприятных состояний, которые в свою очередь определяются запретами на некоторые дуги. Дискреционная политика, как самая распространенная, больше всего подвергалась исследованиям. Существует множество разновидностей этой политики. Однако многих проблем защиты эта политика решить не может. Одна из самых существенных слабостей этого jкласса политик - то, что они не выдерживают атак при помощи "Троянского коня". Это означает, в частности, что система защиты, реализующая дискреционную политику, плохо защищает от проникновения вирусов в систему и других средств скрытого разрушающего воздействия. Покажем на примере принцип атаки "Троянским конем" в случае дискреционной политики.

Пример 1. Пусть U_1 - некоторый пользователь, а U_2 - пользователь-злоумышленник, O_1 - объект, содержащий ценную информацию, O_2 - программа с "Троянским конем" T, и M - матрица доступа, которая имеет вид:



Проникновение программы происходит следующим образом. Злоумышленник U_2 создает программу O_2 и, являясь ее собственником, дает U_1 запускать ее и писать в объект O_2 информацию. После этого он инициирует каким-то образом, чтобы U_1 запустил эту программу (например, O_2 - представляет интересную компьютерную игру, которую он предлагает U_1 для развлечения). U_1 запускает O_2 и тем самым запускает скрытую программу T, которая обладая правами U_1 (т.к.была запущена пользователем U_1), списывает в себя информацию, содержащуюся в O_1. После этого хозяин U_2 объекта O_2, пользуясь всеми правами, имеет возможность считать из O_2 ценную информацию объекта O_1.

Следующая проблема дискреционной политики -это автоматическое определение прав. Так как объектов много, то задать заранее вручную перечень прав каждого субъекта на доступ к объекту невозможно. Поэтому матрица доступа различными способами агрегируется, например, оставляются в качестве субъектов только пользователи, а в соответствующую ячейку матрицы вставляются формулы функций, вычисление которых определяет права доступа субъекта, порожденного пользователем, к объекту O. Разумеется, эти функции могут изменяться во времени. В частности, возможно изъятие прав после выполнения некоторого события. Возможны модификации, зависящие от других параметров.

Одна из важнейших проблем при использовании дискреционной политики - это проблема контроля распространения прав доступа. Чаще всего бывает,что владелец файла передает содержание файла другому пользователю и тот, тем самым, приобретает права собственника на информацию. Таким образом, права могут распространяться, и даже, если исходный владелец не хотел передавать доступ некоторому субъекту S к своей информации в O, то после нескольких шагов передача прав может состояться независимо от его воли. Возникает задача об условиях, при которых в такой системе некоторый субъект рано или поздно получит требуемый ему доступ. Эта задача иследовалась в модели "take-grant", когда форма передачи или взятия прав определяются в виде специального права доступа (вместо own). Некоторые результаты этих исследований будут приведены в главе "Математические методы анализа политики безопасности".

3.3. ПОЛИТИКА MLS.

Многоуровневая политика безопасности (политикаMLS) принята всеми развитыми государствами мира. В повседневном секретном делопроизводстве госсектор России также придерживается этой политики.

Решетка ценностей SC, введенная в параграфе 1.3 является основой политики MLS. Другой основой этой политики является понятие информационного потока (см. 1.4). Для произвольных объектов X и Y пусть

имеется информационный поток X->$_{\alpha}$Y, где X -источник, Y - получатель информации. Отображениес: O->SC считается заданным. Если c(Y)>c(X), то Y -более ценный объект, чем X.

Определение. Политика MLS считает информационный поток X->Y разрешенным тогда и только тогда, когда c(Y)>c(X) в решетке SC.

Таким образом, политика MLS имеет дело с множеством информационных потоков в системе и делит их на разрешенные и неразрешенные очень простым условием. Однако эта простота касается информационных потоков, которых в системе огромное количество. Поэтому приведенное выше определение неконструктивно. Хотелось бы иметь конструктивноеопределение на языке доступов. Рассмотрим класс систем с двумя видами доступов r и w (хотя могут быть и другие доступы, но они либо не определяют информационных потоков, либо выражаются через w и r). Пусть процесс S в ходе решения своей задачипоследовательно обращается к объектам $O_1, O_2, ..., O_n$ (некоторые из них могут возникнуть в ходе решения задачи). Пусть

$$S \xrightarrow{r} O_{i1}, S \xrightarrow{r} O_{i2}, S \xrightarrow{r} O_{ik}, S \xrightarrow{w} O_{i1}, S \xrightarrow{w} O_{jn-k} \qquad (1)$$

Тогда из параграфа 1.3 следует, что при выполнении условий c(S)>c(Oi_t), t=l,....k, соответствующие потоки информации будут идти в разрешенном политикой MLS направлении, а при c(S)<c(Oj_t), t=l,.., п-k, потоки,определяемые доступом w, будут идти в разрешенном направлении. Таким образом, в результате выполнения задачи процессом S, информационные потоки, с ним связанные, удовлетворяют политике MLS. Такого качественного анализа оказывается достаточно, чтобы классифицировать почти все процессы и принять решение о соблюдении или нет политики MLS. Если где-то политика MLS нарушается, то соответствующий доступ не разрешается. Причем разрешенность цепочки (1) вовсе не означает, что субъект S не можетсоздать объект O такой, что c(S)>c(0). Однако он не может писать туда информацию. При передаче управления поток информации от процесса S или к нему прерывается (хотя в него другие процессы могут записывать или считывать информацию как в объект). При этом, если правила направления потока при r и w выполняются, то MLS соблюдается, если нет, то соответствующий процесс не получает доступ. Таким образом, мы приходим к управлению потоками через контроль доступов. В результате для определенного класса систем получим конструктивное описание политики MLS.

Определение. В системе с двумя доступами r и w политика MLS определяется следующими правилами доступа

$$X \xrightarrow{r} Y \Leftrightarrow c(X) \geq c(Y),$$

$$X \xrightarrow{w} Y \Leftrightarrow c(X) \leq c(Y).$$

Структура решетки очень помогает организации поддержки политики MLS. В самом деле, пусть имеется последовательная цепочка информационных потоков

$$O_1 \xrightarrow[\alpha]{} O_2 \xrightarrow[\beta]{} O_3 \xrightarrow[\gamma]{} \dots \xrightarrow[\delta]{} O_k$$

Если каждый из потоков разрешен, то свойства решетки позволяют утверждать, что разрешен сквозной поток $O_1 \xrightarrow[\alpha]{} \dots \xrightarrow[\delta]{} O_k$. Действительно, если информационный поток на каждом шаге разрешен, то $c(O_{i+1}) \geq c(O_i)$, тогда по свойству транзитивности решетки $c(O_1) \leq c(O_k)$, то есть сквозной поток разрешен.

MLS политика в современных системах защиты реализуется через мандатный контроль (или, также говорят, через мандатную политику). Мандатный контроль реализуется подсистемой защиты на самом низком аппаратно-программном уровне, что позволяет эффективно строить защищенную среду для механизма мандатного контроля. Устройство мандатного контроля, удовлетворяющее некоторым дополнительным, кроме перечисленных, требованиям, называется монитором обращений. Мандатный контроль еще называют обязательным, так как его проходит каждое обращение субъекта к объекту, если субъект и объект находятся под защитой системы безопасности. Организуется мандатный контроль следующим образом. Каждый объект O имеет метку с информацией о классе c(O). Каждый субъект также имеет метку, содержащую информацию о том, какой класс доступа c(S) он имеет. Мандатный контроль сравнивает метки и удовлетворяет запрос субъекта S к объекту O на чтение, если c(S)>c(0) и удовлетворяет запрос на запись, если c(S)<c(0). Тогда согласно изложенному выше мандатный контроль реализует политику MLS.

Политика MLS устойчива к атакам "Троянским конем". На чем строится защита от таких атак поясним на примере, являющимся продолжением примера1 из параграфа 3.2.

Пример 1. Пусть пользователи U_1 и U_2 находятся на разных уровнях, то есть $c(U_1)>c(U_2)$. Тогда, если U_1 может поместить в объект O_1 ценную информацию, то он может писать туда и $c(U_2)<c(U_1)<c(0_1)$, то есть $c(U_2)<c(0_1)$. Тогда любой "Троянский конь" T, содержащийся в объекте O_2, который может считать информацию в O_1, должен отражать соотношение

$$c(O_2) \geq c(O_1).$$

Тогда $c(O_2)>c(U_2)$ и пользователь U_2 не имеет право прочитать в O_2, что делает съем в O_1 и запись в O_2 бессмысленным.

Несколько слов о реализации политики безопасности MLS в рамках других структур, внесенных в информацию. Опять обратимся к примеру реляционной базы данных. Пусть структура РМ и структура решетки ценностей MLS согласованы, как это было сделано в параграфе 1.6. Пусть в системе реализован мандатный контроль, который при обращении пользователя U к базе данных на чтение позволяет извлекать и формировать "обзор" только такой информации, класс которой <c(U). Процедура генерации такого "обзора" была описана в параграфе 1.6. Аналогично, мандатный контроль и правила декомпозиции позволяют поддерживать в нужном направлении информационные потоки в процессе функционирования базы данных. В результате получаем, что при наличии мандатного контроля построенная в 1.6 реляционная многоуровневая база данных поддерживает политику MLS.

Политика MLS создана, в основном, для сохранения секретности информации. Вопросы целостности при помощи этой политики не решаются или решаются как побочный результат защиты секретности. Вместе с тем, пример 2 параграфа 3.1 показывает, что они могут быть противоречивы.

Пример 2. (Политика целостности Biba). Предположим, что опасности для нарушения секретности не существует, а единственная цель политики безопасности - защита от нарушений целостности информации. Пусть, по-прежнему, в информацию внесена решетка ценностей SC. В этой связи любой информационный поток X -> Y может воздействовать на целостность объекта Y и совершенно не воздействовать на целостность источника X. Если в Y более ценная информация, чем в X, то такой поток при нарушении целостности Y принесет более ощутимый ущерб, чем поток в обратном направлении от более ценного объекта Y к менее ценному X. Biba предложил в качестве политики безопасности для защиты целостности следующее.

Определение. В политике Biba информационныйпоток X $-->_{\alpha}$Y разрешен тогда и только тогда, когда

$$c(Y) \le c(X)$$

Можно показать, что в широком классе систем эта политика эквивалентна следующей.

Определение. Для систем с доступами w и r политика Biba разрешает доступ в следующих случаях:

$$S \xrightarrow{r} O \Leftrightarrow c(S) \le c(O),$$

$$S \xrightarrow{w} O \Leftrightarrow c(S) \ge c(O).$$

Очевидно, что для реализации этой политики также подходит мандатный контроль.

Глава 4
Классификация систем защиты

В главе IV мы рассмотрим вопросы, связанные с оценкой возможностей системы защиты поддерживать политику безопасности. В параграфе 4.1 в общих чертах изложен метод анализа систем поддержки политики. В параграфе 4.2 построен пример гарантированно защищенной системы. В параграфе 4.3 изложены требования к системам защиты, поддерживающим дискреционную и MLS политики, сформулированные в американском стандарте защиты "Оранжевая книга". В этом же параграфе кратко излагается классификация систем защиты так, как она принята в указанном стандарте. В параграфе 4.4 кратко изложены принципы выбора класса защиты согласно классификации "Оранжевой книги".

4.1. ДОКАЗАТЕЛЬНЫЙ ПОДХОД К СИСТЕМАМ ЗАЩИТЫ . СИСТЕМЫ ГАРАНТИРОВАННОЙ ЗАЩИТЫ.

Пусть задана политика безопасности **P**. Тогда система защиты - хорошая, если она надежно поддерживает **P**, и - плохая, если она ненадежно поддерживает **P**. Однако надежность поддержки тоже надо точно определить. Здесь снова обратимся к иерархической схеме. Пусть политика P выражена на языке $\boldsymbol{Я}_1$, формулы которого определяются через услуги $U_1, \ldots, U_k$.

Пример 1. Все субъекты **S** системы разбиты на два множества $\mathbf{S}_1$ и $\mathbf{S}_2$, $\mathbf{S}_1 \cup \mathbf{S}_2 = \mathbf{S}$, $\mathbf{S}_1 \cap \mathbf{S}_2 = \varnothing$. Все объекты, к которым может быть осуществлен доступ, разделены на два класса $\mathbf{O}_1$ и $\mathbf{O}_2$ $\mathbf{O}_1 \cup \mathbf{O}_2 = \mathbf{O}$, $\mathbf{O}_1 \cap \mathbf{O}_2 = \varnothing$. Политика безопасности **P** -тривиальная: субъект S может иметь доступ $\alpha \in \mathbf{R}$ к объекту **O** тогда и только тогда, когда $S \in S_i$, $O \in O_i$, i = 1, 2. Для каждого обращения субъекта S на доступ к объекту O система защиты вычисляет функции принадлежности

$$I_x = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases}$$

для субъекта и объекта: $I_s(\mathbf{S}_1)$, $I_s(\mathbf{S}_2)$, $I_0(\mathbf{O}_1)$, $I_0(\mathbf{O}_2)$. Затем вычисляется логическое выражение:

$$(I_s(\mathbf{S}_1) \wedge I_0(\mathbf{O}_1)) \vee (I_s(\mathbf{S}_2) \wedge I_0(\mathbf{O}_2)).$$

Если полученное значение - 1 (истинно), то доступ разрешен. Если - 0 (ложно), то - неразрешен. Ясно, что язык $\boldsymbol{Я}_1$, на котором мы выразили политику безопасности **P**, опирается на услуги:

- вычисление функций принадлежности $I_x(A)$;
- вычисление логического выражения;
- вычисление оператора "если x=1, то S ->0, если x=0, то S -+>O".

Для поддержки услуг языку $\boldsymbol{Я}_1$, требуется свой язык $\boldsymbol{Я}_2$, на котором мы определим основные выражения для предоставления услуг языку верхнего уровня. Возможно, что функции $\boldsymbol{Я}_2$ необходимо реализоватьопираясь на язык $\boldsymbol{Я}_3$ более низкого уровня и т.д.

Пусть услуги, описанные на языке $\boldsymbol{Я}_2$ мы умеем гарантировать. Тогда надежность выполнения политики **Р** определяется полнотой ее описания в терминах услуг $U_1,...,U_k$. Если модель **Р** - формальная, то есть язык $\boldsymbol{Я}_1$, формально определяет правила политики **Р**, то можно доказать или опровергнуть утверждение, что множество предоставленных услуг полностью и однозначно определяет политику **Р**. Гарантии выполнения этих услуг равносильны гарантиям соблюдения политики. Тогда более сложная задача сводится к более простым и к доказательству того факта, что этих услуг достаточно для выполнения политики. Все это обеспечивает доказанность защиты с точки зрения математики, или гарантированность с точки зрения уверенности в поддержке политики со стороны более простых функций.

Одновременно, изложенный подход представляет метод анализа систем защиты, позволяющий выявлять слабости в проектируемых или уже существующих системах. При этом иерархия языков может быть неоднозначной, главное - удобство представления и анализа.

Однако проводить подобный анализ в каждой системе дорого. Кроме того, методика проведения анализа государственных систем - конфиденциальная информация. Выход был найден в том, что условия теорем, доказывающих поддержку политики безопасности (включая соответствующую политику), формулировать без доказательства в виде стандарта. Такой подход американцы впервые применили в 1983 году, опубликовав открыто проект стандарта по защите информации в ЭСОД ("Оранжевая книга"), где сформулированы требования гарантированной поддержки двух классов политик - дискреционной и политики MLS. Затем этот метод они применили в 1987 г. для описания гарантированно защищенных распределенных сетей, поддерживающих те же политики, и в 1991 г.для описания требований гарантированно защищенных баз данных. Этот же путь использовали канадцы и европейские государства, создав свои стандарты защиты.

4.2. ПРИМЕР ГАРАНТИРОВАННО ЗАЩИЩЕННОЙ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ.

Построим пример гарантированно защищенной системы обработки информации. Сначала определим модель Σ системы, которая оперирует с ценной информацией. Считаем, что время дискретно и принимает значения из множества $N=\{1, 2,... \}$, информация в системе Σ, включая описание самой системы, представима в форме слов некоторого гипотетического языка ***Я*** над некоторым конечным алфавитом А. Напомним, что объект в Σ - это конечное множество слов из **Я**, состояние объекта - выделенное слово из множества, определяющего этот объект. С понятием объекта связано агрегирование информации в Σ и о Σ. Например, объектом является принтер, который можно рассматривать как автомат с конечным множеством состояний, а эти состояния - суть слова языка ***Я***. Другой пример объекта - файл. Множество слов, которые могут быть записаны в файле, является конечным и определяет объект, а состояния объекта -это текущая запись в файле, которая тоже является словом в языке *Я.*

Принято считать, что вся информация о Σ в данный момент может быть представлена в виде состояний конечного множества объектов. Поэтому будем считать, что состояние системы Σ - это набор состояний ее объектов. Объекты могут создаваться и уничтожаться, поэтому можно говорить о множестве объектов системы Σ в момент t, которое мы будем обозначать O_t, $|O_t|<\infty$.

Для каждого $t\in N$ выделим в O_t подмножество S_t субъектов. Любой субъект $S\in S_t$ есть описание некоторого преобразования информации в системе Σ. Для реализации этого преобразования в Σ необходимо выделить определенные ресурсы (домен) и организовать определенное взаимодействие ресурсов, приводящее к преобразованию информации, которое назовем процессом. Тогда каждый субъект может находиться в двух состояниях: в

форме описания, в котором субъект называется неактивизированным, и в форме (домен, процесс), в которой субъект называется активизированным.

Активизировать субъект может только другой активизированный субъект. Для каждого t∈N на множестве S_t можно определить орграф Γ_t, где S_1 и S_2 из S_t соединены дугой S_1->S_2 тогда и только тогда, когда в случае активизации S_1 возможна активизация S_2. Если субъект S - такой, что для каждого Γ_t в вершину S не входит дуг, то такой субъект будем называть пользователем. Для простоты положим, что в системе S всего два пользователя: U_1 и U_2. Пользователи считаются активизированными по определению и могут активизировать другие субъекты.

Если в любой момент t в графе Γ_t в вершину S не входят дуги и не выходят дуги, то такие субъекты исключаем из рассмотрения.

Обозначим $S_1 \xrightarrow{a} S_2 \quad S_1, S_2 \in S_t$, процедуру активизации процессом S_1 субъекта S_2.

Предположение 1. Если субъект S активизирован в момент t, то существует единственный активизированный субъект S' в $\mathbf{S}_t$, который активизировал S. В момент t=0 активизированы только пользователи.

Лемма. 1. *Если в данный момент t активизирован субъект S, то существует единственный пользователь U, от имени которого активизирован субъект S, то есть существует цепочка*

$$U \xrightarrow{a} S_1 \xrightarrow{a} S_2 \xrightarrow{a} \ldots \xrightarrow{a} S_k \xrightarrow{a} S$$

Доказательство. Согласно предположению 1 существует единственный субъект S_k, активизирующий S. Если $S_k=U_1$, или $S_k=U_2$, то лемма доказана. Если $S_k \neq U_i$, i=1, 2, то существует единственный субъект S_{k-1}, активизировавший S_k. В силу конечности времени работы системы S и того факта, что в начальный момент активизированы могут быть только пользователи, получаем в начале цепочки одного из них. На этом, согласно определению пользователей, цепочка обрывается. Лемма доказана.

Предположение 1 требует единственности идентификации субъектов. Далее будем предполагать, что каждый объект в системе имеет уникальное имя.

Кроме активизации в системе Σ существуют и другие виды доступа активизированных субъектов к объектам. Будем обозначать множество всех видов доступов через **R** и считать |R|<∞. Если p⊆**R**, то будем обозначать множество доступов p активизированного субъекта S к объекту O через $S \xrightarrow{p} P$. Если в некоторый промежуток времени [t, t+k] реализована последовательность доступов

$$U \xrightarrow{a} S_1 \xrightarrow{a} S_2 \xrightarrow{a} \ldots \xrightarrow{a} S_k \xrightarrow{p} S$$

то будем считать, что произошел доступ $S \xrightarrow{p}{}^{*} O$ от имени субъекта S к объекту O. Нас не интересует, какую задачу решает система Σ, мы лишь моделируем функционирование системы последовательностью доступов.

Предположение 2. Функционирование системы Σ описывается последовательностью доступов множеств субъектов к множествам объектов в каждый момент времени t∈N.

Обобщим введенный орграф Γ_t добавив дуги S ->O, обозначающие возможность любого доступа субъекта S к объекту O в момент t, в случае активизации S. Обозначим $D_t(S)$ = {O | S -->*0 в момент t}, гдеS->*O означает возможность осуществления цепочки доступов S ->S_1->S_2->...S_k->O (возможность доступа к O от имени S). Тогда для любого t∈N в системе определены $D_t(U_1)$ и $D_t(U_2)$. Будем считать, что **D**= $D_t(U_1)\cap D_t(U_2)$ фиксировано для всех t, O_t= $D_t(U_1)\cup D_t(U_2)$, в начальный момент t = 0: $\mathbf{O}_0$={U1,U2}∪**D**

Определение. Множество объектов O называется общими ресурсами системы.

В частности, средствами из **D** пользователь может создавать объекты и уничтожать объекты, не принадлежащие **D**. Создание и уничтожение каких-либо объектов является доступом в **R** к некоторым объектам из **O** (и к уничтожаемым объектам).

Мы считаем, что из объектов системы Σ построена некоторая подсистема, которая реализует доступы. Будем полагать, что любое обращение субъекта S за доступом p к объекту O в эту подсистему начинается с запроса, который мы будем обозначать $S \xrightarrow{p?} O$.

При порождении объекта субъект S обращается к соответствующей процедуре, в результате которой создается объект с уникальным именем. Тогда в силу леммы 1, существует единственный пользователь, от имени которого активизирован субъект, создавший этот объект. Будем говорить, что соответствующий пользователь породил данный объект. Обозначим через $O_t(U)$ множество объектов из $\mathbf{O}_t$, которые породил пользователь U. Естественно, будем считать, чтоU∈O_t(U).

Лемма 2. Для каждого t∈N, для каждого O∈$\mathbf{O}_t$, O∉**D**, существует единственный пользователь U такой, что O∈O_t(U) .

Доказательство. Поскольку $\mathbf{O}_0$={U_1, U_2}∪D, то объект O∉**D**, O∈O_t, порожден в некоторый момент s, 0<s<t. Тогда в O существовал активизированный субъект S, создавший O. Тогда, как отмечено ранее, существует единственный пользователь U, породивший O. Лемма доказана.

Обратимся теперь к вопросам безопасности информации в системе. Если в **R** есть доступы read и write, то ограничимся опасностью утечки информации через каналы по памяти, которые могут возникнуть при доступах к объектам. Таким каналом может бытьследующая последовательность доступов при s<t

$U_i \xrightarrow{w} O$ в момент s и $U_j \xrightarrow{r} O$ в момент t, $i \neq j$

При определенных условиях может оказаться опасным доступ от имени пользователя:

$U_i \xrightarrow{w} *O$ в момент s и $U_j \xrightarrow{r} *O$ в момент t, s<t, $i \neq j$

С некоторой избыточностью мы исчерпаем возможные каналы по памяти, если будем считать неблагоприятными какие-либо доступы $p_1, p_2 \subseteq \mathbf{R}$ вида

$U_i \xrightarrow{p1} *O$, $U_j \xrightarrow{p2} *O$, $i \neq j$ (1)

которые мы и считаем каналами утечки.

Предположение 3. Если O∈**D**, то доступы в (1) при любых p_1 и p_2 не могут создать канал утечки.

Это значит, что мы предположили невозможным отразить какую-либо ценную информацию в объектах общего доступа. Это очень сильное предположение и оно противоречит, по крайней мере, возможности присваивать уникальные имена объектам. В самом деле, если объектам присвоены уникальные имена, то в **D** необходимо иметь информацию о уже присвоенных именах, что противоречит предположению 3 о том, что доступы от имени пользователей не отражаются в информации объектов общего пользования. В случае имен можно выйти из положения, используя случайные векторы, вероятности совпадения которых за обозримый период работы системы можно сделать как угодно малыми. Все построения и выводы возможны при стохастическом способе присвоения имен, но должны содержать элементы вероятностной конструкции. Чтобы не усложнять систему стохастическими элементами мы будем следовать сделанным выше предположениям.

Тогда в (1) достаточно ограничиться объектами O, не лежащими в **D**. Это значит, что в одном из доступов в (1) имеется $U_i \xrightarrow{pk} *O$, где $O \in O_t(U_j)$, $i \neq j$. Таким образом, в системе считаются неблагоприятными доступы вида:

$$\exists t, \exists p \subseteq R, p \neq \varnothing, \exists U_i, \exists O \in O_t,$$

$$U_i \xrightarrow{p} *O, \quad O \in O_t(U_j), \quad i \neq j. \qquad (2)$$

то есть доступы от имени какого-либо пользователя к объекту, созданному другим пользователем. Такие доступы будем далее называть утечкой информации.

Предположение 4. Если некоторый субъект S, S∈**D**, активизирован от имени пользователя U_i (т.е. $U_i \xrightarrow{a} *S$), в свою очередь субъекту S предоставлены в момент t доступ к объекту O, то либо O∈**D**, либо $O \in O_t(U_i)$, либо система прекращает работу и выключается. Определим следующую политику безопасности (ПБ):

Если $S \xrightarrow{p?} O$, то при $S, O \in O_t(U)$ доступ $S \xrightarrow{p} O$ разрешается, если $S \in O_t(U_i)$, $O \in O_t(U_j)$, $i \neq j$, то доступ $S \xrightarrow{p} O$ невозможен.

Теорема 1. *Пусть в построенной системе выполняются предположения 1-4. Если все доступы осуществляются в соответствии с ПБ, то утечка информации (2) невозможна.*

Доказательство. Предположим противное, то есть

$$\exists t, \exists p \subseteq R, p \neq \varnothing, \exists U_i, \exists O \in O_t, U_i \xrightarrow{p} *O, O \in O_t(U_j), i \neq j$$

Пусть $S_1, \ldots, S_m$ - все активизированные субъекты, имеющие доступы $\beta \supseteq p$, $i=1,\ldots,m$, в момент t к объекту O. Тогда согласно лемме 2 множество этих субъектов разбивается на три непересекающиеся множества:

$$A = \{S_1 | S_1 \in D\},$$
$$B = \{S_1 | S_1 \in O_t(U_i)\},$$
$$C = \{S_1 | S_1 \in O_t(U_j), i \neq j\}.$$

Согласно лемме 1 для любого S_1, l=1.....m, существует единственный пользователь, от имени которого активизирован субъект S_1. Если $S_1 \in A$, то согласно предположению 4 и условию теоремы 1, что доступ $S_l \xrightarrow{\beta_\varepsilon} O$ - разрешен, получаем, что S, активизирован от имени U_j. Это противоречит предположению.

Если $S_1 \in B$, то $S_l \xrightarrow{\beta_\varepsilon} O$ невозможен согласно политике безопасности. Значит, если $U_i \xrightarrow{p} *O$, то существует цепочка длины (k+1)

$$U_i \xrightarrow{p1} S^{(1)} \xrightarrow{p2} S^{(2)} \xrightarrow{p3} \dots \xrightarrow{p4} S^{(k)} \xrightarrow{p} O,$$

и субъект $S^{(k)} \in C$. Тогда существует цепочка длины k такая, что

$$U_i \xrightarrow{a} *O', O' \in O_{t-1}(U_j), i \neq j, \alpha \subseteq R.$$

Повторяя эти рассуждения, через k шагов получим, что

$$U_i \xrightarrow{\beta} *O'', O'' \in O_{t-k}(U_j), i \neq j, \beta \subseteq R.$$

Последний доступ невозможен, если выполняется ПБ. Поэтому предположение неверно и теорема 1 доказана.

Теперь построим удобное для реализации множество "услуг" более низкого уровня, поддерживающих ПБ. То есть мы хотим определить множество условий, реализованных в системе 2, таких, что можно доказать теорему о достаточности выполнения этих условий для выполнения правил ПБ.

Условие 1. (Идентификация и аутентификация). Если для любых $t \in \mathbf{N}$, $p \subseteq \mathbf{R}$, S, $O \in \mathbf{O}_t$, $S \xrightarrow{p?} O$, то вычислены функции принадлежности S и O к множествам $O_t(U_1)$, $O_t(U_2)$, **D**.

Условие 2. (Разрешительная подсистема). Если $S \in O_t(U_i)$, $O \in O_t(U_j)$ и $S \xrightarrow{p?} O$ в момент t, то из i=j следует $S \xrightarrow{p} O$, и из i≠j следует $S \xrightarrow{p} O$ (не разрешается доступ).

Условие 3. (Отсутствие обходных путей политики безопасности). Для любых $t \in \mathbf{N}$, $p \subseteq \mathbf{R}$, если субъект S, активизированный к моменту t, получил в момент t доступ $S \xrightarrow{p} O$, то в момент t произошел запрос на доступ $S \xrightarrow{p?} O$.

Теорема 2. *Если в построенной системе Σ выполняются предположения 1-4 и условия 1-3, то выполняется политика безопасности.*

Доказательство. Утверждение теоремы следует из двух утверждений:

а) Если для произвольного $p \subseteq \mathbf{R}$ $S \xrightarrow{p?} O$, $S \in O_t(U_i)$, $O \in O_t(U_i)$, то доступ $S \xrightarrow{p} O$ разрешен.

б) Если $S \in O_t(U_i)$, $O \in O_t(U_j)$, i≠j, то какой-либо доступ в момент t субъекта S к объекту O невозможен.

Докажем а). Если $S \xrightarrow{p?} O$, то по условию 1 вычислены функции принадлежности и определено, что $S \in O_t(U_i)$, $O \in O_t(U_j)$. Если i = j, то выполнена посылка условия 2. Тогда согласно условию 2 доступ $S \xrightarrow{p} O$ разрешен.

Докажем теперь б). Если $S \xrightarrow{p?} O$, вычислены функции принадлежности и определено, что $S \in O_t(U_i)$, $O \in O_t(U_j)$, i≠j. Тогда по условию 2 доступ $S \xrightarrow{p} O$ не разрешен.

Если доступ $S \xrightarrow{p} O$ стал возможен, минуя запрос $S \xrightarrow{p?} O$, и S - активизирован к моменту t, то сделанное предположение противоречит условию 3. Если S - не активизирован, то наличие доступа $S \xrightarrow{p} O$ противоречит определению доступа. Теорема доказана.

Теорема означает, что гарантировав выполнение условий 1-3, мы гарантируем выполнение политики безопасности.

Рассмотрим вопрос о создании системы, в которой можно с достаточной степенью уверенности поддерживать функции 1-3. Для этого рассмотрим следующую архитектуру:

1. В каждый момент только один пользователь может работать с системой. Физическое присутствие другого исключено.
2. При смене пользователей системы друг другом уходящий :

- записывает во внешнюю память все объекты, которые он хочет сохранить для дальнейших сеансов;

- выключает питание системы, после чего все содержимое оперативной памяти стирается, остаются записи на внешней памяти и ПЗУ, где хранятся объекты общего доступа.

3. Новый пользователь организует свою работу с включения системы и вызывает свои объекты из внешней памяти, опираясь на объекты общего пользования.
4. На шлюзе внешней памяти стоит шифратор К, который зашифровывает на текущем ключе k всю информацию, записываемую на внешнюю память, включая названия файлов. Наоборот, вся информация, поступающая из внешней памяти, расшифровывается на текущем ключе k. Внешняя память не имеет опции "просмотр директории", а любой запрос на выдачу файла функционирует так, что название запрашиваемого файла шифруется на текущем ключе k. При смене пользователей текущий ключ k автоматически стирается (вместе с содержимым оперативной памяти), а новый пользователь в качестве текущего устанавливает свой ключ.

Покажем, что функционирование системы данной архитектуры позволяет реализовать все описанные выше свойства и, в частности, выполнить условия теорем 1 и 2.

Предположение 1 и другие допущения в описании системы вполне приемлемы для рассматриваемой архитектуры. Так как неблагоприятные состояния системы и политики безопасности выражены в терминах доступов, то для приемлемости предположения 2 достаточно, чтобы возможные вычислительные процессы однозначно отражались в терминах последовательностей доступов и значений функций принадлежности объектов к множествам $O_t(U_1)$, $O_t(U_2)$,**D**. Если объект только создан и находится в оперативной памяти, то доступ к нему со стороны процессов от имени создавшего пользователя автоматически разрешен и можно считать, что функция принадлежности вычислена. Если объект вызван из внешней памяти, то сам вызов и доступ к информации в объекте возможны, если установлен правильный ключ, что эквивалентно вычислению функции принадлежности к $O_t(U)$. Предположения 3 и 4 выполняются, так как вновь подключенный пользователь работает один и вызывает из ПЗУ функции и объекты **D**. В системе нет субъекта, реализующего разрешительную систему, она естественно реализована за счет того, что расшифрованная информация читается тогда и только тогда, когда в шифраторе К установлен нужный ключ. Если пользователь или процесс от его имени обращается за доступом к объекту на внешней памяти, то любой доступ разрешен, если ключ зашифрования объекта (ключ создателя объекта) совпадает с ключом текущего пользователя. Наоборот, при несовпадении ключей допуск автоматически не разрешается, так как имя объекта и его содержание не расшифровываются правильно.

Таким образом, автоматически вычисляются функции принадлежности процесса и объекта при обращении через внешнюю память, что обеспечивает выполнение условия 1. Также автоматически выполняется условие 2 о работе разрешительной системы. Условия 1 и 2 не касаются обращений процессов из **D** и к **D**. Поэтому вопросы идентификации здесь решаются за счет разделения сеансов пользователей и указанные условия выполняются.

Доступ к объекту возможен лишь при обращении к внешней памяти через шифратор, или в случае, когда объект создан в течение текущего сеанса, или вызван из ПЗУ. Если считать, что доступ к объектам в оперативной памяти автоматически опирается на данное пользователю разрешение на доступ к ним, а активизированными могут быть субъекты от его же имени, то можно считать, что любой доступ в этом случае выполняется в соответствии с условиями 1 и 2.

Что касается условия 3, то невозможность получить доступ минуя разрешительную систему определяется разнесенностью работы пользователей, отсутствием подслушивания, необходимостью расшифровывать информацию для получения доступа к ней. Это не касается объектов из **D**, или только созданных, где нет проблем из-за разнесенности сеансов. Также мы считаем, что отсутствует физическое проникновениеи модификация системы.

В результате получим, что данная архитектура реализует условия теорем 1 и 2 и поддерживает политику безопасности.

Суммируем то, что обеспечивает гарантии в построенной системе (то есть, что поддерживает условия теорем 1 и 2):

- обеспечение работы только одного пользователя (охрана);
- отключение питания при смене пользователей;
- стойкость шифратора К и сохранность в тайне ключей каждого пользователя;
- недопустимость физического проникновения в аппаратную часть или подслушивание (охрана).

Известно, как обеспечивать эти требования, а гарантии их обеспечения являются гарантией защищенности системы в смысле выбранной политики безопасности.

Отметим некоторые стороны построенной модели, которые будут встречаться далее.

1) Гарантии построены при четкой политике безопасности.

2) Для поддержки политики потребовалась идентификация объектов (+ вычисление добавочной идентификационной функции).

3) Для поддержки политики потребовалась аутентификация субъекта, обращающегося за допуском.

4) Значительная часть условий определяет функциональную защищенность самой системы защиты.

Далее в обзоре "Оранжевой книги" мы встретим такие же условия.

4.3. "ОРАНЖЕВАЯ КНИГА"(ОК).

ОК принята стандартом в 1985 г. Министерством обороны США (DOD). Полное название документа "Department of Defense Trusted Computer System Evaluation Criteria".

ОК предназначается для следующих целей:

1. Предоставить производителям стандарт, устанавливающий, какими средствами безопасности следует оснащать свои новые и планируемые продукты, чтобы поставлять на рынок доступные системы, удовлетворяющие требованиям гарантированной защищенности (имея в виду, прежде всего, защиту от раскрытия данных) для использования при обработке ценной информации.

2. Предоставить DOD метрику для военной приемкии оценки защищенности ЭСОД, предназначенных для обработки служебной и другой ценной информации.

3. Обеспечить базу для исследования требований к выбору защищенных систем.

Рассматривают два типа оценки:

- без учета среды, в которой работает техника,
- в конкретной среде (эта процедура называется аттестованием).

В 1992 году Гостехкомиссия России издала от своего имени документы, аналогичные по задачам и содержанию ОК.

Во всех документах DOD, связанных с ОК, принято одно понимание фразы обеспечение безопасности информации. Это понимание принимается как аксиома и формулируется следующим образом: безопасность= контроль за доступом.

Аксиома. *ЭСОД называется безопасной, если она обеспечивает, контроль за доступом информации так, что только надлежащим образом уполномоченные лица или процессы, которые функционируют от их имени, имеют право читать, писать, создавать или уничтожать информацию.*

Из этой аксиомы вытекает шесть фундаментальных требований к защищенным ЭСОД. Прежде, чем их формулировать, напомним и введем некоторые определения.

Определение. Политика безопасности - это набор норм, правил и практических приемов, которые регулируют управление, защиту и распределение ценной информации в данной организации.

Определение. Идентификация - это распознавание имени объекта. Идентифицируемый объект есть однозначно распознаваемый.

Определение. Аутентификация это подтверждение того, что предъявленное имя соответствует объекту.

Определение. TCB (Trusted Computing Base) - совокупность механизмов защиты в вычислительной системе (включая аппаратную и программную составляющие), которые отвечают за поддержку политики безопасности.

Определение. Аудит или отслеживание, подотчетность - это регистрация событий, позволяющая восстановить и доказать факт происшествия этих событий.

Теперь сформулируем 6 основных требований.

ПОЛИТИКА.

Требование 1 - Политика обеспечения безопасности - Необходимо иметь явную и хорошо определенную политику обеспечения безопасности. При задании идентифицированных субъектов и объектов необходимо иметь набор правил, используемых системой, для того, чтобы определить, можно ли разрешить указанному субъекту доступ к конкретному объекту. Представленные на аттестацию для использования в DOD вычислительные системы должны предусматривать реализацию мандатного контроля обеспечения безопасности, в рамках которого допускается эффективная реализация правил доступа, ориентированных на обработку конфиденциальной (например, секретной) информации. Эти правила включают такие требования: ни одно лицо, не обладающее надлежащим для допущенного персонала диапазоном полномочий, не получит доступа к секретной информации. Кроме того, также необходимы дискреционные (т.е. допускаемые по собственному усмотрению) средства управления безопасностью, которые гарантируют, что только выбранные пользователи или группы пользователей могут получить доступ к данным (реализация принципа "только те, которым необходимо знать").

Требование 2 - Маркировка - Метки, управляющие доступом, должны быть установлены и связаны с объектами. Для того, чтобы управлять доступом к информации в соответствии с правилами мандатной политики, должна быть предусмотрена возможность маркировать каждый объект меткой, которая надежно идентифицирует степень ценности объекта (например, секретности) и/или режимы допуска, предоставленные тем субъектам, которые потенциально могут запросить доступ к объекту.

ПОДОТЧЕТНОСТЬ.

Требование 3 - Идентификация - Субъекты индивидуально должны быть идентифицированы. Каждый доступ к информации должен быть рассмотрен на

предмет того, кто запрашивает доступ к информации и на какие классы информации он имеет право получить доступ. Такого типа идентифицирующая и санкционирующая информация должна присутствовать и надежно защищаться вычислительной системой, а также быть связана с каждым активным элементом, выполняющим действия, имеющие отношениек безопасности системы.

Требование 4 - Подотчетность - Аудиторская информация должна селективно храниться и защищаться так, чтобы со стороны ответственной за это группы имелась возможность отслеживать действия, влияющие на безопасность. Гарантированно защищенная система должна быть в состоянии регистрировать в аудиторском файле появление событий, имеющих отношение к безопасности системы. Возможность отбирать подлежащие регистрации отслеживаемые события необходима для того, чтобы сократить издержки на аудит и обеспечить эффективный анализ. Аудиторская информация должна быть защищена от модификации, несанкционированного уничтожения, чтобы позволять выявлять и "постфактум" исследовать нарушения правил безопасности.

ГАРАНТИИ.

Требование 5 - Гарантии - Вычислительная система в своем составе обязана иметь аппаратно-программные механизмы, допускающие независимую оценку для получения достаточного уровня гарантий того, что система обеспечивает выполнение изложенных выше требований с первого по четвертое. Для того, чтобы гарантировать, что указанные четыре требования по безопасности - политика, маркировка, идентификация и аудит - реализуются вычислительной системой, необходимо предусмотреть корректно определенный и объединенный в единое целое набор программных и аппаратных средств управления, реализующий указанные функции. Указанные механизмы стандартным образом встраиваются в операционную систему и проектируются так, чтобы выполнить порученные задачи безопасным образом. Основа гарантированности таких системных механизмов при задании исходных рабочих условий должна быть явно задокументирована так, что становится возможным независимый анализ доказательства достаточности проведенных оценок.

Требование 6 - Постоянная защита - Гарантированно защищенные механизмы, реализующие указанные базовые требования, должны быть постоянно защищены от "взламывания" и/или несанкционированного внесения изменений. Никакая вычислительная система не может считаться действительно безопасной, если базовые аппаратные и программные механизмы, реализующие принятую политику, сами могут быть подвергнуты несанкционированному внесению изменений или исправлений. Выполнение требований по непрерывной защите подразумевается в течение всего жизненного цикла вычислительной системы.

ИТОГОВАЯ ИНФОРМАЦИЯ ПО КЛАССАМ КРИТЕРИЕВ ОЦЕНКИ.

Классы систем, распознаваемые при помощи критериев оценки гарантированно защищенных вычислительных систем, определяются следующим образом. Они представлены в порядке нарастания требований с точки зрения обеспечения безопасности ЭВМ.

Класс (D): Минимальная защита. Этот класс зарезервирован для тех систем, которые были подвергнуты оцениванию, но в которых не удалось достигнуть выполнения требований более высоких классов оценок.

Класс (C1): Защита, основанная на разграничении доступа (DAC).

Гарантированно защищающая вычислительная база (TCB) систем класса (C1) обеспечивает разделение пользователей и данных. Она включает средства управления, способные реализовать ограничения по доступу, чтобы защитить проект или частную информацию и не дать другим пользователям случайно считывать или разрушать их данные. Предполагается, что среда класса (C1) является такой средой, в которой могут кооперироваться пользователи, обрабатывающие данные, принадлежащие одному и тому же уровню секретности.

ПОЛИТИКА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ.

TCB должна определять и управлять доступом между поименованными пользователями и поименованными объектами (например, файлами и

программами) в системах автоматической обработки данных. Реализующий политику механизм (например, матрица доступа) должен позволять пользователям определять порядок и управлять использованием объектов поименованными лицами или определенными группами, а также теми и другими совместно.

ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ.

ТСВ должна требовать от пользователей, чтобы те идентифицировали себя перед тем, как начинать выполнять какие-либо действия, в которых ТСВ предполагается быть посредником. Более того, ТСВ обязательно должна использовать один из механизмов защиты (например, пароли) для того, чтобы проверять подлинность идентификации пользователей (аутентификация). ТСВ должна защищать аутентификационные данные таким образом, чтобы доступ к ним со стороны пользователя, не имеющего на это полномочий, был невозможен.

ГАРАНТИИ НА ПРАВИЛЬНУЮ РАБОТУ СИСТЕМЫ.
АРХИТЕКТУРА СИСТЕМЫ.

ТСВ должна содержать домен, который должен обеспечивать ее собственную работу и защищать ее от внешнего воздействия или от внесения в нее несанкционированных изменений (например, от модификаций ее кодов или структур данных). Ресурсы, контролируемые ТСВ, могут составлять подмножество объектов ЭСОД.

ЦЕЛОСТНОСТЬ СИСТЕМЫ.

Должны быть предусмотрены аппаратные и/или программные средства, предназначенные для периодической проверки на правильность и корректность функционирования аппаратных и микропрограммных элементов ТСВ.

ГАРАНТИИ НА ЖИЗНЕННЫЙ ЦИКЛ.
ТЕСТИРОВАНИЕ ФУНКЦИЙ БЕЗОПАСНОСТИ.

Механизм защиты должен соответствовать тем нормам, которые отражены в документации.

ДОКУМЕНТАЦИЯ

1. Руководство пользователя по использованию средств обеспечения безопасности.

2. Руководство администратору системы на гарантированные средства защиты.

В руководстве, ориентированном на администратора системы автоматической обработки данных, должно содержаться описание мер предосторожности и полномочий, которые необходимо контролировать в процессе функционирования средства, имеющего отношение к обеспечению режима секретности.

3. Документация по тестам.

Разработчик системы должен предусмотреть для лиц, занятых оцениванием безопасности системы, документ, в котором дается описание плана тестирования, процедур тестирования, излагающих каким способом проверяются механизмы обеспечения безопасности, и где представлены итоговые результаты функционального тестирования механизмов обеспечения безопасности.

4. Документация по проекту. В наличии должна быть документация, в которой имеется описание основополагающих принципов защиты, выбранных изготовителем данного средства или системы, а также объяснение того, как эти принципы транслируются в гарантированно защищающую вычислительную базу. Если ТСВ составлена непосредственно из модулей, то должно быть дано описание интерфейсов между этими модулями.

Класс (С2): Защита, основанная на управляемом контроле доступом.

Все требования к классу (С1) переносятся на класс(С2). Кроме того, системы этого класса реализуют структурно более "тонкое" управление доступом, в сравнении с системами класса (С1), за счет дополнительных средств управления разграничением доступа и распространением прав, а также за счет системы регистрации событий (аудит), имеющих отношение к безопасности системы и разделению ресурсов. Специально вводится требование по "очищению" ресурсов системы при повторном использовании другими процессами.

Класс(BI): Мандатная защита, основанная на присваивании меток объектам и субъектам, находящимся под контролем ТСВ.

Требования для систем класса (B1) предполагают выполнение всех требований, какие были необходимы в классе (C2). Помимо этого, необходимо представить неформальное определение модели, на которой строится политика безопасности, присваивание меток данным и мандатное управление доступом поименованных субъектов к объектам. В системе необходимо иметь средство, которое позволяет точно и надежно присваивать метки экспортируемой информации.

Класс (B2): Структурированная защита. Все требования класса (B1) должны выполняться для системы класса (B2). В системах класса (B2) TCB основана на четко определенной и формально задокументированной модели, в которой управление доступом, распространяется теперь на все субъекты и объекты данной системы автоматической обработки данных. Помимо этого, должен быть проведен анализ,связанный с наличием побочных каналов утечек. Необходимо провести разбиение структуры TCB по элементам, критическим с точки зрения защиты, и некритическим, соответственно. Интерфейс TCB хорошо определен, а проект и реализация гарантированно защищающей вычислительной базы (TCB) выполнены так, что они позволяют проводить тщательное тестирование и полный анализ. Механизмы аутентификации усилены, управление защитой предусматривается в виде средств, предназначенных для администратора системы и для оператора, а на управление конфигурацией накладываются жесткие ограничения. Система относительно устойчива к попыткам проникновения в нее.

Класс (B3): Домены безопасности.

Все требования для систем класса (B2) включены в требования к системам класса (B3). TCB класса (B3) должна реализовывать концепцию монитора обращений (RM):

- RM гарантированно защищен от несанкционированных изменений, порчи и подделки;
- RM обрабатывает *все* обращения;
- RM прост для анализа и тестирования на предмет правильности выполнения обработки обращений (должна быть полная система тестов, причем полнота должна быть доказана).

С этой точки зрения структура TCB выбирается такой, чтобы исключить коды, не существенные для реализации принятой политики обеспечения безопасности, одновременно с проработкой в процессе проектирования и реализации TCB системных инженерно-технических аспектов, направленных на минимизацию ее сложности. Предусматривается введение администратора безопасности системы, механизмы контроля (аудит) расширены так, чтобы обеспечить обязательную сигнализацию о всех событиях, связанных с возможным нарушением установленных в системе правил безопасности. Обязательным также является наличие процедур, обеспечивающих восстановление работоспособности системы. Система данного класса в высшей степени устойчива относительно попыток проникновения в нее.

Класс (A1): Верифицированный проект.

Системы этого класса (A1) функционально эквивалентны системам класса (B3) в том отношении,
что в них не появляются какие-либо новые требования к политике обеспечения безопасности. Отличительной чертой систем данного класса является анализ TCB, основанный на формальной спецификации проекта и верификации TCB, а, в итоге, высокая степень уверенности в том, что гарантированно защищающая вычислительная база реализована правильно. Такого рода гарантия, по своей природе технологическая, начинается уже с формальной модели политики обеспечения безопасности и формальной спецификации проекта высокого уровня. Наряду с широким и глубоким анализом процесса проектирования и разработки TCB, который необходимо проводить для систем класса (A1), необходимо также выполнение более строгих мер по управлению конфигурацией и специальных процедур по безопасному размещению систем в местах их дислокации. Предусматривается введение администратора безопасности системы.

4.4. О ВЫБОРЕ КЛАССА ЗАЩИТЫ.

Продолжим изучение вопросов, связанных с американским стандартом защиты информации "Оранжевая книга". В предыдущем параграфе были определены семь классов систем защиты информации, причем требования в классах от C к A монотонно возрастали. Американцы не опубликовали детальный анализ риска, который определяет эти требования. Однако, одновременно со стандартом, был опубликован документ "Computer Security Requirements-Guidance for Applying the DoD Trusted Computer System Evaluation Criteria in Specific Environment" (далее будем называть его "Требования"), в котором изложен порядок выбора класса систем в различных условиях. Этот документ частично отражает результаты анализа риска, основания для выбора политики безопасности в связи с этими рисками и меры обеспечения гарантий соблюдения политики безопасности. Всюду далее предполагаем, что в информацию внесена MLS решетка ценностей. Выбор требуемого класса безопасности систем определяется следующими основными факторами, характеризующими условия работы системы.

1. Безопасность режима функционирования системы. Американцы различают 5 таких режимов:

a. Режим, в котором система постоянно обрабатывает ценную информацию одного класса в окружении, которое обеспечивает безопасность для работы с этим классом.

в. Режим особой секретности самой системы. Все пользователи и элементы системы имеют один класс и могут получить доступ к любой информации. Этот режим отличается от предыдущего тем, что здесь обрабатывается информация высших грифов секретности.

c. Многоуровневый режим, который позволяет системе обработку информации двух или более уровней секретности. Причем не все пользователи имеют допуск ко всем уровням обрабатываемой информации.

d. Контролирующий режим. Это многоуровневый режим обработки информации, при котором нет полной гарантии защищенности TCB. Это накладывает ограничения на допустимые классы ценной информации, подлежащей обработке.

e. Режим изолированной безопасности. Этот режим позволяет изолированно обрабатывать информацию различных классов или классифицированную и неклассифицированную информацию. Причем возможно, например, что безопасно обрабатывается только информация класса TS, а остальная информация не защищена вовсе.

2. Основой для выбора класса защиты является индекс риска. Он определяет минимальный требуемый класс.

Отобразим классы секретности в числа согласно таблице: U-0, C-1, S-2, TS-3. Эта таблица не совсем точно отражает соответствие из "Требований". Но здесь мы просто объясняем идею подхода. Определим R_{min} - минимальный уровень допуска пользователя в системе, и R_{max} – максимальный класс ценности информации, присутствующий в системе. В большинстве случаев индекс риска определяется по формуле:

$$\text{Risk Index}=R_{max}-R_{min}$$

Исключения касаются случая, когда $R_{\min} \geq R_{\max}$, тогда

$$\text{RiskIndex} = \begin{cases} 1, & \text{если есть категории, к которым кто-либо из пользователей не имеет доступа} \\ 0, & \text{в противном случае} \end{cases}$$

И также некоторые исключения есть в случае обработки TS-информации.

Пример 1. Если минимальный допуск пользователя в системе - C, а максимальный гриф обрабатываемой информации - S, то $R_{min}=2$, $R_{max}=3$. Тогда RiskIndex = 1 .

В результате учета всех ценностей и определения дополнительных классов у американцев получается восемь значений индекса риска от 0 до 7. Для этих значений индекса риска устанавливается следующее соответствие с минимальными требуемыми классами систем в случае, когда система функционирует во враждебном окружении.

RiskIndex	Безопасность режима	Минимальный класс по

	функционирования	классификации “Оранжевой книги”
0	a	нет обязательного минимума
0	b	B1
1	c, d, e	B2
2	c, d, e	B3
3	c, d	A1
4	c	A1
5	c	*
6	c	*
7	c	*

Символ * означает, что в момент издания книги (1985 г.) минимальные требования по защите информации при данном значении индекса риска выше достигнутого уровня технологии.

Если система функционирует в окружении, которое можно назвать "безопасным периметром", то требования к минимальным классам значительно ниже.

Глава 5

МАТЕМАТИЧЕСКИЕ МЕТОДЫ АНАЛИЗА ПОЛИТИКИ БЕЗОПАСНОСТИ

Доказательство того факта, что соблюдение политики безопасности обеспечивает то, что траектории вычислительного процесса не выйдут в неблагоприятное множество, проводится в рамках некоторой модели системы. В этой главе мы рассмотрим некоторые модели и приведем примеры результатов, которые доказываются в данной области. В параграфе 5.1 рассматривается модель распространения прав доступа в системе с дискреционной политикой безопасности. Параграф 5.2 посвящен описанию модели Белла-Лападула и доказательству теоремы BST (основной теоремы безопасности для многоуровневых систем). К сожалению, полное описание модели Белла-Лападула остается секретным и недоступным. Поэтому приведенный вариант модели и теоремы касаются случая, когда уровень объекта не меняется. В параграфе 5.3 приводится пример (модель Low WaterMark), когда уровень объекта может меняться. В параграфе 5.4 описан подход Гогена и Месгауэра в моделировании безопасных систем. Наконец, в параграфе 5.5 приводится пример модели нарушителя, который используется при анализе политики аудита в реляционных базах данных.

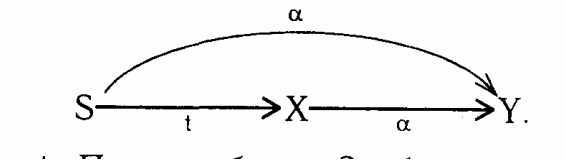
5.1. МОДЕЛЬ "TAKE-GRANT" .

Будем по-прежнему описывать функционирование системы при помощи графов доступов G_t и траекторий в фазовом пространстве $\varsigma=\{G\}$. Единственное дополнение - правила преобразования графов. Будем считать, что множество доступов R={r, w, c} , где r - читать, w - писать, c - вызывать. Допускается,что субъект X может иметь права $\alpha \subseteq R$ на доступ к объекту Y, эти права записываются в матрице контроля доступов. Кроме этих прав мы введем еще два: право take (t) и право grant (g), которые также записываются в матрицу контроля доступов субъекта к объектам. Можно считать, что эти права определяют возможности преобразования одних графов состояний в другие. Преобразование состояний, то есть преобразование графов доступов, проводятся при помощи команд. Существует 4 вида команд, по которым один граф доступа преобразуется в другой.

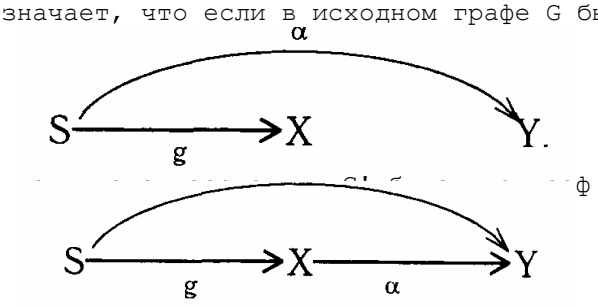
1. Take. Пусть S - субъект, обладающий правом t к объекту X и $\alpha \subseteq R$ некоторое право доступа объекта X к объекту Y. Тогда возможна команда "S take α forY from X". В результате выполнения этой команды в множество прав доступа субъекта S к объекту Y добавляется право α. Графически это означает, что, если в исходном графе доступов G был подграф

$$S \xrightarrow{t} X \xrightarrow{\alpha} Y$$

то в новом состоянии G', построенном по этой команде t, будет подграф



2. Grant. Пусть субъект S обладает правом *g* к объекту X и правом $\alpha \subseteq R$ к объекту Y. Тогда возможна команда "S grant α for Y to X". В результате выполнения этой команды граф доступов G преобразуется в новый граф G', который отличается от G добавленной дугой (X Y). Графически это означает, что если в исходном графе G был подграф



ф

3. Create. Пусть S - субъект, $\beta \subseteq R$. Команда "S create P for new object X" создает в графе новую вершину X и определяет P как права доступов S к X. То есть по сравнению с графом G в новом состоянии G' добавляется подграф вида

$$S \xrightarrow{\beta} X$$

4. Remove. Пусть S - субъект и X - объект, $\beta \subseteq R$. Команда "S remove P for X" исключает права доступа P из прав субъекта S к объекту X. Графически преобразования графа доступа G в новое состояние G' в результате этой команды можно изобразить следующим образом:

$$S \xrightarrow{p} X, S \xrightarrow{p/\beta} X$$

в G в G'

Далее будем обозначать $G|-_c G'$, если команда c преобразует G в G', а также G |- G', если существует команда c, что G|-G'. Будем понимать под безопасностью возможность или невозможность произвольной фиксированной вершине P получить доступ $\alpha \in R$ к произвольной фиксированной вершине X путем преобразования текущего графа G некоторой последовательностью команд в граф G', где указанный доступ разрешен.

Определение. В графе доступов G вершины P и S называются tg-связными, если существует путь в G, соединяющий P и S, безотносительно ориентации дуг, но такой, что каждое ребро этого пути имеет метку, включающую t или *g*.

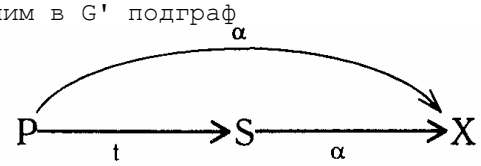
Теорема 1. *Субъект P может получить доступа к, объекту X, если существует субъект S, имеющий доступ a, к вершине X такой, что субъекты P и S связаны произвольно ориентированной дугой, содержащей хотя бы одно из прав t или g*

Доказательство. Возможны 4 случая.

1. В G есть подграф

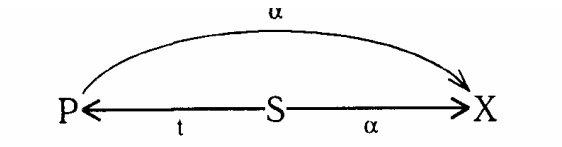
$$P \xrightarrow{t} S \xrightarrow{a} X$$

Тогда имеем право применить команду "P take α for X from S" и получим в G' подграф



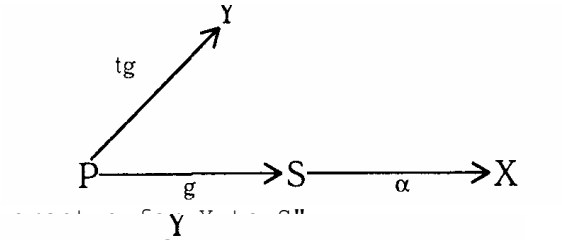
2. В G есть подграф $P \xleftarrow{g} S \xrightarrow{a} X$

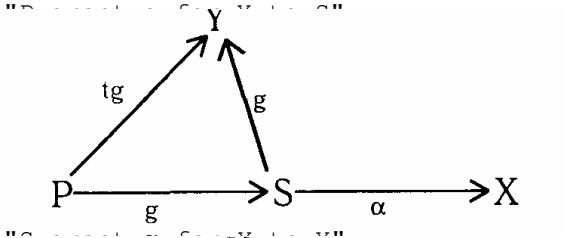
Тогда имеем право применить команду "S grant a for X to P" и получим в G' подграф

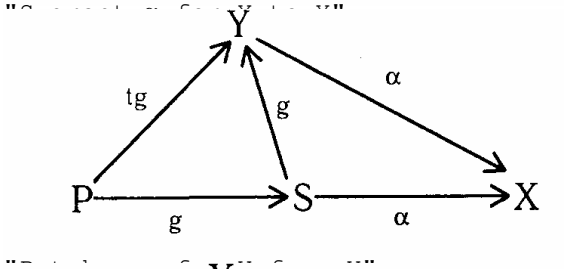


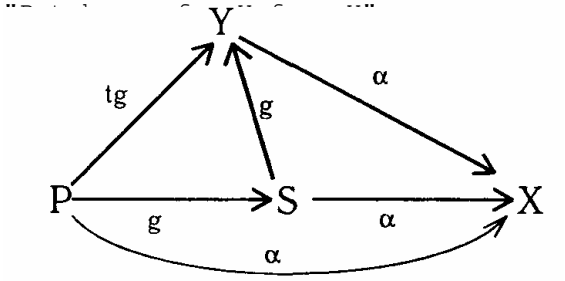
3. В графе G есть подграф $P \xrightarrow{g} S \xrightarrow{a} X$

Тогда применяем следующую последовательность разрешенных команд для преобразования графа G:"P create tg for new object Y"



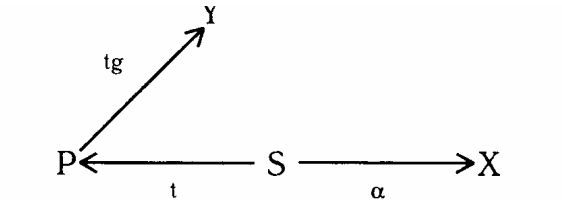




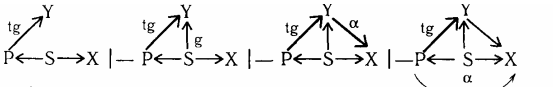


4. В графе G есть подграф $P \xleftarrow{t} S \xrightarrow{a} X$

Тогда применяем следующую последовательность разрешенных команд для преобразования графа G в граф G' с дугой (P X). "P create tg for new object Y"



Далее будем записывать преобразования графов коротко



Теорема доказана.

Замечание. Метка с правом a на дуге в рассматриваемых графах не означает, что не может быть других прав. Это сделано для удобства.

Теорема. 2. *Пусть в системе все объекты являются субъектами. Тогда субъект P может получить доступ a к субъекту X тогда и только тогда, когда выполняются условия:*

I. *Существует субъект S такой, что в текущем графеG есть дуга* $S \xrightarrow{\alpha} X$.

2. S tg-связна с P.

Доказательство. 1. Достаточность.

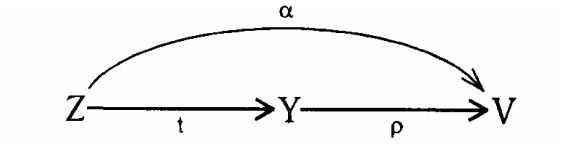
Доказательство будем вести индукцией по длине n tg-пути, соединяющего S и P. При n=1 утверждение доказано в теореме 1. Пусть длина tg-пути в G, соединяющего S и P равна n>1. Пусть также есть вершина Q на этом tg-пути, которая смежна с S. Тогда по теореме 1 можно перейти к графу G', в котором $Q \xrightarrow{\alpha} X$. Ясно, что проводимые при этом команды не уничтожают tg-пути, ведущего из P в Q. При этом длина пути из P в Q равна (n-1), что позволяет применить предположение индукции. Тогда возможен переход от G' к G", в котором есть дуга $P \xrightarrow{\alpha} X$. Сквозной переход от G к G’ доказывает достаточность.

2. Необходимость.

Пусть для пары вершин P и X в графе G нет дуги $P \xrightarrow{\alpha} X$, а после выполнения некоторой последовательности команд в графе G' есть дуга $P \xrightarrow{\alpha} X$. Если в G нет ни одной вершины S, для которой существует дуга $S \xrightarrow{\alpha} X$, то для любой команды c преобразования графа G в графе G’ полученном $G|-_c G'$ при помощи c, также нет ни одной вершины S, из которой выходит дуга $S \xrightarrow{\alpha} X$. Это следует из просмотра всех четырех допустимых команд. Тогда для любой последовательности команд в графе G’, полученном из G применением этой последовательности команд, также нет какой-нибудь вершины S с дугой $S \xrightarrow{\alpha} X$. Тогда такой вершины нет в графе G', что противоречит условию. Следовательно, в графе G есть S такая, что $S \xrightarrow{\alpha} X$.

Пусть G' такой граф, когда впервые появляется дуга $P \xrightarrow{\alpha} X$. Пусть G’ такой граф, из которого по некоторой команде получился G'. Тогда просмотр команд позволяет заключить, что дуга $P \xrightarrow{\alpha} X$ возникла применением к некоторому $S \xrightarrow{\alpha} X$, команды take или grant. Это значит, что в графе G’ от P к S существует tg-путь длины 1.

Пусть в графе G вершины P и S не связаны tg-путем. Тогда при любой команде c в графе G’, полученном из G командой c $G|-_c G'$, также нет tg-пути из P в S. В самом деле, возьмем take



Если в p не было take или grant, то новая дуга не увеличивает количество дуг с правом take или grant в новом графе, поэтому новый tg-путь возникнуть не может. Если в p есть t или g, то между V и Z существовал tg-путь и новая дуга не увеличила числа tg-связных вершин и поэтому не могла связать P и S. Аналогично, если Y и Z были связаны дугой grant. Команда create также не может связать существующие вершины P и S tg-путем.

Значит при любой последовательности команд $c_1,...c_n$, если в G нет tg-пути из P в S, то их нет в G′ полученном из G $G|-_{c1,...cn}G'$. Но это противоречит сделанному выше заключению о наличии такого пути длины 1 в графе G′. Теорема доказана.

5.2. МОДЕЛЬ БЕЛЛА-ЛАПАДУЛА (Б-Л).

Модель Б-Л построена для обоснования безопасности систем, использующих политику MLS. Материалы, в которых опубликована модель в 1976г., до сих пор недоступны. Поэтому в изложении модели Б-Л будем следовать работе J. McLean (1987), в которой классы объектов предполагаются неизменными. Для описания модели нам потребуется несколько другое описание самой вычислительной системы. Пусть определены конечные множества **S, O, R, L.**

S - множество субъектов системы;
O - множество объектов, не являющихся субъектами;
R - множество прав доступа; **R** = {read (r), write(w), execute (e), append (a)};
L - уровни секретности.

Множество **V** состояний системы определяется произведением множеств:

$$V = B \times M \times F \times H,$$

где сомножители определяются следующим образом. **B** - множество текущих доступов и есть подмножество множества подмножеств произведения $S \times O \times R$. Множество подмножеств будем обозначать $P(S \times O \times R)$ элементы множества **B** будем обозначать b и они представляют в текущий момент t графы текущего доступа (в каждый момент субъект может иметь только один вид доступа к данному объекту).

M - матрица разрешенных доступов, $M=|M_{ij}|$, $M_{ij} \subseteq \mathbf{R}$. **F** - подмножество множества $L^s \times L^s \times L^0$, где каждый $f = (f_s, f_0, f_c), f \in F$, - вектор, который состоит из трех компонент, каждая из которых тоже вектор (или отображение).

f_s - уровень допуска субъектов (это некоторое отображение f: **S->L);**

f_o - уровень секретности объектов (это некоторое отображение f: **O->L);**

f_c - текущий уровень секретности субъектов (этотоже некоторое отображение f_c: **S->L).**

Элементы подмножества F, которые допущены для определения состояния, должны удовлетворять соотношению:

$$\forall S \in Sf_s(S) \geq f_c(S)$$

H - текущий уровень иерархии объектов, в работе McLean этот уровень не изменяется, совпадает с f_0 и далее не рассматривается.

Элементы множества **V** состояний будут обозначаться через v. Пусть определены множество **Q** - запросов в систему и множество **D** - решений по

поводу этих запросов (**D** = {yes, no, error}). Определим множество **W** действий системы как

$$W \subseteq Q \times D \times V \times V = \{(q, d, v_2, v_1)\} .$$

Каждое действие системы (q, d, v_2, v_1) имеет следующий смысл: если система находилась в данный момент в состоянии v_1, поступил запрос q, то принято решение d и система перешла в состояние v_2.

Пусть **T** - множество значений времени (для удобства будем считать, что **T=N** - множество натуральных чисел). Определим набор из трех функций (x, y, z)

x: T->Q,
y: T->D,
z: T->V,

и обозначим множества таких функций **X**, **Y**, **Z** соответственно.

Рассмотрим $X \times Y \times Z$ и определим понятие системы в модели Б-Л.

Определение. Системой Σ(**Q**, **D**, **W**, z_0) называетсяподмножество $X \times Y \times Z$ такое,что,

(x, y, z)$\in\Sigma$(**Q**, **D**, **W**, z_0)$\Leftrightarrow$(x_t, y_t, z_t, z_{t-1})$\in$**W**

для каждого значения t$\in$**T**, где z_0- начальное состояние системы.

Определение. Каждый набор (x, y, z)$\in\Sigma$(**Q**, **D**, **W**, z_0) называется реализацией системы.

Определение. Если (x, y, z) - реализация системы, то каждая четверка (x_t, y_t, z_t, z_{t-1}) называется действием системы.

Нетрудно видеть, что при отсутствии ограничений на запросы таким образом определен некоторый автомат, у которого входной алфавит **Q**, а выходной **D**, а множество внутренних состояний **V**. Автомат задается множеством своих реализаций. Перейдем к определению понятий, связанных с безопасностью системы.

Определение. Тройка (S, O, X)$\in$**S×O×R** удовлетворяет свойству простой секретности (ss-свойство) относительно f, если X=execute, или X=append, или, если X=read и $f_s(S)>f_o(0)$, или X=write и $f_s(S)>f_0(S)$.

Определение. Состояние v=(b, M, f, h) обладает ss-свойством, если для каждого элемента (S,O,X)$\in$B этот элемент обладает ss-свойством относительно f.

Определение. Состояние v=(b, M, f, h) обладает *-свойством, если для каждого (S, O, X)$\in$B при X=write текущий уровень субъекта $f_c(S)$ равен уровню объекта $f_0(O)$, или при X=read $f_c(S)>f_0(O)$, или при X=append $f_o(O)>f_c(S)$.

Определение. Состояние обладает *-свойством относительно множества субъектов **S'**, **S'$\in$S**, если оно выполняется для всех троек (S, O, X) таких, что S$\in$S'

Определение. Субъекты из множества **S\S'** называются доверенными.

***Лемма**. Из * -свойства для состояния v=(b, M, f, h) следует ss-свойство относительно f для всех (S, O,X)$\in$B.*

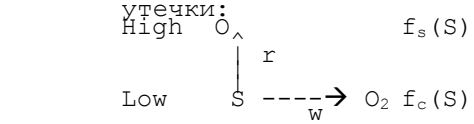
Доказательство. Утверждение следует из условия $f_s(S)>f_c(S)$.

Определение. Состояние v=(b, M, f, h) обладает ds -свойством, если $\forall$(S, O, X)$\in$b=>X$\in m_{so}$, где M=$||m_{so}||$- матрица доступа состояния v.

Определение. Состояние v = (b, M., f, h) называется безопасным, если оно обладает одновременно ss-свойством, *- свойством относительно **S'** и ds-свойством.

Напомним формулировку политики MLS, связанной с решеткой ценностей **SC×L**, где **L** - линейный порядок, **SC** - решетка подмножеств в информации: информационный поток между двумя объектами называется разрешенным, если класс объекта источника доминируется классом объекта получателя. Из определения ss-свойства следует, что в безопасном состоянии возможно чтение вниз, что согласуется с эквивалентным определением MLS политики. Кроме того, ss-свойство определяет ограничение на возможность модификации, которое связано с write:
$f_s(S)>f_o(0)$.

Объясним *-свойство. Если субъект может понизить свой текущий допуск до $f_c(S)=f_0(O)$, то, согласно *-свойству, он может писать в объект. При этом он не может читать объекты на более высоких уровнях, хотя допуск $f_s(S)$ ему это может позволить. Тем самым исключается возможный канал утечки:



Таким образом, при записи информационный поток опять не может быть направлен вниз. Исключение возможно только для доверенных субъектов, которым разрешено строить информационный поток вниз. При этом доверенность субъекта означает безопасность такого потока вниз (поэтому эти потоки считаются разрешенными). Сказанное выше означает, что безопасное состояние модели Б-Л поддерживает политику MLS. Значит, для того, чтобы доказать, что любой поток на траектории вычислительной системы разрешен, достаточно показать, что выходя из безопасного состояния и следуя допустимым действиям мы опять приходим в безопасное состояние, тем самым любая реализация процесса будет безопасной. Проведем строгое обоснование этого вывода.

Определение. Реализация (x, y, z) системы Σ**(Q, D,W, z)** обладает ss-свойством (*-свойством, ds- свойством), если в последовательности (z_0, z_1,.....) каждое состояние z_n обладает ss-свойством (* -свойством , ds-свойством).

Определение. Система обладает ss-свойством (соответственно, *-свойством, ds-свойством), если каждая ее реализация обладает ss-свойством (соответственно, *-свойством, ds-свойством).

Определение. Система называется безопасной, если она обладает одновременно ss-свойством, *-свойством, и ds - свойством.

Теорема A1. *Σ(Q, D, W, z_0) обладает ss -свойством для любого начального z_0, которое обладает ss-свойством тогда и только тогда, когда W удовлетворяет следующим условиям для каждого действия (q, d, (b*, M*, i*, h*), (b, M, f, h)):*

(I) $\forall$(S, 0, X)$\in$b| b обладает ss-свойством относительно f*.*

(2) если (S, O, X)$\in$b и не обладает ss-свойством относительно f, то (S, O, X)$\notin$b**

Доказательство. $\forall$(S, O, X)$\in$b* возможно либо (S,O, X)$\in$b, или (S, O, X)$\in$b*\b. Условие (1) означает, что состояние (b*, M*, f*, h*) пополнилось элементами (S, O, X), обладающими ss - свойством относительно f*. Условие (2) означает, что элементы b*, перешедшие из b, обладают ss - свойством относительно f*. Следовательно, $\forall$(S, O, X)$\in$b* обладает ss-свойством относительно f*. Пусть любое состояние обладает ss-свойством относительно своего *f.* Тогда (1) выполняется, т.к. ss-свойство выполняется для всех (S, O,X) из b*. И, если (S, O, X)$\in$b и перешло в b*, то, в силу ss-свойства (S, O, X) обладает ss - свойством относительно f*. Что и требовалось доказать.

Теорема A2. *Система Σ(R, D, W, z_0) обладает *- свойством относительно S' для любого начального состояния z_0, обладающего *-свойством относительно S' тогда и только тогда, когда **W** удовлетворяет следующим условиям для каждого действия (q, d, (b*, M*,f*, h*), (b, M, f, h)):*

*(I) $\forall$S$\in$**S'**, $\forall$(S, O, X)$\in$b*\ b обладает *-свойством относительно f*;*

*(II) $\forall$S$\in$**S'**, $\forall$(S, O, X)$\in$b и (S, O, X) обладает *-свойством относительно f*, то (S, O, X)$\notin$b*.*

Доказательство проводится аналогично.

Упражнение. Доказать теорему A2.

Теорема A3. *Система Σ(Q, D, W, z_0) обладает ds-свойством тогда и только тогда, когда для любого начального состояния, обладающего ds-свойством, **W** удовлетворяет следующим условиям для любого действия (q, d, (b*, M*, f*, h*), (b, M, f, h)):*

(I) (S, O, X)$\in$b| b то X$\in T_{so}$*,*

(II) $(S, 0, X) \in b^* \mid b\ X \notin m_{so}^*$, *то* $(S, 0, X) \notin b^*$

Доказательство проводится аналогично.

Упражнение. Доказать теорему А3.

Теорема (Basic Security Theorem). *Система* $\Sigma(Q, D, W, z_0)$ *- безопасная тогда и только тогда, когда* z_0 *- безопасное состояние и* ***W*** *удовлетворяет условиям теорем А1, А2, А3 для каждого действия.*

Доказательство. Теорема BST следует из теорем А1, А2, А3.

5.3. МОДЕЛЬ LOW-WATER-MARK (LWM).

Данная модель является конкретизацией модели Б-Л, а также дает пример того, что происходит, когда изменения уровня секретности объекта возможны. Политика безопасности прежняя: все объекты системы классифицированы по узлам решетки ценностей (MLS) и поток информации разрешен только "снизу вверх" .

В рассматриваемой системе один объект (неактивный), три операции с объектом, включающие запросы на доступ:

- read,
- write,
- reset.

Эти операции используются несколькими субъектами (процессами), имеющими фиксированные уровни секретности (для простоты - классы секретности образуют линейный порядок). Напомним формальное требование политики о том, что информация может двигаться только "снизу вверх". Поток информации возможен тогда и только тогда, когда реализуется доступ субъекта к объекту вида w или r. При помощи r поток считается разрешенным, если $f_s(S) > f_o(0)$.

При команде w поток считается разрешенным, еслисубъект S не может прочитать информацию в объекте уровня $f_s(S) < f_0(0)$ и записать в объект, для которого $f_s(S) > f_0(O)$, причем хотя бы в одном из этих соотношений неравенство строгое (напомним, что по условию текущие уровни субъектов $fc(S) = f_s(S)$ для любого S). Из этих свойств следует, что в системе должны выполняться условия ss и *. Условие ds автоматически выполняется, так как нет ограничений на доступ, кроме перечисленных.

Таким образом, условия ss в данной системе выглядят стандартно: если X=w или r, то могут быть разрешены доступы (S, 0, X) при выполнении $f_s(S) > f_0(0)$.

Условие *:

если X=w, то $f_s(S) = f_0(0)$,

если X=r, то $f_s(S) > f_o(0)$.

Опишем функционирование системы и докажем,что выполняются условия ss и *. Уровень объекта O в LWM может меняться: при команде write может снизиться, а при команде reset подняться следующим образом. По команде reset класс объекта поднимается и становится максимальным в линейном порядке. После этого все субъекты приобретают право w, но read имеют право только субъекты, находящие на максимальном уровне решетки. При команде write гриф объекта снижается до уровня субъекта, давшего команду w. При снижении уровня секретности объекта вся прежняя информация в объекте стирается и записывается информация процессом, вызвавшем команду write. Право write имеет любой субъект, у которого $f_s(S) < f_o(0)$, где $f_0(O)$ - текущий уровень объекта. Право reset имеет только тот субъект, который не имеет право write. Право read имеет любой субъект, для которого $fs(S) > f_o(0)$.Суммируем вышесказанное в следующей таблице.

Операция	Организация доступа	Результат операции
Read	$f_s(S) > f_o(0)$	Процесс S получает содержимое объекта O
Write (данные)	$f_s(S) < f_o(0)$	Уровень объекта становится равным уровню S. Данные от S

		записываются в O.
Reset	$f_s(S)>f_o(0)$	Уровень объекта устанавливается выше всех уровней процессов.

*Лемма. Для любого состояния системы LWMвыполнены условия ss и *.*

Доказательство. Если $f_s(S)>f_o(0)$, то r доступ S к O разрешен. Если $f_s(S)=f_o(0)$, то w доступ S к O разрешен. Таким образом ss и * выполнены. Что и требовалось доказать.

Однако все рассуждения останутся теми же, если отказаться от условия, что при команде write в случае снижения уровня объекта все стирается. То есть здесь также будут выполняться условия * и ss, но ясно, что в этом случае возможна утечка информации. В самом деле, любой процесс нижнего уровня, запросив объект для записи, снижает гриф объекта, а получив доступ w, получает возможность r. Возникает канал утечки ($\downarrow$w, $\rightarrow$r), при этом в предыдущей модели свойство * выбрано для закрытия канала ($\uparrow$ r, $\rightarrow$w). Данный пример показывает, что определение безопасного состояния в модели Б-Л неполное и смысл этой модели только в перекрытии каналов указанных видов. Если "доверенный" процесс снижения грифа объекта работает неправильно, то система перестает быть безопасной.

5.4. МОДЕЛИ J.GOGUEN, J.MESEGUER (G-M).

Модели G-M - автоматные модели безопасных систем. Начнем с простейшего случая системы с "фиксированной" защитой. Пусть **V** - множество состояний системы (**V** - конечное и определяется программами, данными, сообщениями и пр.), **C** - множество команд, которые могут вызвать изменения состояния (также конечное множество), **S** - множество пользователей (конечное множество). Смена состояний определяется функцией:

$$do: \mathbf{V} \times \mathbf{S} \times \mathbf{C} \rightarrow \mathbf{V}.$$

Некоторые действия пользователей могут не разрешаться системой. Вся информация о том, что разрешено ("возможности" пользователей) пользователям сведена в c-таблицу t. В рассматриваемом случае "возможности" в c-таблице t совпадают с матрицей доступа. Если пользователь не может осуществить некоторую команду c, то

$$do\ (v,\ S,\ c) = v.$$

Предположим, что для каждого пользователя S и состояния v определено, что "выдается" этому пользователю (т.е., что он видит) на выходе системы. Выход определяется функцией

$$out: \mathbf{V} \times \mathbf{S} \rightarrow \mathbf{Out},$$

где **Out** - множество всех возможных выходов (экранов, листингов и т.д.).

Мы говорим о выходе для пользователя S, игнорируя возможности S подсмотреть другие выходы.

Таким образом получили определение некоторого класса автоматов, которые будут встречаться далее.

Определение. Автомат M состоит из множеств:

S - называемых пользователями;

V - называемых состояниями;

C - называемых командами;

Out - называемых выходами;

и функций:

- выходной функции out: **V× S → Out**, которая "говорит, что данный пользователь видит, когда автомат находится в данном состоянии ";

• функции переходов do: **V×S×C→V**, которая "говорит, как изменяется состояние автомата под действием команд";

и начального состояния v_0.

Системы с изменяющимися "возможностями" защиты определяют следующими образами.

Пусть **Capt** - множество всех таблиц "возможностей", **CC** - множество с-команд (команд управления "возможностями"). Их эффект описывается функцией:

cdo: Capt× S × CC→Capt.

При отсутствии у пользователя S права на с-команду положим cdo(t, S, c)=t. Пусть **VC** - множество команд, изменяющих состояние. Теперь можем определить С-автомат, который лежит в основе дальнейшего.

Определение. С-автомат M определяется множествами:

S - "пользователи";
V - "состояния";
VC - "команды состояния";
Out - "выходы";
Capt - "с-таблицы";
CC - "с-команды",

и функциями:

• выхода out: **V×Capt×S→Out**, которая "говорит, что данный пользователь видит, когда автомат находится в данном состоянии v, а допуски определяются с-таблицей";

• переходов do: **V×Capt×S×VC→**V, которая "говорит, как меняются состояния под действием команд";

• изменения с-таблиц cdo: **Capt×S×CC→Capt**, которая "говорит, как меняется с - таблица под действием с", и начального состояния, которое определяется с-таблицей t и состоянием v.

Будем считать

C=CC∪VC.

То, что мы определили на языке теории автоматов, называется последовательным соединением автоматов .

Определение. Подмножества множества команд **C** называются возможностями Ab=2^C.

Если дан С-автомат M, мы можем построить функцию переходов всей системы в множестве состояний **V×Capt:**

cvdo: **V** ×**Capt**×**S**×**C→V**×**Capt**,

где

cvdo(v, t, S, c) = (do(v, t, S, c), t), если c∈**VC**,

cvdo(v, t, S, c) = (v, cdo(t, S, c)), если c∈**CC**.

Стандартно доопределяется функция cvdo на конечных последовательностях входов

cvdo: **V×Capt×(S×C)*→V×Capt**

следующим образом:

cvdo(v, t, Nil) = (v, t),

если входная последовательность пустая;

cvdo(v, t, W(S, c))=cvdo(cvdo(v, t, W), S, c)),

где W(S, C) - входное слово, кончающееся на (S,C) и начинающееся подсловом W.

Определение. Если W входное слово, то[[W]] = (..., (v_i t_i),...), где последовательность состоянии вычисляется в соответствии с определенной выше функцией переходов под воздействием входной последовательности W.

Введем понятие информационного влияния одной группы на другую, смысл которого состоит в том, что используя некоторые возможности одна группа пользователей не влияет на то, что видит каждый пользователь другой группы. Для этого определим $[[W]]_s$ - выход для S при выполнении входного слова W C -автомата M:

$$[[W]]_s=out([[W]], S),$$

где

$$out([[W]]_s, S) = (... out(v, t, S)...),$$
$$[[W]] = (..., (v_i, t_i),...).$$

Пусть **G⊆S, A⊆C, W∈(S×C)*.**

Определение. Pg(W) - подпоследовательность W, получающаяся выбрасыванием всех пар (S, c) при S∈G, P_a(W) - подпоследовательность W, получающаяся выбрасыванием из W всех пар (S, c) при c∈A, P_{GA}(W) - подпоследовательность W, получающаяся выбрасыванием пар (S, c), S∈G и c∈A.

Пример 1. G={S, P}, A={c_1, c_2}.

P_{GA}((S', c), (S, cз), (S, c_2), (P', c))= (S', c), (S, c_3), (P', c).

Определим несколько вариантов понятия независимости .

Пусть **G⊆S, G'⊆S.**

Определение. G информационно не влияет на G' (обозначается G: | G'), если ∀W∈(S×C)* и ∀S∈G' $[[W]]_s = [[P_A(W)]]_s$

Аналогично определяется невлияние для возможностей **A** (или группы **G** и возможностей **A**).

Определение. **A** информационно не влияет на **G** (обозначается **A: | G**) , если ∀W∈(S×C)* и ∀S∈G $[[W]]_s= [[P_A(W)]]_s$

Определение. Пользователи **G**, используя возможности **A**, информационно не влияют на **G'** (обозначается **A.G:** | **G'**), если ∀W∈(S×C)* и ∀S×**G'** $[[W]]_s=[[P_A(W)]]_s$.

Пример 2. Если A: | {S}, то команды из A не влияют на выход, выданный S. Если A={create, write, modify, deleted} для файла F, то A: | {S} означает, что информация читаемая S в F не может измениться любой из команд в A. Если F не существовал, то для S будет всегда выдаваться информация, что F не существует.

Определение. Политика безопасности в модели G-M - это набор утверждений о невлиянии.

Пример 3. MLS политика. Пусть L - линейно упорядоченное множество уровней секретности и задано отображение

level: **S->L.**

Определим: ∀x∈**L**

S[-∞, x] = {Se**S** | level(S)<x)

S[x,+∞] = {SeS| level(S)>x}.

Определение. MLS политика в модели G-M определяется следующим набором утверждений о невлиянии:

$\forall x \in \mathbf{L}$, $\forall x' \in \mathbf{L}$, $x > x'$,

$S[x, +\infty] :| S[-\infty, x']$.

Говорят, что $\mathbf{G} \subseteq \mathbf{S}$ невидимо для остальных пользователей, если $G :| \overline{G}$, где $\overline{G} = \mathbf{S \backslash G}$.

Используя это понятие легко обобщить определение MLS политики на случай, когда **L** - решетка.

Определение. MLS политика в модели G-M определяется следующим набором утверждений о невлиянии:

$\forall x \in \mathbf{L}$ $S \backslash S[-\infty, x]$ - невидимо для остальных пользователей.

Пример 4. Одним из важнейших примеров политики безопасности, легко выражаемой в G-M модели, является режим изоляции.

Определение. Группа **G** называется изолированной,если $\mathbf{G} :| \overline{G}$ *и* $\overline{G} :| \mathbf{G}$. Система полностью изолирована, если каждый пользователь изолирован.

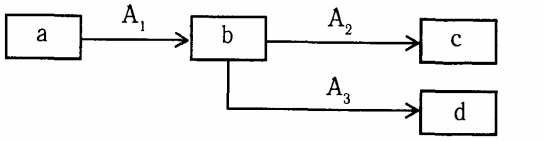
Пример 5. Контроль канала. В модели G-M канал определяется как набор команд $\mathbf{A} \subseteq \mathbf{C}$.

Пусть $\mathbf{G}, \mathbf{G}' \subseteq \mathbf{S}$.

Определение. G и G' могут связываться только через канал A тогда и только тогда, когда

$\overline{A}, G :| G'$ и $\overline{A}, G' :| G$

Пример 6. Информационный поток Пусть a, b, c, d -процессы и A_1, A_2, Аз - каналы такие, что a, b, c, d могут связываться только по схеме



Эта картинка описывается следующими утверждениями о невлиянии:

{b, c, d}: | {a}	A_1, {a}: | {b, c, d}
{c, d}: | {b}	A_2, {b}: | {c}
{c}: | {d}	A_3, {b}: | {d}
{d} : | {c}	

5.5. МОДЕЛЬ ВЫЯВЛЕНИЯ НАРУШЕНИЯ БЕЗОПАСНОСТИ.

Один из путей реализации сложной политики безопасности, в которой решения о доступах принимаются с учетом предыстории функционирования системы, - анализ данных аудита. Если такой анализ возможно проводить в реальном масштабе времени, то аудиторская информация (АИ) совместно с системой принятия решений превращаются в мощное средство поддержки политики безопасности. Такой подход представляется перспективным с точки зрения использования вычислительных средств общего назначения, которые не могут гарантировано поддерживать основные защитные механизмы. Но, даже не в реальном масштабе времени, АИ и экспертная система, позволяющая вести анализ АИ, являются важным механизмом выявления нарушений или попыток нарушения политики безопасности, так как реализуют механизм ответственности пользователей за свои действия в системе.

По сути анализ АИ имеет единственную цель выявлять нарушения безопасности (даже в случаях, которые не учитываются политикой безопасности). Далее мы изложим пример организации такого анализа, который известен из литературы под названием "Модель выявления нарушения безопасности". Эта модель, опубликованная D.Denning в 1987

г., явилась базисом создания экспертной системы IDES для решения задач выявления нарушений безопасности. Модель включает 6 основных компонент:

- субъекты, которые инициируют деятельность в системе, обычно - это пользователи;
- объекты, которые составляют ресурсы системы - файлы, команды, аппаратная часть;
- АИ - записи, порожденные действиями или нарушениями доступов субъектов к объектам;
- профили - это структуры, которые характеризуют поведение субъектов в отношении объектов в терминах статистических и поведенческих моделей;
- аномальные данные, которые характеризуют выявленные случаи ненормального поведения;
- правила функционирования экспертной системы при обработке информации, управление.

Основная идея модели - определить нормальное поведение системы с тем, чтобы на его фоне выявлять ненормальные факты и тенденции. Определению субъектов и объектов мы уделили достаточное вниманиев параграфе 1.1. Остановимся подробнее на описаниии примерах остальных элементов модели.

АИ - это совокупность записей, каждая из которых в модели представляет 6-мерный вектор, компоненты которого несут следующую информацию:

<субъект; действие; объект; условия для предоставления исключения; лист использования ресурсов; время>,

где смысл компонент следующий.

Действие - операция, которую осуществляет субъект и объект.

Условия для предоставления исключения, если они присутствуют, определяют, что дополнительно надо предпринять субъекту, чтобы получить требуемый доступ.

Лист использования ресурсов может содержать, например, число строчек, напечатанных принтером, время занятости CPU (центрального процессора) и т.д.

Время - уникальная метка времени и даты, когда произошло действие.

Так как АИ связана с субъектами и объектами, то данные АИ подобны по организации матрице доступа, где указаны права доступа каждого субъекта к любому объекту. В матрице АИ в клетках описана активность субъекта по отношению к объекту.

Пример 1. Рассмотрим команду

Copy=Game.exe to <Library> Game.exe,

которую использовал пользователь Smith для того, чтобы скопировать файл "Game" в библиотеку; копирование не выполнено, так как Smith не имеет праваписать в библиотеку. АИ, соответствующая этому примеру будет состоять из записей:

(Smith, execute, < Library> Copy.exe, O,

CPU = 00002, 11.05.85.21678)

(Smith, Read, <Smith> Game.exe, 0, Records = 0,

11.05.85.21679)

(Smith, Write, <Library> Game.exe,

Write - violation. Records = 0, 1 1 .05.85.2 1 680).

Рассмотрим подробнее понятие "профиль". Профили описывают обычное поведение субъектов по отношению к объектам. Это поведение характеризуется набором статистических характеристик, вычисленных по наблюдениям за действиями субъекта по отношению к объекту, а также некоторой статистической моделью такого поведения. Приведем примеры таких статистических характеристик.

1. Частоты встречаемости событий. Например, частота встречаемости заданной команды в течение часа работы системы и т.д.

2. Длина временного промежутка между осуществлением некоторых событий.

3. Количество ресурсов, которые были затрачены в связи с каким-либо событием. Например, время работы CPU при запуске некоторой программы, число задействованных элементов аппаратной части и др.

Статистические модели строятся по наблюденным значениям статистических характеристик с учетом статистической обработки данных. Например, некоторый процесс характеризуется устойчивым средним числом встречаемости данной команды, которая может быть уверенно заключена в доверительный интервал в 3σ, где σ - стандартное отклонение частоты встречаемости этого события.

Чаще всего статистические модели являются многомерными и определяются многомерными статистическими методами. Наиболее сложные модели - это случайные процессы, характеристики моделей являются некоторыми функционалами от случайных процессов. Например, моменты остановки программы и т.д.

В нормальных условиях статистические характеристики находятся в границах своих значений. Если происходит ненормальное явление, то возможно наблюдать статистически значимое отклонение от средних параметров статистических моделей.

Пример 2. Реализация канала утечки по времени, приведенного в примере 4 параграфа 2.1, требует повторяющегося запроса на принтер, а затем отказа от него. Ясно, что в реализации такого канала частота обращения к принтеру возрастает, что, естественно, приведет к значимому отклонению этой характеристики от модели повседневного использования принтера данным пользователем.

Описание профиля или его структура состоит из 10 компонентов, которые, кроме собственно статистической модели, определяют АИ с ней связанную. Структура профиля может быть представлена следующим вектором:

<Имя переменной; отражаемые действия; имеющиеся исключения; данные использования ресурсов; период измерений; тип статистики; порог допустимых значений; субъект; объект; значение последнего наблюдения модели>.

К сожалению, анализ на уровне субъект - объект в значительной степени затруднен. Поэтому аналогичные структуры (профили) создаются для агрегированных субъектов и объектов. Например, для некоторого фиксированного множества субъектов в отношении всех возможных объектов. Другой пример: действия всех пользователей в отношении данного объекта.

Основной сложностью при внедрении этой моделиявляется выбор и построение профилей.

Пример 3. Рассмотрим систему с 1000 пользователями; у каждого пользователя в среднем 200 файлов, что дает 200000 файлов в системе. Тогда имеем 200 млн. возможных комбинаций: пользователь-файл. Даже, если предположить, что каждый пользователь осуществляет доступ к 300 файлам, то необходимо создать 300 000 профилей.

Пример 3 показывает, что необходимо применять специальные приемы для сокращения информации.

Если обнаружено ненормальное поведение, то немедленно делается запись в сборнике аномальных фактов. Каждая запись в этом сборнике - трехмерный вектор, имеющий следующие компоненты:

<событие; время; в отношении какого профиля получено отклонение > .

Применение рассмотренной модели дало хорошие результаты. Вместе с тем требуют исследований такие вопросы:

• насколько надежно предложенный метод выявляет нарушения безопасности;

• какова доля нарушений безопасности, для которых работает метод;

•выбор инструментов статистической обработки данных и моделей профилей требует обоснования;

• идеология самого быстрого обнаружения ещене ясна.

Глава 6 ГАРАНТИРОВАННО ЗАЩИЩЕННЫЕРАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

Глава VI посвящена вопросам защиты сетей межмашинного обмена информацией. Основной аспект нашего изучения - распространение стандарта "Оранжевая книга" на распределенные системы обработки информации. Базисом для предложенного ниже подхода является "Trusted Network Interpretation", которая была выпущена DoD в 1987 г. в составе "радужной" серии. Эта книга известна под названием "Красная книга" (КК). В главе приведены результаты, заимствованные из КК и связанные с методами анализа защищенных сетей и синтеза гарантированно защищенных распределенных систем обработки информации.

В отличии от "монолитных" вычислительных систем, имеющих заданный периметр и спецификацию, распределенные системы не имеют ограниченного периметра, а число компонент их может меняться. Между компонентами в сетях существуют каналы, которые могут проходить по незащищенной или враждебной территории. Этими чертами различия междумонолитной и распределенными вычислительнами системами исчерпываются. Следовательно, если как-то учтены перечисленные различия, то все вопросы защиты вычислительных и распределенных систем одинаковы. Таким образом, к распределенным системам можно попытаться применить критерии гарантированной защищенности вычислительных систем, например, "Оранжевой книги".

Возможны два подхода к анализу и оценке защищенности распределенной системы.

1. Каждая компонента распределенной системы есть самостоятельная защищенная система, а, в целом, сеть представляет множество взаимодействующих, защищенных порознь систем. Такая сеть не есть одно целое и вопросы ее гарантированной защиты сводятся к доказательству защищенности компонент в условиях рассматриваемого окружения и организации защищенных шлюзов для взаимодействия компонент. Однако, никто не отвечает за информацию в сети целиком.

2. Все компоненты и связи между ними составляют единое целое. В этом случае существует лицо (центр), которое берет на себя обязательство обеспечить безопасность в сети. Здесь эта безопасность относится к сети в целом, несмотря на неопределенный периметр и изменяемую конфигурацию. Тогда должна существовать некоторая политика безопасности и средства сети, поддерживающие эту политику (NTCB). При этом средства поддержки безопасности в сети вовсе не должны составлять полный комплекс (удовлетворяющий стандарту ОК) механизмов защиты в каждой отдельной компоненте. Однако они, в целом, должны составлять единый механизм защиты, который в случае использования дискреционной и мандатной политики может анализироваться на предмет соответствия стандарту ОК. В частности,

для класса В3 и выше NTCB должна реализовывать монитор обращения во всей сети. Отсюда возникает задача синтеза из отдельных компонент NTCB, поддерживающую политику в сети, а также задача оценки защитных функций компонент, из которых NTCB возможно синтезировать. В КК все рассмотренные вопросы поставлены с точки зрения проведения оценки распределенной системы. Предложенный подход (он же применим в анализе и синтезе таких систем) состоит в том, чтобы по сети построить гипотетическую монолитную систему с TCB, совпадающей по функциям с NTCB, и ее оценивать. С другой стороны, при создании распределенной системы можно сначала создать гипотетический проект монолитной защищенной системы, а затем провести декомпозицию его по компонентам распределенной системы с сохранением защитных свойств. И, наконец, всем известна "слабость" ОК, состоящая в том, что слабо отработана проблема контроля защищенности при модификациях или замене подсистем. Если для монолитной вычислительной системы эта слабость была преодолима, то в распределенных системах проблема наращивания компонент без переоценки всего в целом становится принципиальной. В параграфе 6.1 на основе теоремы о синтезе монитора обращения в NTCB из мониторов обращений компонент строится главная составляющая NTCB - контроль за доступами. Отсюда следует описанная вкратце декомпозиция монолитных систем, в которых есть монитор обращения, и синтез распределенной системы в монолитную вычислительную систему с TCB, реализующей монитор обращения. В параграфе 6.2 описаны подходы КК к анализу компонент и примеры встраивания таких компонентв гарантированно защищенную распределенную систему, в которой функции NTCB распределены по компонентам.

6.1. СИНТЕЗ И ДЕКОМПОЗИЦИЯ ЗАЩИТЫ В РАСПРЕДЕЛЕННЫХСИСТЕМАХ.

Рассмотрим сеть, компоненты которой связаны каналами связи, а каждая из компонент несет некоторые функции защиты. Основное требование при анализе безопасности распределенной системы как одного целого - является общая политика безопасности. Тогда вопрос защищенности распределенной системы как одного целого состоит в организации согласованного действия компонент защиты по поддержке политики безопасности всей сети. Решению этой проблемы противостоят опасности нарушения защиты информации при передаче ее по каналам связи, а также опасности асинхронного функционирования компонент защиты.

В связи с этим предположим, что TCB компонент в каждом локальном случае поддерживают функции монитора обращения. Единая политика безопасности в сети не означает, что все TCB компонент поддерживают одну и ту же политику и соответствуют требованиям одного класса. Например, одна компонента может быть классифицирована по классу С2 (хотя в ней тоже мы требуем дополнительно наличие монитора обращения), а другая - по классу В3. При этом обе компоненты поддерживают единые дискреционную и мандатную политики, хотя первая компонента - одноуровневая (соответствует одному классу обрабатываемой информации), а вторая - поддерживает MLS политику в полном объеме. Кроме того, в обоих компонентах предполагается единая система категорий и единые ограничения на распространение прав в дискреционной политике (по крайней мере она должна вкладываться в единую систему категорий и прав).

Рассмотрим вопрос, когда совокупность мониторов обращения в подсистемах реализует монитор обращения во всей распределенной сети. В КК формулируются условия, позволяющие это сделать.

Теорема. *Пусть выполнены следующие условия.*

1. Каждый субъект сети существует только в одной компоненте на протяжении всего жизненного цикла.

2. Каждый субъект может иметь доступ только к объектам своей компоненты.

3. Каждая компонента содержит отнесенный к этой компоненте монитор обращений, который рассматривает только обращения субъектов этой компоненты к объектам этой компоненты.

4. Все каналы, связывающие компоненты, не компрометируют безопасность информации, в них проходящей.

Тогда совокупность мониторов обращения компонент является монитором обращения в сети.

Доказательство. Этот результат потребует дополнительного уточнения некоторых основных понятий. Рассмотрим сеть из компонент, связанных безопасными каналами связи. Напомним, что любой объект представляет собой конечное непустое множество слов некоторого языка. Добавим к этому, что объект существует только при условии, что возможен доступ к содержанию объекта, то есть мы предполагаем наличие хотя бы одного слова из множества, определяющего объект, к которому возможен в данный момент доступ хотя бы одного субъекта. Тогда информация, передаваемая по безопасному каналу связи, не является объектом, так как до момента окончания приема, нет ни одного слова на приемном конце, к которому хотя бы один субъект может получить доступ. В самом канале, если он не может компрометировать информацию при передаче, мы считаем невозможным доступ к информации и поэтому это не объект. На передающем конце информация передается из некоторого объекта и при передаче мы считаем, что есть некоторый субъект на передающем конце, который в течение передачи имеет доступ к объекту на передающем конце и представляет собой интерфейс с многоуровневым прибором ввода/вывода (концепция такого экспорта информации изложена в ОК). После окончания передачи на приемном конце сформировался новый объект, который, вообще говоря, не имеет отношения к объекту на передающем конце и из которого шла перекачка информации.

Рассмотрим формальное объединение мониторов обращения компонент. Тогда из вышесказанного следует, что нет объектов вне локальных компонент, а дистанционный доступ происходит за счет создания нового объекта, который полностью контролируется локальной ТСВ компоненты. Покажем, что формальное объединение мониторов обращения компонент-монитор обращения сети М.

1) Каждое обращение субъекта к объекту в системе происходит через М. В самом деле, каждый субъект существует только в одной компоненте по усл. 1 и может обращаться к объектам только своей компоненты по усл. 2. Поэтому все обращения в системе ограничены рамками компонент. А тогда каждое обращение обрабатывается монитором обращения соответствующей компоненты по определению монитора обращения.

2) Монитор обращения каждой компоненты по определению гарантированно защищен. Поэтому объединение их гарантированно защищено.

3) Функционирование всех мониторов обращения компонент полностью тестируется (то есть существует полная система тестов). Тогда совокупность тестов компонент полностью тестирует М.

Теорема доказана.

Теперь рассмотрим вопрос о синтезе единой вычислительной системы из компонент таким образом,что анализ защищенности сети эквивалентен анализу такой вычислительной системы. Пусть вычислительная система обладает следующими свойствами. Это многоуровневая, многопрограммная система, удовлетворяющая условиям соответствующего класса ОК(например, ВЗ). В системе информация ТСВ распределена среди одновременно работающих процессоров, которые соединены одной шиной. В системе функционирует одна операционная система, которая поддерживает процесс

на любом процессоре. Каждый процесс может использовать внешние приборы через запрос в ТСВ, где реализован монитор обращения. Можно показать, что единая NTCB в распределенной системе, эквивалентной описанной выше вычислительной системе, реализует в компонентах мониторы обращения, объединение которых дает монитор обращения NTCB (по доказанной теореме). А ТСВ вычислительной системы эквивалентна NTCB сети после декомпозиции этой вычислительной системы.

Этот подход позволяет проводить анализ распределенной системы как единой вычислительной системы. Можно действовать наоборот. Создать проект монолитной защищенной вычислительной системы описанного типа, а затем реализовать ее представление в виде распределенной сети.

Заметим, что некоторые компоненты монитора обращений NTCB могут быть вырожденными. Кроме того, наличие монитора обращений вовсе не означает, что в компоненте есть все функции ТСВ системы защиты по какому-либо классу. Единая распределенная система тем и хороша, что ТСВ сети можно построить из компонент, не содержащих в отдельности все функции защиты.

В заключение сформулируем требования, которым должен удовлетворять использованный в теореме безопасный канал связи:

1. Безопасность связи - устойчивость к несанкционированным раскрытию или модификации передаваемой ценной информации.
2. Надежность связи - не допускает отказ от доставки сообщения, неправильную доставку, доставку ошибочных данных.
3. Имитозащита - не допускает изменений в критичной для этого информации (метки и т.д.).
4. Не допускает скрытые каналы утечки за счет модуляции параметров канала.

6.2. АНАЛИЗ КОМПОНЕНТ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ.

Анализ и оценка защиты распределенных систем, как единого целого, предполагает анализ частей, а затем построение оценки защищенности всей системы в целом. Анализ компонент и синтез единой оценки защищенности всей системы необходим также при модернизации системы, при замене старых компонент новыми, при синтезе системы из блоков или частей, для того, чтобы иметь возможность использовать разработки различных производителей, для доказательства существования NTCB, удовлетворяющей требованиям ОК.

При анализе возникают две проблемы.

1 . Как разделить сеть так, чтобы из анализа и оценки компонент можно построить оценку защищенности системы в целом.

2. Какими критериями надо пользоваться при анализе компонент и как из результатов для компонент синтезировать общую оценку.

В предыдущем параграфе мы наметили контуры ответа на первый вопрос. В случае, когда декомпозиция происходит так, что в каждой компоненте реализован монитор обращения, то, при выполнении условий теоремы предыдущего параграфа, во всей системе есть монитор обращения. Тогда гипотетическоеобъединение распределенной системы в единую вычислительную систему, как это было обозначено выше, позволяет провести анализ наличия всех остальных функций NTCB. И наоборот, декомпозиция единой гарантированно защищенной вычислительной системы из предыдущего параграфа так, что в любой компоненте реализован монитор обращений, позволяет рассредоточить функции NTCB по различным компонентам.

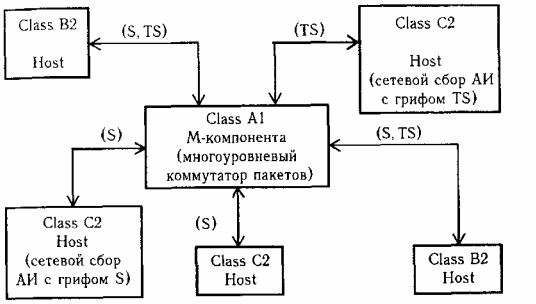
В "Красной книге" допускается, что ТСВ любого класса (с соответствующими оговорками) может быть синтезирована из реализации 4 функций:

- поддержки дискреционной политики (Д);
- поддержки мандатного контроля (М);
- функции идентификации/аутентификации (I);
- аудита (А).

Исходя из этого предполагается, что любая подсистема защиты, подлежащая отдельной оценке и экспертизе на предмет встраивания в распределенную систему, должна удовлетворять внутри себя условиям теоремы параграфа 6.1 и выполнять некоторый набор из перечисленных

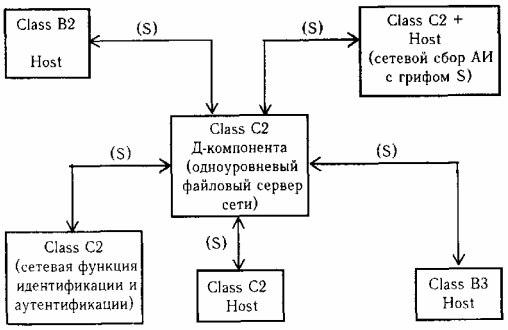
функций (всего имеется 16 вариантов таких наборов). При наличии этих свойств подсистема может быть компонентой распределенной сети и входить в NTCB. Приведем примеры включения таких подсистем.

Пример 1. Пусть дана М - компонента (то есть подсистема, единственной функцией которой является поддержка мандатного контроля доступа). Пусть также эта подсистема обладает монитором обращений, оценена как самостоятельная система по классу А1. Тогда ее можно включить как компоненту в гарантированно защищенную распределенную систему обработки информации, например, для выполнения функций многоуровневой коммутации пакетов. Покажем это на схеме, взятой из "Красной книги".



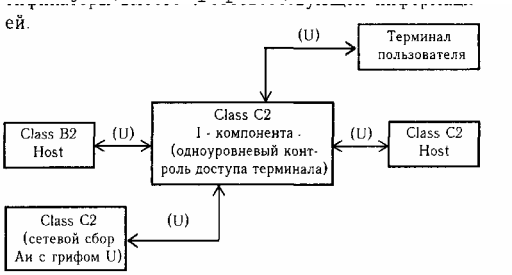
На приведенной схеме показана взаимосвязь М -компоненты с другими компонентами и, в частности, с подсистемами TCB. Минимальное взаимодействиенеобходимо с системой аудита.

Пример 2. Данный пример из "Красной книги" показывает использование Д-компоненты в качестве одноуровневого файлового сервера сети.



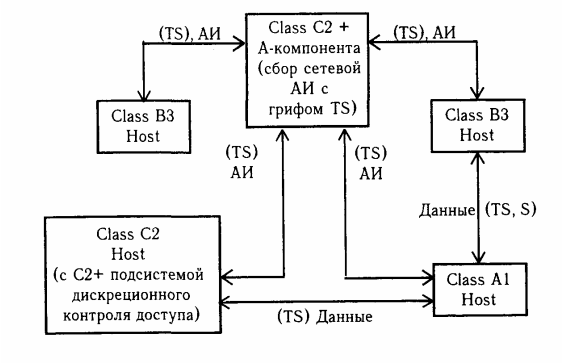
В этом примере обозначение C2+ показывает, что система может быть оценена по классу С2, но иметь дополнительные функции, которые присущи дискреционной политике и аудиту в классе В3 и выше. (Дополнительно требуется выполнение функции блокировки при превышении числа опасных событий выше порога, матрица запрещенных доступов и т.д.)

Пример 3. В данном примере мы покажем использование компоненты I для организации интерфейса с несекретными пользователями. Основная функция компоненты - контроль доступа терминала. Вся работа по аутентификации пользователя проводится в этой компоненте, а затем в разрешение доступа и аудиторскую информацию переписываются только идентификаторы вместе с соответствующей информацией.



Пример 4. В рассматриваемом примере, также взятом из "Красной книги", А - компонента выполняет функцию сбора одноуровневой АИ в сети.

Обозначение C2+ показывает, что в системе, классифицированной уровнем C2, выполняются дополнительные (по классу B3) функции.



Глава 7 ПРОБЛЕМА

ПОСТРОЕНИЯ ГАРАНТИРОВАННО ЗАЩИЩЕННЫХ БАЗ ДАННЫХ

Создание гарантированно защищенных баз данных связано с некоторыми общими проблемами синтеза систем защиты. Эти проблемы отражены в "Розовой книге" (Trusted Database Management System Interpretation of the Trasted Computer System EvaluationCriteria, 1991). Промежуток в 4 года между второй, "Красной книгой", и "Розовой книгой" показывает, что за эти годы решались трудные задачи теории зашиты информации. В самом деле, если политика безопасности в базе данных не включает вопросов, связанных с взаимным выводом информации и каналов утечки, основанных на этом выводе, вопросов восстановления зашумленной информации путем повторных запросов в базу данных, то такая политика не является адекватной для безопасности информации. Однако, все эти вопросы нельзя включать в политику безопасности, поддерживаемую самой вычислительной системой, что приведет к симбиозу операционной системы и системы управления базой данных. Вместе с тем, сложилась такая практика, что производителями операционных систем и систем управления базами данных являются разные фирмы или организации, что делает такой симбиоз невозможным. Поэтому политика безопасности частично должна поддерживаться самой базой данных. Аналогичные проблемы возникают при модернизации защищенных систем и могут быть сформулированы как противоречие между единой оценкой защищенности всей системы и многопрофильностью подсистем, создаваемых различными производителями. С аналогичной проблемой мы сталкивались в главе VI. Однако теперь существенным образом возникает зависимость частей защиты друг от друга (ТСВ вычислительной системы управляет субъектами системы защиты (ТСВ), поддерживающей политику безопасности базы данных). Такая структура не вкладывается в систему независимых защищенных компонент распределенной сети. В параграфе 7.1 приводится описание модели ТСВ-подмножеств, иерархически связанных друг с другом, которые при выполнении некоторых дополнительных условий могут удовлетворять условиям ТСВ системы. В параграфе 7.2 приведены наиболее распространенные архитектуры защищенных баз данных, в которых теория параграфа 7.1 может быть реализована.

7.1. ИЕРАРХИЧЕСКИЙ МЕТОД ПОСТРОЕНИЯ ЗАЩИТЫ .

В данном параграфе рассматривается еще один пример иерархической декомпозиции сложных систем. Если в параграфе 1.2 мы рассматривали примеры, не связанные непосредственно с задачей защиты, то сейчас основное внимание будет посвящено иерархическому построению ТСВ в электронных системах обработки данных. Как и раньше, основная задача

ТСВ - поддержка монитора обращений. Однако, в отличие от теории предыдущего параграфа, где мониторы обращения были независимыми, сейчас мы покажем, что одни ТСВ-подмножества могут использовать ресурсы в других ТСВ-подмножествах. А именно, рассмотрим следующую модель.

Определение. ТСВ-подмножество М есть совокупность программно-аппаратных ресурсов системы, которые управляют доступами множества S субъектов к множеству О объектов на основе четко определенной политики Р и удовлетворяет свойствам:

1) М определяет и контролирует каждый доступ кобъектам из О со стороны субъектов из S;

2) М гарантировано защищено;

3)М достаточно просто устроено, чтобы существовала возможность проанализировать его системой тестов, полнота которых доказана.

Зависимость ТСВ-подмножеств состоит в том, что М использует ресурсы точно определенного множества более примитивных ТСВ-подмножеств (то есть предполагается заданным некоторый частичный порядок на ТСВ-подмножествах), для того, чтобы создать объекты из О, создать и поддерживать структуры данных и поддерживать политику Р.

Если более примитивных ТСВ-подмножеств нет, то такое ТСВ-подмножество опирается в решении тех же задач на аппаратную часть.

Отметим, что кроме монитора обращений необходимо существование механизма поддержки этого монитора. Рассмотрим условия, при которых из ТСВ-подмножеств удается собрать ТСВ системы (то есть доказать, что выполняются все требования к ТСВ).

Пусть даны ТСВ-подмножества М(i), которые управляют доступом субъектов S(i) к объектам 0(i) в соответствии с политикой безопасности Р(i). Предположим, что нет объектов в системе, которые контролируются более, чем одним ТСВ-подмножеством. Здесь принимается та же точка зрения, как в главе VI, состоящая в том, что ТСВ-подмножества могут экспортировать объекты, подлежащие управлению другим ТСВ-подмножеством. Однако принятый и переданный объекты - это разные объекты, контролируемые разными ТСВ-подмножествами. При этом политика безопасности Р(i) ТСВ-подмножества М(i) может отличаться от политики безопасности Р(j) ТСВ-подмножества М(j). Однако все вместе должны составлять единую политику Р системы, причем каждое правило из Р должно поддерживаться определенной системой ТСВ-подмножеств.

Необходимо также предполагать, что каждый доверенный субъект, то есть субъект, который может нарушать правила Р(i) является частью ТСВ-подмножества М(i).

Рассмотрим ограничения на зависимости ТСВ-подмножеств.

Определение. ТСВ-подмножество А прямо зависит (в своей правильности) от ТСВ-подмножества В тогда и только тогда, когда доводы о подтверждении правильности А (верификация установки А обозначается vA) частично или полностью основаны на предположении, что В установлена верно в соответствии соспецификацией В (обозначается sB).

Определение. ТСВ-подмножество А менее примитивно, чем ТСВ-подмножество В, если:

а) А прямо зависит от В;

б) существует цепочка ТСВ-подмножеств от А к В такая, что каждый элемент цепи прямо зависит от предыдущего элемента цепи.

Рассмотрим примеры, поясняющие понятие зависимости, взятые из "Розовой книги".

Пример 1. Пусть ТСВ-подмножество В предоставляет услугу в виде "файла", которым В управляет в соответствии с политикой Р(В), а ТСВ-подмножествоА использует В-файл для хранения информации. Если vA использует факт, что различные В-файлы действительно различаются и

доступ к ним определяется политикой P(B), то vA полагается на факт, что sB и B соответствуют друг другу. Тогда A прямо зависит от B.

Пример 2. Пусть A и B взаимно не доверяющие друг другу системы бронирования авиабилетов, расположенные отдельно и принадлежащие различным организациям. Предположим sA и sB дают возможность получить заказ на бронирование по сети от других систем, используя взаимно согласованный протокол. Пусть этот протокол полностью определен и соответствует sA и sB. Пусть также vA и vB с заданной степенью уверенности подтверждают, что A и B соответствуют своим спецификациям sA и sB. A не зависит в правильности своей установки от правильности установки B, так как sA - полная, то есть какая бы последовательность бит не пришла от B, sA определяет, что A должна делать, а vA демонстрирует, что именно это и делается. Аналогично, B не зависит от правильности A. Поэтому A и B не зависят одна отдругой.

Пример 3. Пусть A является сервером электронной почты, а B управляет запросами в базу данных. Спецификация sA может совсем не упоминать систему управления базой данных, она просто определяет интерфейс почтовой системы. Однако в vA мы находим, что программа обеспечения A использует таблицы, предоставляемые B, для хранения сообщений A и B на различных, связанных машинах. Ни sB, ни vB не упоминают о почтовой системе. Как и в предыдущем примере, sB полностью определяет поведение B для всех получаемых последовательностей бит. Здесь A прямо зависит от B, но B не зависит от A. Отметим, что информационные потоки в обоих направлениях являются законными и никоим образом не компрометируют целостность B. Зависимость находится в другой плоскости от потока данных.

Этот пример замечателен тем, что анализ структуры элементов не позволяет выявить существование зависимости, при этом архитектура системы внешне совпадает с приведенной в примере 2 архитектурой взаимно независимых систем. Кроме того, этот пример показывает, что описания интерфейса не достаточно для выявления зависимостей.

Пример 4. Пусть A и B взаимно зависимые системы. A зависит от B и B зависит от A. Это значит, что правильность установки vA доказывается из предположения, что B установлена правильно в соответствии с sB. Также правильность установки vB доказывается из предположения, что A установлена правильно в соответствии с sA. Пусть vA и vB подтверждают правильность установки A и B. Однако отсюда не следует, что A и B функционируют правильно.

В самом деле, если A и B функционируют правильно относительно sA и sB, то vA и vB поддерживают правильность установки. Если A установлено неправильно по отношению к sA и B установлено неправильно по отношению к sB, то никто не мешает возникновению ситуации, когда vA подтверждает правильность исходя из того, что B не соответствует sB. Аналогично, vB подтверждает правильность установки sB, хотя A не соответствует sA. Тогда можно доказать, что vA и vB подтверждают правильность A и B, хотя A и B установлены неверно.

Для того, чтобы понять возникающую здесь коллизию, рассмотрим две системы бронирования билетов на самолеты A и B, как в примере 2. Предположим, что A содержит информацию об исходных пунктах и времени вылета всех полетов в США, а B - в Европе. Пусть sA включает утверждение, что A обеспечивает пассажирам услугу в подборе вылета по транзиту, минимизирующем время ожидания следующего полета. Пусть sB предлагает аналогичные услуги. Из утверждения A соответствует sA и B соответствует sB можно вывести истинность утверждения, что A соответствует спецификации sA в предоставлении услуги подбора транзитного рейса в Европе и Америке. Аналогичное утверждение можно вывести для B. Однако, если A хранит информацию в местном времени, а B - во времени по Гринвичу, то транзитный маршрут, составленный A и B на основе информации, полученной друг от друга, будет неправильным из-за различия во времени. То есть каждая система функционирует правильно, но результат неверен. Это происходит из-за того, что A и B зависимы.

Гарантии защищенности ТСВ-подмножеств имеют большое значение в проблеме построения единой ТСВ системы из ТСВ-подмножеств. В случае зависимых ТСВ-подмножеств менее примитивное ТСВ-подмножество может использовать объекты и субъекты, предоставленные более примитивным ТСВ-подмножеством. Потому первая проблема состоит в доказательстве того факта, что невозможны никакие изменения данных, критических для политики безопасности, или кодов ТСВ-подмножества. То есть никакая внешняя по отношению к ТСВ-подмножеству система (кроме, может быть, более примитивных ТСВ-подмножеств) не может инициировать произвольное изменение кодов ТСВ-подмножества или его структур данных. Вторая проблема состоит в доказательстве того, что данные, хранящиеся в ТСВ-подмножестве и критичные для политики безопасности, не могут изменяться иначе, чем в соответствии с логикой ТСВ-подмножества. Разумеется, эти доказательства возможны при условии правильности той информации, которая вносится в ТСВ-подмножество более примитивным ТСВ-подмножеством.

Ясно, что в доказательстве возможности построения единой ТСВ системы из ТСВ-подмножеств присутствуют условия не только на локальные свойства ТСВ-подмножеств, но также интегральные требования.

Суммируем перечень всех условий, которые необходимо выполнить, чтобы из семейства ТСВ-подмножеств можно было бы синтезировать ТСВ системы. Если, кроме нижеприведенных требований, выполняются требования какого-либо класса в стандарте "Оранжевая книга", то построенная таким образом система может быть аттестована по соответствующему классу.

Условия:

1. ТСВ-подмножества четко определены.
2. Политика безопасности системы распределена по ТСВ-подмножествам.
3. Каждое ТСВ-подмножество M(i) включает доверенные процессы по отношению к своей политике безопасности P(i).
4. Архитектура ТСВ-подмножеств четко определена.
5. Все ТСВ-подмножества занимают различные домены.
6. Во всех случаях примитивные ТСВ-подмножества поддерживают правильное функционирование монитора обращений в менее примитивном ТСВ-подмножестве.

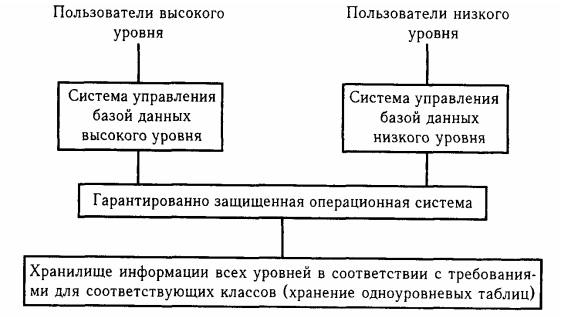
7.2. ГАРАНТИРОВАННО ЗАЩИЩЕННЫЕ БАЗЫ ДАННЫХ.

Будем предполагать, что гарантированно защищенная база данных может быть инсталирована только на платформе гарантированно защищенной вычислительной системы. Тогда ТСВ всей системы получается из ТСВ-подмножества вычислительной системы и ТСВ-подмножества базы данных. Причем в возникающей здесь иерархической структуре ТСВ-подмножество вычислительной системы является более примитивным,чем ТСВ-подмножество базы данных.

Рассмотрим несколько примеров архитектур баз данных, позволяющих говорить о гарантированной защищенности.

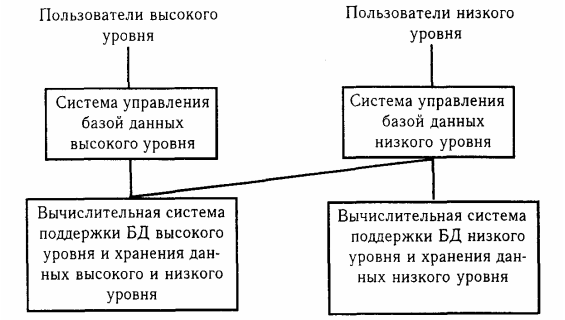
Пример 1. Одноуровневая база данных. Пусть все пользователи имеют одинаковый допуск ко всей информации, содержащейся в базе данных и в вычислительной системе. В этом случае возможно применение базы данных общего назначения на охраняемой вычислительной системе. В случае высокого класса хранящейся в базе данных информации, согласно "Требованиям" к применению "Оранжевой книги" должна использоваться система класса C2 (наличие аудита).

Пример 2. Многоуровневая база данных с ядерной архитектурой. Эта архитектура подробно рассматривалась в параграфах 1.6-1.7, 5.5 и в ряде примеров. Исследования по данной архитектуре проводились System Research Institute группой ученых под руководством D.Denning и T.Lunt. Цель исследований: построить гарантированно защищенную базу данных по классу A1. Данная архитектура может быть представлена следующей схемой.



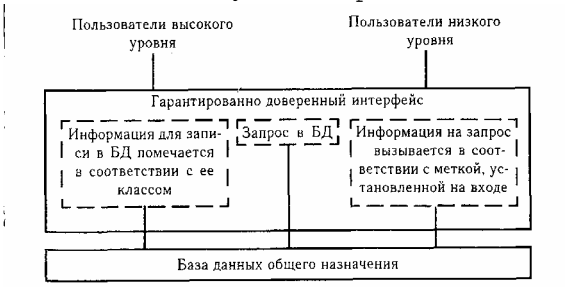
На этой схеме ТСВ вычислительной системы разрешает формировать отношения, содержащие информацию, разрешенную для данного класса пользователя. Сама система управления базой данных является системой общего пользования. Однако в данной архитектуре серьезной проблемой является полиинстантинация. Вместе с тем, здесь на системном уровне видно, как доказывать, что база данных поддерживает MLS-политику (на уровне ядра реализована модель невлияния G - M).

Пример 3. Модификацией архитектуры примера 2 является дублирующая архитектура. Она основана на гарантированно правильном разделении пользователей.



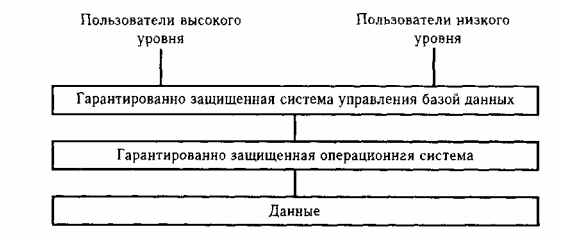
Фактически система представляет композицию одноуровневых баз данных примера 1. Здесь также на системном уровне видно, как доказывать защищенность системы.

Пример 4. Исторически одной из первых была предложена архитектура "базы данных с замком целостности". Это разработка ВВС США, анонсированная в1983 г. Схематически данная архитектура может быть представлена следующим образом.



Недостатком этой архитектуры является то, что *-свойство приходится нарушать почти в каждой транзакции.

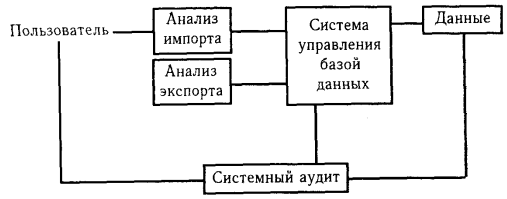
Пример 5. Базы данных, которые американцы называют архитектурой с "гарантированно защищенным субъектом". Эти системы предполагают наличие ТСВ-подмножеств в самой базе данных, причем ТСВ базы данных реализует свою составляющую политики безопасности.



К этому классу относятся системы управлениябазами данных ORACLE, INFORMIX, INGRES и т.д.

Основная сложность в том, что описания политик безопасности и доказательства их поддержки недоступны. Кроме того, большой объем программного обеспечения не позволяет провести достаточный анализ.

Пример 6. Базы данных с распределенным доменом защиты. Используется идея "Красной книги" о распределенной NTCB, как совокупность TCB компонент. Этот подход можно изобразить на схеме.



Каждая компонента на этой схеме гарантировано защищена.

Литература

I. Department of Defense Trusted Computer System Evaluation Criteria, DoD, 1985.

2. Computer Security Requirements. Guidence for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, DoD, 1985.

3. Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC, 1987.

4. Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria, NCSC, 1991.

5. D. Denning, Cryptography and Data Security, Addison Wesley, Reading (MA), 1983.

6. Data Communications and Networks 3, Edited byR.L. Brewster, Redwood Books, Trowbridge, Wiltshire. 1994.

7. D.Russell, G.T.Gangemi Sr., Computer Security Basics, O'Reilly & Associates, Inc., 1991.

8. D. Denning, An Intrution Detection Model, IEEE Trunsactions on Software Engineering, v. SE-13,№ I, 1987, pp 222-232.

9. J.A. Goguen, J. Meseguer, Securiti Policies and Securiti Models, Proceedings of the 1982 Sympsium
on Security and Privacy, IEEE, April 20-21, 1982.Oakland, CA, IEEE Catalog №82CHI753, pp. I 1-26.

10. J. McLean, Reasoning About Security Models, Proceedings, IEEE Symposium on Privasy and Security, Oakland, CA, April 27-29, 1987, IEEE Computer Society Press, 1987, pp. 123-131.
11. J Fellows, J. Hemenway, The Architecture of a Distributed Trusted Computing Base, Proceedings, 10th National Computer Security Conference, Baltimore, MD, September 21-24, 1987, National Bureau of Standards/National Computer SecurityCenter, 1987, pp. 68-77.
12. D. Denning, et al., Views for Multilevel Database Security, IEEE Trunsactions on Software Engineering, v. SE-13, № 2, 1987, pp 129-140.
13. D. Denning, et al., Multilevel Relational DataModel, Proceedings, IEEE Symposium on Privasy and Security, Oakland, CA, April 27-29, 1987, IEEE Computer Society Press, 1987, pp. 220-242.
14. R.R Kenning, S.A Walker, Computer Architecture and Database Security, Proceedings, 9th National Computer Security Conference, Gaithersburg, MD, September 15-18, 1986, National Bureau of Standards/National Computer Security Center,1986, pp. 216-230.
15. R. Schell, D. Denning, Integrity in Trusted Database Systems, Proceedings, 9th National Computer Security Conference, Gaithersburg, MD, September 15-18, 1986, National Bureau of Standards/National Computer Security Center,1986, pp. 30-36.
16. S.G. Aki, D. Denning, Checking Classification Constraints for Consistency and Completeness, Proceedings, IEEE Symposium on Privasy and Security, Oakland, CA, April 27-29, 1987, IEEE Computer Society Press, 1987, pp. 196-201.
17. T-A. Su, G. Ozsoyoglu, Data Dependencies and Inference Control in Multilevel Relational Database Systems, Proceedings, IEEE Symposiumon Privasy and Security, Oakland, CA, April 27-29,1987, IEEE Computer Society Press, 1987, pp. 202-211.