СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

Н. А. Богульская, М. М. Кучеров

МОДЕЛИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Учебное пособие

ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Министерство науки и высшего образования Российской Федерации
Сибирский федеральный университет

Н. А. Богульская, М. М. Кучеров

МОДЕЛИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Учебное пособие

Красноярск
СФУ
2019

УДК 004.4.056(07)
ББК 32.973.2я73
Б748

Р е ц е н з е н т ы:
В. Е. Зобов, доктор физико-математических наук, главный научный сотрудник ИФ СО РАН;
А. А. Родионов, доктор физико-математических наук, профессор кафедры МАиДУ СФУ

Богульская, Н. А.
Б748 Модели безопасности компьютерных систем : учеб. пособие / Н. А. Богульская, М. М. Кучеров. – Красноярск : Сиб. федер. ун-т, 2019. – 206 с.

ISBN 978-5-7638-4008-7

Изложены основные понятия, стандарты и критерии информационной безопасности. Описаны классические модели безопасности компьютерных систем.

Предназначено для студентов, обучающихся по специальности 10.05.01 «Компьютерная безопасность».

Электронный вариант издания см.:
http//catalog.sfu-kras.ru

УДК 004.4.056(07)
ББК 32.973.2я73

ISBN 978-5-7638-4008-7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информация – это сведения о фактах, событиях, процессах и явлениях, о состоянии объектов (их свойствах, характеристиках) в некоторой предметной области, воспринимаемые человеком или специальным устройством и используемые (необходимые) для оптимизации принимаемых решений в процессе управления данными объектами.

Введем основные понятия, которые будем использовать в учебном пособии.

Информация может существовать в различных формах в виде совокупностей некоторых знаков (символов, сигналов и т. п.) на носителях различных типов. В связи с развивающимся процессом информатизации общества все большие объемы информации накапливаются, хранятся и обрабатываются в автоматизированных системах, построенных на основе современных средств вычислительной техники и связи. В дальнейшем будут рассматриваться только те формы представления информации, которые используются при ее автоматизированной обработке.

Одним из основополагающих понятий курса является информационная безопасность, это понятие в разных контекстах может иметь различный смысл. В Доктрине информационной безопасности Российской Федерации термин «информационная безопасность» используется в широком смысле. Имеется в виду состояние защищенности национальных интересов в информационной сфере, определяемых совокупностью сбалансированных интересов личности, общества и государства.

В Законе РФ «Об участии в международном информационном обмене» информационная безопасность определяется аналогичным образом – как состояние защищенности информационной среды общества, обеспечивающее ее формирование, использование и развитие в интересах граждан, организаций, государства.

Под *информационной безопасностью* мы будем понимать защищенность информации от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести неприемлемый ущерб субъектам информационных отношений, в том числе владельцам и пользователям информации и поддерживающей инфраструктуры. Защита информации – это комплекс мероприятий, направленных на обеспечение информационной безопасности.

Таким образом, правильный с методологической точки зрения подход к проблемам информационной безопасности начинается с выявления

субъектов информационных отношений и интересов этих субъектов, связанных с использованием автоматизированных систем. Угрозы информационной безопасности – это оборотная сторона использования информационных технологий.

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

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

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

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

Цель защиты систем обработки информации – противодействие угрозам безопасности. Следовательно, безопасная или защищенная система – это система, обладающая средствами защиты, которые успешно и эффективно противостоят угрозам безопасности.

Безопасность – это свойство информационной системы, характеризующее взаимодействие данного программного продукта с окружающей его средой. Производительность, масштабируемость, надежность являются непосредственными свойствами конкретных программных решений, в отличие от них безопасность зависит от взаимодействующих с системой

механизмов и может изменяться. Таким образом, безопасность является условной характеристикой.

Можно определить связь безопасности и надежности. Под надежностью понимается устойчивость к сбоям, вызванным внешними и/или внутренними причинами. В отличие от безопасности надежность не зависит напрямую от внешних воздействий. Когда говорят о безопасности, понимается, что процесс не должен быть связан с угрозами, приводящими к нежелательным последствиям.

Существует три подхода к информационной безопасности: нормативный, теоретический и практический (экспериментальный).

Нормативный подход появился в 80-е гг. XX в. Название подхода происходит от слова «норма», означающего некий эталон. Существуют различные стандарты информационной безопасности, документы в которых определяют признаки, свойства и требования к безопасным информационным системам, а также шкалу, с помощью которой все системы можно оценить на предмет безопасности.

Теоретический подход основан на построении модели безопасности – некоего абстрактного представления реальной системы с точки зрения безопасности. Полученные модели должны быть теоретически и математически обоснованы. Чем сильнее математическая и теоретическая база построенной модели, тем безопаснее система.

Зарождение *практического* подхода определения безопасности можно связывать с началом эпохи Интернета. Безопасность системы при этом проверяется на практике: одна система более безопасна, чем другая, если она более устойчива к внешним воздействиям и лучше противостоит угрозам.

Из вышеизложенного можно сделать следующий вывод: ни один из вышеперечисленных подходов не является исчерпывающим, поэтому при определении безопасности информационной системы используются все три подхода.

Цель данного курса – выяснить, что скрывается за понятием «информационная безопасность», так как без конструктивного определения этого понятия невозможно рассматривать основные принципы функционирования защищенных систем и технологии их создания.

1. НОРМАТИВНЫЙ ПОДХОД К ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

1.1. КЛАССИЧЕСКИЕ СТАНДАРТЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

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

Для ответа на эти вопросы и согласования всех точек зрения на проблему создания защищенных систем разработаны и продолжают разрабатываться стандарты информационной безопасности. Это документы, регламентирующие основные понятия и концепции информационной безопасности на государственном или межгосударственном уровне, определяющие понятие «защищенная система» посредством стандартизации требований и критериев безопасности, образующих шкалу оценки степени защищенности ВС.

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

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

1.2. РОЛЬ СТАНДАРТОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

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

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

Уровень 1. Система для обработки информации с грифом не выше «для служебного пользования».

Уровень 2. Система для обработки информации с грифом не выше «секретно» и т. д.

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

Производители также нуждаются в стандартах как средстве сравнения возможностей своих продуктов и в применении процедуры сертификации как механизма объективной оценки их свойств, а также в стандартизации определенного набора требований безопасности, который мог бы ограничить фантазию заказчика конкретного продукта и заставить его выбирать требования из этого набора. С точки зрения производителя требования должны быть максимально конкретными и регламентировать необходимость применения тех или иных средств, механизмов, алгоритмов и т. д. Кроме того, требования не должны противоречить существу-

ющим парадигмам обработки информации, архитектуре вычислительных систем и технологиям создания информационных продуктов. Этот подход также не может быть признан в качестве доминирующего, так как не учитывает нужд пользователей (а ведь это главная задача разработчика) и пытается подогнать требования защиты под существующие системы и технологии, а это далеко не всегда возможно осуществить без ущерба для безопасности.

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

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

Стандарт – это договоренность группы людей, организаций и т. д. Стандарт – не обязательно истина, но обязательно будет восприниматься

всеми как эталон, а следовательно, одинаково. Стандарт позволяет договориться о том, какие технологии, средства являются безопасными, а какие – нет.

Стандарт должен:

1) определить, что понимать под безопасностью, и ввести концептуальную базу;

2) выдвинуть набор требований к безопасной системе;

3) предложить шкалу оценки безопасности, определенной в первой части стандарта.

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

«*Оранжевая книга*» является первым стандартом информационной безопасности, он был разработан Министерством обороны США в 1983 г.

К области применения «Оранжевой книги» относились системы военного и государственного назначения. Стандарт был составлен для больших информационных систем (mainframe), оснащенных терминалами и информационными хранилищами, систем, предназначенных для совместной работы нескольких пользователей.

Под безопасностью понимается выполнение правил обработки и категорирования информации, принятые в организации. Информация определяется как конфиденциальная (только определенные люди имеют доступ к информации) и категорированная (существуют различные категории информации, и доступ к ним осуществляется в зависимости от обладания привилегиями). Таким образом, в данном стандарте безопасность можно определить как доступ к информации только для авторизованных пользователей.

С точки зрения «Оранжевой книги» система является безопасной, если она осуществляет обработку информации в соответствии с правилами, принятыми в данном учреждении, а также противодействует угрозам нарушения конфиденциальности, целостности и доступности информации.

В данном стандарте определена классификация безопасных информационных систем (A, B, C, D). В каждом классе свои функции безопасности (контроль доступа, аутентификация, аудит и т. д.), которые определяются с учетом приведенных угроз. Также существуют так называемые assurance – набор мер, подтверждающих, что все методы обеспечения безопасности реализованы корректно.

Требования определяют, какие функции обеспечения безопасности можно считать достаточными для данного класса. Таким образом, формируется следующая шкала:

1) D – минимальная защита. Этот класс является пустым, т. е. не содержит никаких требований к функциям защиты, следовательно, любая система может быть сертифицирована по данному классу;
2) C – дискреционная защита;
3) B – мандатная защита;
4) A – верифицированная защита;

Рассмотрим классы более подробно.

Требования к системам **класса C** формируются на основе требований к дискреционной защите и дискреционному контролю доступа. Для осуществления контроля доступа субъектов к объектам нужно определить, от кого к кому осуществляется доступ. Поэтому необходимы функции идентификации и аутентификации сущностей в системе.

Существуют подклассы C1 и C2, отличающиеся гранулярностью защиты, т. е. тем, насколько детально будет осуществляться контроль доступа. Так как часто доступ к ресурсу системы может контролироваться на разных уровнях, например, доступ к дереву файлов и каталогов системы и доступ к отдельному файлу, доступ к базе данных в целом и доступ к отдельной записи БД.

В подклассе C2 выдвинуты требования регистрации и учета событий, а также выделения ресурсов. Аудит хотя и не является средством защиты, однако тесно связан с контролем доступа. Так как в некоторых случаях средства защиты системы могут дать сбой, то необходим механизм для получения информации об операциях, приведших к сбою, для противодействия подобным угрозам в дальнейшем.

В качестве правил доступа к системам **класса B** используются правила модели Белла – ЛаПадулы. Существуют три подкласса B1, B2, B3, отличающиеся тем, на что распространяется контроль доступа.

Класс A отличается тем, что требования гарантированности (assurance) для реализованной системы должны быть доказаны формально. Тестирование информационной системы сложнее ее реализации. Задача тестирования включает среду функционирования и входные/выходные данные, связанные с алгоритмом. Таким образом, задачи достижения гарантированности начинают превосходить по сложности задачи достижения требуемой функциональности системы. Так как в данном случае надо доказать, что система не только делает то, что должна, но и не делает того, чего не должна делать. Поэтому необходимо уделять внимание верификации на всех этапах жизненного цикла системы.

На практике большинство систем сертифицировано по классу C и гораздо меньше по классу B.

Появление стандарта позволило:

1) потребителю и производителю использовать единую терминологию;

2) потребителю формулировать конкретные требования к необходимому им продукту информационных технологий (ИТ-продукту);

3) сравнивать уровень безопасности существующих ИТ-продуктов по единой шкале;

4) производителю использовать процедуру сертификации для получения объективной оценки свойств разработанных продуктов.

Руководящие документы при президенте РФ были выпущены в 1992 г. Гостехкомиссией при президенте РФ (в настоящее время Федеральная служба по техническому и экспортному контролю – ФСТЭК).

Существует несколько категорий информации:

1. Государственная тайна – информация, имеющая гриф «секретно».

2. Конфиденциальная информация – информация ограниченного распространения.

3. Персональные данные. Конфиденциальность персональных данных человека должна быть обеспечена теми, кому доверяет этот человек. Существует закон о персональных данных.

В Руководящих документах основной угрозой считается несанкционированный доступ к информации (НСД).

Основными документами являются следующие:

1) «Концепция защиты средств вычислительной техники от несанкционированного доступа к информации»;

2) «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации»;

3) «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требований по защите информации».

Руководящие документы предлагают две группы критериев: показатели защищенности средств вычислительной техники (СВТ) от несанкционированного доступа (НСД) и критерии защищенности автоматизированных систем (АС).

Автоматизированные системы – комплексные системы, состоящие из средств вычислительной техники и направленные на решение прикладных задач. Примерами являются автоматизированные системы управления; учебный класс информатики (решает задачу обеспечения учебного процесса).

Для оценки безопасности АС требуется оценить безопасность ее компонентов.

Существуют семь классов защищенности СВТ от НСД. Самый низкий – седьмой, самый высокий – первый.

В документе «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации» приведены следующие показатели защищенности СВТ от НСД:

1. Дискреционный принцип контроля доступа.
2. Мандатный принцип контроля доступа.
3. Очистка памяти (используется для мандатного КД).
4. Изоляция модулей.
5. Маркировка документов – расстановка грифов.
6. Защита ввода/вывода на отчужденный носитель информации.
7. Сопоставление пользователя с устройством.
8. Идентификация и аутентификация.
9. Гарантии проектирования (assurance): документирование, тестирование, представление подтверждающей документации.
10. Регистрация событий.
11. Взаимодействие пользователя с КСЗ.
12. Надежное восстановление (действия, предпринимаемые при возникновении сбоя).
13. Целостность КСЗ.
14. Контроль модификации сертифицированного продукта (при нахождении ошибок нужен механизм, позволяющий модифицировать продукт).
15. Контроль дистрибуции.
16. Гарантии архитектуры – формальное доказательство.
17. Тестирование.
18. Руководство пользователя.
19. Конструкторская и проектная документация. При этом в четвертом классе защищенности появляется мандатный контроль. Гарантии архитектуры появляются только в первом классе.

Рассмотрим *классы защищенности автоматизированных систем*. Группа АС определяется на основании следующих признаков:

- наличие информации различного уровня конфиденциальности;
- уровень полномочий пользователей на доступ к информации;
- индивидуальный/коллективный доступ к информации.

Соответственно выделяют три группы АС, в переделах каждой группы определена иерархия классов защищенности АС.

Третья группа, содержащая классы 3А и 3Б, включает в себя АС, где работает один пользователь, допущенный ко всей информации, размещенной на носителях одного уровня конфиденциальности.

Вторая группа, содержащая классы 2А и 2Б, включает АС, в которых пользователи имеют одинаковые полномочия доступа к информации на носителях различного уровня конфиденциальности.

Первая группа, содержащая классы 1Д, 1Г, 1В, 1Б и 1А, включает многопользовательские системы, содержащие информацию различных уровней конфиденциальности. Не все пользователи имеют одинаковые права доступа.

Требования к средствам защиты АС от НСД сгруппированы вокруг реализующих их подсистем: управления доступом, регистрации и учета, криптографической подсистемы, а также подсистемы обеспечения целостности. В пределах каждой группы выделены подгруппы требований (табл. 1.1).

Таблица 1.1

Требования к подсистемам средств защиты АС от НСД

Подсистемы средств защиты АС от НСД	Требования к подсистемам
Управление доступом	Идентификация, проверка подлинности и контроль доступа субъектов в систему, к терминалам, ЭВМ, к программам, к томам, каталогам, файлам, записям, полям записей. Управление потоками информации
Регистрация и учет	Регистрация и учет входа/выхода субъектов доступа в/из системы (узла сети); выдача графических выходных документов; запуск/завершение программ и процессов; доступ программ к защищенным файлам, включая их создание и удаление, передачу по линиям и каналам связи; доступа программ к терминалам ЭВМ, узлам сети, каналам связи ЭВМ, программам, каталогам, файлам, записям, полям записей. Учет носителей информации. Очистка (обнуление, обезличивание) освобождаемых областей памяти и внешних накопителей. Сигнализация попыток нарушения защиты
Криптографическая подсистема	Шифрование конфиденциальной информации. Шифрование информации, принадлежащей различным субъектам доступа (разным группам) на различных ключах. Использование сертифицированных криптографических средств
Обеспечение целостности	Обеспечение целостности программных средств и обрабатываемой информации. Физическая охрана СВТ и носителей информации. Наличие администратора ЗИ или службы ЗИ. Периодическое тестирование СЗИ НСД. Наличие средств восстановления СЗИ НСД. Использование сертифицированных средств ЗИ

Мы не можем численно сравнивать классы защищенности информационных систем. Классы АС различаются в зависимости от того, однородна ли информация в системе, много ли пользователей в системе и совпадают ли права доступа пользователей. Класс защищенности системы определяется тем требованием, которое хуже всего реализовано.

Требования носят обобщенный характер, поэтому не все требования могут быть применимы.

Все требования к подсистеме управления доступом и к подсистеме регистрации/учета действуют полностью начиная с класса В. Все требования к криптографической подсистеме действуют полностью, начиная с класса Б.

Контрольные вопросы

1. Что такое «Оранжевая книга»?
2. В чем заключаются отличия класса А от класса В стандарта «Оранжевая книга»?
3. Какие руководящие документы по вопросу информационной безопасности действуют в РФ?
4. Какие существуют категории информации согласно руководящим документам РФ?
5. На основании каких признаков выделяется группа АС?

2. ЕДИНЫЕ КРИТЕРИИ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

2.1. ГОСТ Р ИСО 15408

Важное место в системе стандартов в области информационной безопасности занимает стандарт ГОСТ Р ИСО 15408.

Разработкой данного стандарта занималась международная организация по стандартизации ISO. Стандарт разрабатывался в 1993–1999 гг., первая версия появилась в 1996 г. В данном документе рассматривается не только IT-продукт, но и процесс разработки. Пользователями данного документа являются три группы специалистов: потребители, верификаторы (evaluators) и разработчики.

Потребитель, пользуясь положениями стандарта, выдвигает требования к разработчику, а также оценивает существующие продукты с помощью стандарта.

Разработчики должны использовать стандарт как базу, терминологический словарь. Стандарт является платформой для общения между потребителем и разработчиком.

Эксперты по квалификации используют стандарт в качестве критерия определения соответствия средств защиты ИТ-продукта требованиям, предъявленным к нему потребителями, и угрозам, действующим в среде эксплуатации продукта.

Безопасность – совокупность конфиденциальности, целостности и доступности информации, обрабатываемой IT-продуктами; также ставится требование безопасности продукта в среде (при определенных условиях безопасности). Некоторые угрозы являются актуальными для определенной среды. Продукт считается безопасным, когда он отражает угрозы, считающиеся актуальными для среды его эксплуатации. Следовательно, требования к продукту выдвигаются в некотором контексте. Основными документами, описывающими все аспекты безопасности ИТ-продукта, с точки зрения пользователей и разработчиков являются соответственно Профиль защиты и Проект защиты.

Профиль защиты – документ, определяющий требования потребителя к производителю, т. е. требования к определенному классу IT-продуктов (для решения определенных задач).

Рассмотрим структуру Профиля защиты. Введение содержит идентификатор профиля и обзор содержания. Описание ИТ-продукта – краткая характеристика продукта – позволяет понять, стоит ли этот профиль применять к данному продукту. Раздел «Описание среды эксплуатации» включает описание среды функционирования с точки зрения безопасности. Он состоит из следующих подразделов:

- условия безопасности;
- угрозы безопасности – описание внешних угроз;
- политика безопасности – описание правил политики безопасности, реализованных в ИТ-продукте.

Раздел «Задачи защиты» отражает потребности потребителей в противодействии угрозам и защите IT-продукта. Он состоит из следующих подразделов:

- защита IT-продукта;
- задачи, которые могут быть решены с помощью IT-продукта.

Раздел «Требования безопасности» содержит список требований, которым должен удовлетворять ИТ-продукт. Он включает три подраздела:

- подраздел функциональных требований, в котором определены ссылки на требуемые функции стандарта;
- подраздел требований адекватности, содержащий ссылки на требования адекватности стандарта;
- подраздел требований к среде эксплуатации, которые должны предъявляться не к продукту, а к компонентам среды его эксплуатации.

Используются не все требования, а только выбранные разработчиками профиля, с учетом задач защиты.

Раздел «Дополнительные сведения» является необязательным и содержит дополнительную информацию, полезную для проектирования, разработки, анализа.

Раздел «Обоснования» состоит из следующих подразделов:

- обоснование задач защиты;
- обоснование требований безопасности. Данный подраздел показывает, что требования безопасности позволяют эффективно решить задачи защиты, поскольку есть:

а) совместимость целей, соответствие отдельных функциональных требований, соответствие задачам защиты;

б) согласованность требований безопасности;

в) оправданность выбора требований;

г) соответствие задачам защиты выбранного набора требований и уровень адекватности.

Взаимосвязь всех вышеуказанных компонентов показана на рис. 2.1.

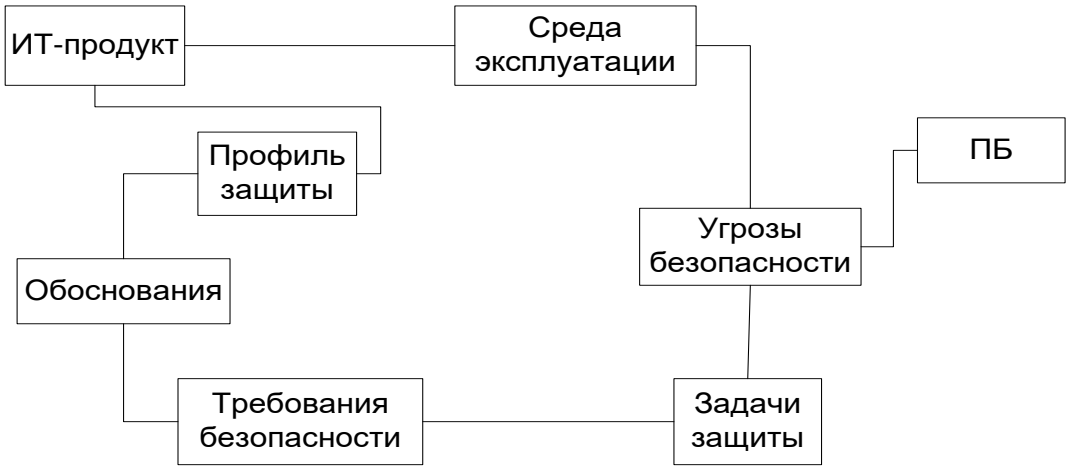

Рис. 2.1. Взаимосвязь компонентов профиля защиты

Задание по безопасности(Security Task) – это документ, описывающий программу для облегчения процесса сертификации продукта информационных технологий. Задание по безопасности дополняет Профиль защиты, так как содержит обоснование функциональных возможностей средств защиты продукта и подтверждение степени их адекватности.

Общие спецификации ИТ-продукта – это документ, описывающий механизмы осуществления задач защиты. Общие спецификации содержат подразделы спецификаций функций защиты и уровня адекватности для данного продукта.

На рис. 2.2 приведена схема взаимосвязи между компонентами, дополненная с учетом задания по безопасности.

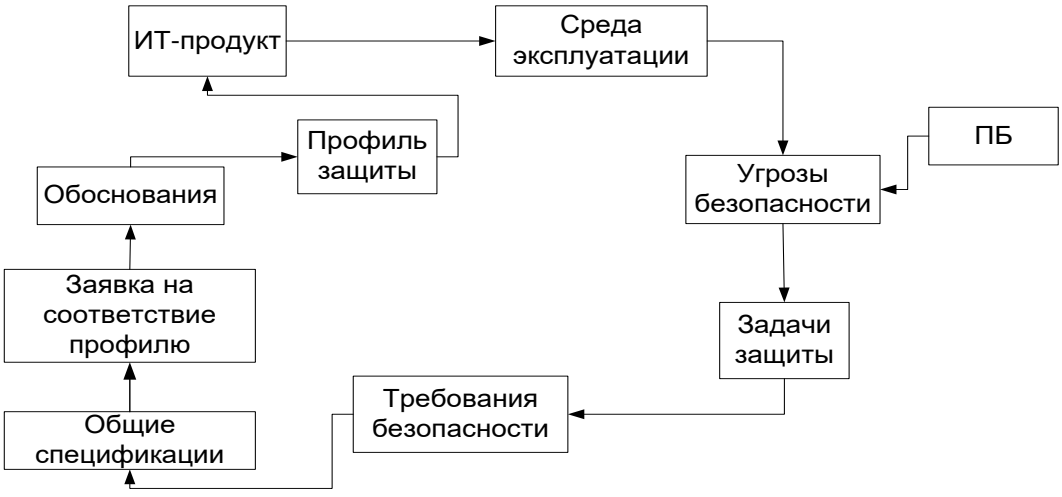

Рис. 2.2. Взаимосвязь компонентов профиля защиты
с учетом задания по безопасности

Функциональные требования единых критериев разбиты на 11 классов, которые, в свою очередь, состоят из разделов.

Классы функциональных требований: аудит; подтверждение приема/передачи информации; криптография; защита информации; идентификация/аутентификация; управление безопасностью; конфиденциальность работы в системе; надежность средств защиты; контроль за использованием ресурсов; контроль доступа к системе; прямое взаимодействие (Trusted Path).

Требования в одном разделе упорядочены. При задании Профиля защиты указывается само требование.

Рассмотрим эти классы подробнее.

1. Аудит включает автоматическое реагирование на попытки нарушения безопасности, регистрацию и учет событий, анализ протокола аудита, доступ к протоколу аудита, отбор событий для регистрации и учета, протокол аудита.

2. Подтверждение приема и передачи информации предусматривает предотвращение отречения от факта передачи информации, подтверждение факта передачи информации по требованию, автоматическое подтверждение факта передачи информации, предотвращение отречения от факта получения информации, подтверждение факта получения информации по требованию, автоматическое подтверждение факта получения информации.

3. Криптография включает управление ключами (генерация ключей заданного размера; распределение ключей; осуществление доступа к ключам; уничтожение ключей с использованием методов, определенных в стандартах) и криптографические средства.

4. Защита информации. Этот класс состоит из следующих разделов:

- политика управления доступом;
- средства управления доступом;
- аутентификация информации (подтверждение аутентификации информации, содержащейся в объекте доступа); аутентификация информации с идентификацией субъектов и осуществлением проверки подлинности;
- экспорт информации из системы (без атрибутов безопасности; вместе с атрибутами безопасности);
- политика управления информационными потоками (для частичного; для всего);
- средства управления информационными потоками (управление на основании атрибутов субъектов и объектов, на основании иерархии атрибутов безопасности (модели Белла – Лападулы); контроль скрытых информационных потоков; частичный запрет скрытых информационных потоков; полный запрет скрытых информационных потоков; мониторинг

скрытых информационных потоков и ограничение их информационной способности;

- импорт информации;
- защита информации при передаче по внутренним каналам (базовая защита; передача данных с различными атрибутами безопасности; контроль целостности; контроль целостности с различными атрибутами; уничтожение остаточной информации: для определенного множества объектов при их создании и удалении; для всех объектов при их создании и удалении);
- откат (RollBack) (По отношению к ограниченному количеству информации; По отношению ко всему количеству информации).
- контроль целостности информации в процессе хранения (обнаружение нарушений целостности; обнаружение нарушений и определение реакции на ошибки);
- защита внутрисистемной передачи информации при использовании внешних каналов;
- целостность внутрисистемной передачи информации при использовании внешних каналов (повторная передача; определение каналов информации).

5. Идентификация/Аутентификация. В этот класс входят:

- реакция на неудачные попытки аутентификации (средства идентификации и аутентификации должны перестать воспринимать попытки аутентификации после заданного количества неудач*);*
- атрибуты безопасности пользователей;
- аутентификационные параметры;
- аутентификация пользователей (обязательная аутентификация; распознование поддельных аутентификационных параметров; использование одноразовых аутентификационных параметров; использование множества параметров; повторная аутентификация);
- идентификация пользователей (обязательная идентификация; невозможность осуществления действий без идентификации; соответствие пользователей и субъектов доступа).

6. Управление безопасностью включает:

- управление средствами защиты, атрибутами безопасности, параметрами и конфигурацией средств защиты;
- отзыв атрибутов безопасности в соответствии с установленными правилами;
- ограничение времени действия атрибутов безопасности;
- требования к административным ролям (использование ролей; упорядочение ролей; предоставление ролевых полномочий по запросу пользователя).

7. Конфиденциальность работы в системе предусматривает:

- анонимность пользователей (анонимность субъектов, идентификаторов пользователей для средств защиты);
- использование псевдонимов (Alias) (контроль действий анонимных пользователей с помощью псевдонимов (гость, администратор); установление личности пользователя по псевдониму; назначение псевдонимов в соответствии с правилами);
- анонимность сеанса работы в системе;
- защита от мониторинга сеансов работы в системе (защита от мониторинга; рассредоточение критической информации между компонентами СЗ; запрет СЗ на запрос конфиденциальной информации у пользователя; мониторинг использования ресурсов системы только при наличии полномочий).

8. Надежность средств защиты предполагает:

- тестирование аппаратно-программной платформы (проверка корректности функционирования аппаратно-программной платформы);
- защита от сбоев;
- готовность средств защиты к обслуживанию удаленных клиентов (готовность к обслуживанию удаленных клиентов с заданной вероятностью (коэффициент готовности));
- конфиденциальность передаваемой информации при работе с удаленными клиентами;
- целостность передаваемой информации при работе с удаленными клиентами (обнаружение искажений; обнаружение и исправление искажений);
- защита внутренних каналов информационного обмена между средствами защиты (наличие средств безопасности; разделение трафика средств защиты и трафика приложений; контроль целостности информации при взаимодействии средств защиты);
- физическая защита (обнаружение физических атак; уведомление администратора; активное противодействие атакам);
- безопасное восстановление после сбоев (ручное; автоматическое; автоматическое с уменьшением потерь; автоматическое путем осуществления отката в безопасное состояние);
- распознавание повторных передач информации и имитации событий;
- контроль взаимодействий;
- разделение доменов (выделение отдельных доменов для СЗ; выделение отдельных доменов для функций СЗ; выделение отдельных доменов для всех функций СЗ);
- синхронизация (подтверждение приема информации; синхронизация состояний у участников в ходе осуществления обмена информацией);

- использование средств защиты надежного таймера;
- согласованность обмена информацией между средствами защиты (корректное преобразование информации при передаче);
- репликация информации, используемой СЗ;
- самотестирование средств защиты.

9. Контроль за использованием ресурсов включает:

- отказоустойчивость (обеспечение работоспособности на заданном уровне при возникновении сбоев; обеспечение нормальной работы в случае возникновения указанных сбоев);
- распределение ресурсов на уровне приоритетов (CPU time, memory, и т.д; распределение не всех ресурсов; распределение всех ресурсов);
- квотирование ресурсов.

10. Контроль доступа к системе предполагает:

- ограничение на использование атрибутов безопасности (ограничение множества атрибутов безопасности, используемого в рамках одной сессии; в зависимости от атрибутов безопасности пользователя);
- ограничение числа одновременных сеансов;
- блокировка сеанса работа с системой (вручную; автоматическое завершение сеанса);
- объявления, предупреждения, приглашения и подсказки;
- протокол сеансов работы пользователей;
- управление сеансами работы с системой.

11. Прямое взаимодействие (Trusted path) предусматривает взаимодействие между средствами защиты и доверенное взаимодействие системы с пользователем.

Требования адекватности позволяют определить, насколько выполняются требования функциональности. Мы заменяем характеристику продукта характеристикой технологии его создания.

Создание ИТ-продукта включает семь этапов (табл. 2.1):

1. Управление проектом.
2. Дистрибуция (распределение).
3. Разработка.
4. Документация.
5. Ход проекта разработки.
6. Тестирование.
7. Анализ защиты.

Используется шкала EAL – evaluated assurance scheme. Предлагается семь уровней адекватности:

- функциональное тестирование;
- структурное тестирование;
- методическое тестирование;

- методическое тестирование, разработка и анализ;
- полуформальные методы тестирования;
- полуформальные методы верификации разработки и тестирования;
- формальные методы верификации разработки и тестирования.

Требования адекватности «Единых критериев» регламентируют все этапы проектирования ИТ-продукта.

Таблица 2.1

Этапы создания ИТ-продукта

Управление проектом	
Средства управления проектом	Применение автоматизированных средств. Полная автоматизация управления проектом и контроля версий
Управление версиями	Нумерация версий. Идентификация компонентов. Контроль целостности версий. Авторизация разработчиков при обновлении версий. Контроль целостности и подлинности дистрибутива
Конфигурация проекта	Основные компоненты проекта. Обнаруженные ошибки и уязвимости. Инструментальные средства разработки
Дистрибуция – распространение	
Поставка	Регламентация поставки. Обнаружение искажений. Защита от искажений во время поставки
Установка, настройка, запуск	Регламентация установки, настройки, запуска. Протоколирование установки, настройки, запуска
Разработка	
Общие функциональные спецификации	Неформальные спецификации для СЗ. Неформальные спецификации для всех интерфейсов СЗ. Полуформальные спецификации для СЗ. Формальные спецификации для СЗ.
Архитектура защиты	Описание архитектуры. Соответствие архитектуры ПБ. Полуформальное описание архитектуры. Соответствие полуформального описания архитектуры и ПБ. Формальное описание и его соответствие ПБ
Форма представления продукта на сертификацию	Описание реализации ограниченного подмножества средств защиты. Полное описание всех СЗ. Структурированное описание всех СЗ
Структура СЗ	Использование модульного подхода. Использование иерархии модулей. Минимизация сложности структуры СЗ

Продолжение табл. 2.1

Разработка	
Частные спецификации средств защиты	Неформальные частные спецификации СЗ. Полуформальные частные спецификации СЗ. Формальные частные спецификации СЗ
Соответствие описаний различного уровня	Неформальное подтверждение соответствия. Полуформальное подтверждение. Формальное доказательство
Политика безопасности	Неформальное описание ПБ. Полуформальное описание ПБ. Формальная модель ПБ
Документация	
Руководство администратора	Описание администрирования СЗ
Руководство пользователя	Описание использования СЗ
Процесс разработки	
Безопасность среды разработки	Применение мер безопасности при разработке. Подтверждение достаточности мер безопасности
Исправление ошибок и устранение уязвимостей	Исправление ошибок и устранение уязвимостей. Регулярное исправление ошибок и устранение уязвимостей. Гарантированное исправление ошибок и устранение уязвимостей в течение заданного времени
Технология разработки	Определенная разработчиком технология разработки. Использование стандартизированной технологии. Использование технологии разработки, позволяющей оценить качество разрабатываемого продукта
Средства разработки	Использование ограниченного набора СР. Использование ограниченного набора СР, отвечающего стандарту. Использование СР, отвечающего стандарту
Тестирование	
Полнота тестирования (множество входных компонентов)	Обоснование полноты. Анализ полноты (объяснить). Строгий анализ (доказательство)
Глубина тестирования	Тестирование на уровне архитектуры. Функциональные спецификации. Реализация
Методика тестирования	Функциональное тестирование и протоколирование результатов тестов. Тестирование в соответствии с определенной методикой
Независимое тестирование (проводимое независимой стороной)	Готовность продукта к независимому тестированию. Избирательное независимое тестирование. Полное независимое тестирование

Окончание табл. 2.1

Анализ защиты	
Анализ скрытых каналов	Поиск и документирование СКУИ. Поиск СКУИ на основе определенных методов. Полный поиск СКУИ
Анализ возможностей неправильного использования СЗ	Анализ руководств администратора. Подтверждение полноты и безопасности применения руководств. Независимый анализ возможностей неправильного использования СЗ
Анализ стойкости СЗ	
Анализ продукта на наличие уязвимостей	Выявление уязвимостей разработчиком. Независимый анализ. Систематический анализ уязвимостей. Исчерпывающий анализ уязвимостей

Каждый уровень адекватности характеризуется определенным набором требований.

1. Функциональное тестирование (минимальный уровень) содержит требования к управлению версиями, установке, настройке и запуску, общим функциональным спецификациям, соответствию описаний различного уровня, руководствам администратора и пользователя, независимому тестированию и некоторые другие.

2. Структурное тестирование дополняет первый уровень:

- появление требования 2 в группе управления версиями;
- появление требования по поставке;
- архитектура защиты (требование 1);
- полнота и методика тестирования (требование 1);
- анализ стойкости и поиск уязвимостей (требование 1).

3. Методическое тестирование дополняет второй уровень:

- появление требования 3 к управлению версиями;
- конфигурация проекта (требование 1);
- архитектура защиты, полнота тестирования, независимое тестирование (требование 2);
- безопасность среды разработки, глубина тестирования, анализ возможностей неправильного использования (требование 1).

4. Методическое тестирование, разработка и анализ дополняют третий уровень:

- средства управления проектом, форма представления продукта, частные спецификации, ПБ, технология и средства разработки (требование 1);
- управление версиями (требование 4);

- конфигурация проекта, поставка, общие функциональные спецификации, анализ возможностей неправильного использования, анализ на наличие уязвимостей (требование 2).

5. Полуформальные методы тестирования дополняют четвертый уровень:

- структура средств защиты, анализ СКУИ (требование 1);
- конфигурация проекта, общие функциональные спецификации, архитектура защиты, анализ на наличие уязвимостей (требование 3);
- форма представления продукта, соответствие описаний различного уровня, ПБ, технология и средства разработки, глубина тестирования (требование 2).

6. Полуформальные методы верификации разработки и тестирования дополняют пятый уровень:

- средства управления проектом, структура средств защиты, частные спецификации средств защиты, безопасность среды разработки, методика тестирования, анализ СКУИ (требование 2);
- управление версиями (требование 5)
- архитектура защиты, анализ на наличие уязвимостей (требование 4);
- форма представления продукта, средства разработки, полнота тестирования, анализ возможностей неправильного использования (требование 3).

7. Формальные методы верификации разработки и тестирования дополняют шестой уровень до максимума:

- поставка, структура средств защиты, соответствие описаний различного уровня, ПБ, технология разработки, глубина тестирования, независимое тестирование (требование 3);
- общие функциональные спецификации (требование 4);
- архитектура защиты (требование 5).

Единые критерии являются наиболее широко используемым в мире стандартом, содержащим профиль защиты и задание по безопасности.

2.2. СЕРТИФИКАЦИЯ СРЕДСТВ ЗАЩИТЫ В РОССИЙСКОЙ ФЕДЕРАЦИИ

В РФ существует три основных службы по сертификации ИТ-продуктов: ФСТЭК, ФСБ и Министерство обороны РФ. При этом ФСБ занимается криптографическими средствами защиты, АС для органов государственной власти, средствами связи и телекоммуникационного оборудования. Министерство обороны занимается только сертификацией средств защиты, используемых им. А ФСТЭК занимается сертификацией всех остальных средств защиты.

Система сертификации в РФ приведена на рис. 2.3, где НСД – несанкционированный доступ к информации, НДВ – недокументированные возможности, РДВ – реальные декларированные возможности.

При подаче заявки на сертификацию ИТ-продукта во ФСТЭК продукт сначала передается органам сертификации, а затем тестируется в испытательных лабораториях. Результаты тестирования передаются органам сертификации, которые подтверждают соответствие ИТ-продукта профилю защиты (рис. 2.4).

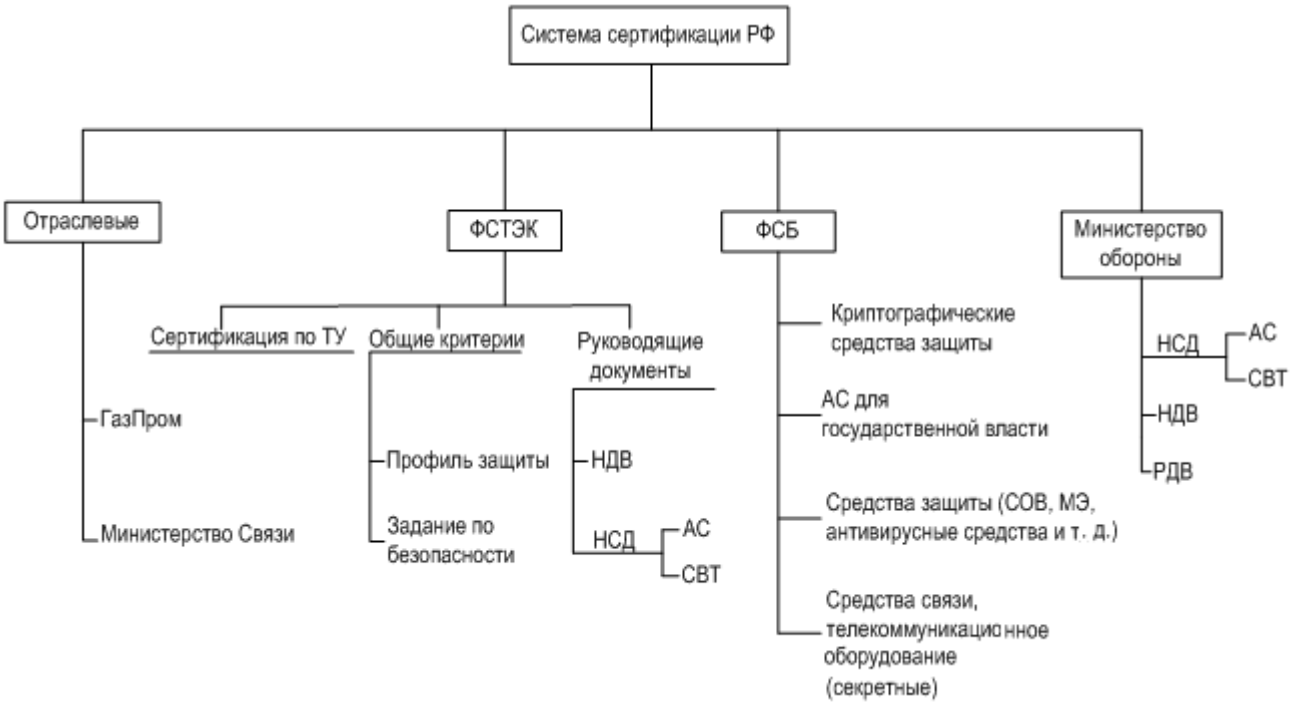

Рис. 2.3. Система сертификации в РФ

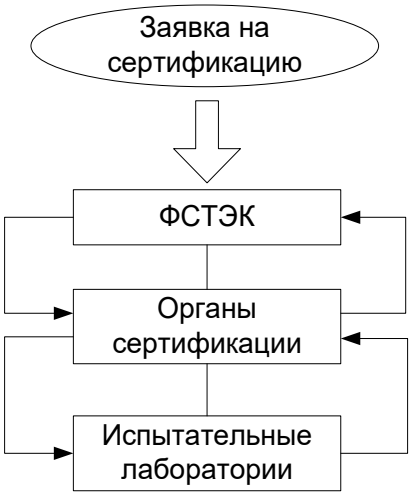

Рис. 2.4. Схема сертификации ИТ-продукта

В качестве обобщенных показателей, характеризующих стандарты информационной безопасности, предлагается использовать следующие:

1) универсальность, которая определяется множеством ИТ-продуктов, к которым применимы стандарты;

2) гибкость, т. е. возможность и удобство применения стандарта к постоянно развивающимся информационным технологиям;

3) гарантированность, которая определяется предусмотренным стандартом механизмом подтверждения выполнения требований стандарта.

4) реализуемость требований стандарта на практике;

5) актуальность – то, какие угрозы безопасности учитываются стандартом, насколько они современны.

Контрольные вопросы

1. Что такое Профиль защиты? Какова структура Профиля защиты?
2. Что позволяют определить требования адекватности?
3. Что такое задание по безопасности?
4. Какие службы сертификации ИТ-продуктов действуют в РФ?

3. ТЕОРЕТИЧЕСКИЙ ПОДХОД К ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

3.1. ФОРМАЛЬНЫЕ МОДЕЛИ БЕЗОПАСНОСТИ

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

Для чего же нужны модели безопасности? Неужели нельзя обойтись неформальным описанием политики безопасности, ведь составление формальных моделей требует существенных затрат и привлечения высококвалифицированных специалистов, они трудны для понимания и требуют определенной интерпретации для применения в реальных системах. Тем не менее формальные модели необходимы и используются достаточно широко, потому что только с их помощью можно доказать безопасность системы, опираясь при этом на объективные и неопровержимые постулаты математической теории. По своему назначению модели безопасности аналогичны аэродинамическим моделям самолетов и моделям плавучести кораблей –

и те, и другие позволяют обосновать жизнеспособность системы и определяют базовые принципы ее архитектуры и используемые при ее построении технологические решения. Основная цель создания политики безопасности информационной системы и описания ее в виде формальной модели – это определение условий, которым должно подчиняться поведение системы, выработка критерия безопасности и проведение формального доказательства соответствия системы этому критерию при соблюдении установленных правил и ограничений. На практике это означает, что только соответствующим образом уполномоченные пользователи получат доступ к информации и смогут осуществлять с ней только санкционированные действия.

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

Производители защищенных информационных систем используют модели безопасности в следующих случаях:

- при составлении формальной спецификации политики безопасности разрабатываемой системы;
- при выборе и обосновании базовых принципов архитектуры защищенной системы, определяющих механизмы реализации средств защиты;
- в процессе анализа безопасности системы в качестве эталонной модели;
- при подтверждении свойств разрабатываемой системы путем формального доказательства соблюдения политики безопасности.

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

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

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

1. Система является совокупностью взаимодействующих сущностей – *субъектов* и *объектов*. *Объекты* можно интуитивно представлять в виде

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

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

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

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

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

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

3.2. МОДЕЛЬ МАТРИЦЫ ДОСТУПОВ ХАРРИСОНА – РУЗЗО – УЛЬМАНА

Модель Харрисона – Руззо – Ульмана (ХРУ) используется для анализа систем защиты, реализующих дискреционную политику управления доступом.

В модели ХРУ используются следующие обозначения: O – множество объектов системы (сущности-контейнеры в модели ХРУ не рассматриваются); S – множество субъектов системы ($S \subseteq O$); R – множество видов прав доступа субъектов к объектам, например, права на чтение (*read*), на запись (*write*), владения (*own*); M – матрица доступов, строки которой соответствуют субъектам, а столбцы – объектам, $M[s, o] \subseteq R$ – права доступа субъекта s к объекту o.

Определение 3.1. Автомат, построенный согласно описанию модели ХРУ, назовем системой ХРУ.

Функционирование системы рассматривается только с точки зрения изменений в матрице доступа. Возможные изменения определяются шестью видами примитивных операторов, представленных в табл. 3.1. В результате выполнения примитивного оператора α осуществляется переход из состояния $q = (S, O, M)$ в результирующее состояние $q' = (S', O', M')$. Данный переход обозначим через $q \vdash_\alpha q'$.

Таблица 3.1

Примитивные операторы модели ХРУ

Примитивный оператор	Исходное состояние $q = (S,O,M)$	Результирующее состояние $q' = (S', O', M')$
«Внести» право r в $M[s, o]$	$s \in S, o \in O, r \in R$	$S' = S, O' = O, M'[s, o] = M[s, o] \cup \{r\}$, для $(s', o') \neq (s, o)$ выполняется равенство $M'[s', o'] = M[s', o']$
«Удалить» право r из $M[s, o]$	$s \in S, o \in O, r \in R$	$S' = S, O' = O, M'[s, o] = M[s, o] \setminus \{r\}$, для $(s', o') \neq (s, o)$ выполняется равенство $M'[s', o'] = M[s', o']$
«Создать» субъект s'	$s' \notin O$	$S' = S \cup \{s'\}, O' = O \cup \{s'\}$, для $(s, o) \in S \times O$ выполняется равенство $M'[s, o] = M[s, o]$, для $o \in O'$ выполняется равенство $M'[s', o] = \varnothing$, для $s \in S'$ выполняется равенство $M'[s, s'] = \varnothing$
«Создать» объект o'	$o' \notin O$	$S' = S, O' = O \cup \{o'\}$, для $(s, o) \in S \times O$ выполняется равенство $M'[s, o] = M[s, o]$, для $s \in S'$ выполняется равенство $M'[s, o'] = \varnothing$
«Уничтожить» субъект s'	$s' \in S$	$S' = S \setminus \{s'\}, O' = O \setminus \{s'\}$, для $(s, o) \in S' \times O'$ выполняется равенство $M'[s, o] = M[s, o]$
«Уничтожить» объект o'	$o' \in O, o' \notin S$	$S' = S, O' = O \setminus \{o'\}$, для $(s, o) \in S' \times O'$ выполняется равенство $M'[s, o] = M[s, o]$

Из примитивных операторов составляется конечное число команд системы ХРУ. Каждая команда включает две части: условия, при котором выполняется команда; последовательность примитивных операторов.

Таким образом, запись команды имеет следующий вид:

command $c(x_1, \ldots, x_k)$
    if $(r_1 \in M[x_{s1}, x_{o1}])$ *and* … *and* $(r_m \in M[x_{sm}, x_{om}])$ *then*
        α_1;
        …
    α_n;
    endif; end,

где $r_1, \ldots, r_m \in R$ – права доступа, $\alpha_1, \ldots, \alpha_n$ – последовательность примитивных операторов, параметрами которых являются параметры команды $x_1, \ldots, x_k$. Следует отметить, что наличие условия в теле команды не является обязательным.

При выполнении команды $c(x_1, \ldots, x_k)$ система осуществляет переход из состояния q в новое состояние q'. Данный переход обозначим

$$q \vdash_{c(x1, \ldots, xk)} q',$$

при этом $q' = q$, если одно из условий команды $c(x_1, \ldots, x_k)$ не выполнено, $q' = q_n$, если условие команды $c(x_1, \ldots, x_k)$ выполнено, и существуют состояния $q_1, \ldots, q_n$ такие, что $q = q_0 \vdash_{\alpha 1} q_1 \vdash_{\alpha 2} \ldots \vdash_{\alpha n} q_n$.

Пример 3.1. Команда создания субъектом *s* личного файла *f*:

command CreateFile(*s*, *f*)
«создать» объект *f*;
«внести» право владения *own* в *M*[*s*, *f*];
«внести» право на чтение *read* в *M*[*s*, *f*];
«внести» право на запись *write* в *M*[*s*, *f*];
end.

Пример 3.2. Команда передачи субъекту s' права *read* к файлу *f* его владельцем субъектом *s*:

command GrantRead(*s*, s', *f*)
if (*own* ∈*M*[*s*, *f*]) *then*
    «внести» право *read* в *M*[s', *f*];
endif
end.

Проведем анализ безопасности систем ХРУ. Согласно требованиям основных критериев оценки безопасности КС их системы защиты должны строиться на основе формальных моделей. С использованием формальных моделей должно быть теоретически обосновано соответствие системы защиты требованиям заданной политики безопасности. Для решения этой задачи необходим алгоритм, позволяющий осуществлять данную проверку. Однако, как показывают результаты анализа модели ХРУ, задача построения алгоритма проверки безопасности КС, реализующих дискреционную политику управления доступом, не может быть решена в общем случае. Дадим определения.

Определение 3.2. Будем считать, что в состоянии q системы ХРУ возможна утечка права доступа $r \in R$ в результате выполнения команды $c(x_1, \ldots, x_k)$, если при переходе системы $q \vdash_{c(x1, \ldots, xk)} q'$ выполняется примитивный оператор, вносящий право доступа r в ячейке матрицы доступов M, до этого r не содержавшей.

Определение 3.3. Начальное состояние q_0 системы ХРУ называется безопасным относительно некоторого права доступа $r \in R$, если невозможен переход системы в такое состояние q, в котором возможна утечка права r. Иными словами, начальное состояние q_0 системы ХРУ называется безопасным относительно некоторого права доступа r, если невозможен переход системы в состояние, в котором право доступа r появилось в ячейке матрицы доступов M, до этого r не содержавшей.

Рассмотрим класс систем ХРУ, для которых существует алгоритм проверки безопасности.

Определение 3.4. Система ХРУ называется монооперационной, если каждая команда системы содержит один примитивный оператор.

Теорема 3.1. Существует алгоритм, проверяющий является ли начальное состояние произвольной монооперационной системы ХРУ безопасным относительно некоторого права доступа $r \in R$.

Доказательство. Для доказательства достаточно показать, что является конечной максимальная длина последовательности команд монооперационной системы, каждая из которых вносит новые права доступа в матрицу доступов. В этом случае алгоритм проверки безопасности будет заключаться в применении к начальному состоянию данной последовательности команд и в проверке конечного состояния на отсутствие утечки права доступа r.

Заметим, что нет необходимости строить последовательность с командами, содержащими примитивные операторы вида «удалить»… и «уничтожить»…, так как в условиях команд и при утечке проверяется наличие прав доступа, а не их отсутствие.

Возможны три случая.

Первый случай: в начальном состоянии системы $q_0 = (S_0, O_0, M_0)$ выполняются следующие условия:

- множество субъектов $S_0 \neq \varnothing$;
- существует $(s, o) \in S_0 \times O_0$ такой, что $r \notin M_0[s, o]$.

Тогда нет необходимости включать в последовательность команды, содержащие примитивный оператор вида «создать»…. Это обусловлено тем, что все последовательности команд, которые проверяют или вносят права доступа в новые ячейки матрицы доступов, могут быть заменены последовательностями команд над существующими субъектами и объектами.

Число различных примитивных операторов вида «внести»… равно

$$n = |R| \cdot |S_0| \cdot |O_0|.$$

В каждой команде возможна проверка не более n различных условий на наличие в некоторой ячейке матрицы доступов некоторого права доступа. Следовательно, максимальное число различных команд, содержащих примитивный оператор вида «внести»…, с различными наборами параметров не превосходит $n \cdot 2^n$.

Алгоритм построения последовательности команд, позволяющей для первого случая проверить возможность утечки права доступа r, состоит из выполнения следующих шагов:

Шаг 1. Проверить выполнение условий первого случая.

Шаг 2. Построить список L всех команд, содержащих примитивный оператор вида «внести»… с различными наборами параметров ($|L| \leq n \cdot 2^n$).

Шаг 3. Перейти к началу списка L.

Шаг 4. Если список L пройден или пуст, то утечка права доступа r невозможна. В этом случае надо закончить выполнение алгоритма. Иначе – выбрать из списка L очередную команду. Если условие команды выполнено, то следует перейти на шаг 5, иначе – перейти на шаг 4.

Шаг 5. Удалить выбранную на шаге 4 команду из списка L и применить ее. Если в результате этого произошла утечка права доступа r, то надо закончить выполнение алгоритма. Иначе – перейти на шаг 3.

Данный алгоритм выполняется за конечное число шагов, в результате чего начиная с начального состояния системы реализуется конечная последовательность команд, позволяющая в первом случае проверить возможность утечки права доступа r.

Второй случай: в начальном состоянии системы $q_0 = (S_0, O_0, M_0)$ выполняется условие $S_0 = \varnothing$. Тогда необходимо применить одну команду, содержащую примитивный оператор вида «создать» субъекта, и перейти к первому случаю. Алгоритм построения последовательности команд, поз-

воляющей для второго случая проверить возможность утечки права доступа r, состоит из выполнения следующих шагов:

Шаг 1. Проверить выполнение условий второго случая.

Шаг 2. Рассмотреть все команды системы (их конечное число), содержащие примитивный оператор вида «создать» субъекта. Если все такие команды содержат проверку условий, то их применение невозможно, следовательно, невозможна утечка права доступа r. В этом случае надо закончить выполнение алгоритма. Если существует команда, содержащая примитивный оператор вида «создать» субъекта, то следует применить ее.

Шаг 3. Выполнить алгоритм, построенный для первого случая. При этом выполняется равенство $n = |R| \cdot (|S_0| + 1) \cdot (|O_0| + 1)$.

Данный алгоритм выполняется за конечное число шагов, в результате чего, начиная с начального состояния системы, реализуется конечная последовательность команд, позволяющая во втором случае проверить возможность утечки права доступа r.

Третий случай: в начальном состоянии системы $q_0 = (S_0, O_0, M_0)$ выполняются следующие условия:

- множество субъектов $S_0 \neq \varnothing$;
- для всех $(s, o) \in S_0 \times O_0$ выполняется условие $r \in M_0[s, o]$.

Так как нет необходимости включать в последовательность более одной команды, содержащей примитивный оператор вида «создать»…, следует на основе алгоритма для первого случая с использованием команд, содержащих примитивный оператор вида «внести»…, обеспечить (если это возможно) применение одной команды, содержащей примитивный оператор вида «создать»…, после чего перейти к первому случаю. Заметим, что максимальное число различных команд, содержащих примитивный оператор вида «создать»… («создать» субъекта и «создать» объект), с различными наборами параметров не превосходит $2 \cdot 2^m$, где $m = = |R| \cdot |S_0| \cdot |O_0|$.

Алгоритм построения последовательности команд, позволяющей для третьего случая проверить возможность утечки права доступа r, состоит из выполнения следующих шагов:

Шаг 1. Проверить выполнение условий третьего случая.

Шаг 2. Построить список L_1 всех команд системы, содержащих примитивный оператор вида «внести»… с различными наборами параметров ($|L_1| \leq k \cdot 2^k$, где $m = |R| \cdot |S_0| \cdot |O_0|$). Построить список L_2 всех команд системы, содержащих примитивный оператор вида «создать»…, с различными наборами параметров ($|L_2| \leq 2^{m+1}$, где $m = |R| \cdot |S_0| \cdot |O_0|$).

Шаг 3. Перейти к началу списка L_1.

Шаг 4. Если список L_1 пройден или пуст, то утечка права доступа r невозможна. В этом случае следует закончить выполнение алгоритма.

Иначе выбрать из списка L_1 очередную команду. Если условие команды выполнено, то перейти на шаг 5, иначе перейти на шаг 4.

Шаг 5. Удалить выбранную на шаге 4 команду из списка L_1 и применить ее. Последовательно по списку L_2 проверить возможность применения хотя бы одной команды, содержащей примитивный оператор вида «создать»…. Если такая команда найдена, то следует применить ее и перейти на шаг 6. Иначе – перейти на шаг 3.

Шаг 6. Выполнить алгоритм, построенный для первого случая. При этом выполняются условия:

- если на шаге 5 применена команда, содержащая примитивный оператор вида «создать» субъекта, то выполняется равенство

$$n = |R \cdot (|S_0| + 1) \cdot (|O_0| + 1);$$

- если на шаге 5 применена команда, содержащая примитивный оператор вида «создать» объект, то выполняется равенство

$$n = |R| \cdot |S_0| \cdot (|O_0| + 1).$$

Таким образом, для каждого из трех случаев описаны алгоритмы, позволяющие за конечное число шагов построить последовательность команд конечной длины. В результате применения последовательности, начиная с начального состояния, система переходит в состояние, которое назовем конечным. Если в конечном состоянии произошла утечка права доступа r, то начальное состояние системы не является безопасным относительно права доступа r, в противном случае начальное состояние системы является безопасным относительно права доступа r.

Теорема доказана.

Следствие 3.1. Алгоритм проверки безопасности монооперационных систем имеет экспоненциальную сложность.

Теорема 3.2. Задача проверки безопасности произвольных систем ХРУ алгоритмически неразрешима.

Доказательство. Воспользуемся фактом, обоснованным в теории машин Тьюринга: не существует алгоритма проверки для произвольной машины Тьюринга и произвольного начального слова для определения того, остановится ли машина Тьюринга в конечном состоянии или нет.

Под машиной Тьюринга понимается способ переработки слов в конечных алфавитах. Слова записываются на бесконечную в обе стороны ленту, разбитую на ячейки.

Для элементов и команд машины Тьюринга используем обозначения:

$A = \{a_0, a_1, \ldots, a_m\}$ – внешний алфавит, где $a_0 = \wedge$ – пустой символ;

$Q = \{q_0, q_1, \ldots, q_k\}$ – внутренний алфавит, где q_1 – начальное состояние, q_0 – конечное состояние;

$D = \{r, l, e\}$ – множество действий, где r – шаг вправо управляющей головки, l – шаг влево, e – управляющая головка не перемещается;

$C: Q \times A \to Q \times A \times D$ – функция, задающая команды машины Тьюринга. Если $C(q_{i0}, a_{i0}) = (q_{i1}, a_{i1}, d)$, то это означает, что когда машина находится в состоянии q_{i0} и управляющая головка указывает на ячейку ленты, содержащую символ a_{i0}, тогда выполняется шаг машины, в результате которого в эту ячейку записывается символ a_{i1}, машина переходит в состояние q_{i1}, а управляющая головка смещается по ленте согласно действию d.

Для того чтобы воспользоваться указанным выше фактом, представим все элементы и команды машины Тьюринга в виде элементов и команд системы модели ХРУ.

Пусть машина Тьюринга выполнила некоторое количество шагов. Занумеруем все ячейки, пройденные считывающей головкой (включая те, в которые была изначально занесена информация) числами от 1 до n. Тогда $(a_{s1}, \ldots, a_{sn})$ – заполнение ленты, где $a_{si} \in A$, $s_i \in \{0, 1, \ldots, m\}$ для $i = 1, \ldots, n$. Пусть считывающая головка указывает на ячейку с номером $i \in \{1, \ldots, n\}$, содержащую символ $a_{si} \in A$, где $s_i \in \{0, 1, \ldots, m\}$, состояние машины $q_{ij} \in Q$, где $i_j \in \{1, \ldots, k\}$, и должна быть выполнена команда $C(q_{ij}, a_{si}) = (q_{ij,} a_{si'}, d)$, где $q_{ij'} \in Q, a_{si'} \in A, s_i' \in \{0, 1, \ldots, m\}, ij' \in \{1, \ldots, k\}, d \in D$.

Каждой ячейке ленты поставим в соответствие субъекта системы ХРУ, при этом будем считать, что $O = S = \{s_1, \ldots, s_n\}$. Зададим матрицу доступов M для текущего состояния. Пусть множество прав доступа $R = Q \cup A \cup \{own, left, right\}$ и в матрицу доступов внесены права:

- $a_{sj} \in M[s_j, s_j]$, для $j = 1, \ldots, n$, – заполнение ленты;
- $own \in M[s_j, s_{j+1}]$, для $j = 1, \ldots, n - 1$, – упорядочивание субъектов, соответствующих ячейкам ленты;
- $q_{ij} \in M[s_i, s_i]$ – управляющая головка указывает на ячейку с номером i;
- $left \in M[s_1, s_1]$ – признак крайней левой из пройденных ячеек на ленте;
- $right \in M[s_n, s_n]$ – признак крайней правой из пройденных ячеек на ленте.

Таким образом, текущему состоянию машины Тьюринга соответствует матрица доступов M системы ХРУ (рис. 3.1).

	s_1	s_2	s_3	…	s_i	…	s_{n-1}	s_n
s_1	a_{s1}, *left*	*own*						
s_2		a_{s2}	*own*					
s_3			a_{s3}					
…								
s_i					a_{si}, q_{ij}			
…								
s_{n-1}							$a_{s\,n-1}$	*own*
s_n								a_{sn}, *right*

Рис. 3.1. Заполнение матрицы доступов системы ХРУ

Построим гомоморфизм машины Тьюринга в систему ХРУ. Для каждой команды машины Тьюринга $C(q_{ij}, a_{it}) = (q_{ij'}, a_{it'}, d)$ зададим соответствующую ей команду модели ХРУ:

- если $d = e$, то

command $Eq_{ij}a_{it}q_{ij'}a_{it'}$ (s)
if (q_{ij}∈$M[s, s]$) *and* (a_{it}∈$M[s, s]$) *then*
«удалить» право q_{ij} из $M[s, s]$;
«удалить» право a_{it} из $M[s, s]$;
«внести» право $q_{ij'}$ в $M[s, s]$;
«внести» право $a_{it'}$ в $M[s, s]$;
endif
end;

- если $d = r$, то необходимо задать две команды для случаев, когда считывающая головка указывает или не указывает на самую правую ячейку ленты:

command $R1q_{ij}a_{it}q_{ij'}a_{it'}$ (s, s')
if (q_{ij}∈$M[s, s]$) *and* (a_{it}∈$M[s, s]$) *and* (*own* ∈$M[s, s']$) *then*
«удалить» право q_{ij} из $M[s, s]$;
«удалить» право a_{it} из $M[s, s]$;
«внести» право $a_{it'}$ в $M[s, s]$;
«внести» право $q_{ij'}$ в $M[s', s']$;
endif
end;

command $R2q_{ij}a_{it}q_{ij'}a_{it'}(s, s')$
if ($q_{ij} \in M[s, s]$) *and* ($a_{it} \in M[s, s]$) and (*right* $\in M[s, s]$) *then*
«удалить» право q_{ij} из $M[s, s]$;
«удалить» право a_{it} из $M[s, s]$;
«удалить» право *right* из $M[s, s]$;
«внести» право $a_{it'}$ в $M[s, s]$;
«создать» субъект s';
«внести» право *own* в $M[s, s']$;
«внести» право a_0 в $M[s', s']$;
«внести» право $q_{ij'}$ в $M[s', s']$;
«внести» право *right* в $M[s', s']$;
endif
end;

- если $d = l$, то две команды для этого случая задаются аналогично командам для случая $d = r$.

Если машина Тьюринга останавливается в своем конечном состоянии q_0, то в соответствующей системе ХРУ происходит утечка права доступа q_0. Из алгоритмической неразрешимости задачи проверки остановки машины Тьюринга в конечном состоянии следует аналогичный вывод для задачи проверки безопасности соответствующей ей системы ХРУ. Таким образом, в общем случае для систем дискреционного управления доступом, построенных на основе модели ХРУ, задача проверки безопасности алгоритмически неразрешима.

Теорема доказана.

Приведенные теорема 3.1 и 3.2 указывают на имеющуюся проблему выбора у разработчиков систем защиты. С одной стороны, общая модель ХРУ может выражать большое разнообразие политик дискреционного управления доступом, но при этом не предоставляет алгоритма проверки их безопасности. С другой стороны, можно использовать монооперационные системы, для которых алгоритм проверки безопасности существует, но данный класс систем является слишком узким. Например, монооперационные системы не могут выразить политику, дающую субъектам право владения на созданные ими объекты, так как не существует одного примитивного оператора, который одновременно и создает объект, и помечает его как принадлежащий создающему субъекту.

Дальнейшие исследования модели ХРУ велись в основном в направлении определения условий, которым должна удовлетворять система, чтобы для нее задача проверки безопасности была алгоритмически разрешимой. Так, в 1976 г. авторами модели было доказано, что задача проверки безопасности алгоритмически разрешима для систем, в которых нет при-

митивных операторов вида «создать»…. В 1978 г. они показали, что таковыми могут быть системы монотонные и моноусловные, т. е. системы, команды которых не содержат операторов вида «уничтожить»…или «удалить»…, и содержат условия, состоящие из не более одной проверки вида $r \in M[s, o]$. В том же году было показано, что задача проверки безопасности для систем с конечным множеством субъектов разрешима, но вычислительно сложна.

3.3. МОДЕЛЬ ТИПИЗИРОВАННОЙ МАТРИЦЫ ДОСТУПОВ

Другая дискреционная модель, получившая название «Типизированная матрица доступов» (ТМД), представляет собой развитие модели ХРУ, дополненной концепцией типов, что позволяет несколько смягчить те условия, для которых возможно доказательство безопасности системы.

Формальное описание модели ТМД включает в себя следующие элементы:

O – множество объектов системы;

S – множество субъектов системы ($S \subseteq O$);

R – множество прав доступа субъектов к объектам;

M – матрица доступов;

C – множество команд;

T – множество типов объектов;

t: $O \rightarrow T$ – функция, ставящая в соответствие каждому объекту его тип;

$q = (S, O, t, M)$ – состояние системы;

Q – множество состояний системы.

Состояния системы изменяются в результате применения к ним команд из множества C. Команды в модели ТМД имеют тот же формат, что и в модели ХРУ, при этом для всех параметров команд указывается их тип:

command $c(x_1: t_1, \ldots, x_k: t_k)$
    if $(r_1 \in M[x_{s1}, x_{o1}])$ *and* … *and* $(r_m \in M[x_{sm}, x_{om}])$ *then*
        α_1;
        …
        α_n;
    endif
end

Перед выполнением команды происходит проверка типов фактических параметров, и, если они не совпадают с указанными в определении команды, то команда не выполняется. В модели ТМД используются шесть видов примитивных операторов, отличающихся от аналогичных операторов модели ХРУ только использованием типизированных параметров (табл. 3.2).

Таблица 3.2

Примитивные операторы модели ТМД

Примитивный оператор	Исходное состояние $q = (S, O, t, M)$	Результирующее состояние $q' = (S', O', t', M')$
«Внести» право r в $M[s, o]$	$s \in S$, $o \in O$, $r \in R$	$S' = S$, $O' = O$, $t'(o) = t(o)$ для $o \in O$, $M'[s, o] = M[s, o] \cup \{r\}$, для $(s', o') \neq (s, o)$ выполняется равенство $[s', o'] = M[s', o']$
«Удалить» право r из $M[s, o]$	$s \in S$, $o \in O$, $r \in R$	$S' = S$, $O' = O$, $t'(o) = t(o)$ для $o \in O$, $M'[s, o] = M[s, o] \setminus \{r\}$, для $(s', o') \neq (s, o)$ выполняется равенство $M'[s', o'] = M[s', o']$
«Создать» субъекта s' с типом t_s	$s' \notin S$	$S' = S \cup \{s'\}$, $O' = O \cup \{s'\}$, для $o \in O$ выполняется равенство $t'(o) = t(o)$, $t'(s') = t_s$, для $(s, o) \in S \times O$ выполняется равенство $M'[s, o] = M[s, o]$, для $o \in O'$ выполняется равенство $M'[s', o] = \varnothing$, для $s \in S'$ выполняется равенство $M'[s, s'] = \varnothing$
«Создать» объект s' с типом t_o	$s' \notin O$	$S' = S$, $O' = O \cup \{o'\}$, $t'(o') = t_o$, для $(s, o) \in S \times O$ выполняется равенство $M'[s, o] = M[s, o]$, для $s \in S'$ выполняется равенство $M'[s, o'] = \varnothing$
«Уничтожить» субъекта s'	$s' \in S$	$S' = S \setminus \{s'\}$, $O' = O \setminus \{s'\}$, для $o \in O'$ выполняется равенство $t'(o) = t(o)$, для $(s, o) \in S' \times O'$ выполняется равенство $M'[s, o] = M[s, o]$
«Уничтожить» объект o'	$o' \in O$, $o' \notin S$	$S' = S$, $O' = O \setminus \{o'\}$, для $o \in O'$ выполняется равенство $t'(o) = t(o)$, для $(s, o) \in S' \times O'$ выполняется равенство $M'[s, o] = M[s, o]$

Таким образом, с одной стороны, ТМД является обобщением модели ХРУ, которую можно рассматривать как частный случай ТМД с одним единственным типом для всех объектов и субъектов. С другой стороны, любую систему ТМД можно выразить через систему ХРУ, введя для обозначения типов специальные права доступа, а проверку типов в командах заменив проверкой наличия соответствующих прав доступа.

Определение 3.5. Пусть $c(x_1: t_1, \ldots, x_k: t_k)$ – некоторая команда ТМД. Будем говорить, что x_i является дочерним параметром, а t_i является дочер-

ним типом в $c(x_1: t_1, \ldots, x_k: t_k)$, если в ней имеется один из следующих примитивных операторов:

- «создать» субъекта x_i с типом t_i;
- «создать» объект x_i с типом t_i.

В противном случае будем говорить, что x_i является родительским параметром, а t_i является родительским типом в команде $c(x_1: t_1, \ldots, x_k: t_k)$.

Заметим, что в одной команде тип может быть одновременно и родительским, и дочерним. Например:

command foo(s_1: *u*, s_2: *u*, s_3: *v*, o_1: *w*, o_2: *b*)
«создать» субъекта s_2 с типом *u*;
«создать» субъекта s_3 с типом *v*;
end.

Здесь *u* является родительским типом относительно s_1 и дочерним типом относительно s_2. Кроме того, *w* и *b* являются родительскими типами, а *v* – дочерним типом. Появление в каждой команде дополнительных неявных условий, ограничивающих область применения команды только объектами соответствующих типов, позволяет несколько смягчить жесткие условия классической модели, при которых критерий безопасности является разрешимым.

Определение 3.6. Система монотонной ТМД (МТМД) – система ТМД, в командах которой отсутствуют немонотонные примитивные операторы вида «удалить»… и «уничтожить»….

Определение 3.7. Каноническая форма системы МТМД (КФМТМД) – система МТМД, в которой команды, содержащие примитивные операторы вида «создать»…, не содержат условий и примитивных операторов вида «внести»….

Терема 3.3. Для каждой системы МТМД существует эквивалентная ей система КФМТМД.

Доказательство. Пусть задана система МТМД, в которой определены множества *R, T, Q, C*. Построим эквивалентную ей систему КФМТМД, определив множества *R*, T*, Q*, C**.

Пусть

$R^* = R \cup \{active\}$;
$T^* = T \cup \{t_{active}\}$.

В каждом состоянии $q^* = (S^*, O^*, t^*, M^*)$, соответствующем состоянию $q = (S, O, t, M)$, справедливы равенства:

$S^* = S \cup \{s_{active}\}$;
$O^* = O \cup \{s_{active}\}$.

Пусть также для каждого $o \in O$ справедливо равенство $t^*(o) = t(o)$ и s_{active} – единственный субъект такой, что $t^*(s_{active}) = t_{active}$. Кроме того, для $s \in S$, $o \in O$ справедливо равенство $M^*[s, o] = M[s, o]$, и в начальном состоянии системы для $o \in O_0$ справедливо равенство $M_0^*[s_{active}, o] = \{active\}$.

Таким образом, право доступа *active* обозначает активизированные субъекты и объекты КФМТМД. Каждую команду $c(x_1: t_1, \ldots, x_k: t_k) \in C$ системы МТМД, не содержащую примитивные операторы «создать»…, представим командой $c(x_1: t_1, \ldots, x_k: t_k, s: t_{active})$ системы КФМТМД, полученной из исходной команды добавлением условий проверки $active \in M[s, x_i]$, для $i = 1, \ldots, k$.

Каждую команду $c(x_1: t_1, \ldots, x_k: t_k)$ системы МТМД, содержащую примитивные операторы «создать»…, представим монотонными командами КФМТМД:

- $c'_{xi}(x_{j1}: t_{j1}, \ldots, x_{jk}: t_{jk})$ – команды без проверки условий, каждая их которых соответствует одному дочернему параметру x_i команды $c(x_1: t_1, \ldots, x_k: t_k)$, содержит все ее родительские параметры и параметр x_i и содержит соответствующий примитивный оператор вида «создать»…;
- $c''(x_1: t_1, \ldots, x_k: t_k, s: t_{active})$ – команда, содержащая условия и примитивные операторы «внести»… команды $c(x_1: t_1, \ldots, x_k: t_k)$, условия проверки $active \in M[s, x_i]$, для всех родительских (не создаваемых в команде $c(x_1: t_1, \ldots, x_k: t_k)$) объектов x_i, примитивные операторы «внести» право *active*в $M[s, x_i]$ для всех объектов x_i, создаваемых в команде $c(x_1: t_1, \ldots, x_k: t_k)$.

Таким образом, только «активизированные» объекты (в том числе и субъекты) системы КФМТМД соответствуют объектам системы МТМД, а все преобразования над ними в системе КФМТМД соответствуют преобразованиям системы МТМД. Теорема доказана.

Для того чтобы сформулировать ограничения, достаточные для алгоритмической разрешимости задачи проверки безопасности в системах МТМД, опишем взаимосвязи между различными типами с помощью графа, определяющего отношение «наследственности» между типами, устанавливаемые через команды порождения объектов и субъектов.

Определение 3.8. Граф создания системы МТМД – ориентированный граф с множеством вершин T, в котором ребро от вершины u к вершине v существует тогда и только тогда, когда в системе имеется команда, в которой u является родительским типом, а v – дочерним типом.

Граф создания для каждого типа позволяет определить:

- объекты каких типов должны существовать в системе, чтобы в ней мог появиться объект или субъект заданного типа;
- объекты каких типов могут быть порождены при участии объектов заданного типа.

Определение 3.9. Система МТМД (КФМТМД) называется ациклической (АМТМД или, соответственно, АКФМТМД) тогда и только тогда, когда ее граф создания не содержит циклов; в противном случае говорят, что система является циклической.

Например, граф создания для приведенной выше команды *foo*, содержит следующие ребра: {(*u*, *u*), (*u*, *v*), (*w*, *u*), (*w*, *v*), (*b*, *u*), (*b*, *v*)}. Система МТМД, содержащая эту команду, будет циклической, поскольку тип *u* является для нее одновременно и родительским, и дочерним, что приводит к появлению в графе цикла (*u*, *u*). Алгоритм проверки безопасности систем АМТМД будет строиться для эквивалентных им систем АКФМТМД. Он состоит из трех шагов.

Алгоритм 3.1. Алгоритм проверки безопасности системы АМТМД.

Шаг 1. Для системы АМТМД построить эквивалентную ей систему АКФМТМД.

Шаг 2. Используя команды, содержащие только примитивные операторы «создать»… и не содержащие условий, перейти из начального состояния системы в некоторое «развернутое» состояние, обеспечивающее минимально необходимый и достаточный для распространения прав доступа состав объектов.

Шаг 3. Используя команды, не содержащие примитивные операторы «создать»…, перейти из развернутого состояния в «замкнутое» состояние, в котором дальнейшее применение таких команд не приводит к изменениям в матрице доступов.

Третий шаг алгоритма, очевидно, всегда будет иметь конечную сложность. Так как множество объектов не изменяется, то необходимо перебрать все последовательности различных команд (по аналогии с доказательством теоремы 3.1), а их конечное число.

Наибольшую трудность в общем случае представляет разработка алгоритма построения «развернутого» состояния. Однако для АМТМД или АКФМТМД такой алгоритм существует. Перед его рассмотрением дадим определение.

Определение 3.10. Пусть α и β – две различные команды системы МТМД, содержащие примитивные операторы «создать»…. Будем считать, что $\alpha < \beta$ тогда и только тогда, когда для некоторого дочернего типа команды α в графе создания найдется путь в некоторый родительский тип команды β.

Лемма 3.1. В системе АМТМД отношение «<» на множестве команд является отношением строгого порядка (обладает свойствами антирефлексивности, транзитивности и антисимметричности).

Доказательство. Так как граф создания систем АМТМД не содержит циклов, то для любой команды α не выполняется сравнение $\alpha < \alpha$.

Следовательно, отношение «<» обладает свойством антирефлексивности.

Пусть для команд α, β и γ выполняются сравнения $\alpha < \beta$ и $\beta < \gamma$. Тогда для некоторого дочернего типа команды α в графе создания существует путь в некоторый родительский тип команды β, и для некоторого дочернего типа команды β в графе создания существует путь в некоторый родительский тип команды γ. Кроме того, по определению графа создания, для каждого родительского типа команды β существует путь в каждый дочерней тип команды β. Следовательно, для некоторого дочернего типа команды α в графе создания существует путь в некоторый родительский тип команды γ. Таким образом, выполняется сравнение $\alpha < \gamma$ и отношение «<» обладает свойством транзитивности.

Пусть для команд α и β выполняются сравнения $\alpha < \beta$ и $\beta < \alpha$. Предположим противное, что выполняется неравенство $\alpha \neq \beta$. Тогда, повторяя рассуждения, выполненные при обосновании транзитивности отношения «<», получаем, что для некоторого дочернего типа команды α в графе создания существует путь в некоторый родительский тип команды α. По определению графа создания для каждого родительского типа команды α существует путь в каждый дочерней тип команды α. Следовательно, в графе создания существует цикл. Это противоречие. Таким образом, отношение «<» обладает свойством антирефлексивности.

Лемма доказана.

Алгоритм 3.2. Алгоритм построения «развернутого» состояния для системы АКФМТМД.

Шаг 1. Упорядочить линейно в списке все команды, содержащие примитивные операторы вида «создать»… (команда α следует в списке перед командой β тогда и только тогда, когда или $\alpha < \beta$, или α и β несравнимы).

Шаг 2. Начиная с начального состояния применять по созданному на шаге 1 списку все команды, при этом каждая команда применяется со всеми возможными для нее наборами родительских объектов.

Лемма 3.2. Алгоритм 3.2 заканчивает работу за конечное время на любом начальном состоянии произвольной системы АКФМТМД.

Доказательство. Число команд в списке конечно, для каждой команды из списка на каждом шаге работы алгоритма существует конечный набор объектов, которые могут являться ее параметрами. Следовательно, алгоритм 3.2 всегда заканчивает работу за конечное время.

Лемма доказана.

Определение 3.11. Пусть $(q_0, q_1, \ldots, q_i, \ldots)$ некоторая история системы МТМД. Для каждого состояния $q_i = (S_i, O_i, t_i, M_i)$, $i = 0, 1, \ldots$ на множестве O_i рекурсивно определим функцию порождения объектов π:

- для каждого $o \in O_0 \subset O_i$ положим $\pi(o) = o$,
- если объект $o \in O_k \subset O_i$ создан на шаге $0 < k \le I$ истории командой α_k, где $o_1, o_2, \ldots, o_m \in O_{k-1}$ – последовательность входящих в нее параметров объектов родительских типов, то положим $\pi(o) = \alpha_k(\pi(o_1), \pi(o_2), \ldots, \pi(o_m))$.

Пример 3.3. Пусть система МТМД с двумя командами:

command cv(*x*: *u*, *y*: *v*)
«создать» субъекта *y* с типом *v*;
end,
command cw(*x*: *u*, *y*: *v*, *z*: *w*)
«создать» объект *z* с типом *w*;
end,

и история имеет вид $q_0 \vdash_{cv(x, y)} q_1 \vdash_{cw(x, y, z)} q_2$, где $q_0 = (S_0, O_0, t_0, M_0) = (\{x\}, \{x\}, \{(x, u)\}, M_0)$. Тогда в состоянии $q_2 = (S_2, O_2, t_2, M_2) = (\{x, y\}, \{x, y, z\}, \{(x, u), (y, v), (z, w)\}, M_2)$ функция π будет иметь значения:

$\pi(x) = x$,
$\pi(y) = cv(x)$,
$\pi(z) = cw(x, cv(x))$.

Лемма 3.3. В «развернутом» состоянии системы АМТМД для каждого возможного значения функции порождения объектов π существует ровно один соответствующий объект.

Доказательство. Без ограничения общности по теореме 3.3 будем рассматривать системы АКФМТМД.

Пусть $(q_0, q_1, \ldots, q_l)$ – история системы, полученная в результате применения к начальному состоянию q_0 алгоритма 3.2, Π_j – множества значений функции π от всех объектов в состоянии q_j, где $0 \le j \le l$. При этом Π_l – множество значений функции π от всех объектов в «развернутом» состоянии q_l. Очевидно, что выполняются включения $\Pi_0 \subset \Pi_1 \subset \ldots \subset \Pi_l$.

Докажем от противного, что в «развернутом» состоянии q_l для каждого возможного значения функции π существует соответствующий объект. Пусть $(g_0, g_1, \ldots, g_i)$ некоторая история системы АКФМТМД, где $g_0 = q_0$, такая, что в состоянии $g_i = (S^g_i, O^g_i, t^g_i, M^g_i)$, где $i \ge 0$, существует объект $o^{(i)} \in O^g_i$ такой, что $\pi(o^{(i)}) \notin \Pi_l$. Если объект $o^{(i)} \in O_0$, то выполняется условие $\pi(o^{(i)}) = o^{(i)} \in \Pi_l$, это противоречие. Следовательно, существуют $0 \le k < I$ и команда α такие, что $g_k \vdash_\alpha g_{k+1}$ и объект $o^{(i)}$ создается командой α. При этом

объекты o_1, o_2, …, $o_m \in O^g_k$ – последовательность входящих в команду α параметров-объектов родительских типов, и по определению 3.11 выполняется равенство $\pi(o^{(i)}) = \alpha(\pi(o_1), \pi(o_2), \ldots, \pi(o_m))$.

Команда α содержится в списке, созданном при выполнении шага 1 алгоритма 3.2. Пусть q_n – состояние, к которому при выполнении алгоритма 3.2 впервые применена команда α, где $0 \le n \le l$. Так как $\pi(o^{(i)}) \notin \Pi_l$, то, как минимум, для одного из значений $\pi(o_1)$, $\pi(o_2)$, …, $\pi(o_m)$ в состоянии q_n должен отсутствовать соответствующий ему объект. Пусть объект $o^{(k)} \in \{o_1, o_2, \ldots, o_m\}$ и выполняется условие $\pi(o^{(k)}) \notin \Pi_n$. По определению алгоритма 3.2 выполняется условие $\pi(o^{(k)}) \notin \Pi_l$.

Многократно повторяя рассуждения, получим, что существует объект $o^{(0)} \in O_0$ такой, что выполняется условие $\pi(o^{(0)}) = o^{(0)} \notin \Pi_l$, это противоречие. Следовательно, в «развернутом» состоянии q_l для каждого возможного значения функции π существует соответствующий объект.

Так как на шаге 2 алгоритма 3.2 при применении каждой команды создается один объект с новым значением функции π, то в «развернутом» состоянии q_l для каждого возможного значения функции π существует ровно один соответствующий объект.

Лемма доказана.

Теорема 3.4. Существует алгоритм проверки безопасности систем АМТМД.

Доказательство. Без ограничения общности по теореме 3.3 будем рассматривать системы АКФМТМД. Пусть f – начальное состояние системы, q – «развернутое» состояние, полученное из f по алгоритму 3.1, m – «замкнутое» состояние, полученное из q с использованием всех команд, не содержащих примитивные операторы «создать»…, при этом дальнейшее применение таких команд в m не приводит к изменениям в матрице доступов. Так как система монотонная, то для $(s, o) \in S_q \times O_q = S_m \times O_m$ выполняется условие

$$M_q[s, o] \subseteq M_m[s, o]. \qquad (3.1)$$

Покажем, что для систем АКФМТМД для любой истории $(f, \ldots, h, \ldots)$ может быть найдена история $(q, \ldots, g, \ldots)$ без команд, содержащих примитивные операторы вида «создать»…, где для $(s, o) \in S_h \times O_h$ выполняется условие

$$M_h[s, o] \subseteq M_g[\pi(s), \pi(o)]. \qquad (3.2)$$

Заметим, что по лемме 3.3 в «развернутом» состоянии q каждому возможному значению функции π соответствует ровно один объект. Если (3.2) выполняется, то из (3.1) и (3.2) следует, что для $(s, o) \in S_f \times O_f$ выполняется условие

$$M_h[s, o] \subseteq M_g[\pi(s), \pi(o)] \subseteq M_m[\pi(s), \pi(o)],$$

что означает алгоритмическую разрешимость задачи проверки безопасности систем АКФМТМД и, следовательно, систем АМТМД. Например, для начального состояния f заменяем $\pi(s)$ на s, $\pi(o)$ на o и получаем, что для $(s, o) \in S_f \times O_f$ выполняется условие

$$M_h[s, o] \subseteq M_g[s, o] \subseteq M_m[s, o].$$

Обоснуем (3.2), для чего построим алгоритм преобразования последовательности команд истории $(f, \ldots, h, \ldots)$ в последовательность команд истории $(q, \ldots, g, \ldots)$, который состоит из двух шагов.

Шаг 1. В командах истории $(f, \ldots, h, \ldots)$ удалить примитивные операторы вида «создать»….

Шаг 2. Команды вида $c(x_1: t_1, \ldots, x_k: t_k)$ истории $(f, \ldots, h, \ldots)$ заменить на команды $c(\pi(x_1): t_1, \ldots, \pi(x_k): t_k)$ истории $(q, \ldots, g, \ldots)$.

По индукции по длине истории $(f, \ldots, h, \ldots)$ легко показать, что выполняются два условия.

Условие 1. Для каждой команды истории $(f, \ldots, h, \ldots)$, если истинно ее условие, то истинно условие соответствующей ей команды истории $(q, \ldots, g, \ldots)$.

Условие 2. Для состояния h истории $(f, \ldots, h, \ldots)$ и соответствующего ему состояния g истории $(q, \ldots, g, \ldots)$ верно, что для $(s, o) \in S_h \times O_h$ выполняется условие

$$M_h[s, o] \subseteq M_g[\pi(s), \pi(o)].$$

Таким образом, условие (3.2) обосновано. Теорема доказана.

Следствие 3.2. Алгоритм проверки безопасности систем АМТМД имеет экспоненциальную сложность.

Доказательство. При построении «замкнутого» состояния из «развернутого» состояния используется алгоритм, аналогичный алгоритму, использованному в теореме 3.1. По следствию 3.1, получаем, что алгоритм проверки безопасности систем АМТМД имеет экспоненциальную сложность.

Определим подмножество АМТМД, называемое тернарными АМТМД.

Определение 3.12. Тернарными называются АМТМД, в которых каждая команда имеет не более трех параметров.

Для тернарной АМТМД доказано, что алгоритм проверки ее безопасности имеет полиномиальную сложность.

Таким образом, введение строгого контроля типов в дискреционную модель ХРУ позволило доказать критерий безопасности систем для более приемлемых ограничений, что существенно расширило область ее применения.

Контрольные вопросы и задания

1. Как произвольную систему ТМД представить системой ХРУ?
2. Для команды $C(q_{i0}, a_{i0}) = (q_{i1}, a_{i1}, l)$ машины Тьюринга выпишите две представляющие ее команды модели ХРУ.
3. Что такое субъект системы? Что такое объект системы?
4. Каким может являться каждое состояние системы в соответствии с предложенным в модели критерием безопасности?

4. МОДЕЛЬ РАСПРОСТРАНЕНИЯ ПРАВ ДОСТУПА TAKE-GRANT

4.1. ОСНОВНЫЕ ПОЛОЖЕНИЯ КЛАССИЧЕСКОЙ МОДЕЛИ TAKE-GRANT

Классическая модель *Take-Grant* ориентирована на анализ путей распространения прав доступа в системах дискреционного управления доступом. Основными элементами модели *Take-Grant* являются:

O – множество объектов;

$S \subseteq O$ – множество субъектов;

$R = \{r_1, r_2, \ldots, r_m\} \cup \{t, g\}$ – множество видов прав доступа, где t(*take*) – право брать права доступа, g(*grant*) – право давать права доступа;

$G = (S, O, E)$ – конечный помеченный ориентированный без петель граф доступов, описывающий состояние системы. Элементы множеств S и O являются вершинами графа, которые будем обозначать, соответственно, $\otimes$ – объекты (элементы множества $O \setminus S$), $\bullet$ – субъекты (элементы множества S). Элементы множества $E \subseteq O \times O \times R$ являются ребрами графа. Каждое ребро помечено непустым подмножеством множества видов прав доступа R.

Состояние системы описывается соответствующим ему графом доступов. В отличие от модели ХРУ в модели *Take-Grant* возможно наличие прав доступа не только у субъектов к объектам, но и у объектов к объектам.

Основная цель классической модели *Take-Grant* – определение и обоснование алгоритмически проверяемых условий проверки возможности утечки права доступа по исходному графу доступов, соответствующему некоторому состоянию системы.

Порядок перехода системы модели *Take-Grant* из состояния в состояние определяется правилами преобразования графа доступов, которые в классической модели носят название де-юре правил. Преобразование графа G в граф G' в результате выполнения правила *op* обозначим

$$G \vdash_{op} G'.$$

В классической модели *Take-Grant* рассматриваются четыре де-юре правила преобразования графа, выполнение каждого из которых может

быть инициировано только субъектом, являющимся активной компонентой системы (рис. 4.1–4.4):

- *take* – брать права доступа;
- *grant* – давать права доступа;
- *create* – создавать новый объект или субъект, при этом субъект создатель может взять на созданный субъект любые права доступа (по-умолчанию предполагается, что при выполнение правила *create* создается объект, случаи, когда создается субъект, оговариваются особо);
- *remove* – удалять права доступа.

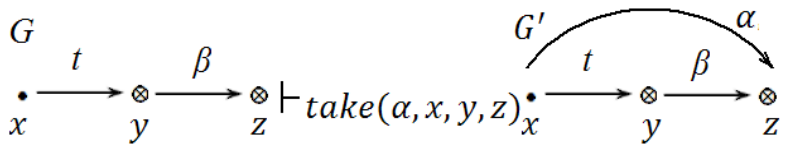

Рис. 4.1. Применение правила take(α, x, y, z)

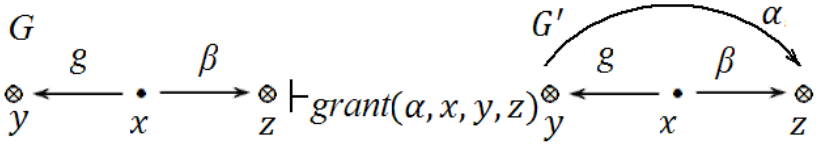

Рис. 4.2. Применение правила grant(α, x, y, z)

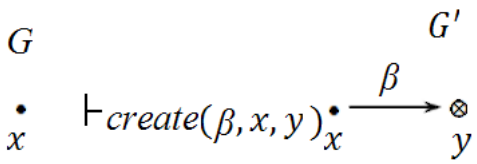

Рис. 4.3. Применение правила *create(β, x, y)*

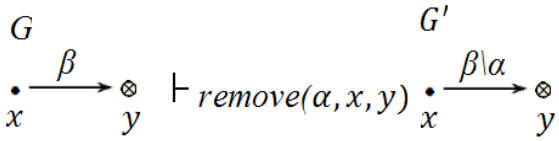

Рис. 4.4. Применение правила *remove*(α, x, y)

Рассмотрим условия применения де-юре правил в исходном состоянии $G = (S, O, E)$ и результаты их применения в результирующем состоянии $G' = (S', O', E')$ (табл. 4.1).

Таблица 4.1

Де-юре правила классической модели *Take-Grant*

Правила	Исходное состояние $G = (S, O, E)$	Результирующее состояние $G' = (S', O', E')$
Take(α, x, y, z)	$x \in S, y, z \in O,$ $(x, y, \{t\}) \subset E,$ $(y, z, \beta) \subset E, x \neq z, \alpha \subseteq \beta$	$S' = S, O' = O,$ $E' = E \cup \{(x, z, \alpha)\}$
Grant(α, x, y, z)	$x \in S, y, z \in O,$ $(x, y, \{g\}) \subset E,$ $(x, z, \beta) \subset E, y \neq z, \alpha \subseteq \beta$	$S' = S, O' = O,$ $E' = E \cup \{(y, z, \alpha)\}$
Create(β, x, y)	$x \in S, y \notin O, \beta \neq \varnothing$	$O' = O \cup \{y\}$, если y субъект, то $S' = S \cup \{y\}$, иначе $S' = S, E' = E \cup \{(x, y, \beta)\}$
Remove(α, x, y)	$x \in S, y \in O,$ $(x, y, \beta) \subset E, \alpha \subseteq \beta$	$S' = S, O' = O,$ $E' = E \setminus \{(x, y, \alpha)\}$

В модели *Take-Grant* основное внимание уделяется определению условий, при которых в системе возможно распространение прав доступа для случая, когда не рассматриваются ограничения на кооперацию субъектов при передаче прав доступа, и случая, когда на кооперацию субъектов наложены ограничения. Рассмотрим данные случаи подробнее.

1. Условия передачи прав доступа при отсутствии ограничений на кооперацию субъектов

Данный случай характеризуется тем, что при передаче прав доступа не накладываются ограничения на кооперацию субъектов системы, участвующих в этом процессе.

Определение 4.1. Пусть $x, y \in O_0$, $x \neq y$ – различные объекты графа доступов $G_0 = (S_0, O_0, E_0)$, $\alpha \subseteq R$. Определим предикат *can_share*(α, x, y, G_0), который будет истинным тогда и только тогда, когда существуют графы $G_1 = (S_1, O_1, E_1)$, …, $G_N = (S_N, O_N, E_N)$ и правила op_1, …, op_N такие, что $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x, y, \alpha) \subset E_N$, где $N \geq 0$.

Определение истинности предиката *can_share*(α, x, y, G_0) непосредственно по определению является в общем случае алгоритмически неразрешимой задачей, так как требует проверки всех траекторий функционирования системы. По этой причине для проверки истинности предиката *can_share*(α, x, y, G_0) следует определить необходимые и достаточные

условия, проверка которых возможна. Решение данной задачи будет выполнено в два этапа. На первом этапе будут определены и обоснованы условия истинности предиката *can_share*(α, *x*, *y*, G_0) для графов, все вершины которых являются субъектами, на втором этапе условия истинности предиката *can_share*(α, *x*, *y*, G_0) будут определены и обоснованы для произвольных графов.

Определение 4.2. Пусть $G = (S, S, E)$ – граф доступов, все вершины которого являются субъектами. Говорят, что вершины графа доступов являются *tg*-связными или они соединены *tg*-путем, если без учета направления ребер в графе между ними существует путь такой, что каждое ребро этого пути помечено *t* или *g*.

Теорема 4.1. Пусть $G_0 = (S_0, S_0, E_0)$ – граф доступов, содержащий только вершины субъекты, $x, y \in S_0$, $x \neq y$. Тогда предикат *can_share*(α, *x*, *y*, G_0) истинен тогда и только тогда, когда выполняются условия 1 и 2.

Условие 1. Существуют субъекты $s_1, \ldots, s_m \in S_0$:

$(s_i, y, \gamma_i) \subset E_0$, где $i = 1, \ldots, m$ и $\alpha = \gamma_1 \cup \ldots \cup \gamma_m$.

Условие 2. Субъекты *x* и s_i являются *tg*-связными в графе G_0, где $i = 1, \ldots, m$.

Доказательство. Проведем доказательство теоремы для $m = 1$, так как схему доказательства для этого случая легко продолжить для случая $m > 1$.

При $m = 1$ условия 1 и 2 формулируются следующим образом:

Условие 1. Существует субъект *s* такой, что выполняется включение $(s, y, \alpha) \subset E_0$.

Условие 2. Субъекты *x* и *s* соединены *tg*-путем в графе G_0.

Достаточность. Пусть выполнены условия 1 и 2. Доказательство проведем индукцией по длине *tg*-пути, соединяющего субъектов *x* и *s*.

Пусть $N = 0$. Тогда, $x = s$, $(x, y, \alpha) \subset E_0$ и предикат *can_share*(α, *x*, *y*, G_0) истинен. Пусть $N = 1$. Тогда существует $(s, y, \alpha) \subset E_0$, и *x* и *s* соединены ребром *t* или *g* в графе G_0. Возможны четыре случая такого соединения *x* и *s*, для каждого из которых указана последовательность преобразований графа, требуемая для передачи прав доступа (рис. 4.5–4.8).

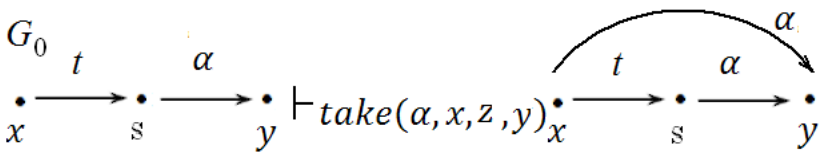

Рис. 4.5. Первый случай

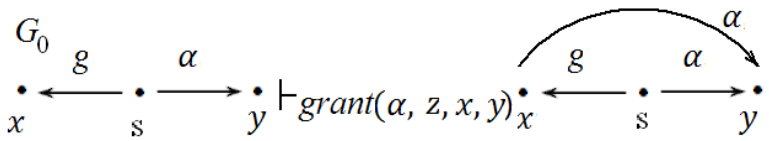

Рис. 4.6. Второй случай

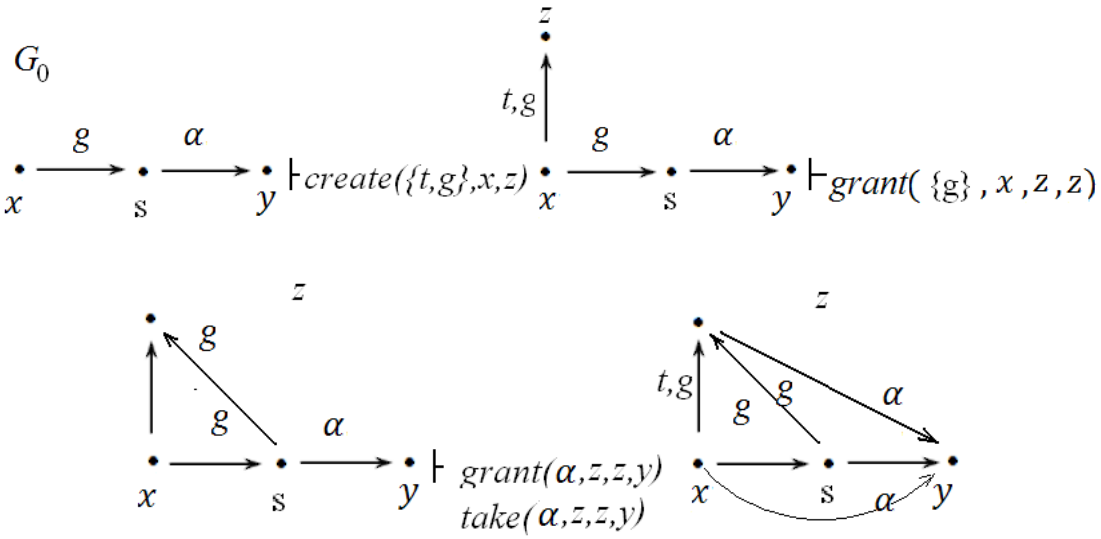

Рис.4.7. Третий случай

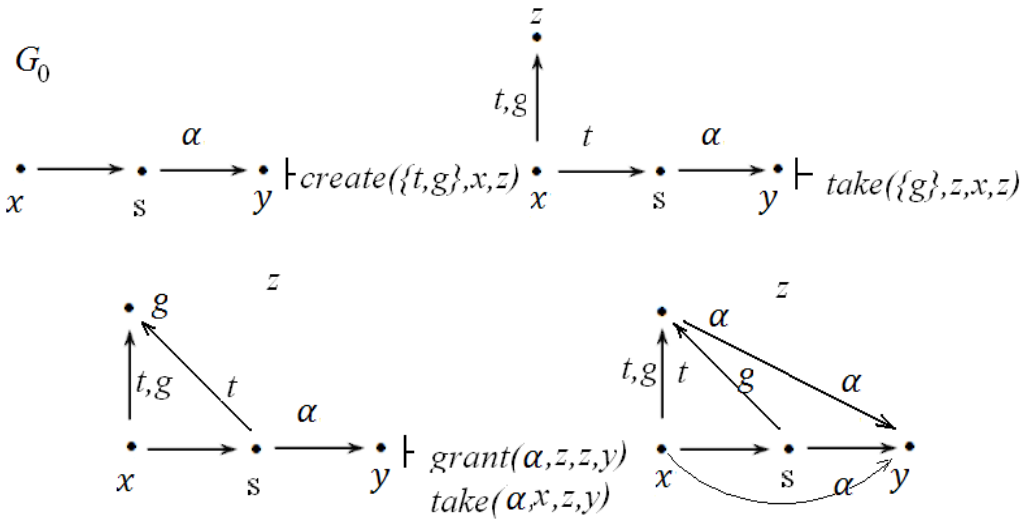

Рис.4.8. Четвертый случай

Пусть длина пути $N > 1$. Пусть вершина z находится на tg-пути между x и s и является смежной с s в графе G_0. Тогда, исходя из доказанного, для случая $N = 1$ существует последовательность преобразований графов доступов $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opK} G_K$ такая, что $(z, y, \alpha) \subset E_k$, и длина tg-пути

между z и x равна $N - 1$. Это позволяет применить предположение индукции. Достаточность условий теоремы доказана.

Необходимость. Пусть истинен предикат $can\_share(\alpha, x, y, G_0)$.

По определению истинности предиката существует последовательность графов доступов $G_1 = (S_1, O_1, E_1)$, …, $G_N = (S_N, O_N, E_N)$ такая, что $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x, y, \alpha) \subset E_N$. Среди всех таких последовательностей выберем ту, у которой длина N является минимальной.

Докажем необходимость *условий* 1 и 2 индукцией по N.

При $N = 0$ очевидно, что $(x, y, \alpha) \subset E_0$. Таким образом, $s = x$, и условия теоремы выполнены.

При $N = 1$ выполняется условие $(x, y, \alpha) \not\subset E_0$, и из определения де-юре правил модели следует, что существует субъект s такой, что справедливо $(s, y, \alpha) \subset E_0$ и ($op_1 = take(\alpha, x, s, y)$ или $op_1 = grant(\alpha, s, x, y)$). Таким образом, s и x являются tg-связными в графе G_0, и условия теоремы выполнены.

Пусть $N > 1$ и утверждение теоремы истинно для $k < N$. Тогда $(x, y, \alpha) \not\subset E_{N-1}$, и ребро (x, y, α) появляется в графе доступов G_N в результате применения к графу G_{N-1} некоторого правила op_N. Очевидно, это не правила *create* или *remove*. Следовательно, существует субъект $s' \in S_{N-1}$ такой, что $(s', y, \alpha) \subset E_{N-1}$ и $((x, s', t) \in E_{N-1}$ и $op_N = take(\alpha, x, s', y))$ или $((s', x, g) \in E_{N-1}$ и $op_N = grant(\alpha, s', x, y))$.

Возможны два случая: $s' \in S_0$ и $s' \notin S_0$.

Рассмотрим первый случай. Пусть $s' \in S_0$, тогда истинен предикат $can\_share(\alpha, s', y, G_0)$, при этом число преобразований графов меньше N. Следовательно, по предположению индукции существует $s \in S_0$: $(s, y, \alpha) \in E_0$, и s' соединен с stg-путем в графе G_0. Кроме того, если $op_N = take(\alpha, x, s', y)$, то истинен предикат $can\_share(\{t\}, x, s', G_0)$, а если $op_N = grant(\alpha, s', x, y)$, то истинен предикат $can\_share(\{g\}, s', x, G_0)$; при этом число преобразований графов меньше N. Следовательно, по предположению индукции существует $s'' \in S_0$, такой, что если $op_N = take(\alpha, x, s', y)$, то $(s'', s', t) \in E_0$ и s'' соединен с xtg-путем в графе G_0, а если $op_N = grant(\alpha, s', x, y)$, то $(s'', x, g) \in E_0$ и s'' соединен с $s'tg$-путем в графе G_0. Таким образом, существует $s \in S_0$: $(s, y, \alpha) \subset E_0$, и субъекты x и s соединены tg-путем в графе G_0. Для случая $s' \in S_0$ индуктивный шаг доказан (рис. 4.9).

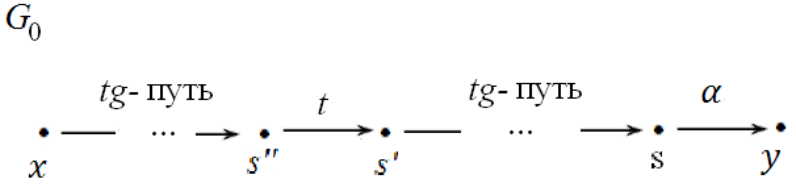

Рис. 4.9. Случай $s' \notin S_0$ и $op_N = take(\alpha, x, s', y)$

Рассмотрим второй случай. Пусть $s' \notin S_0$. Заметим, что число преобразований графов N минимально, поэтому преобразования графов удовлетворяют следующим требованиям:

- каждый субъект графа G_0 создает не более одного субъекта,
- субъект создатель берет на созданный субъект необходимый набор прав $\{t, g\}$;
- созданный субъект не создает новых субъектов.

Справедливость перечисленных требований следует из того, что если субъект графа G_0 создает более одного субъекта или созданный субъект создаст нового субъекта, то в преобразованиях графов доступов такие дополнительные субъекты могут быть заменены имеющимися. Следовательно, не будет являться необходимым применение правил создания дополнительных субъектов, что противоречит условию минимальности числа преобразований графов. Требование к субъекту-создателю брать на созданный субъект необходимый набор прав $\{t, g\}$ является, очевидно, необходимым.

Из перечисленных требований следует, что существуют $M < N - 1$, $s'' \in S_0$: $op_M = create(\{t, g\}, s'', s')$. Среди всех последовательностей преобразований длины N можно выбрать такую, что $M = 1$. Тогда рассмотрим граф G_1; при этом $S_1 = S_0 \cup \{s'\}$, $E_1 = E_0 \cup \{(s'', s', \{t, g\})\}$, истинен предикат $can\_share(\alpha, s', y, G_1)$ с числом преобразований меньше N и $s' \in S_1$. Следовательно, существует $s \in S_1$: $(s, y, \alpha) \subset E_1$, при этом s и s' соединены tg-путем в графе G_1. Очевидно, что $s \in S_0$, $(s, y, \alpha) \subset E_0$; при этом s и s' соединены tg-путем в графе G_0. Кроме того, если $op_N = take(\alpha, x, s', y)$, то истинен предикат $can\_share(\{t\}, x, s', G_1)$, а если $op_N = grant(\alpha, s', x, y)$, то истинен предикат $can\_share(\{g\}, s', x, G_1)$; при этом число преобразований меньше N. Так как s' – единственный субъект в графе G_1, обладающий правами доступа к субъекту s', то x и s'' соединены tg-путем в графе G_1 и, соответственно, в графе G_0.

Таким образом, существует $s \in S_0$: $(s, y, \alpha) \subset E_0$; при этом x и s соединены tg-путем в графе G_0. Для случая $s' \notin S_0$ индуктивный шаг доказан (рис. 4.10).

Теорема доказана.

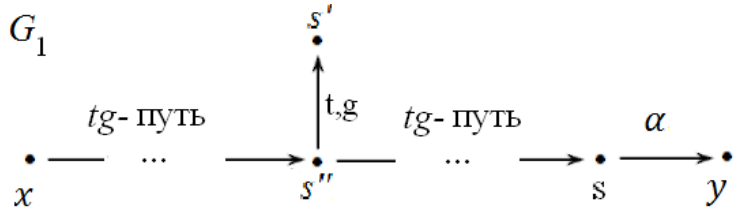

Рис. 4.10. Случай $s' \notin S_0$ и $opN=take(\alpha, x, s', y)$

Для определения истинности предиката *can_share*(α, x, y, G_0) в произвольном графе дадим определения.

Определение 4.3. Островом в произвольном графе доступов G_0 называется его максимальный *tg*-связный подграф, состоящий только из вершин субъектов.

Определение 4.4. Мостом в графе доступов G_0 называется *tg*-путь, концами которого являются вершины-субъекты и который проходит через вершины-объекты. Словарная запись его имеет вид

$$\vec{t}*, \overleftarrow{t}*, \vec{t}*\vec{g}\,\overleftarrow{t}*, \vec{t}*\overleftarrow{g}\overleftarrow{t}*,$$

где символ «*» означает многократное (в том числе нулевое) повторение.

Определение 4.5. Начальным пролетом моста в графе доступов G_0 называется *tg*-путь, началом которого является вершина-субъект, концом – объект, проходящий через вершины-объекты, словарная запись которого имеет вид $\vec{t}*\vec{g}$.

Определение 4.6. Конечным пролетом моста в графе доступов G_0 называется *tg*-путь, началом которого является вершина-субъект, концом – объект, проходящий через вершины объекты, словарная запись которого имеет вид: $\vec{t}*$.

Теорема 4.2. Пусть $G_0 = (S_0, O_0, E_0)$ – произвольный граф доступов, $(x,y)\in O_0$, $x\neq y$. Предикат *can_share*(α, x, y, G_0) истинен тогда и только тогда, когда или $(x, y, \alpha)\subset E_0$ или выполняются условия 1–3:

Условие 1. Существуют объекты $s_1, \ldots, s_m\in O_0$:

$(s_i, y, \gamma_i)\subset E_0$ для $i = 1, \ldots, m$ и $\alpha = \gamma_1\cup \ldots \cup\gamma_m$.

Условие 2. Существуют субъекты $x_1', \ldots, x_m', s_1', \ldots, s_m'\in S_0$:

1) $x = x_i'$ или x_i'соединен с x начальным пролетом моста в графе G_0, где $i = 1, \ldots, m$;

2) $s = s_i'$ или s_i'соединен с s конечным пролетом моста в графе G_0, где $i = 1, \ldots, m$.

3) *Условие* 3. В графе G_0 для каждой пары (x_i', s_i'), где $i = 1, \ldots, m$, существуют острова $I_{i,1}, \ldots, I_{i,ui}$, $u_i \geq 1$ такие, что $x_i'\in I_{i,1}$, $s_i'\in I_{i,ui}$, и существуют мосты между островами $I_{i,j}$ и $I_{i,j+1}$, где $j = 1, \ldots, u_i - 1$.

Доказательство. Проведем доказательство теоремы для $m = 1$, так как схему доказательства для этого случая легко продолжить на случай $m > 1$. При $m = 1$ условия 1–3 выполнены (рис. 4.11).

Условие 1. Существует объект $s\in O_0$: $(s, y, \alpha)\subset E_0$.

Условие 2. Существуют субъекты $x', s'\in S_0$:

а) $x = x'$ или x' соединен с x начальным пролетом моста в графе G_0;

б) $s = s'$ или s' соединен с s конечным пролетом моста в графе G_0.

Условие 3. В графе G_0 существуют острова I_1, …, I_u, $u \geq 1$, такие, что $x' \in I_1$, $s' \in I_u$, и существуют мосты между островами I_j и I_{j+1}, где $j = 1,\ldots, u-1$.

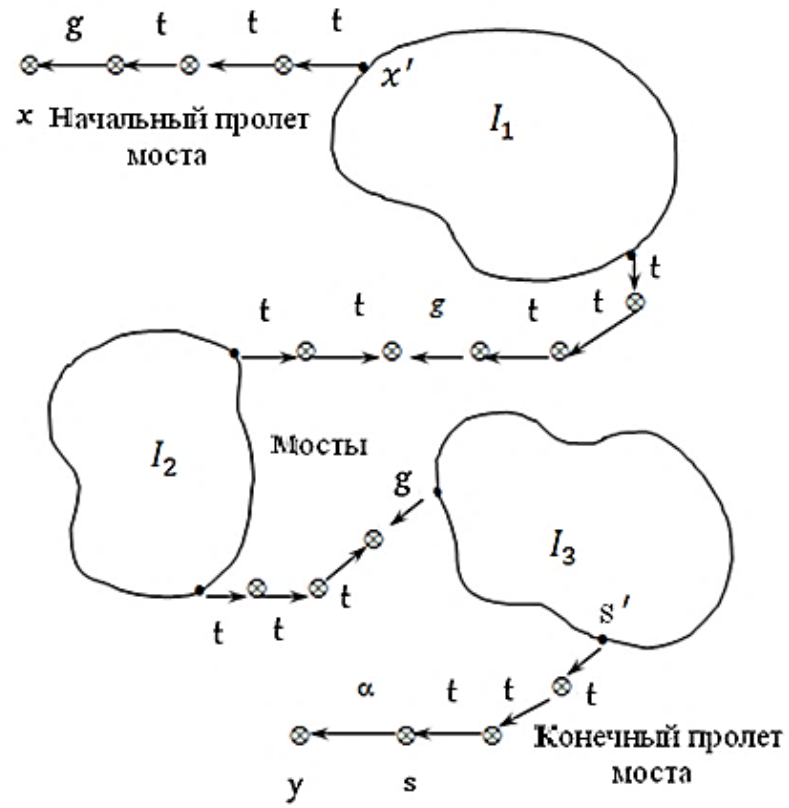

Рис. 4.11. Пример пути передачи объекту *x* прав доступа α к объекту *y*

Достаточность. Если $(x, y, \alpha) \subset E_0$, то предикат *can_share*(α, x, y, G_0) истинен.

Пусть выполнены условия 1–3 теоремы. Является очевидным тот факт, что по мостам возможна передача прав доступа между субъектами, являющимися его концами. В качестве примера разобрана последовательность преобразований графа доступов при передаче прав по мосту вида $\vec{t}\ \vec{g}\ \overleftarrow{t}$ (рис. 4.12).

Также очевидно, что по конечному пролету моста субъект может забрать права доступа у объекта, а по начальному пролету – передать их объекту.

По условию 1 существует объект *s*, который обладает правами α к объекту *y*. По условию 2б существует субъект *s′*, который либо совпадает с *s*, либо по конечному пролету моста может забрать у субъекта *s* права α к объекту *y*.

По теореме 4.1 права доступа, полученные одним субъектом, принадлежащим острову, могут быть переданы любому другому субъекту острова. По условию 3 между островами существуют мосты, по которым возможна передача прав доступа. По условию 2а существует субъект *x′*, который или совпадает с *x*, или, получив права доступа, может передать их *x* по начальному пролету моста.

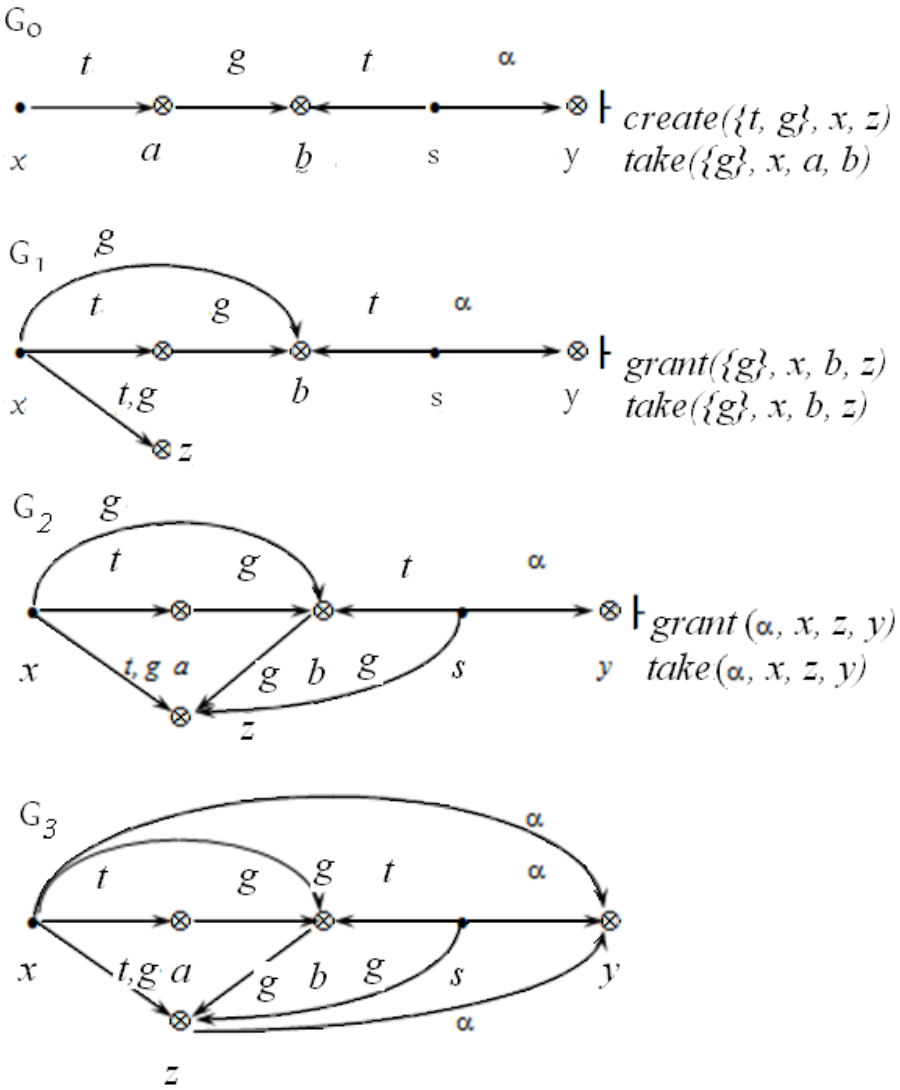

Рис. 4.12. Пример передачи прав доступа по мосту

Необходимость. Пусть истинен предикат *can_share*(α, *x*, *y*, G_0). По определению истинности предиката существует последовательность графов доступов $G_1 = (S_1, O_1, E_1)$, …, $G_N = (S_N, O_N, E_N)$ такая, что $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x, y, \alpha) \subset E_N$, при этом $N \geq 0$ выбирается минимальным. Докажем необходимость выполнения условий теоремы индукцией по *N*.

При $N = 0$ очевидно, что $(x, y, \alpha) \subset E_0$. Следовательно, условия теоремы выполнены.

При $N = 1$ из определения де-юре правил модели следует:

- либо существует субъект $s \in S_0$ такой, что справедливо условие $(s, y, \alpha) \in E_0$ и $op_1 = grant(\alpha, s, x, y)$;
- либо существует объект $s \in O_0$ такой, что справедливо условие $(s, y, \alpha) \in E_0$ и $op_1 = take(\alpha, x, s, y)$.

Таким образом, для $N = 1$ условия теоремы выполняются.

Пусть $N > 1$, и утверждение теоремы истинно для $k < N$. Тогда $(x, y, \alpha) \notin E_{N-1}$ и ребро (x, y, α) появляется в графе доступов G_N в результате применения к графу G_{N-1} некоторого правила op_N. Возможны два случая: $x \notin S_0$ и $x \in S_0$.

Если $x \notin S_0$, то существует $x_1 \in S_{N-1}$: $op_N = grant(\alpha, x_1, x, y)$. Также возможны два случая: $x_1 \in S_0$ или $x_1 \notin S_0$.

Если $x_1 \in S_0$, тогда истинен предикат $can\_share(\{g\}, x_1, x, G_0)$ с числом преобразований графов меньшим N. Следовательно, по предположению индукции существуют:

- $x_2 \in O_0$: $(x_2, x, g) \in E_0$;
- $x' \in S_0$, соединенный с x_2 конечным пролетом моста;
- острова $I_1, \ldots, I_t$, где $t \geq 1$, такие, что $x_1 \in I_t$, $x' \in I_1$, и существуют мосты между островами I_j и I_{j+1} для $j = 1, \ldots, t - 1$.

Кроме того, истинен предикат $can\_share(\alpha, x_1, y, G_0)$ с числом преобразований графов меньшим N. Тогда по предположению индукции существуют:

- $s \in O_0$: $(s, y, \alpha) \subset E_0$;
- $s' \in S_0$: $s = s'$ или s' соединен с s конечным пролетом моста;
- острова $I_t, \ldots, I_u$, где $t - u \geq 1$, такие, что $x_1 \in I_t$, $s' \in I_u$, и существуют мосты между островами I_j и I_{j+1} для $j = t, \ldots, u - 1$.

Заметим, что путь, соединяющий вершины x', x_2, x, есть начальный пролет моста. Если $x_1 \notin S_0$, то с учетом замечаний, сделанных при доказательстве теоремы 4.1, получаем, что существуют $M < N - 1$, $x_2 \in S_0$ такой, что $op_M = create(\{t, g\}, x_2, x_1)$. Среди всех последовательностей преобразований длины N можно выбрать такую, что $M = 1$. Далее используем технику доказательства теоремы 4.1 и доказательства индуктивного шага для случая $x_1 \in S_0$.Таким образом, для случая $x \notin S_0$ условия 1–3 выполняются, и индуктивный шаг доказан.

Если $x \in S_0$, то условие 2а очевидно выполняется. Многократно применяя технику доказательства, использованную выше и в теореме 4.1, доказываем индуктивный шаг и в данном случае.

Теорема доказана.

Замечание 4.1. При доказательстве теоремы 4.2 можно показать, что не существует путей, отличных от мостов между двумя субъектами, проходящих через вершины объекты, по которым возможна передача прав доступа. Для этого рассмотрим все пути (с учетом симметрии) длины 2 между двумя субъектами, проходящие через вершины объекты, по которым возможна передача прав доступа (рис. 4.13, *а*) и невозможна передача прав доступа (рис. 4.13, *б*).

Очевидно, что любой путь, соединяющий двух субъектов, проходящий через объекты, по которому возможна передача прав доступа, не должен содержать фрагменты, приведенные на рис. 4.13, *б*. В то же время, очевидно, что любой такой путь, состоящий только из фрагментов, приведенных на рис. 4.13, *а*, является мостом.

Определение 4.7. Пусть x, $y \in O_0$, $x \neq y$ – различные объекты графа доступов $G_0 = (S_0, O_0, E_0)$, $\alpha \subseteq R$. Определим предикат *can_steal*(α, x, y, G_0), который будет истинным тогда и только тогда, когда $(x, y, \alpha) \cap E_0 = \varnothing$ и существуют графы $G_1 = (S_1, O_1, E_1)$, …, $G_N = (S_N, O_N, E_N)$ такие, что $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x, y, \alpha) \subset E_N$, при этом, если существует $(s, y, \gamma) \subset E_0$, где $\gamma \subseteq \alpha$, то справедливо неравенство $op_K \neq grant(\gamma, s, z, y)$, где $z \in O_{K-1}$ для $K = 1, \ldots, N$.

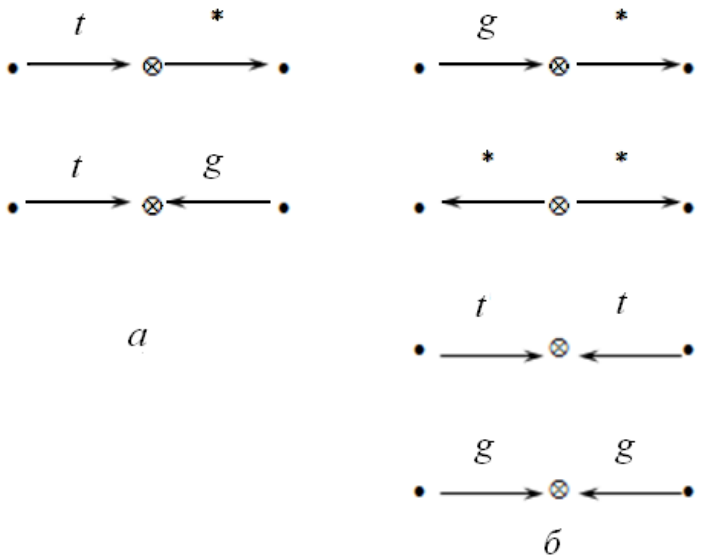

Рис. 4.13. Виды путей длины 2(*– или *t*, или *g*): *а*, *б* – пути, по которым, соответственно, возможна и невозможна передача прав доступа

Теорема 4.3. Пусть $G_0 = (S_0, O_0, E_0)$ – произвольный граф доступов, $x, y \in O_0$, $x \neq y$. Предикат *can_steal*(α, x, y, G_0) истинен тогда и только тогда, когда выполняются условия 1–4.

Условие 1. $(x, y, \alpha) \cap E_0 = \varnothing$.

Условие 2. Существуют объекты $s_1, \ldots, s_m \in O_0$:

$(s_i, y, \gamma_i) \subset E_0$ для $i = 1, \ldots, m$ и $\alpha = \gamma_1 \cup \ldots \cup \gamma_m$.

Условие 3. Являются истинными предикаты $can\_share(\{t\}, x, s_i, G_0)$, где $i = 1, \ldots, m$.

Условие 4. Для $i = 1, \ldots, m$ граф доступов G_0 не является графом вида, приведенного на рис. 4.14.

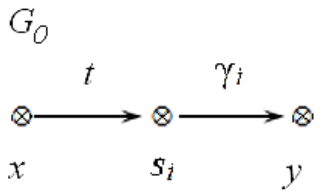

Рис. 4.14. Условие 4 теоремы 4.3

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 4.2.

4.2. РАСШИРЕННАЯ МОДЕЛЬ TAKE-GRANT. НАПРАВЛЕНИЯ РАЗВИТИЯ МОДЕЛИ TAKE-GRANT

Рассмотренные в классической модели *Take-Grant* способы анализа путей распространения прав доступа в системах с дискреционным управлением доступом имеют в большей степени теоретическое значение, так как, как правило, в реальных КС не реализуются столь сложные графы доступов, для анализа которых необходимо использовать теоремы 4.1–4.3. В то же время на основе классической модели были разработаны ее расширения, которые развивают идеи классической модели, предлагая новые механизмы анализа, в большей степени применимые к современным системам защиты информации.

Рассмотрим три расширения модели.

1. Де-факто правила, предназначенные для поиска и анализа информационных потоков.

2. Алгоритм построения замыкания графа доступов и информационных потоков.

3. Способы анализа путей распространения прав доступа или информационных потоков.

Вместо прав доступа *take* и *grant* в расширенной модели в первую очередь рассматриваются права доступа *read* и *write*, наличие которых

у субъектов системы является причиной возникновения информационных потоков.

Расширенная модель *Take-Grant* строится на основе классической модели. Ее основными элементами являются:

O – множество объектов;

$S \subseteq O$ – множество субъектов;

$R = \{r_1, r_2, \ldots, r_m\} \cup \{t, g\} \cup \{r, w\}$ – множество видов прав доступа и видов информационных потоков, где r(*read*) – право на чтение или информационный поток на чтение, w(*write*) – право на запись или информационный поток на запись;

$G = (S, O, E \cup F)$ – конечный помеченный ориентированный без петель граф доступов и информационных потоков, описывающий состояние системы. Элементы множеств S, O являются вершинами графа. Элементы множества $E \subseteq O \times O \times R$ являются «реальными» ребрами графа, соответствующими правам доступа, и в графе доступов обозначаются сплошными линиями. Элементы множества $F \subseteq O \times O \times \{r, w\}$ являются «мнимыми» ребрами, соответствующими информационным потокам, и в графе доступов обозначаются пунктирными линиями. Каждое «реальное» ребро помечено непустым подмножеством множества видов прав доступа R, каждое «мнимое» ребро помечено непустым подмножеством множества $\{r, w\}$.

Порядок перехода системы расширенной модели *Take-Grant* из состояния в состояние определяется де-юре и де-факто правилами преобразования графа доступов и информационных потоков. Преобразование графа G в граф G' в результате выполнения правила *op* обозначим следующим образом:

$$G \vdash_{op} G'.$$

Определение де-юре правил *take*, *grant*, *create*, *remove* совпадает с определением этих правил в классической модели *Take-Grant*. Де-юре правила применяются только к «реальным» ребрам (элементам множества E).

Де-факто правила применяются к «реальным» или «мнимым» ребрам (элементам множества $E \cup F$), помеченным r или w. Результатом применения де-факто правил является добавление новых «мнимых» ребер во множество F. Рассматриваются шесть де-факто правил: два вспомогательных и четыре основных.

Рассмотрим порядок применения де-юре правил преобразования графа доступов. В отличие от де-юре правил для применения трех из шести де-факто правил требуется участие двух субъектов (рис. 4.15–4.20).

Рассмотрим условия применения де-факто правил в исходном состоянии $G = (S, O, E \cup F)$ и результаты их применения в результирующем состоянии $G' = (S, O, E \cup F')$ (табл. 4.2).

Таблица 4.2

Де-факто правила расширенной модели *Take-Grant*

Правила	Исходное состояние $G = (S, O, E \cup F)$	Результирующее состояние $G' = (S, O, E \cup F')$
Первое правило	$x \in S, y \in O,$ $(x, y, r) \in E \cup F$	$F' = F \cup \{(y, x, w), (x, y, r)\}$
Второе правило	$x \in S, y \in O,$ $(x, y, w) \in E \cup F$	$F' = F \cup \{(y, x, r), (x, y, w)\}$
spy(x, y, z)	$x, y \in S, x \neq z,$ $\{(x, y, r), (y, z, r)\} \subset E \cup F$	$F' = F \cup \{(x, z, r), (z, x, w)\}$
find(x, y, z)	$x, y \in S, z \in O, x \neq z,$ $\{(x, y, w), (y, z, w)\} \subset E \cup F$	$F' = F \cup \{(x, z, w), (z, x, r)\}$
post(x, y, z)	$x, y \in S, z \in O, x \neq y,$ $\{(x, z, r), (y, z, w)\} \subset E \cup F$	$F' = F \cup \{(x, y, r), (y, x, w)\}$
pass(x, y, z)	$x \in S, y, z \in O, y \neq z$ $\{(x, y, w), (x, z, r)\} \subset E \cup F$	$F' = F \cup \{(y, z, r), (z, y, w)\}$

Из определения де-факто правил следует, что для анализа информационных потоков достаточно рассматривать потоки одного вида либо на чтение, либо на запись. В дальнейшем будем рассматривать только информационные потоки на запись. Будем также предполагать, что при возникновении информационного потока не накладывается ограничений на кооперацию субъектов системы, участвующих в этом процессе.

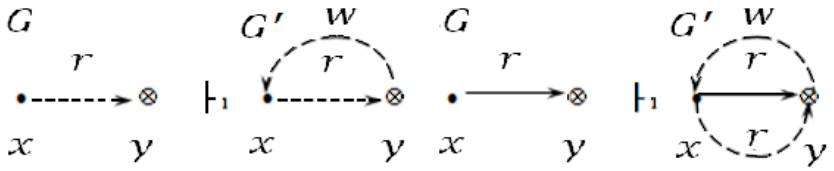

Рис. 4.15. Применение первого де-факто правила

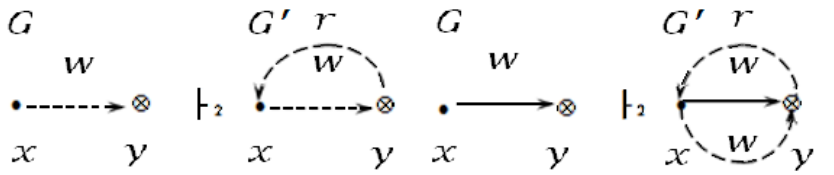

Рис. 4.16. Применение второго де-факто правила

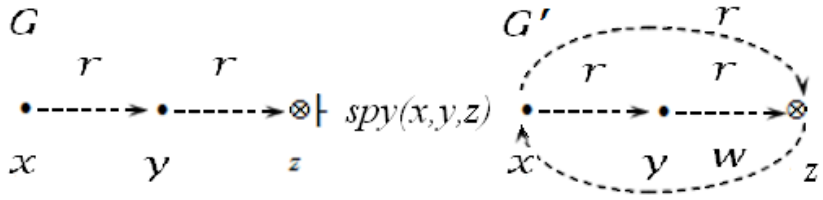

Рис. 4.17. Применение правила *spy(x, y, z)*

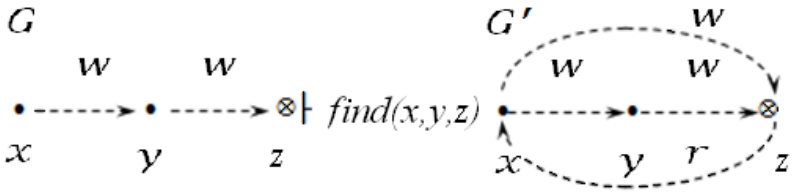

Рис. 4.18. Применение правила *find(x, y, z)*

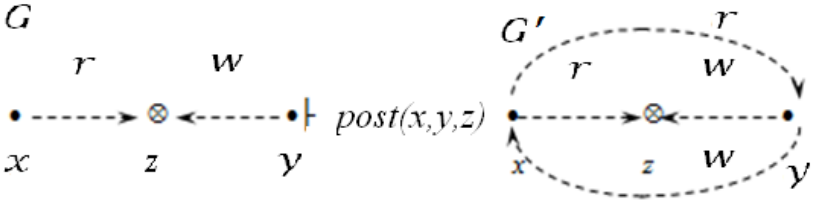

Рис.4.19. Применение правила *post(x, y, z)*

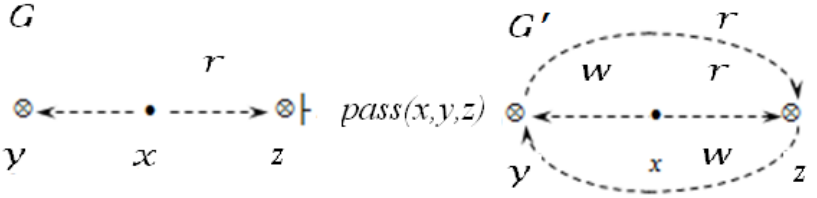

Рис. 4.20. Применение правила *pass(x, y, z)*

Определение 4.8. Пусть $x, y \in O_0$, $x \neq y$ – различные объекты графа доступов и информационных потоков $G_0 = (S_0, O_0, E_0 \cup F_0)$. Определим предикат *can_write*(x, y, G_0), который будет истинным тогда и только тогда, когда существуют графы $G_1 = (S_1, O_1, E_1 \cup F_1)$, …, $G_N = (S_N, O_N, E_N \cup F_N)$ и де-юре или де-факто правила op_1, …, op_N такие, что $G_0 \vdash_{op1} G_1 \vdash_{op2} \ldots \vdash_{opN} G_N$ и $(x, y, w) \in F_N$, где $N \geq 0$.

Для проверки истинности предиката *can_write*(x, y, G_0) также следует определить необходимые и достаточные условия, задача проверки которых алгоритмически разрешима.

Теорема 4.4. Пусть $G_0 = (S_0, O_0, E_0 \cup F_0)$ граф доступов и информационных потоков, $x, y \in O_0$, $x \neq y$. Тогда предикат *can_write*(x, y, G_0) истинен тогда и только тогда, когда существуют объекты $o_1, \ldots, o_m \in O_0$, где $o_1 = x$, $o_m = y$, такие, что или $m = 2$ и $(x, y, w) \in F_0$, или для $i = 1, \ldots, m - 1$ выполняется одно из условий:

- $o_i \in S_0$ и истинен предикат *can_share*$(\{w\}, o_i, o_{i+1}, G_0)$ или $(o_i, o_{i+1}, w) \in E_0 \cup F_0$;
- $o_{i+1} \in S_0$ и истинен предикат *can_share*$(\{r\}, o_{i+1}, o_i, G_0)$ или $(o_{i+1}, o_i, r) \in E_0 \cup F_0$;
- $o_i, o_{i+1} \in S_0$ и истинен предикат *can_share*$(\alpha, o_i, o_{i+1}, G_0)$ или истинен предикат *can_share*$(\alpha, o_{i+1}, o_i, G_0)$, где $\alpha \in \{t, g\}$.

Доказательство. Доказательство теоремы осуществляется аналогично доказательству теорем 4.2 и 4.3.

4.3. ПОСТРОЕНИЕ ЗАМЫКАНИЯ ГРАФА ДОСТУПОВ И ИНФОРМАЦИОННЫХ ПОТОКОВ

Для проверки истинности предиката *can_share*(α, x, y, G_0) или *can_write*(x, y, G_0) для многих пар вершин неэффективно использовать алгоритмы проверки условий теорем 4.2, 4.3, 4.4. Эффективнее применять алгоритмы, позволяющие осуществить проверку истинности данных предикатов сразу для всех пар вершин. Такие алгоритмы реализуют преобразование графа доступов и информационных потоков в его замыкание.

Определение 5.9. Пусть $G = (S, O, E \cup F)$ – граф доступов и информационных потоков такой, что для каждого $s \in S$ существует $o \in O$ и при этом $(s, o, \{t, g, r, w\}) \subset E$. Тогда замыканием (или де-факто-замыканием) графа G называется граф доступов и информационных потоков $G^* = (S, O, E^* \cup F^*)$, полученный из G применением последовательности правил *take*, *grant* и де-факто правил. При этом применение к графу G^* указанных правил не приводит к появлению в нем новых ребер.

Алгоритм построения замыкания графа доступов состоит из трех этапов: построение *tg*-замыкания, построение де-юре-замыкания, построение замыкания.

Определение 4.10. Пусть $G = (S, O, E \cup F)$ – граф доступов и информационных потоков такой, что для каждого $s \in S$ существует $o \in O$ и при этом $(s, o, \{t, g, r, w\}) \subset E$. Тогда *tg*-замыканием графа G называется граф доступов и информационных потоков $G^{tg} = (S, O, E^{tg} \cup F)$, полученный из G

применением последовательности правил *take* или *grant*. При этом каждое ребро $(o_1, o_2, \alpha) \in E^{tg} \backslash E$ имеет вид (o_1, o_2, t) или (o_1, o_2, g), и применение к графу G^{tg} правил *take* или *grant* не приводит к появлению в нем новых ребер указанного вида.

Определение 4.11. Пусть $G = (S, O, E \cup F)$ граф доступов и информационных потоков такой, что для каждого $s \in S$ существует $o \in O$ и при этом $(s, o, \{t, g, r, w\}) \subset E$. Тогда де-юре-замыканием графа G называется граф доступов и информационных потоков $G^{\text{де-юре}} = (S, O, E^{\text{де-юре}} \cup F)$, полученный из G применением последовательности правил *take* или *grant*. При этом применение к графу $G^{\text{де-юре}}$ правил *take* или *grant* не приводит к появлению в нем новых ребер.

Алгоритм 4.1. Алгоритм построения *tg*-замыкания графа доступов и информационных потоков $G = (S, O, E \cup F)$ состоит из пяти шагов.

Шаг 1. Для каждого $s \in S$ выполнить правило *create*$(\{t, g, r, w\}, s, o)$; при этом создаваемые объекты занести во множество O, создаваемые ребра – во множество E.

Шаг 2. Инициализировать: $L = \{(x, y, \alpha) \in E: \alpha \in \{t, g\}\}$ – список ребер графа доступов и информационных потоков и $N = \varnothing$ – множество вершин.

Шаг 3. Выбрать из списка L первое ребро (x, y, α). Занести x, y во множество N. Удалить ребро (x, y, α) из списка L.

Шаг 4. Для всех вершин $z \in N$ проверить возможность применения правил *take* или *grant* на тройке вершин x, y, z с использованием ребра (x, y, α), выбранном на шаге 3. Если в результате применения правил *take* или *grant* появляются новые ребра вида (a, b, β), где $\{a, b\} \subset \{x, y, z\}$ и $\beta \in \{t, g\}$, занести их в конец списка L и множество E.

Шаг 5. Пока список L не пуст, перейти на шаг 3.

Очевидно, что вычислительная сложность данного алгоритма пропорциональна $|O|^3$.

Теорема 4.5. Алгоритм 4.1 корректно строит *tg*-замыкание графа доступов и информационных потоков.

Доказательство. Пусть $G' = (S, O, E' \cup F)$ – граф доступов и информационных потоков, полученный из $G = (S, O, E \cup F)$ после применения алгоритма 4.1. Пусть $G^{tg} = (S, O, E^{tg} \cup F)$ – *tg*-замыкание графа G. Необходимо доказать, что $G' = G^{tg}$.

Докажем от противного. Пусть существует ребро $(x, y, \alpha) \in E^{tg} \backslash E'$, где $\alpha \in \{t, g\}$. Тогда существуют два ребра (a, b, β), $(c, d, \gamma) \in E^{tg}$, использованные в правиле *take* или *grant*, применение которого привело к появлению ребра (x, y, α), где $\{x, y\} \subset \{a, b, c, d\}$, $|\{a, b, c, d\}| = 3$, $\beta, \gamma \in \{t, g\}$.

Если (a, b, β), $(c, d, \gamma) \in E'$, то согласно описанию алгоритма 4.1 ребро $(x, y, \alpha) \in E'$ – противоречие. Следовательно, или $(a, b, \beta) \in E^{tg} \backslash E'$, или

$(c, d, \gamma) \in E^{tg} \backslash E'$. В то же время граф G^{tg} получен из G в результате применения конечной последовательности правил *take* и *grant*. Ребра (a, b, β), (c, d, γ) должны быть получены раньше, чем ребро (x, y, α). Таким образом, повторяя рассуждения для ребер, с использованием которых получены ребра (a, b, β), (c, d, γ), придем к противоречию, так как все ребра графа G изначально содержатся в графе G'. Следовательно, $E^{tg} = E'$ и $G' = G^{tg}$. Теорема доказана.

Алгоритм 4.2. Алгоритм построения де-юре-замыкания графа доступов и информационных потоков $G = (S, O, E \cup F)$ состоит из четырех шагов.

Шаг 1. Выполнить алгоритм 4.1.

Шаг 2. Для каждой пары ребер вида (x, y, t), $(y, z, \alpha) \in E^{tg}$, где $x \in S$, применить правило *take*(α, x, y, z) и, если полученное ребро $(x, z, \alpha) \notin E^{tg}$, то занести его во множество E^{tg}.

Шаг 3. Для каждой пары ребер вида (x, y, g), $(x, z, \alpha) \in E^{tg}$, где $x \in S$, применить правило *grant*(α, x, y, z) и, если полученное ребро $(y, z, \alpha) \notin E^{tg}$, то занести его во множество E^{tg}.

Шаг 4. Для каждой пары ребер вида (x, y, t), $(y, z, \alpha) \in E^{tg}$, где $x \in S$, применить правило *take*(α, x, y, z) и, если полученное ребро $(x, z, \alpha) \notin E^{tg}$, то занести его во множество E^{tg}.

Теорема 4.6. Алгоритм 4.2 корректно строит де-юре замыкание графа доступов и информационных потоков.

Доказательство. После построения *tg*-замыкания на первом шаге алгоритма 4.2 всем субъектам графа доступов необходимо выполнить две последовательности действий:

- используя правило *grant*, раздать права доступа и, используя правило *take*, забрать права доступа (рис. 4.21, *а*);
- используя правило *take*, забрать права доступа и, используя правило *grant*, раздать права доступа (рис. 4.21, *б*).

Объединяя указанные последовательности, получаем шаги 2–4 алгоритма.

Теорема доказана.

а *б*

Рис. 4.21. Примеры для последовательностей шагов алгоритма 5.2:
а – граф доступов для последовательности действий grant, take;
б – граф доступов для последовательности действий take, grant

Алгоритм 4.3. Алгоритм построения де-факто-замыкания графа доступов и информационных потоков $G = (S, O, E \cup F)$ состоит из шести шагов:

Шаг 1. Выполнить алгоритм 4.2.

Шаг 2. Для всех ребер $(x, y, \alpha) \in E^{\text{де-юре}} \cup F$, где $x \in S$, $\alpha \in \{w, r\}$, применить первые два де-факто правила. Если будут получены новые ребра, то занести их во множество F.

Шаг 3. Инициализировать: $L = \{(x, y, \alpha) \in E^{\text{де-юре}} \cup F: \alpha \in \{w, r\}\}$ – список ребер графа доступов и информационных потоков и $N = \varnothing$ – множество вершин.

Шаг 4. Выбрать из списка L первое ребро (x, y, α). Занести x, y в множество N. Удалить ребро (x, y, α) из списка L.

Шаг 5. Для всех вершин $z \in N$ проверить возможность применения де-факто правил на тройке вершин x, y, z с использованием ребра (x, y, α). Если в результате применения де-факто правил *spy*, *find*, *post*, *pass* появляются новые ребра вида (a, b, β), где $\{a, b\} \subset \{x, y, z\}$ и $\beta \in \{r, w\}$, то занести их в конец списка L и множество F.

Шаг 6. Пока список L не пуст, перейти на шаг 4.

Теорема 4.7. Алгоритм 4.3 корректно строит де-факто-замыкание графа доступов и информационных потоков.

Доказательство. Алгоритм 4.3 аналогичен алгоритму 4.1. Следовательно, доказательство теоремы 4.7 аналогично доказательству теоремы 4.5.

Рассмотрим задачу поиска и анализа информационных потоков в ином свете. Допустим, факт нежелательной передачи прав или информации уже состоялся. Каков наиболее вероятный путь его осуществления? В классической модели *Take-Grant* не дается прямого ответа на этот вопрос. Можно говорить, что есть возможность передачи прав доступа или возникновения информационного потока, но нельзя определить, какой из путей при этом использовался.

Рассмотрим подходы к решению задачи определения возможных путей передачи прав доступа или возникновения информационных потоков.

Предположим, что чем больше узлов на пути между вершинами, по которому произошла передача прав доступа или возник информационный поток, тем меньше вероятность использования этого пути.

Рассмотрим пример, представленный на рис. 4.22.

Интуитивно ясно, что наиболее вероятный путь передачи информации от субъекта z к субъекту x лежит через объект y. В то же время нарушитель для большей скрытности может специально использовать более длинный путь через a, b, c. Особенно если предположить, что информация в объекте y контролируется администратором безопасности системы.

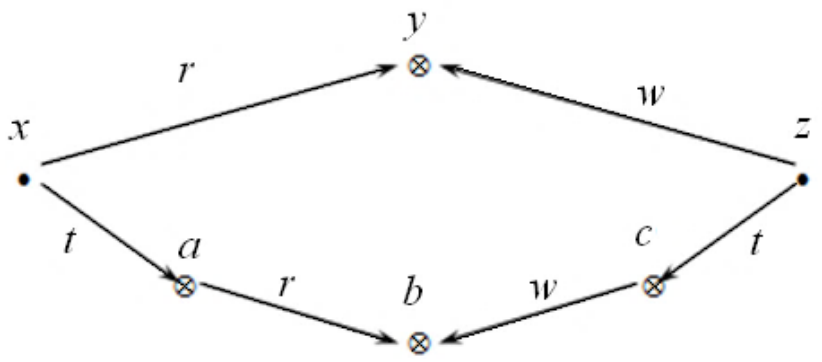

Рис. 4.22. Возможные пути возникновения информационного потока на запись от *z* к *x*

Рассмотрим другой пример. Какой из двух путей возникновения информационного потока, представленных на рис. 4.23, более вероятный?

Путь, представленный на рис. 4.23, *в* реализуется за счет активных действий субъекта *x*, заинтересованного в возникновении информационного потока. По этой причине наиболее вероятным, как правило, будет именно этот путь.

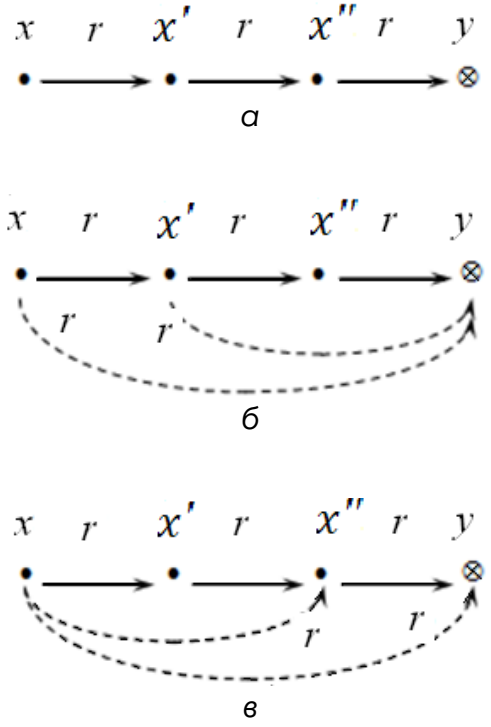

Рис. 4.23. Исходный граф (*а*) и пути (*б*, *в*) возникновения информационных потоков на чтение от *x* к *y*

Таким образом, в расширенную модель *Take-Grant* можно включить понятие вероятности или стоимости пути передачи прав доступа или информации. Путям меньшей стоимости соответствует наивысшая вероятность, и их надо исследовать в первую очередь. Есть два основных подхода к определению стоимости путей.

Первый подход основан на присваивании стоимости ребрам графа доступов, находящимся на пути передачи прав доступа или возникновения информационного потока. В этом случае стоимость ребра определяется в зависимости от прав доступа, которыми оно помечено, а стоимость пути есть сумма стоимостей пройденных ребер.

Второй подход основан на присваивании стоимости каждому используемому де-юре или де-факто правилу. Стоимость правила при этом может быть выбрана, исходя из условий функционирования системы *Take-Grant* и может:

- являться константой;
- зависеть от специфики правила;
- зависеть от числа и состава участников при применении правила;
- зависеть от степени требуемого взаимодействия субъектов.

Стоимость пути в этом случае определяется как сумма стоимостей примененных правил.

4.4. ПРЕДСТАВЛЕНИЕ СИСТЕМ TAKE-GRANT СИСТЕМАМИ ХРУ

Решение задачи представления систем классической модели *Take-Grant* системами модели ХРУ позволяет лучше изучить основные свойства двух моделей систем дискреционного управления доступом.

Построим гомоморфизм системы *Take-Grant* и системы ХРУ.

Пусть состояние системы *Take-Grant* описывается графом $G = (S_{tg}, O_{tg}, E)$, а R_{tg} – множество прав доступа системы *Take-Grant*.

Положим для системы ХРУ:

$R = R_{tg} \cup \{own\}$ – множество прав доступа;

$S = O = O_{tg}$ – множество субъектов и объектов системы ХРУ;

$M_{|S|\times|S|}$ – матрица доступов, где для $x, y \in O_{tg}$, если $(x, y, r) \in E$, то $r \in M[x, y]$, и для $s \in S_{tg}$ выполняется условие $own \in M[s, s]$;

$q = (S, O, M)$ – состояние системы.

Переход системы ХРУ в соответствии с правилами модели *Take-Grant* осуществляется в результате применения команд пяти видов для каждого $\alpha = \{r_1, \ldots, r_k\} \subset R_{tg}$.

1. *command take_*$\alpha(x, y, z)$
if ($own \in M[x, x]$) *and* ($t \in M[x, y]$) *and* ($r_1 \in M[y, z]$) *and* …
and ($r_k \in M[y, z]$) *then*
«внести» право r_1 в $M[x, z]$;
…
«внести» право r_k в $M[x, z]$;
endif
end.
2. *command grant_*$\alpha(x, y, z)$
if ($own \in M[x, x]$) *and* ($g \in M[x, y]$) *and* ($r_1 \in M[x, z]$) *and* …
and ($r_k \in M[x, z]$) *then*
«внести» право r_1 в $M[y, z]$;
…
«внести» право r_k в $M[y, z]$;
endif
end.
3. *command create_object_*$\alpha(x, y)$
if ($own \in M[x, x]$) *then*
«создать» субъекта y;
«внести» право r_1 в $M[x, y]$;
…
«внести» право r_k в $M[x, y]$;
endif
end.
4. *command create_subject_*$\alpha(x, y)$
if ($own \in M[x, x]$) *then*
«создать» субъекта y;
«внести» право *own* в $M[y, y]$;
«внести» право r_1 в $M[x, y]$;
…
«внести» право r_k в $M[x, y]$;
endif
end.
5. *command remove_*$\alpha(x, y)$
if ($own \in M[x, x]$) *then*
«удалить» право r_1 из $M[x, y]$;
…
«удалить» право r_k из $M[x, y]$;
endif
end.

В приведенных командах, реализующих правила *take* и *grant*, не учитывается случай, когда выполняется условие $x = z$. В модели *Take-Grant* не допускается появление петель в графе доступов. Таким образом, в командах *take*_α(x, y, z) и *grant*_α(x, y, z) системы ХРУ следует осуществить проверку условия $x \neq z$. Непосредственная реализация проверки данного условия потребует значительного усложнения рассмотренного представления систем *Take-Grant* системами ХРУ и выходит за рамки рассматриваемых в пособии вопросов моделирования управления доступом и информационными потоками в КС.

Кроме того, если исключить из определения модели *Take-Grant* требование отсутствия петель в графе доступов, то рассмотренные в модели условия передачи прав доступа и реализации информационных потоков существенно не изменятся.

Контрольные вопросы и задания

1. Выразите команду *spy* расширенной модели *Take-Grant* через ее другие де-факто команды.
2. Как по аналогии с определением безопасного начального состояния в модели ХРУ и с использованием определения предиката can_share (α, x, y, G_0) определить безопасное начальное состояние в модели *Take-Grant*?
3. Как представить систему, построенную на основе классической модели *Take-Grant*, системой ТМД?
4. Как представить систему, построенную на основе расширенной модели *Take-Grant*, системой ТМД?
5. Является ли алгоритмически разрешимой задача проверки безопасности систем, построенных на основе классической или расширенной модели *Take-Grant*?

5. МОДЕЛИ КОМПЬЮТЕРНЫХ СИСТЕМ С МАНДАТНЫМ УПРАВЛЕНИЕМ ДОСТУПОМ. МОДЕЛЬ БЕЛЛА – ЛАПАДУЛЫ

5.1. КЛАССИЧЕСКАЯ МОДЕЛЬ БЕЛЛА – ЛАПАДУЛЫ

Классическая модель Белла – Лападулы (*Bell – LaPadula*) описана в 1975 г. и в настоящее время является основной моделью, предназначенной для анализа систем защиты, реализующих мандатное управление доступом.

В классической модели Белла – Лападулы рассматриваются условия, при выполнении которых в КС невозможно возникновение информационных потоков от объектов с большим уровнем конфиденциальности к объектам с меньшим уровнем конфиденциальности. Основными элементами классической модели Белла – Лападулы являются:

S – множество субъектов;

O – множество объектов;

$R = \{$*read, write, append* (доступ на запись в конец объекта), *execute*$\}$ – множество видов доступа и видов прав доступа;

$B = \{b \subseteq S \times O \times R\}$ – множество возможных множеств текущих доступов в системе;

$(L, \leq)$ – решетка уровней конфиденциальности, например: $L = \{U$ (*unclussified*), C(*confidential*), S (*secret*), TS (*topsecret*)$\}$, где $U < C < S < TS$;

$M = \{m_{|S| \times |O|}\}$ – множество возможных матриц доступов, где $m_{|S| \times |O|}$ – матрица доступов, $m[s, o] \subseteq R$ – права доступа субъекта sк объекту o;

$(f_s, f_o, f_c) \in F = L^s \times L^o \times L^s$ – тройка функций (f_s, f_o, f_c), задающих: f_s: $S \to L$ – уровень доступа субъекта; f_o: $O \to L$ – уровень конфиденциальности объекта; f_c: $S \to L$ – текущий уровень доступа субъекта, при этом для любого $s \in S$ выполняется неравенство $f_c(s) \leq f_s(s)$;

$V = B \times M \times F$– множество состояний системы;

Q – множество запросов системе;

D – множество ответов по запросам, например: $D = \{$*yes, no, error*$\}$;

$W \subseteq Q \times D \times V \times V$ – множество действий системы, где четверка $(q, d, v^*, v) \in W$ означает, что система по запросу qс ответом d перешла из состояния vв состояние v^*;

$N_0 = \{0, 1, 2, \ldots\}$ – множество значений времени;

X – множество функций x: $N_0 \to Q$, задающих все возможные последовательности запросов к системе;

Y – множество функций y: $N_0 \to D$, задающих все возможные последовательности ответов системы по запросам;

Z – множество функций z: $N_0 \to V$, задающих все возможные последовательности состояний системы.

Определение 5.1. $\Sigma(Q, D, W, z_0) \subseteq X \times Y \times Z$ называется системой, если для каждого $(x, y, z) \in \Sigma(Q, D, W, z_0)$ выполняется условие: для $t \in N_0$, $(x_t, y_t, z_{t+1}, z_t) \in W$, где z_0 начальное состояние системы. При этом каждый набор $(x, y, z) \in \Sigma(Q, D, W, z_0)$ называется реализацией системы, а $(x_t, y_t, z_{t+1}, z_t) \in W$ – действием системы в момент времени $t \in N_0$.

В классической модели Белла – Лападулы рассматриваются следующие запросы, входящие во множество Q:

1) запросы изменения множества текущих доступов b;
2) запросы изменения функций f;
3) запросы изменения прав доступа в матрице m.

Следующий список описывает изменения каждого элемента состояния системы. Конкретное решение по запросу включает возможность производить следующие изменения в состоянии системы:

1. Изменение текущих доступов:

- *получить доступ* (добавить тройку (субъект, объект, вид доступа) в текущее множество доступов b);
- *отменить доступ* (удалить тройку из текущего множества доступов b).

2. Изменение значений функций уровней конфиденциальности и доступа:

- *изменить уровень конфиденциальности объекта*;
- *изменить уровень доступа субъекта*.

3. Изменение прав доступа:

- *дать разрешение на доступ* (добавить право доступа в соответствующий элемент матрицы доступов m);
- *отменить разрешение на доступ* (удалить право доступа из соответствующего элемента матрицы доступов m).

Безопасность системы определяется с помощью трех свойств: *Ss*-свойства простой безопасности (*simple security*); * – свойства звезда; *ds*-свойства дискреционной безопасности (*discretionary security*).

Определение 5.2. Доступ $(s, o, r) \in S \times O \times R$ обладает *ss*-свойством относительно $f = (f_s, f_o, f_c) \in F$, если выполняется одно из условий:

- $r \in \{\text{execute, append}\}$;
- $r \in \{\text{read, write}\}$ и $f_s(s) \geq f_o(o)$.

Определение 5.3. Состояние системы $(b, m, f) \in V$ обладает *ss*-свойством, если каждый элемент $(s, o, r) \in b$ обладает *ss*-свойством относительно f.

Определение 5.4. Доступ $(s, o, r) \in S \times O \times R$ обладает *-свойством относительно $f = (f_s, f_o, f_c) \in F$, если выполняется одно из условий:

- $r = execute$;
- $r = append$ и $f_o(o) \geq f_c(s)$;
- $r = read$ и $f_c(s) \geq f_o(o)$;
- $r = write$ и $f_c(s) = f_o(o)$.

Определение 5.5. Состояние системы $(b, m, f) \in V$ обладает *-свойством, если каждый элемент $(s, o, r) \in b$обладает *-свойством относительно f.

Определение 5.6. Состояние системы $(b, m, f) \in V$ обладает *-свойством, относительно подмножества $S' \subseteq S$, если каждый элемент $(s, o, r) \in b$, где $s \in S'$, обладает *-свойством относительно f. При этом $S \backslash S'$ называется множеством доверенных субъектов, т. е. субъектов, имеющих право нарушать требования *-свойства.

Определение 5.7. Состояние системы $(b, m, f) \in V$ обладает *ds*-свойством, если для каждого элемента $(s, o, r) \in b$ выполняется условие $r \in m[s, o]$.

Определение 5.8. Состояние системы (b, m, f) называется безопасным, если оно обладает *-свойством относительно S', *ss*-свойством и *ds*-свойством.

Определение 5.9. Реализация системы $(x, y, z) \in \Sigma(Q, D, W, z_0)$ обладает *ss*-свойством (*-свойством, *ds*-свойством), если в последовательности $(z_0, z_1, \ldots)$ каждое состояние обладает *ss*-свойством (*-свойством, *ds*-свойством).

Определение 5.10. Система $\Sigma(Q, D, W, z_0)$ обладает *ss*-свойством (*-свойством, *ds*-свойством), если каждая ее реализация обладает *ss*-свойством (*-свойством, *ds*-свойством).

Определение 5.11. Система $\Sigma(Q, D, W, z_0)$ называется безопасной, если она обладает *ss*-свойством, *-свойством, *ds*-свойством одновременно.

Прокомментируем описанные свойства безопасности системы.

Во-первых, из обладания доступом *-свойством относительно f следует обладание этим доступом *ss*-свойством относительно f.

Во-вторых, из обладания системой *ss*-свойством следует, что в модели Белла – Лападулы выполняется запрет на чтение вверх, требуемый мандатной политикой безопасности. Кроме того, *ss*-свойство не допускает модификацию с использованием доступа *write*, если $f_s(s) < f_o(o)$. Таким образом, функция $f_s(s)$ задает для субъекта s верхний уровень конфиденциальности объектов, к которым он потенциально может получить доступ *read* или *write*.

В-третьих, поясним *-свойство. Если субъект s может понизить свой текущий уровень доступа до $f_c(s) = f_o(o)$, то он может получить доступ *write* к объекту o, но не доступ *read* к объектам o', чей уровень $f_o(o') > f_c(s)$. Хотя при этом, возможно, справедливо неравенство $f_s(s) \geq f_o(o')$, и в каких-то других состояниях системы субъект s может получить доступ *read* к объекту o'. Таким образом, *-свойство исключает появление в системе запрещенного информационного потока «сверху вниз» и соответствует требованиям мандатной политики безопасности (рис. 5.1).

Проверка безопасности системы по определению в большинстве случаев не может быть реализована на практике в связи с тем, что при этом требуется проверка безопасности всех реализаций системы, а их бесконечно много. Следовательно, необходимо определить и обосновать иные условия безопасности системы, которые можно проверять на практике. В классической модели Белла – Лападулы эти условия задаются для множества действий системы W.

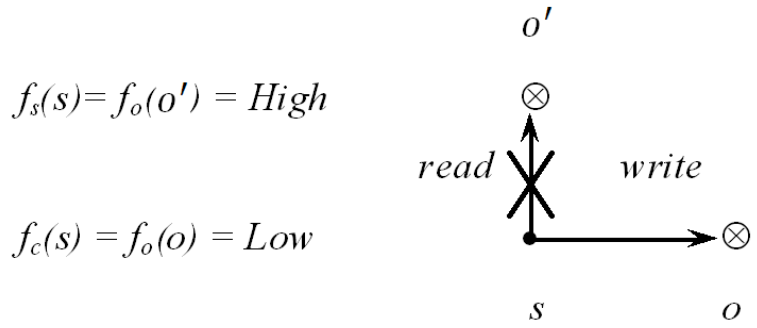

Рис. 5.1. Иллюстрация *-свойства

Теорема 5.1 (*A*1). Система $\Sigma(Q, D, W, z_0)$ обладает *ss*-свойством для любого начального состояния z_0, обладающего *ss*-свойством, тогда и только тогда, когда для каждого действия $(q, d, (b^*, m^*, f^*), (b, m, f)) \in W$ выполняются условия 1, 2.

Условие 1. Каждый доступ $(s, o, r) \in b^* \setminus b$ обладает *ss*-свойством относительно f^*.

Условие 2. Если $(s, o, r) \in b$ и не обладает *ss*-свойством относительно f^*, то $(s, o, r) \notin b^*$.

Доказательство. Сначала докажем достаточность условий.

Достаточность. Пусть выполнены условия 1 и 2 и пусть $(x, y, z) \in \Sigma(Q, D, W, z_0)$ – произвольная реализация системы. Тогда $(x_t, y_t, (b_{t+1}, m_{t+1}, f_{t+1}), (b_t, m_t, f_t)) \in W$, где $z_{t+1} = (b_{t+1}, m_{t+1}, f_{t+1})$, $z_t = (b_t, m_t, f_t)$ для $t \in N_0$.

Для $(s, o, r) \in b_{t+1}$ выполняется одно из условий: или $(s, o, r) \in b_{t+1} \setminus b_t$, или $(s, o, r) \in b_t$. Из условия1 следует, что состояние системы z_{t+1} пополнилось доступами, которые обладают *ss*-свойством относительно f^*. Из условия 2 следует, что доступы из b_t, которые не обладают *ss*-свойством относительно f^*, не входят в b_{t+1}. Следовательно, каждый доступ $(s, o, r) \in b_{t+1}$ обладает *ss*-свойством относительно f^* и по определению состояние z_{t+1} обладает *ss*-свойством для $t \in N_0$. Так как по условию и состояние z_0 обладает *ss*-свойством, то выбранная произвольная реализация (x, y, z) также обладает *ss*-свойством. Достаточность условий теоремы доказана.

Необходимость. Пусть система $\Sigma(Q, D, W, z_0)$ обладает *ss*-свойством. Будем считать, что во множество W входят только те действия системы, которые встречаются в ее реализациях. Тогда для каждого $(q, d, (b^*, m^*, f^*), (b, m, f)) \in W$ существует реализация $(x, y, z) \in \Sigma(Q, D, W, z_0)$ и существует $t \in N_0$: $(q, d, (b^*, m^*, f^*), (b, m, f)) = (x_t, y_t, z_{t+1}, z_t)$. Так как реализация системы (x, y, z) обладает *ss*-свойством, то и состояние $z_{t+1} = (b^*, m^*, f^*)$ обладает *ss*-свойством по определению. Следовательно, условия 1 и 2 выполняются. Необходимость условий теоремы доказана.

Теорема 5.2 (*A*2). Система $\Sigma(Q, D, W, z_0)$ обладает *-свойством относительно $S' \subseteq S$ для любого начального состояния z_0, обладающего *-свойством относительно S', тогда и только тогда, когда множество для каждого действия $(q, d, (b^*, m^*, f^*), (b, m, f)) \in W$ выполняются условия 1 и 2.

Условие 1. Для $s \in S'$ доступ $(s, o, r) \in b^* \setminus b$ обладает *-свойством относительно f^*.

Условие 2. Для $s \in S'$, если доступ $(s, o, r) \in b$ и не обладает *-свойством относительно f^*, то $(s, o, r) \notin b^*$.

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 5.1.

Теорема 5.3 (*A*3). Система $\Sigma(Q, D, W, z_0)$ обладает *ds*-свойством для любого начального состояния z_0, обладающего *ds*-свойством, тогда и только тогда, когда для каждого действия $(q, d, (b^*, m^*, f^*), (b, m, f)) \in W$ выполняются условия 1 и 2.

Условие 1. Для каждого $(s, o, r) \in b^* \setminus b$, выполняется условие $r \in m^*[s, o]$.

Условие 2. Если доступ $(s, o, r) \in b$ и $r \notin m^*[s, o]$, то $(s, o, r) \notin b^*$.

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 5.1.

Теорема 5.4 (базовая теорема безопасности (БТБ)). Система $\Sigma(Q, D, W, z_0)$ безопасна для безопасного z_0 тогда и только тогда, когда множество действий системы W удовлетворяет условиям теорем 5.1–5.3.

Доказательство. Данная теорема является следствием теорем 5.1–5.3.

Описанная классическая модель Белла – Лападулы предоставляет общий подход к построению систем, реализующих мандатную политику безопасности. В модели определяется, какими свойствам должны обладать состояния и действия системы, чтобы система была безопасной согласно данным определениям. В то же время в модели не задается точный порядок действий системы управления доступом по запросам на доступ субъектов к объектам.

Пример 5.1. Пусть субъект *s* запрашивает доступ *read* к объекту *o′*. В данной ситуации система может выбрать один из двух возможных ответов по запросу:

1) запретить субъекту *s* запрашиваемый им доступ *read* к объекту *o′*;

2) закрыть доступ *write* субъекта *s* к объекту *o*, повысить текущий уровень конфиденциальности $f_c(s)$ до *High*, разрешить субъекту *s* запрашиваемый им доступ *read* к объекту *o′*.

Каждый из описанных путей соответствует требованиям безопасности модели Белла – Лападулы. В реальных системах возможны более сложные ситуации, чем ситуация, описанная в примере 5.1. Кроме того, возможно использование в системе каких-то других видов доступа субъектов к объектам, которые потребуют дополнительного определения свойств безопасности, что не всегда легко сделать. В связи с этим большое значение имеет корректное определение свойств безопасности.

5.2. ПРИМЕР НЕКОРРЕКТНОГО ОПРЕДЕЛЕНИЯ СВОЙСТВ БЕЗОПАСНОСТИ

При использовании классической модели Белла – Лападулы важно учитывать тот факт, что в ней отсутствует логическая увязка условий выполнения системой свойств безопасности, данных в их определениях, с заложенными в модель условиями их проверки, необходимость и достаточность которых доказывается в теореме 5.4. В параграфе приводится пример некорректного определения основного свойства безопасности в модели Белла – Лападулы. Вместо *-свойства используется абсурдное с точки зрения здравого смысла **-свойство. Однако при этом не возникает никаких противоречий в логике доказательства теорем, определяющих условия проверки безопасности системы.

Определение 5.12. Доступ $(s, o, r) \in S \times O \times R$ обладает **-свойством относительно $f = (f_s, f_o, f_c) \in F$, если выполняется одно из условий:

- $r \in \{read, append, execute\}$;
- $r = write$ и $f_c(s) \geq f_o(o)$.

Определение 5.13. Состояние системы $(b, m, f) \in V$ обладает **-свойством, если каждый доступ $(s, o, r) \in b$ обладает **-свойством относительно f.

Определение 5.14. Состояние системы $(b, m, f) \in V$ называется **-безопасным, если обладает *ss*-свойством, **-свойством, *ds*-свойством одновременно.

Определение 5.15. Реализация (x, y, z) системы $\Sigma(Q, D, W, z_0)$ обладает **-свойством, если в последовательности $(z_0, z_1, \ldots)$ каждое состояние обладает **-свойством.

Определение 5.16. Система $\Sigma(Q, D, W, z_0)$ обладает **-свойством, если каждая ее реализация обладает **-свойством.

Определение 5.17. Система $\Sigma(Q, D, W, z_0)$ называется **-безопасной, если она обладает *ss*-свойством, **-свойством, *ds*-свойством одновременно.

Теорема 5.5 (*A2*).** Система $\Sigma(Q, D, W, z_0)$ обладает **-свойством для любого начального состояния z_0, обладающего **-свойством, тогда и только тогда, когда для каждого действия $(q, d, (b^*, m^*, f^*), (b, m, f)) \in W$ выполняются условия 1 и 2.

Условие 1. Каждый доступ $(s, o, r) \in b^* \setminus b$ обладает **-свойством относительно f^*.

Условие 2. Если доступ $(s, o, r) \in b$ и не обладает **-свойством относительно f^*, то $(s, o, r) \notin b^*$.

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 5.1.

Теорема 5.6 (БТБ).** Система $\Sigma(Q, D, W, z_0)$ **-безопасна для z_0 **-безопасного состояния тогда и только тогда, множество действий системы W удовлетворяет условиям теорем 5.1, 5.3, 5.5.

Доказательство. Данная теорема является следствием теорем 5.1, 5.3, 5.5.

5.3. ПОЛИТИКА LOW-WATER-MARK В МОДЕЛИ БЕЛЛА – ЛАПАДУЛЫ

Среди проблем использования модели Белла – Лападулы особо выделяется проблема отсутствия в модели определения порядка действий системы при переходе из состояния в состояние. Данная проблема иллюстрируется на примере политики *low-water-mark*.

Политика *low-water-mark* реализуется в рассматриваемой далее модели мандатной политики целостности информации Биба. В то же время

используемый в политике *low-water-mark* подход к заданию правил переходов системы из состояния в состояние может быть использован для демонстрации уязвимости определений безопасности в модели Белла – Лападулы.

При реализации политики *low-water-mark* в модели Белла – Лападулы задается порядок безопасного функционирования системы в случае, когда по запросу субъекта ему всегда необходимо предоставлять доступ на запись в объект.

Основными элементами реализации политики *low-water-mark* в модели Белла – Лападулы являются:

S – множество субъектов системы;

O – множество объектов системы;

$R = \{read, write\}$ – множество видов и прав доступа;

$B = \{b \subseteq S \times O \times R\}$ – множество возможных множеств текущих доступов в системе;

$(L, \leq)$ – решетка уровней конфиденциальности;

$(f_s, f_o) \in F = L^s \times L^o$ – двойка функций (f_s, f_o), задающих: f_s: $S \to L$ – уровень доступа субъекта; f_o: $O \to L$ – уровень конфиденциальности объекта;

$V = B \times F$– множество состояний системы;

$OP = \{Read, Write, Reset\}$ – множество операций (табл. 5.1).

$W \subseteq OP \times V \times V$ – множество действий системы, где тройка $(op, (b^*, f^*), (b, f)) \in W$ означает, что система в результате выполнения операции $op \in OP$ перешла из состояния (b, f) в состояние (b^*, f^*).

Таблица 5.1

Условия и результаты выполнения операций

Операция	Условия выполнения	Результат выполнения операции
Read(*s*, *o*)	$f_s(s) \geq f_o(o)$	$f^* = f$; $b^* = b \cup \{(s, o, read)\}$
Write(*s*, *o*)	$f_s(s) \leq f_o(o)$	$f_s^* = f_s$; $f_o^*(o) = f_s(s)$; для каждого $o' \neq o$ справедливо равенство $f_o^*(o') = f_o(o')$; если $f_o^*(o) < f_o(o)$, то содержимое o очищается; $b^* = b \cup \{(s, o, write)\}$
Reset(*s*, *o*)	$f_s(s) > f_o(o)$	$f_s^* = f_s$; $b^* = b$; для каждого $o' \neq o$ справедливо равенство $f_o^*(o') = f_o(o')$; $f_o^*(o) = \oplus L$ (максимальный уровень конфиденциальности)

В результате выполнения операции *Write* уровень конфиденциальности объекта понижается до уровня доступа субъекта. Если это понижение реально происходит, то вся информация в объекте стирается. В результате выполнения операции *Reset* уровень конфиденциальности объекта становится максимально возможным в системе. При этом в системах, в которых

несколько субъектов одновременно могут запрашивать доступ к одному и тому же объекту, может потребоваться уточнение порядка использования операций *Write* и *Reset*.

Дадим определения *ss*-свойства и *-свойства для реализации политики *low-watermark* в модели Белла – Лападулы.

Определение 5.18. Доступ $(s, o, r) \in S \times O \times R$ обладает *ss*-свойством относительно $f \in F$, если $r \in \{read, write\}$ и $f_s(s) \geq f_o(o)$.

Определение 5.19. Доступ $(s, o, r) \in S \times O \times R$ обладает *-свойством относительно $f \in F$, если он удовлетворяет одному из условий:

- $r = read$ и $f_s(s) \geq f_o(o)$;
- $r = write$ и $f_s(s) = f_o(o)$.

Определение 5.20. Состояние системы $(b, f) \in V$ обладает *ss*-свойством (*-свойством), если каждый элемент $(s, o, r) \in b$ обладает *ss*-свойством (*-свойством) относительно f.

Определение 5.21. Состояние системы называется безопасным, если оно обладает *ss*-свойством и *-свойством одновременно.

Лемма 5.1. Операции *Read*, *Write*, *Reset* переводят систему из безопасного состояния в безопасное состояние.

Доказательство. Из описания операций *Read*, *Write*, *Reset* следует, что в результате их выполнения последующее состояние системы обладает *ss*-свойством и *-свойством, если исходное состояние обладало *ss*-свойством и *-свойством. Лемма доказана.

Заметим, что условие стирания информации при выполнении операции *Write* является существенным. Хотя при его отсутствии лемма 5.1 формально останется истинной. Однако в этом случае система не будет безопасной, так как возможно возникновение запрещенного информационного потока (рис. 5.2).

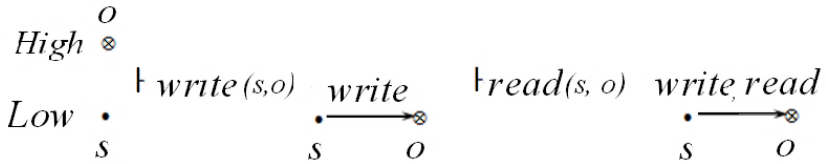

Рис. 5.2. Реализация запрещенного информационного потока

Например, субъект, запрашивая доступ на запись к объекту, понижает его уровень конфиденциальности, после чего он может запросить доступ на чтение к этому объекту.

Данный пример еще раз демонстрирует уязвимость определений безопасности в классической модели Белла – Лападулы.

5.4. ПРИМЕРЫ РЕАЛИЗАЦИИ ЗАПРЕЩЕННЫХ ИНФОРМАЦИОННЫХ ПОТОКОВ

Основной проблемой при реализации мандатной политики безопасности является обеспечение безопасности информационных потоков. Это связано с тем, что крайне сложно в современных КС выявить возможные запрещенные информационные потоки по памяти и, особенно, по времени. Кроме того, разработчиками КС мандатного управления доступом нередко упускается из виду, что система защиты должна не только препятствовать доступу к объектам с высоким уровнем конфиденциальности субъектов, не имеющих на это прав. Она должна также обеспечивать защиту от возникновения информационных потоков от объектов с большим уровнем конфиденциальности к объектам с меньшим уровнем конфиденциальности, реализуемых с использованием кооперации субъектов, имеющих доступ к конфиденциальной информации.

Рассмотрим ряд примеров программ, представленных на некотором абстрактном языке программирования высокого уровня, иллюстрирующих трудности практической реализации механизма безопасности информационных потоков в системах мандатного управления доступом.

Пример 5.2. Запрещенные информационные потоки по памяти с использованием локальных и логических переменных.

Используем обозначения:

f_1 – объект-файл с уровнем конфиденциальности *High*, который может содержать запись «*A*» или запись «*B*»;

f_2 – объект-файл с уровнем конфиденциальности *Low* < *High*.

Приведенные ниже процедуры реализуют запрещенные информационные потоки по памяти через локальную переменную (процедура p_1) и логическую переменную (процедура p_2):

Procedure p_1(*f_1*: *file*, *f_2*: *file*)
Open f_1 *for read*;
Read A from f_1;
Close f_1;
Open f_2 *for write*;
Write A to f_2;
Close f_2;
End.

Procedure p_2(*f_1*: *file*, *f_2*: *file*)
Open f_1 *for read*;
If (f_1 = «*a*») *Then*
Close f_1;
Open f_2 *for write*;
Write «*a*» *to* f_2;
Else
Close f_1;
Open f_2 *for write*;
Write «*b*» *to* F_2;
End If;
Close f_2;
End.

Анализируя процедуры p_1 и p_2, можно предложить следующий простой способ предотвращения реализуемых ими запрещенных информационных потоков. Процесс, единожды получивший доступ на чтение к файлу с некоторым уровнем конфиденциальности, не должен получать доступ на запись к файлам с более низким уровнем конфиденциальности. Такое решение может быть применено для пользовательских процессов в ОС. Однако оно не может быть признано универсальным, так как системные процессы, например монитор ссылок, при мандатном управлении доступом должны одновременно оперировать с данными различных уровней конфиденциальности.

Другой путь – инициализация каждой переменной процесса как объекта доступа – также не может являться оптимальным.

Для предотвращения запрещенных информационных потоков через локальные или логические переменные при написании программ в тех случаях, когда это возможно, следует следовать рекомендациям:

- открывать все файлы, необходимые для работы программы в начале ее выполнения;
- закрывать все файлы в конце выполнения программы;
- непосредственную обработку информации из открытых файлов осуществлять во внутренних процедурах, использующих только локальные переменные.

Пример 5.3. Реализация запрещенного информационного потока по времени.

Используем обозначения:

f_1 – объект-файл с уровнем конфиденциальности *High*, который может содержать запись «*a*» или запись «*в*»;

f_2 – объект-файл с уровнем конфиденциальности *Low*<*High*;

s_1 – субъект с высоким уровнем доступа *High*, работающий по программе:

Process s_1(F_1: *file*)
    Open F_1 *for read*;
    While F_1 = «*A*» *Do End*;
    CloseF_1;
End.

s_2 – субъект-нарушитель с низким уровнем доступа *Low*, работающий по программе:

Process s_2(f_1: *file*, f_2: *file*)
    Open f_2 *for write*;
    If (*Stop* s_1) *Then*
    Write «*B*» *to* f_2;
Else
    Write «*A*» *to* f_2;
End If
    Close f_2;
End;

Предполагается, что выполняются следующие условия:

$f_o(f_1)$ = *High*;
$f_o(f_2)$ = *Low*;
$f_s(s_1) = f_s(s_2)$ = *High*;
$f_c(s_1)$ = *High*;
$f_c(s_2)$ = *Low*.

Субъект-нарушитель s_2 действует одновременно с субъектом s_1, проверяя его состояние. При этом в зависимости от результата работы с конфиденциальным объектом-файлом f_1 субъекта s_1, который либо сразу завершит работу, либо «зависнет», субъект s_2 записывает данные в объект-файл f_2 с низким уровнем конфиденциальности (рис. 5.3).

Для предотвращения запрещенных информационных потоков по времени данного вида можно потребовать запрета получения процессами-субъектами низкого уровня доступа к информации о результатах работы процессов-субъектов высокого уровня доступа. В то же время такое решение не всегда возможно на практике. Кроме того, возможны иные способы реализации запрещенных информационных потоков по времени, для кото-

рых это решение неэффективно. Так, большинство современных ОС позволяет при открытии процессом файла определять параметры совместного с другими процессами его использования. Например, если процесс открывает файл на чтение, то он может разрешить или запретить другим процессам одновременное чтение этого файла. Системная переменная ОС, применяемая для определения условий совместного использования файла, может быть использована для реализации запрещенных информационных потоков по времени.

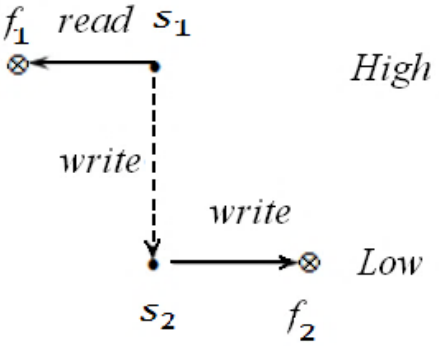

Рис. 5.3. Пример реализации информационного потока по времени с использованием «зависания» субъекта

Анализируя рассмотренные примеры, целесообразно сделать следующие выводы: без дополнительных уточнений свойств политики безопасности, порядка написания кода программ, определения порядка конфигурирования и администрирования невозможна реализация КС с мандатным управлением доступом. В то же время слишком строгая политика безопасности может привести к трудностям или даже невозможности практического использования системы защиты. Кроме того, доопределение и усложнение свойств безопасности также несет в себе угрозу ошибки и, как следствие, угрозу неадекватности реализации формальной модели и политики безопасности в КС.

5.5. БЕЗОПАСНОСТЬ ПЕРЕХОДОВ

Как уже было отмечено, в классической модели Белла – Лападулы не описывается точный порядок действий системы при переходе из состояния в состояние. Частично этот недостаток был устранен в модели

low-water-mark, где предлагается оригинальное определение свойств безопасности модели Белла – Лападулы с использованием вместо множества действий системы функции переходов.

При рассмотрении этого подхода будем считать $R = \{read, write\}$ и для каждого $s \in S$ справедливо равенство $f_s(s) = f_c(s)$. Исключим из рассмотрения матрицу доступов m и множество ответов системы D. Вместо множества действий системы W используем функцию переходов:

T: $Q \times V \to V$, где $T(q, v) = v^*$– переход из состояния vпо команде q в состояние v^*.

В этом случае будем обозначать систему через $\Sigma(V, T, z_0)$.

Далее переопределим *ss*-свойство и *-свойство. Так как основные ограничения на доступ *write*следуют из *-свойства, то в *ss*-свойстве зададим ограничения только на *read*.

Определение 5.22. Доступ $(s, o, r) \in b$ обладает *ss*-свойством относительно f, если выполняется одно из условий:

- $r = write$;
- $r = read$ и $f_s(s) \geq f_o(o)$.

Определение 5.23. Доступ $(s, x, r) \in b$ обладает *-свойством относительно f, если выполняется одно из условий:

- $r = read$ и не существует $y \in O$: $(s, y, write) \in b$ и $f_o(x) > f_o(y)$;
- $r = write$ и не существует $y \in O$: $(s, y, read) \in b$ и $f_o(y) > f_o(x)$.

Заметим особо, что в *-свойстве не рассматривается уровень доступа субъекта посредника s. В этом нет необходимости, так как если требовать выполнения *-свойства и *ss*-свойства одновременно и считать, что субъект не может накапливать в себе информацию, то не возникает противоречий по существу с положениями мандатной политики безопасности.

Субъект может читать информацию из объектов с уровнем конфиденциальности не выше его уровня доступа, и при этом субъект не может реализовать запрещенный информационный поток «сверху вниз».

По аналогии со свойствами классической модели Белла – Лападулы определяются *ss*-свойства и *-свойства для состояния, реализации и системы в целом.

Теорема 5.7 (*A*1-*RW*). Система $\Sigma(V, T, z_0)$ обладает *ss*-свойством для любого начального состояния z_0, обладающего *ss*-свойством, тогда и только тогда, когда функция переходов $T(q, v) = v^*$ удовлетворяет условиям 1 и 2.

Условие 1. Если доступ $(s, o, read) \in b^* \setminus b$, то $f_s^*(s) \geq f_o^*(o)$.

Условие 2. Если доступ $(s, o, read) \in b$ и $f_s^*(s) < f_o^*(o)$, то $(s, o, read) \notin b^*$.

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 5.1.

Теорема 5.8 (*A*2-*RW*). Система $\Sigma(V, T, z_0)$ обладает *-свойством для любого начального состояния z_0, обладающего *-свойством, тогда и только тогда, когда функция переходов $T(q, v) = v^*$ удовлетворяет условиям 1 и 2.

Условие 1. Если $\{(s, x, read), (s, y, write)\} \cap (b^* \setminus b) \neq \varnothing$ и $\{(s, x, read), (s, y, write)\} \subseteq b^*$, то $f_o^*(y) \geq f_o^*(x)$.

Условие 2. Если $\{(s, x, read), (s, y, write)\} \subseteq b$ и $f_o^*(y) < f_o^*(x)$, то $\{(s, x, read), (s, y, write)\} \not\subset b^*$.

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 5.1.

Теорема 5.9 (БТБ-*RW*). Система $\Sigma(V, T, z_0)$ безопасна для безопасного начального состояния z_0 тогда и только тогда, когда выполнены условия теорем 5.7 и 5.8.

Доказательство. Теорема следует из теорем 5.7, 5.8.

В данном подходе *-свойство определено таким образом, что его смысл – предотвращение возникновения запрещенных информационных потоков «сверху вниз» становится более ясным, чем при использовании функции f_c в классической модели Белла – Лападулы. В этом заключена основная ценность рассмотренных определений основных свойств безопасности. Далее определим основные свойства безопасности модели Белла – Лападулы, используя определения свойств безопасности функции переходов T. При этом на T накладывается дополнительное ограничение – за один шаг работы системы вносится только одно изменение в одном из параметров, используемом при определении свойств безопасности, т. е. изменяется на один элемент либо множество текущих доступов, либо одно из значений одной из функций. При этом остальные параметры остаются неизменными.

Определение 5.24. Функция переходов $T(q, (b, f)) = (b^*, f^*)$ обладает *ss*-свойством, если выполнены следующие условия:

- если $(s, o, read) \in b^* \setminus b$, то $f_s(s) \geq f_o(o)$ и $f^* = f$;
- если $f_s(s) \neq f_s^*(s)$, то $f_o^* = f_o$, $b^* = b$, для $s' \neq s$ справедливо равенство $f_s^*(s') = f_s(s')$, и если $(s, o, read) \in b$, то $f_s^*(s) \geq f_o(o)$;
- если $f_o(o) \neq f_o^*(o)$, то $f_s^* = f_s$, $b^* = b$, для $o' \neq o$ справедливо равенство $f_o^*(o') = f_o(o')$, и если $(s, o, read) \in b$, то $f_s(s) \geq f_o^*(o)$.

Определение 5.25. Функция переходов $T(q, (b, f)) = (b^*, f^*)$ обладает *-свойством, если выполнены следующие условия:

- если $\{(s, x, read), (s, y, write)\} \subseteq b^*$ и $\{(s, x, read), (s, y, write)\} \not\subset b$, то $f^* = f$ и $f_o(y) \geq f_o(x)$;
- если $f_o(y) \neq f_o^*(y)$, то $b^* = b$, $f_s^* = f_s$, для $z \neq y$ справедливо равенство $f_o^*(z) = f_o(z)$, и если $\{(s, x, read), (s, y, write)\} \subseteq b$, то $f_o^*(y) \geq f_o(x)$, или если $\{(s, y, read), (s, x, write)\} \subseteq b$, то $f_o(x) \geq f_o^*(y)$.

Определение 5.26. Функция переходов T безопасна, если она обладает *ss*-свойством и *-свойством.

Теорема 5.10 (БТБ-*T*). Система $\Sigma(V, T, z_0)$ безопасна для безопасного начального состояния z_0, если ее функция переходов безопасна.

Доказательство. По аналогии с доказательством теоремы 5.1 необходимо показать, что если функция переходов $T(q, (b, f)) = (b^*, f^*)$ безопасна, то состояние (b^*, f^*) безопасно по условиям определений 5.22 и 5.23. Этот факт следует из условий определений 5.24 и 5.25.

Таким образом, при безопасном начальном состоянии любая реализация системы, а следовательно, и система в целом будут безопасными.

Теорема доказана.

В рамках изучения определений безопасности функции переходов можно рассмотреть вопрос о возможностях смены уровня конфиденциальности субъектов и объектов. Примем следующие обозначения:

- для $x \in S$, $c_s(x) \subseteq S$ – множество субъектов, имеющих право менять уровень доступа субъекта x,
- для $y \in O$, $c_o(y) \subseteq S$ – множество субъектов, имеющих право менять уровень конфиденциальности объекта y.

В этом случае в параметры функции переходов необходимо внести еще один параметр, определяющий субъекта, дающего запрос системе.

Определение 5.27. Функция переходов $T(s, q, (b, f)) = (b^*, f^*)$ безопасна в смысле администрирования, если выполнены условия:

- если $f_s^*(x) \neq f_s(x)$, то $s \in c_s(x)$;
- если $f_o^*(y) \neq f_o(y)$, то $s \in c_o(y)$.

Таким образом, задавая множества $c_s(x)$ и $c_o(y)$, возможно рассмотрение случаев, когда все субъекты могут менять уровни конфиденциальности объектов и уровни доступа субъектов; никто кроме системного администратора не может менять уровни конфиденциальности объектов и уровни доступа субъектов.

5.6. МОДЕЛЬ МАНДАТНОЙ ПОЛИТИКИ ЦЕЛОСТНОСТИ ИНФОРМАЦИИ БИБА

Классическая модель Белла – Лападулы в первую очередь ориентирована на обеспечение защиты от угрозы конфиденциальности информации. В то же время ее математическая основа используется в модели Биба, реализующей мандатную политику целостности.

В модели Биба рассматриваются доступы субъектов к объектам и субъектам:

- *modify* – доступ субъекта на модификацию объекта (аналог доступа *write* в модели Белла – Лападулы);
- *invoke* – доступ на обращение (запись) субъекта к субъекту;
- *observe* – доступ на чтение субъекта к объекту (аналог доступа *read* в модели Белла – Лападулы);
- *execute* – доступ на выполнение.

Основными элементами модели Биба являются:

S – множество субъектов;

O – множество объектов;

$(LI, \leq)$ – решетка уровней целостности, например: $LI = \{I$ (*important*), VI (*veryimportant*), C (*crucial*)$\}$, где $I < VI < C$;

$RI = \{$*modify*, *invoke*, *observe*, *execute*$\}$ – множество видов доступа и видов прав доступа;

$B = \{b \subseteq S \times O \times RI\}$ – множество возможных множеств текущих доступов в системе;

$(i_s, i_o, i_c) \in I = LI^s \times LI^o \times LI^s$ – тройка функций (i_s, i_o, i_c), задающих: i_s: $S \to LI$ – уровень целостности субъекта; i_o: $O \to LI$ – уровень целостности объекта; i_c: $S \to LI$ – текущий уровень целостности субъекта, при этом для каждого $s \in S$ выполняется условие $i_c(s) \leq i_s(s)$;

$V = B \times I$ – множество состояний системы.

Элементы модели, необходимые для реализации дискреционной политики безопасности в модели Биба, не отличаются от аналогичных элементов в модели Белла – Лападулы и рассматриваться не будут. Так же как в классической модели Белла – Лападулы, в модели Биба не анализируются вопросы администрирования уровней целостности субъектов и объектов.

В отличие от требований безопасности классической модели Белла – Лападулы требования безопасности в модели Биба являются динамическими, для их описания используются элементы текущего и последующего состояний системы.

Определение 5.28. Доступ $(s, o, r) \in S \times O \times RI$ соответствует требованиям политики *low-water-mark* для субъектов относительно $i = (i_s, i_o, i_c) \in I$, если выполняется одно из условий:

- $r =$ *execute*;
- $r =$ *modify* и $i_c(s) \geq i_o(o)$;
- $r =$ *invoke*, $o \in S$ и $i_c(s) \geq i_c(o)$;
- $r =$ *observe* и после получения доступа в последующем состоянии системы $i_c'(s) = i_c(s) \otimes i_o(o)$, где $\otimes$ наибольшая нижняя граница элементов

решетки (LI, $\leq$); i_c'(s) – значение функции текущего уровня целостности субъекта в последующем состоянии системы.

Определение 5.29. Доступ $(s, o, r) \in S \times O \times RI$ соответствует требованиям политики *low-water-mark* для объектов относительно $i = (i_s, i_o, i_c) \in I$, если выполняется одно из условий:

- $r \in \{execute, invoke, observe\}$;
- $r = modify$ и после получения доступа в последующем состоянии системы $i_o'(o) = i_c(s) \otimes i_o(o)$, где $i_o'(o)$ –значение функции уровня целостности объекта в последующем состоянии системы.

Динамическое изменение функций уровня целостности субъекта или объекта может потребовать уточнения требований политики *low-water-mark*. Например, если в системе допускается подача субъектом одновременно нескольких запросов на доступ, то результат их выполнения может зависеть от последовательности выполнения запросов. Если субъект запросил одновременно доступы *observe* и *modify*, то предоставление в первую очередь доступа *observe* может привести к понижению уровня целостности субъекта, и следовательно, может сделать невозможным получение им доступа *modify*. Кроме того, понижение уровня целостности объекта при доступе *modify* к нему, может привести к модификации информации с высоким уровнем целостности субъектом с низким уровнем целостности. Аналогичный пример для случая понижения уровня конфиденциальности объекта анализируется при описании политики *low-water-mark* в модели Белла – Лападулы.

Возможным путем уточнения политики *low-water-mark*, является выполнение требования неизменности значений функций (i_s, i_o, i_c) для всех субъектов и объектов во всех состояниях системы. В этом случае всегда справедливо равенство $i_s = i_c$.

Определение 5.30. Доступ $(s, o, r) \in S \times O \times RI$ соответствует требованиям политики *low-water-mark* при неизменных значениях функций (i_s, i_o), если выполняется одно из условий:

- $r \in \{execute, observe\}$;
- $r = modify$ и $i_s(s) \geq i_o(o)$;
- $r = invoke$, $o \in S$ и $i_s(s) \geq i_s(o)$.

Выполнение требований политики *low-water-mark* при неизменных значениях функций (i_s, i_o) позволяет предотвратить модификацию субъектом информации в объекте с более высоким или несравнимым с субъектом уровнем целостности. В то же время отсутствие ограничений на доступ *observe* может позволить субъекту реализовать информационный поток от объекта с меньшим уровнем целостности в объект с большим уровнем целостности. Для предотвращения угрозы возникновения информационных потоков данного вида возможно использование строгой политики *low-water-mark* при неизменных значениях функций (i_s, i_o).

Определение 5.31. Доступ $(s, o, r) \in S \times O \times RI$ соответствует требованиям строгой политики *low-watermark* при неизменных значениях функций (i_s, i_o), если выполняется одно из условий:

- $r = execute$;
- $r = modify$ и $i_s(s) \geq i_o(o)$;
- $r = invoke$, $o \in S$ и $i_s(s) \geq i_s(o)$;
- $r = observe$ и $i_s(s) \leq i_o(o)$.

По аналогии с классической моделью Белла – Лападулы в модели Биба можно определить требования политики *low-water-mark* для состояния и системы в целом. После чего можно сформулировать и доказать для требований политики *low-water-mark* теоремы, аналогичные теоремам 5.1–5.4 классической модели Белла – Лападулы.

Контрольные вопросы и задания

1. Каковы основные недостатки классической модели Белла – Лападулы?

2. Опишите состояния системы Белла – Лападулы со следующими параметрами $S = \{s_1, s_2\}$, $O = \{o_1, o_2\}$, $R = \{read, write\}$, $(L, \leq) = \{Low, High\}$, M – не используется, $f_s(s_1) = f_o(o_1) = Low$, $f_s(s_2) = f_o(o_2) = High$.

3. Известно, что при реализации мандатного управления доступом может допускаться использование правил дискреционного управления доступом (например, с использованием *ds*-свойства безопасности модели Белла – Лападулы). Учитывая это обстоятельство, предложите подход по созданию запрещенного информационного потока от объекта с высоким уровнем конфиденциальности в объект с низким уровнем конфиденциальности, использующий кооперацию субъектов КС.

4. В чем схожесть подходов к определению информационного потока в расширенной модели *Take-Grant* и модели Белла – Лападулы?

5. В чем основное отличие определений свойств безопасности классической модели Белла – Лападулы и модели Биба?

6. МОДЕЛЬ СИСТЕМ ВОЕННЫХ СООБЩЕНИЙ

Построенная на основе модели Белла – Лападулы модель систем военных сообщений СВС (*MMS, Military Message Systems*) была предложена в 1984 г. В первую очередь она ориентирована на системы приема, передачи и обработки почтовых сообщений, реализующих мандатную политику безопасности. В то же время предложенные в модели СВС подходы к определению основных свойств безопасности могут быть использованы для построения и анализа произвольных КС с мандатным управлением доступом.

Определения основных понятий, используемых при описании свойств модели СВС, либо уже давались ранее в модели Белла – Лападулы, либо их смысл интуитивно ясен, например: объект, контейнер, сущность, пользователь, идентификатор пользователя, уровни конфиденциальности, доступ, множество доступов. В то же время дополнительно в рамках модели СВС даются следующие определения.

Определение 6.1. Роль пользователя – совокупность прав пользователя, определяемая характером выполняемых им действий в системе. Пользователь может менять свои роли во время работы в системе.

Определение 6.2. Способ доступа к содержимому контейнера (*CCR*) – атрибут контейнеров, определяющий порядок обращения к его содержимому (с учетом уровня конфиденциальности контейнера или только уровня конфиденциальности сущности контейнера, к которой осуществляется обращение).

Определение 6.3. Идентификатор сущности – уникальный номер или имя сущности.

Определение 6.4. Непосредственная ссылка – ссылка на сущность, совпадающая с идентификатором сущности.

Определение 6.5. Косвенная ссылка – ссылка на сущность, являющуюся частью контейнера, через последовательность двух и более ссылок на сущности, в которой только первая ссылка есть идентификатор (непосредственная ссылка).

Определение 6.6. Сообщение – особый тип сущностей, имеющийся в СВС. В большинстве случаев сообщение есть контейнер, хотя в некоторых системах, только получающих сообщения, оно может быть и объектом. Каждое сообщение как контейнер содержит несколько сущностей, описывающих его параметры, например: кому (*to*), от кого (*from*), информация (*info*), дата-время-группа (*date-time-group*), текст (*text*), безопасность (*security*).

Определение 6.7. Операция – функция, которая может быть выполнена над сущностями. В модели СВС основными операциями над сообщениями являются:

- над входящими сообщениями;
- над исходящими сообщениями;
- хранения и получения сообщений.

В модели СВС описываются четыре постулата безопасности, выполнение которых необходимо для ее корректной работы.

1. Системный офицер безопасности корректно разрешает доступ пользователей к сущностям и назначает уровни конфиденциальности устройств и множества ролей.

2. Пользователь назначает или переназначает корректные уровни конфиденциальности сущностей, когда создает или редактирует в них информацию.

3. Пользователь корректно направляет сообщения по адресатам и определяет множества доступа к созданным им самим сущностям.

4. Пользователь правильно задает атрибут *CCR* контейнеров.

Приведем описание модели СВС. В первой части представлены основные свойства модели, описанные неформально. Во второй части – с использованием формальной модели.

Всего в модели СВС описываются десять неформальных свойств.

1. *Авторизация*. Пользователь может осуществить операцию над сущностями только, если идентификатор пользователя или его роль присутствуют во множестве доступов сущности вместе с данной операцией и правильными индексами операндов сущностей.

2. *Иерархия уровней конфиденциальности*. Уровень конфиденциальности любого контейнера, по крайней мере, не ниже максимума уровней конфиденциальности сущностей, в нем содержащихся.

3. *Безопасный перенос информации*. Информация, извлекаемая из объекта, наследует уровень конфиденциальности объекта. Информация, внедряемая в объект, не должна иметь уровень конфиденциальности выше, чем сам объект.

4. *Безопасный просмотр*. Пользователь может просматривать (на некотором устройстве вывода) сущность с уровнем конфиденциальности не выше уровня доступа пользователя и уровня конфиденциальности устройства вывода.

5. *Доступ к сущностям с атрибутом CCR*. Пользователь может иметь доступ косвенно к сущностям контейнера с атрибутом *CCR* только в том случае, если пользователь имеет уровень доступа не ниже, чем уровень конфиденциальности контейнера.

6. *Доступ по косвенной ссылке*. Пользователь может использовать идентификатор контейнера, чтобы получить через него косвенную ссылку на сущность только в случае, если он авторизован для просмотра этой сущности с использованием этой ссылки.

7. *Пометка вывода*. Любая сущность, просматриваемая пользователем, должна быть помечена ее уровнем конфиденциальности.

8. *Определение доступов, множества ролей, уровней устройств*. Только пользователь с ролью «системный офицер безопасности» может определять права доступа и множество ролей пользователя, а также уровень конфиденциальности устройств вывода информации. Выбор текущей роли из множества авторизованных ролей пользователя может быть осуществлен только самим пользователем или пользователем с ролью «системный офицер безопасности».

9. *Безопасное понижение уровня конфиденциальности*. Никакой уровень конфиденциальности не может быть понижен, за исключением тех случаев, когда понижение выполняет пользователь с соответствующей ролью.

10. *Безопасное отправление сообщений*. Никакое черновое сообщение не может быть отправлено, за исключением случая, когда это делает пользователь с ролью «отправитель».

В модели СВС система представляется конечным автоматом. Основными элементами модели являются:

OP – множество операций;

(L, $\leq$) – решетка уровней конфиденциальности;

UI – множество идентификаторов пользователей;

RL – множество ролей пользователей;

US – множество пользователей. Для любого пользователя $u \in US$ заданы: $CU(u) \in L$ – уровень доступа пользователя u; $R(u) \subseteq RL$ – множество авторизированных ролей пользователя u; $RO(u) \subseteq RL$ – множество текущих ролей пользователя u;

RF – множество ссылок, состоящее из двух подмножеств: *DR* – множества непосредственных ссылок и *IR* – множества косвенных ссылок. Хотя точная природа этих ссылок не важна, положим, что непосредственные ссылки есть целые числа. Каждая косвенная ссылка есть конечная последовательность целых чисел $<n_1, \ldots, n_m>$, где $<n_1>$ – непосредственная ссылка;

VS – множество значений сущностей (например, множество битовых или символьных строк);

TY – множество типов сущностей, включающее в себя типы: *DM* – тип «черновое сообщение», *RM* – тип «отправленное сообщение»;

ES – множество сущностей. При этом для каждой сущности $e \in ES$ задаются:

- $CE(e) \in L$ – уровень конфиденциальности сущности;
- $AS(e) \subseteq (UI \cup RL) \times OP \times N$ – множество троек, задающих множество разрешенных доступов к сущности, где $(u, op, k) \in AS(e)$ тогда и только тогда, когда пользователь с идентификатором u или ролью u авторизован для выполнения операции op над сущностью e, ссылка на которую является k-м параметром операции op;
- $T(e) \in TY$ – тип сущности; при этом, если $T(e_1) = T(e_2)$, тогда по определению обе сущности e_1 и e_2 – или контейнеры, или объекты;
- $V(e) \in VS$ – значение сущности;
- $H(e) = <e_1, \ldots, e_n>$ – структура сущности, где e_i – i-я сущность, непосредственно содержащаяся в контейнере e;
- $CCR(e) \in \{true, false\}$ – атрибут сущностей-контейнеров. $CCR(e) = true$ тогда и только тогда, когда контейнер e помечен CCR, иначе $CCR(e) = false$;
- $RE(e) \in UI$ – идентификатор отправителя для сущности-сообщения.

$O \subset ES$ – множество сущностей-контейнеров, описывающих устройства вывода информации. Для каждого $o \in O$ заданы значения двух функций:

- $D(o)$ – множество упорядоченных пар $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, где каждый x_i есть некоторый пользователь или сущность и соответствующее y_i есть «проявление» x_i или некоторая ссылка, или идентификатор, или результат применения заданных в рамках модели функций к x_i. Таким образом, каждое $D(o)$ описывает, что могут видеть пользователи на устройстве вывода информации. При этом $((x, V(x)) \in D(o)) \Rightarrow (x \in H(o))$, т. е. если значение сущности можно получить на устройстве вывода, то эта сущность входит в состав контейнера устройства вывода;
- $CD(o)$ – максимальный уровень конфиденциальности информации, разрешенный для вывода на o;

$U: UI \rightarrow US$ – функция идентификаторов пользователей;

$E: RF \rightarrow ES$ – функция ссылок на сущности.

При этом для любого $n > 1$ верно равенство $E(<i_1, \ldots, i_n>) = e$ тогда и только тогда, когда $E(<i_1, \ldots, i_{n-1}>) = e^*$, где e^* – контейнер такой, что e – это i_n-й элемент $H(e^*)$. Для любой ссылки $r \in RF$, если $E(r) = e$, то говорим, что r есть ссылка на сущность e. Кроме того, косвенная ссылка $<i_1, \ldots, i_n>$ на сущность e задает последовательность сущностей $e_1, \ldots, e_{n-1}$ таких, что каждая сущность e_j для $1 < j < n$ определяется через непосредственную ссылку $<i_1>$, и e_j есть i_j-я сущность контейнера e_{j-1}. Будем говорить, что такая косвенная ссылка $<i_1, \ldots, i_n>$ основывается на каждой сущности e_j,

где $0 < j < n$; LO: $UI \to RF$ – функция входов пользователей (*logon*) в систему.

Для произвольной функции F: $X \to Y$ обозначим:

- $dom(F)$ – область определения функции F;
- $rng(F)$ – область значений функции F;
- $F^{-1}(Z) = \{x \in X: F(x) \in Z\}$, где $Z \subseteq Y$.

Определение 6.8. Состояние системы есть упорядоченная тройка $s = (U, E, LO)$, где U – функция идентификаторов; E – функция ссылок; LO – функция входов. При этом:

- $dom(LO) \subseteq dom(U)$;
- $rng(LO) \subseteq E^{-1}(O)$;
- для каждой сущности-устройства вывода $o \in rng(E) \cap O$, если $(x, y) \in D(o)$, то выполняется условие $x \in rng(E) \cup rng(U)$ – только информация о пользователях или сущностях, существующих в данном состоянии, может быть выведена на устройствах вывода;
- для каждой ссылки $r \in dom(E)$ выполняется условие $((x, r) \in D(o)) \Rightarrow (E(r) = x)$ – если выводится ссылка, то определена сущность, на которую она указывает;
- для $u_1, u_2 \in dom(LO)$ выполняется условие $(E(LO(u_1)) = E(LO(u_2))) \Rightarrow (u_1 = u_2)$ – в каждом состоянии с одного устройства вывода информации может осуществлять вход в систему только один пользователь.

Если дано состояние системы $s = (U, E, LO)$, то используем следующие обозначения:

- $u_s = U(u)$ – пользователь с идентификатором u в состоянии s;
- $r_s = E(r)$ – сущность по ссылке r в состоянии s;
- $u_s' = E(LO(u))$ – сущность-устройство вывода информации, с которого осуществил вход в систему пользователь с идентификатором u в состоянии s.

Определение 6.9. Состояние $s = (U, E, LO)$ безопасно, если для любых двух сущностей $x, y \in rng(E)$, устройства вывода информации $o \in O \cap rng(E)$, идентификатора пользователя $w \in dom(LO)$ и пользователя $u \in rng(U)$ выполняются условия:

- $(x \in H(y)) \Rightarrow (CE(x) \le CE(y))$ – уровень конфиденциальности сущности в составе контейнера не выше уровня конфиденциальности контейнера;
- $(x \in H(w_s')) \Rightarrow (CU(w_s) \ge CE(x))$ – уровень доступа пользователя с данным идентификатором не ниже, чем уровень конфиденциальности сущностей, выводимых на устройстве вывода информации, с которого он осуществил вход в систему;

- $((x,\ V(x)) \in D(o)) \Rightarrow ((x,\ CE(x)) \in D(o))$ – если выводится значение сущности, то должен быть выведен и ее уровень конфиденциальности;
- $RO(u) \subseteq R(u)$ – текущее множество ролей пользователя должно быть частью множества ролей, на которые он авторизован;
- $CD(o) \geq CE(o)$ – максимальный уровень конфиденциальности сущностей, разрешенный для вывода на устройстве вывода информации, должен быть не ниже текущего уровня конфиденциальности устройства вывода.

Определение 6.10. Система $\Sigma(T, s_0)$ – пара элементов:

- s_0 – начальное состояние системы;
- T: $UI \times I \times S \to S$ – функция переходов системы, где S – множество возможных состояний системы; I – множество запросов к системе. При этом каждый запрос $i \in I$ имеет вид $<op, x_1, \ldots, x_n>$, где $op \in OP$ и $x_j \in RF \cup UI \cup VS$ для $1 \leq j \leq n$, т. е. ссылка на сущность, идентификатор пользователя или значение сущности могут быть параметром запроса.

Определение 6.11. Функция π: $N_0 \to UI \times I \times S$ – история системы, при этом для $n \in N_0$ выполняется условие: если $\pi(n) = (u, i, s)$ и $\pi(n + 1) = (u^*, i^*, s^*)$, то $T(u, i, s) = s^*$.

Определение 6.12. Состояния $s = (U, E, LO)$ и $s^* = (U^*, E^*, LO^*)$ эквивалентны за исключением некоторого множества ссылок $\rho \subset dom(E)$ (обозначим через $s \sim^{\rho} s^*$) тогда и только тогда, когда выполняются условия:

- $U = U^*$;
- $LO = LO^*$;
- $dom(E) = dom(E^*)$;
- для любой функции F, за исключением V (функции значений сущностей), справедливо равенство $F = F^*$;
- для любой ссылки $r \in dom(E) \setminus \rho$ справедливо равенство $V(r_s) = V(r_{s^*})$.

Определение 6.13. Запрос i пользователя с идентификатором u в состоянии s, где $(u, i, s) \in UI \times I \times S$, потенциально модифицирует сущность по ссылке $r \in dom(E)$ тогда и только тогда, когда существуют состояния s_1, s_1^* и $\rho \subset dom(E)$ такие, что $s \sim^{\rho} s_1$, $T(u, i, s_1) = s_1^*$ и для некоторой функции F справедливо неравенство $F(r_{s1}) \neq F(r_{s1^*})$. При этом сущность по ссылке $y \in dom(E)$ называется источником потенциальной модификации тогда и только тогда, когда выполняется одно из условий:

- $y = r$;
- существуют состояния s_2, s_2^* такие, что $s_1 \sim^{\{y\}} s_2$, $T(u, i, s_2) = s_2^*$ и $F(r_{s1^*}) \neq F(r_{s2^*})$.

Потенциальная модификация сущности по ссылке rс источником сущностью по ссылке указывает на реализацию в системе информацион-

ного потока от сущности y_s к сущности r_s при переходе системы из состояния в состояние по заданному запросу (рис. 6.1).

В определении потенциальной модификации также учитывается тот факт, что информационный поток, возникающий в системе, может не приводить к изменению модифицируемой сущности. Например, в случае, когда она по значению совпадает с сущностью-источником.

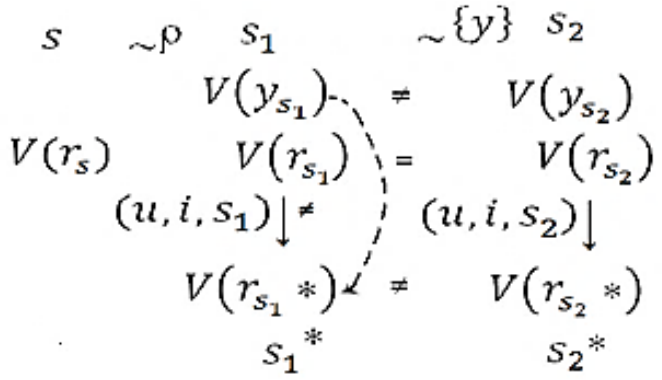

Рис. 6.1. Потенциальная модификация с источником

Дадим восемь определений смыслов безопасности функции переходов.

Определение 6.14. Функция переходов T безопасна в смысле доступов к сущностям тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняется одно из условий: или $s = s^*$ (запрос отвергается), или ($op \in i \cap OP$ и $r_k \in i \cap RF$) $\Rightarrow$ (или $(u, op, k) \in AS((r_k)_s)$, или существует роль $l \in RO(u_s)$: $(l, op, k) \in AS((r_k)_s)$).

Определение 6.15. Функция переходов T безопасна в смысле модификации сущностей тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняется условие (сущность по ссылке x в состоянии s потенциально модифицируется по запросу i пользователя с идентификатором u с источником сущностью по ссылке y) $\Rightarrow$ $(CE(x_s) \geq CE(y_s))$.

Определение 6.16. Функция переходов T безопасна в смысле доступов к контейнерам с атрибутом CCR тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняется условие (сущность по ссылке z в состоянии s потенциально модифицируется по запросу i пользователя с идентификатором u с источником сущностью по косвенной ссылке $r \in i \cap IR$, основанной на сущности по ссылке y с атрибутом $CCR(y_s) = true$) $\Rightarrow (CU(u_s) \geq CE(y_s))$.

Определение 6.17. Функция переходов T безопасна в смысле доступов по косвенной ссылке тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняется условие (для непосредственной ссылки $x \in DR$ выполняется условие $(x_{s^*}, x) \in D(u_{s^*}')$ и существует косвенная ссылка $r \in i \cap RF$: $r_s = x_s$ и сущность по ссылке r основана на контейнере по ссылке z, с атрибутом $CCR(z_s) = true$) $\Rightarrow$ ($CU(u_s) \geq CE(z_s)$) – значение непосредственной ссылки на сущность может быть получено по косвенной ссылке с учетом правил доступа к контейнерам с атрибутом CCR.

Определение 6.18. Функция переходов T безопасна в смысле администрирования тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняются условия:

- для ссылки на устройство вывода $o \in E^{-1}(O)$ выполняется $CD(o_s) \neq CD(o_{s^*})$, или для идентификатора пользователя $x \in dom(U)$ выполняется одно из условий: или $CU(x_s) \neq CU(x_{s^*})$, или $R(x_s) \neq R(x_{s^*}) \Rightarrow$ ($security\_officer \in RO(u_s)$) – максимальный уровень конфиденциальности информации, выводимой на устройстве вывода, уровень доступа пользователя или множество его авторизованных ролей может изменить только пользователь с ролью «системный офицер безопасности» (*security_officer*);
- для идентификатора пользователя $x \in dom(U)$ выполняется условие $RO(x_s) \neq RO(x_{s^*}) \Rightarrow$ (или $x_s = u_s$, или $security\_officer \in RO(u_s)$) – изменить множество текущих ролей пользователя может только сам пользователь или пользователь с ролью «системный офицер безопасности».

Определение 6.19. Функция переходов T безопасна в смысле понижения уровня конфиденциальности сущностей тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняется следующее условие: (для $x \in dom(E) \setminus E^{-1}(\{u_s'\})$ справедливо неравенство $CE(x_s) > CE(x_{s^*})$) $\Rightarrow$ ($downgrader \in RO(u_s)$) – за исключением устройства вывода, с которого вошел в систему пользователь, уровень конфиденциальности сущности может понижать только пользователь с соответствующей специальной ролью (*downgrader*).

Определение 6.20. Функция переходов T безопасна в смысле отправки сообщений тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, выполняются условия:

- ($T(x_s) = RM$) $\Rightarrow$ ($T(x_{s^*}) = RM$ и $RE(x_s) = RE(x_{s^*})$) – у сообщения тип и идентификатор отправителя не могут быть изменены после отправки (RM – тип «отправленное сообщение»);
- ($T(x_s) \neq RM$ и $T(x_{s^*}) = RM$) $\Rightarrow$ ($T(x_s) = DM$, $RE(x_{s^*}) = u$, существует ссылка $r_s = x_s$ и i – операция вида <*release*, r>, где $releaser \in RO(u_s)$) – может быть с использованием некоторой ссылки отправлено только черновое

сообщение (*DM* – тип «черновое сообщение») и только пользователем с ролью «отправитель» (*releaser*); при этом идентификатор этого пользователя устанавливается в соответствующее поле сообщения.

Определение 6.21. Функция переходов T безопасна согласно базовой теореме безопасности (БТБ) тогда и только тогда, когда для u, i, s, s^* таких, что $T(u, i, s) = s^*$, и для $x, y \in RF$, $w \in UI$ выполняются условия:

- $(x_s \notin H(y_s)$ и $x_{s^*} \in H(y_{s^*})) \Rightarrow (CE(x_{s^*}) \leq CE(y_{s^*}))$;
- $(x_s \in H(y_s)$ и $CE(x_{s^*}) > CE(y_{s^*})) \Rightarrow (x_{s^*} \notin H(y_{s^*}))$;
- $(x_s \notin H(w_s')$ и $x_{s^*} \in H(w_{s^*}')) \Rightarrow (CE(x_{s^*}) \leq CU(w_{s^*}))$;
- $(x_s \in H(w_s')$ и $CE(x_{s^*}) > CU(w_{s^*})) \Rightarrow (x_{s^*} \notin H(w_{s^*}'))$;
- $((x_s, V(x_s)) \notin D(w_s')$ и $(x_{s^*}, V(x_{s^*})) \in D(w_{s^*}')) \Rightarrow ((x_{s^*}, CE(x_{s^*})) \in D(w_{s^*}'))$;
- $((x_s, V(x_s)) \in D(w_s')$ и $(x_{s^*}, CE(x_{s^*})) \notin D(w_{s^*}')) \Rightarrow ((x_{s^*}, V(x_{s^*})) \notin D(w_{s^*}'))$;
- $(R(w_s) \neq R(w_{s^*})$ или $RO(w_s) \neq RO(w_{s^*})) \Rightarrow (RO(w_{s^*}) \subseteq R(w_{s^*}))$;
- $(CE(w_s') \neq CE(w_{s^*}')$ или $CD(w_s') \neq CD(w_{s^*}')) \Rightarrow (CD(w_{s^*}') \geq CE(w_{s^*}'))$.

Определение 6.22. Функция переходов T безопасна тогда и только тогда, когда она безопасна согласно определениям 6.14–6.21.

Определение 6.23. История π системы $\Sigma(T, s_0)$ безопасна, если все ее состояния и функция переходов T безопасны.

Определение 6.24. Система $\Sigma(T, s_0)$ безопасна, если все ее истории безопасны.

Лемма 6.1 (теорема БТБ-СВС). Каждое состояние системы $\Sigma(T, s_0)$ безопасно, если состояние s_0 безопасно и функция переходов T безопасна согласно БТБ.

Доказательство. Доказательство осуществляется аналогично доказательству теоремы 5.1 модели Белла – Лападулы.

Теорема 6.1. Система $\Sigma(T, s_0)$ безопасна, если начальное состояние s_0 и функция переходов T безопасны.

Доказательство. Доказательство следует из определений и леммы 6.1.

В заключение рассмотрения модели СВС целесообразно отметить, что в ней не содержится описания механизмов администрирования. В частности, при описании функции переходов пропущены такие возможные действия в системе, как создание новых сущностей, присваивание им уровня конфиденциальности $CE(e)$ и инициализация множества доступов $AS(e)$. С учетом постулатов безопасности предполагается, что данные действия осуществляются корректно. Изменения значения множества доступов $AS(e)$ для существующей сущности могут быть осуществлены с использованием доступов, определенных в самом значении $AS(e)$ для данной сущности. Поэтому для задания правил изменения значений функции AS не требуется изменения определений безопасности функции переходов T.

Контрольные вопросы

1. Каким образом десять неформальных свойств модели СВС реализуются в ее формальном описании?
2. В каком случае система $\Sigma(T, s_0)$ безопасна?
3. Где в определениях безопасности модели СВС реализовано *ss*-свойство безопасности классической модели Белла – Лападулы?
4. Где в определениях безопасности модели СВС реализовано *-свойство безопасности классической модели Белла – Лападулы?
5. Где в определениях безопасности модели СВС реализовано *ds*-свойство безопасности классической модели Белла – Лападулы?

7. МОДЕЛИ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ПОТОКОВ

7.1. АВТОМАТНАЯ МОДЕЛЬ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ПОТОКОВ

В автоматной модели безопасности информационных потоков система защиты представляется детерминированным автоматом, на вход которого поступает последовательность команд пользователей. Основными элементами автоматной модели безопасности информационных потоков являются:

V – множество состояний системы;

U – множество пользователей;

M – множество матриц доступов;

CC – множество команд пользователей, изменяющих матрицу доступов;

VC – множество команд пользователей, изменяющих состояние;

$C = CC \cup VC$ – множество всех команд пользователей;

Out – множество выходных значений;

$cvdo$: $U \times C \times V \times M \to V \times M$ – функция переходов системы.

В зависимости от вида команды при переходе системы в последующее состояние или изменяется состояние системы, или вносятся изменения в матрицу доступов. При этом для каждого пользователя в матрице доступов определяются команды, которые ему разрешено выполнять. Таким образом, автоматная модель безопасности информационных потоков реализует дискреционное управление доступом.

Для каждого пользователя системы задается функция выходов, определяющая, что каждый из них «видит» на устройстве вывода в соответствующем состоянии системы:

out: $U \times V \to Out$ – функция выходов системы.

Таким образом, определен автомат, в котором:

- (v_0, m_0) – начальное состояние;
- $V \times M$ – множество внутренних состояний автомата;
- $U \times C$ – входной алфавит;
- Out – выходной алфавит;
- $cvdo$ – функция переходов автомата;
- out – функция выходов автомата.

Функция *cvdo* может быть по индукции определена для конечных последовательностей команд пользователей:

cvdo: $(U \times C)^* \times V \times M \to V \times M$.

Пусть $w \in (U \times C)^*$ – последовательность команд пользователей; $s \in U$ – пользователь; $G \subset U$ – подмножество множества пользователей; $A \subset C$ – подмножество множества команд. Используем обозначения:

$[[w]]$ – последовательность состояний, полученная из исходного состояния в результате выполнения последовательности команд пользователей w;

$[[w]]_s$ – последовательность выходов, наблюдаемых пользователем s при реализации последовательности состояний $[[w]]$;

$P_G(w)$ – подпоследовательность последовательности w, полученная из нее удалением всех пар (s, c), где $s \in G$;

$P_A(w)$ – подпоследовательность последовательности w, полученная из нее удалением всех пар (s, c), где $c \in A$;

$P_{GA}(w)$ – подпоследовательность последовательности w, полученная из нее удалением всех пар (s, c), где $s \in G$ и $c \in A$.

Определение 7.1. Пусть $G, G' \subset U$. G информационно не влияет на G' (обозначим через $G:| G'$), если для каждой $w \in (U \times C)^*$ и для $s \in G'$ справедливо равенство $[[w]]_s = [[P_G(w)]]_s$.

Определение 7.2. Пусть $A \subset C$, $G' \subset U$. A информационно не влияет на G' (обозначим через $A:| G'$), если для каждой $w \in (U \times C)^*$ и для $s \in G'$ справедливо равенство $[[w]]_s = [[P_A(w)]]_s$.

Определение 7.3. Пусть $A \subset C$, G, $G' \subset U$. G и A информационно не влияют на G' (обозначим через $G, A:| G'$), если для каждой $w \in (U \times C)^*$ и для $s \in G'$ справедливо равенство $[[w]]_s = [[P_{GA}(w)]]_s$.

Определение 7.4. Политика безопасности в автоматной модели безопасности информационных потоков – это набор требований информационного невлияния.

Пример 7.1. Пусть f – файл, $s \in U$ – пользователь, $A = \{create(f), write(f), modify(f), delete(f)\}$ – набор команд. Тогда $\{s\}, A:| \{s\}$ означает, что пользователь s может осуществлять только чтение из файла f.

Пример 7.2. Мандатную политику безопасности в автоматной модели безопасности информационных потоков, которая состоит в том, что пользователи с большим уровнем доступа не должны информационно влиять на пользователей с меньшим уровнем доступа, можно выразить, используя следующие обозначения:

$(L, \leq)$ – шкала уровней доступа к информации;

level: $U \to L$ – функция уровней доступа пользователей;

$U[-\infty, x] = \{s \in U: level(s) \leq x\}$ – множество пользователей с уровнем доступа, не большим $x \in L$;

$U[x, +\infty] = \{s \in U: level(s) \geq x\}$ – множество пользователей с уровнем доступа, не меньшим $x \in L$.

Тогда мандатная политика безопасности есть совокупность требований информационного невлияния:

для $x, x' \in L$, $x < x'$ выполняется $U[x', +\infty] :| U[-\infty, x]$.

Определение 7.5. Пусть $G \subset U$. Пользователи из G невидимы для остальных пользователей, если $G :| (U \setminus G)$. Пользователи из G изолированы от остальных пользователей, если $G :| (U \setminus G)$ и $(U \setminus G) :| G$.

С помощью требований информационного невлияния могут быть определены различные виды политик безопасности, определяющие не только правила управления доступом пользователей системы, но и конфигурацию, порядок взаимодействия между собой распределенных компонент КС.

7.2. ВЕРОЯТНОСТНАЯ МОДЕЛЬ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ПОТОКОВ

В рассматриваемой вероятностной модели безопасности информационных потоков анализируются КС, в которых реализована мандатная политика безопасности. В модели предполагается, что все объекты (в том числе и субъекты) КС объединены по трем группам: объекты Σ, реализующие систему защиты; объекты H, обрабатывающие информацию высокого уровня конфиденциальности; объекты L, обрабатывающие информацию низкого уровня конфиденциальности. Все информационные потоки между H и L проходят через систему защиты Σ (рис. 7.1).

Пусть на множестве значений объектов системы задано вероятностное распределение, т. е. H и L являются случайными величинами.

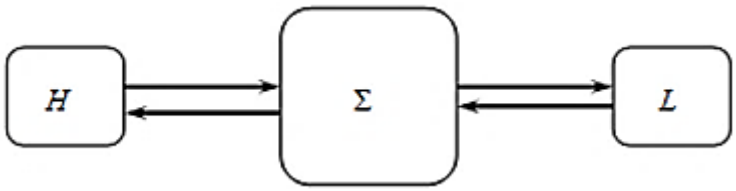

Рис. 7.1. Схема компьютерной системы

Согласно требованиям мандатной политики безопасности любые информационные потоки от H к L запрещены. Однако в большинстве реальных систем реализовать данное жесткое требование не представляется возможным. В модели рассматриваются ряд подходов к определению возможных информационных потоков между H и L, основанных на понятиях информационной невыводимости и информационного невлияния.

Определение 7.6. КС соответствует требованиям информационной невыводимости, если для $p(H) > 0$, $p(L) > 0$ справедливо неравенство $p(H \mid L) > 0$.

Так как при $p(H) > 0$, $p(L) > 0$ справедливо равенство

$$p(L \mid H) = p(H, L) / p(H) = p(H \mid L)\, p(L) / p(H),$$

то в условиях определения из истинности неравенства $p(H \mid L) > 0$ следует истинность $p(L \mid H) > 0$, что предполагает отсутствие информационных потоков от низкоуровневых объектов к высокоуровневым. Таким образом, требования информационной невыводимости являются более строгими, чем требования безопасности классической модели Белла – Лападулы, и фактически предполагают изоляцию друг от друга высокоуровневых и низкоуровневых объектов системы, что является чрезмерно жестким требованием.

В традиционной модели информационного невлияния требуется, чтобы низкоуровневая информация была независима от высокоуровневой, т. е. выполнялось условие следующего определения.

Определение 7.7. КС соответствует требованиям информационного невлияния (без учета времени), если для $p(H) > 0$, $p(L) > 0$ выполняется:

$$p(L \mid H) = p(L).$$

При $p(H) > 0$, $p(L) > 0$ условие определения равносильно условию

$$p(H \mid L) = p(H).$$

Условие определения 7.7 также является жестким. Однако если ввести параметр времени, то возможно данное условие сделать в большей степени применимым для использования в реальных системах. Конечный вид условия информационного невлияния получим в результате последовательности рассуждений.

Если L_t описывает состояния всех низкоуровневых объектов, а H_t всех высокоуровневых объектов в момент времени t = 0, 1, 2, …, то нет необходимости требовать выполнения условия

$$p(H_t \mid L_{t-1}) = p(H_t),$$

т. е. текущее значение низкоуровневых объектов может содержать информацию о последующих значениях высокоуровневых объектов. Например, если низкоуровневым является некий неконфиденциальный файл, обрабатываемый пользователем с низким уровнем доступа, а высокоуровневым объектом является журнал аудита. Тогда значение файла и операции, совершаемые над ним пользователем в момент t, могут отображаться в журнале аудита в момент $t + 1$, о чем «знает» низкоуровневый пользователь.

Учитывая тот факт, что состояние системы влияет на последующие состояния только через информацию, хранимую в объектах системы, казалось бы, необходимо потребовать, чтобы текущее значение низкоуровневых объектов не зависело от значения высокоуровневых объектов в предыдущие моменты работы системы, т. е. выполнялось условие

$$p(L_t \mid H_{t-1}) = p(L_t) \text{ или } p(H_{t-1} \mid L_t) = p(H_{t-1}), \text{ для } t = 1, 2, \ldots.$$

Однако следует отметить, что данный вариант определения соотношения L_t и H_{t-1} является слишком строгим, так как предполагает независимость L_t и H_{t-1}. В то же время значение высокоуровневых объектов на текущем шаге часто оказывает влияние на значение низкоуровневых объектов на последующих шагах работы системы. Например, рассмотрим работу монитора ссылок КС.

Монитор ссылок в реальных системах защиты является высокоуровневым субъектом, принимающим решения по запросам на доступ к объектам, полученным от других субъектов системы. Очевидно, что такое решение, полученное низкоуровневым субъектом в момент t работы системы, содержит информацию о значении высокоуровневого монитора ссылок в предыдущий момент времени $t - 1$.

Более целесообразным представляется подход, обеспечивающий невозможность накопления низкоуровневыми объектами новой информации о значении высокоуровневых объектов. Более формально, необходимо требование, чтобы знание значений L_{t-1} и L_t не давало бы новой информации о H_{t-1}, т. е. должно выполняться условие

$$p(L_t \mid H_{t-1}, L_{t-1}) = p(L_t \mid L_{t-1}), \text{ для } t = 1, 2, \ldots$$

или

$$p(H_{t-1} \mid L_t, L_{t-1}) = p(H_{t-1} \mid L_{t-1}), \text{ для } t = 1, 2, \ldots.$$

Таким образом, запрещается обратный информационный поток из L_t в H_{t-1}, но не запрещается поток из L_t в H_{t+1}. Кроме того, следует отметить, что при решении проблем, обозначенных в рассмотренных выше примерах, последнее правило блокирует возникновение запрещенных информационных потоков по времени.

Дадим определение информационного невлияния с учетом времени.

Определение 7.8. КС соответствует требованиям информационного невлияния (с учетом времени), если для $p(L_s) > 0$, $p(L_t) > 0$, $p(H_s) > 0$ выполняется условие

$$p(L_t \mid H_s, L_s) = p(L_t \mid L_s), \text{ где } s, t = 0, 1, 2, \ldots \text{ и } s < t.$$

Пример 7.3. Согласно описанию автоматной модели безопасности информационных потоков система защиты представляется детерминированным автоматом, на вход которого поступает последовательность команд пользователей. Для каждого пользователя системы задана функция выходов, определяющая, что каждый из них «видит» на устройстве вывода в соответствующем состоянии системы. Если всех пользователей системы разделить на две группы (низкоуровневые и высокоуровневые), то требование информационного невлияния автоматной модели является требованием независимости низкоуровневого вывода системы от ее высокоуровневого ввода.

Рассмотрим систему автоматной модели как совокупность четырех объектов: высокоуровневых и низкоуровневых портов ввода/вывода *high-in, high-out, low-in, low-out*. При этом системный вывод полностью определяется системным вводом. Вероятностные отношения исключаются.

Однако если интерпретировать детерминированность автомата, используя события с вероятностью {0, 1}, то информационное невлияние можно определить следующим образом.

Определение 7.9. Пусть система автоматной модели безопасности информационных потоков представляется совокупностью четырех событий с вероятностью {0, 1}: $high\text{-}in_t$, $high\text{-}out_t$, $low\text{-}in_t$, $low\text{-}out_t$ для $t = 0, 1, 2, \ldots$. Тогда система автоматной модели соответствует требованиям информационного невлияния, если выполняется условие

$$p(low\text{-}out_t \mid high\text{-}in_s, low\text{-}in_s) = p(low\text{-}out_t \mid low\text{-}in_s),$$

где $s, t = 0, 1, 2, \ldots$ и $s < t$.

Контрольные вопросы

1. Какой вид политики управления доступом используется в качестве основы автоматной модели безопасности информационных потоков?
2. В каких случаях в КС с мандатным управлением доступом нецелесообразно предотвращение возможности реализации всех информационных потоков от устройств ввода пользователей с высоким уровнем доступа к устройствам вывода пользователей с низким уровнем доступа?
3. В чем отличие информационной невыводимости от информационного невлияния?
4. Почему использование определения требований информационного невлияния (с учетом времени) позволяет обеспечить возможность функционирования в КС монитора ссылок?
5. В каких случаях может являться эффективным моделирование безопасности информационных потов с использованием вероятностных подходов?

8. МОДЕЛИ КОМПЬЮТЕРНЫХ СИСТЕМ С РОЛЕВЫМ УПРАВЛЕНИЕМ ДОСТУПОМ. БАЗОВАЯ МОДЕЛЬ РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ

8.1. ПОНЯТИЕ РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ

Ролевое управление доступом представляет собой развитие политики дискреционного управления доступом, при этом права доступа субъектов системы к объектам группируются с учетом специфики их применения, образуя роли.

Ролевое управление доступом является составляющей многих современных КС. Как правило, оно применяется в системах защиты СУБД, отдельные элементы ролевого управления доступом реализуются в сетевых ОС.

Задание ролей позволяет определить более четкие и понятные для пользователей КС правила управления доступом. При этом ролевое управление доступом наиболее эффективно используется в КС, для пользователей которых четко определен круг их должностных полномочий и обязанностей. Роль является совокупностью прав доступа к объектам КС. Однако ролевое управление доступом не является частным случаем дискреционного управления доступом, так как его правила задают порядок предоставления прав доступа субъектам (пользователям) КС в зависимости от сессии его работы и от имеющихся или отсутствующих у него ролей в каждый момент времени, что является характерным для систем мандатного управления доступом. В то же время правила ролевого управления доступом являются более гибкими, чем правила мандатного управления доступом, построенные на основе жестко определенной решетки (шкалы) ценности информации.

Для анализа и изучения свойств КС с ролевым управлением доступом используются формальные модели, в основе которых лежит базовая модель ролевого управления доступом.

8.2. БАЗОВАЯ МОДЕЛЬ РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ

Основными элементами базовой модели ролевого управления доступом (*RBAC – Role-Based Access Control*) являются:

- U – множество пользователей;
- R – множество ролей;

- P – множество прав доступа к объектам КС;
- S – множество сессий пользователей;
- PA: $R \to 2^P$ – функция, задающая для каждой роли множество прав доступа; при этом для каждого права доступа $p \in P$ существует роль $r \in R$, такая что $p \in PA(r)$;
- UA: $U \to 2^R$ – функция, задающая для каждого пользователя множество ролей, на которые он может быть авторизован;
- $user$: $S \to U$ – функция, задающая для каждой сессии пользователя, от имени которого она активизирована;
- $roles$: $S \to 2^R$ – функция, задающая для пользователя множество ролей, на которые он авторизован в данной сессии, при этом в каждый момент времени для каждой сессии $s \in S$ выполняется условие $roles(s) \subseteq UA(user(s))$.

Заметим, что могут существовать роли, на которые не авторизован ни один пользователь.

В базовой модели ролевого управления доступом предполагается, что множества U, R, P и функции PA, UA не изменяются с течением времени. Множество ролей, на которые авторизуется пользователь в течение одной сессии, модифицируется самим пользователем. При этом отсутствуют механизмы, позволяющие одной сессии активизировать другую сессию. Все сессии активизируются только пользователем. Для обеспечения возможности большего соответствия реальным КС, каждый пользователь которых занимает определенное положение в служебной иерархии, на множестве ролей реализуется иерархическая структура.

Определение 8.1. Иерархией ролей в базовой модели ролевого управления доступом называется заданное на множестве ролей R отношение частичного порядка «≤». При этом по определению выполняется условие для пользователя $u \in U$, если роли $r, r' \in R$, $r \in UA(u)$ и $r' \le r$, то $r' \in UA(u)$.

Таким образом, наряду с данной ролью пользователь должен быть авторизован также на все роли, меньшие ее в иерархии.

Отношение частичного порядка на множестве ролей не обязательно задает на нем решетку. Другим важным механизмом базовой модели ролевого управления доступом являются ограничения, накладываемые на множества ролей, на которые может быть авторизован пользователь или на которые он авторизуется в течение одной сессии. Данный механизм также необходим для широкого использования ролевого управления доступом, так как обеспечивает большее соответствие используемым в существующих КС технологиям обработки информации.

Дадим определения для ряда распространенных видов ограничений.

Определение 8.2. В базовой модели ролевого управления доступом заданы ограничения статического взаимного исключения ролей или прав доступа, если выполняются условия

$$R = R_1 \cup \ldots \cup R_n, \text{ где } R_i \cap R_j = \varnothing, \text{ для } 1 \le i < j \le n;$$

$$|UA(u) \cap R_i| \le 1 \text{ для } u \in U,\ i \in 1, 2, \ldots, n;$$

$$P = P_1 \cup \ldots \cup P_m, \text{ где } P_i \cap P_j = \varnothing \text{ для } 1 \le i < j \le m,$$

$$|PA(r) \cap P_i| \le 1 \text{ для } p \in U,\ i \in 1, 2, \ldots, m.$$

Множество ролей и множество прав доступа разделяются на непересекающиеся подмножества. При этом каждый пользователь может обладать не более чем одной ролью из каждого подмножества ролей, а каждая роль – не более чем одним правом доступа из каждого подмножества прав доступа.

Определение 8.3. В базовой модели ролевого управления доступом задано ограничение динамического взаимного исключения ролей, если выполняются условия

$$R = R_1 \cup \ldots \cup R_n, \text{ где } R_i \cap R_j = \varnothing, \text{ для } 1 \le i < j \le n;$$

$$|roles(s) \cap R_i| \le 1, \text{ для } s \in S,\ i \in 1, 2, \ldots, n.$$

Множество ролей разделяется на непересекающиеся подмножества. При этом в каждой сессии пользователь может обладать не более чем одной ролью из каждого подмножества ролей.

Определение 8.4. В базовой модели ролевого управления доступом заданы статические количественные ограничения на обладание ролью или правом доступа, если определены две функции:

$$\alpha: R \to N_0,$$

$$\beta: P \to N_0,$$

где N_0 – множество натуральных чисел с нулем, и выполняются условия

$$|UA^{-1}(r)| \le \alpha(r), \text{ для } r \in R;$$

$$|PA^{-1}(p)| \le \beta(p), \text{ для } p \in P.$$

Для каждой роли устанавливается максимальное число пользователей, которые могут быть на нее авторизованы, а для каждого права доступа устанавливается максимальное число ролей, которые могут им обладать.

Определение 8.5. В базовой модели ролевого управления доступом задано динамическое количественное ограничение на обладание ролью, если определена функция

$$\gamma: R \to N_0$$

и выполняется условие

$$|roles^{-1}(r)| \le \gamma(r), \text{ для } r \in R.$$

Для каждой роли устанавливается максимальное число сессий пользователей, которые могут одновременно быть на нее авторизованы.

Определение 8.6. В базовой модели ролевого управления доступом заданы статические ограничения необходимого обладания ролью или правом доступа, если определены две функции:

$$\alpha: R \to 2^R,$$

$$\beta: P \to 2^P$$

и выполняются условия

$$\text{для } u \in U, \text{ если } r, r' \in R,\ r \in UA(u) \text{ и } r' \in \alpha(r), \text{ то } r' \in UA(u);$$

$$\text{для } r \in R, \text{ если } p, p' \in P,\ p \in PA(r) \text{ и } p' \in \beta(p), \text{ то } p' \in PA(r).$$

Для каждой роли для того, чтобы на нее мог быть авторизован пользователь, могут быть определены роли, на которые пользователь также должен быть авторизован. Для каждого права доступа для того, чтобы им обладала роль, могут быть определены права доступа, которыми данная роль также должна обладать.

Определение 8.7. В базовой модели ролевого управления доступом задано динамическое ограничение необходимого обладания ролью, если определена функция $\gamma: R \to 2^R$ и выполняется условие для $s \in S$, если $r, r' \in R$, $r \in roles(s)$ и $r' \in \gamma(r)$, то $r' \in roles(s)$. Для каждой роли для того, чтобы на нее мог быть авторизован пользователь в некоторой сессии, могут быть

определены роли, на которые пользователь также должен быть авторизован в данной сессии. Общая структура элементов базовой модели ролевого управления доступом показана на рис. 8.1.

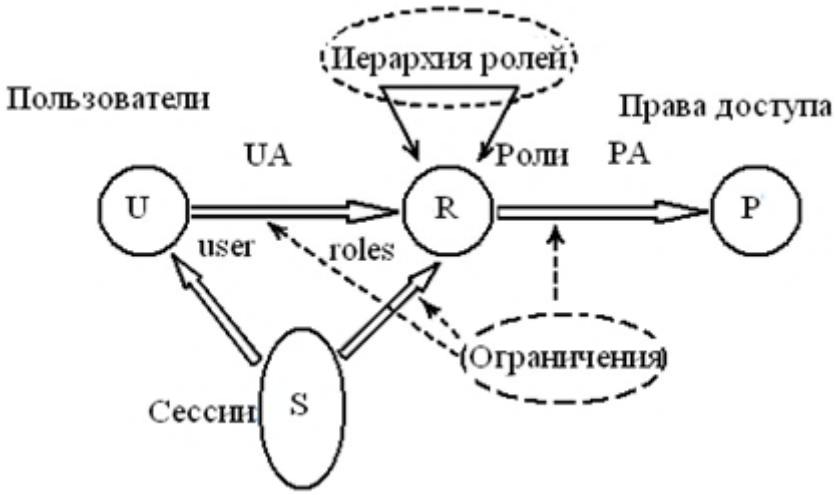

Рис. 8.1. Структура базовой модели ролевого управления доступом

Контрольные вопросы и задания

1. Какие основные проблемы задания правил изменения иерархии ролей рассматриваются в модели администрирования ролевого управления доступом?
2. Что такое роль?
3. Что называется иерархией ролей в базовой модели ролевого управления доступом?
4. Перечислите основные элементы модели *RBAC*.
5. Какого вида ограничения, описанные в базовой модели ролевого управления доступом, могут быть использованы при определении требований либерального или строгого мандатного управления доступом?

9. МОДЕЛЬ АДМИНИСТРИРОВАНИЯ РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ. МОДЕЛЬ МАНДАТНОГО РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ

9.1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

В базовой модели ролевого управления доступом предполагается, что множества U, R, P и функции PA, UA не изменяются с течением времени или существует единственная роль – «офицер безопасности», которая предоставляет возможность изменять эти множества и функции. В реальных КС, в которых одновременно могут работать сотни и тысячи пользователей, а структура ролей и прав доступа может быть очень сложной, проблема администрирования является чрезвычайно важной задачей. Для решения этой задачи рассматривается построенная на основе базовой модели модель администрирования ролевого управления доступом.

В дополнение к используемым элементам базовой модели в модели администрирования ролевого управления доступом рассматриваются следующие элементы:

- AR – множество административных ролей ($AR \cap R = \varnothing$);
- AP – множество административных прав доступа ($AP \cap P = \varnothing$);
- APA: $AR \rightarrow 2^{AP}$ – функция, задающая для каждой административной роли множество административных прав доступа, при этом для каждого права доступа $p \in AP$ существует роль $r \in AR$ такая, что $p \in APA(r)$;
- AUA: $U \rightarrow 2^{AR}$ – функция, задающая для каждого пользователя множество административных ролей, на которые он может быть авторизован.

Кроме того, переопределяется функция:

$roles$: $S \rightarrow 2^R \cup 2^{AR}$ – функция, задающая для пользователя, множество ролей, на которые он авторизован в данной сессии, при этом в каждый момент времени для каждой сессии $s \in S$ выполняется условие $roles(s) \subseteq UA(user(s)) \cup AUA(user(s))$.

Как и в базовой модели, в модели администрирования ролевого управления доступом реализуются иерархия административных ролей и механизмы ограничений.

Определение 9.1. Иерархией ролей в модели администрирования ролевого управления доступом называется заданное на множестве ролей *AR* отношение частичного порядка «≤». При этом выполняется условие

для $u \in U$, если $r, r' \in AR$, $r \in AUA(u)$ и $r' \leq r$, то $r' \in AUA(u)$.

Административные роли по своему назначению могут быть разделены на три группы:

- администрирование множеств авторизованных ролей пользователей;
- администрирование множеств прав доступа, которыми обладает роли;
- администрирование иерархии ролей.

Общая структура элементов модели администрирования ролевого управления доступом имеет вид, показанный на рис. 9.1.

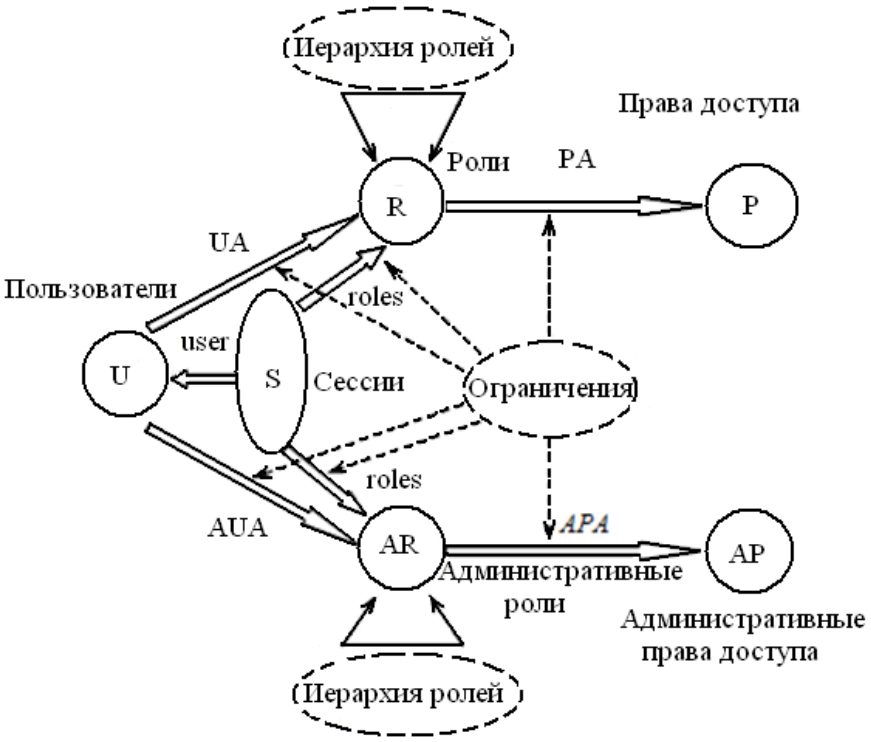

Рис. 9.1. Структура элементов модели администрирования ролевого управления доступом

Как правило, каждой административной роли назначают подмножество иерархии ролей, параметры которых данная административная роль позволяет изменять. Рассмотрим подробнее каждую из групп административных ролей.

9.2. АДМИНИСТРИРОВАНИЕ МНОЖЕСТВ АВТОРИЗОВАННЫХ РОЛЕЙ ПОЛЬЗОВАТЕЛЕЙ

При администрировании множеств авторизованных ролей пользователей изменяются значения функции *UA*, для чего определяются специальные административные роли из множества *AR*. При этом следует задать:

- для каждой административной роли множество ролей, множества авторизованных пользователей которых она позволяет изменять;
- для каждой роли предварительное условие, которому должны соответствовать пользователи, прежде чем они будут включены во множество ее авторизованных пользователей.

Пример 9.1. Пусть заданы иерархия ролей (рис. 9.2) и иерархия административных ролей (рис. 9.3).

Рис. 9.2. Пример иерархии ролей пользователей

Рис. 9.3. Пример иерархии административных ролей пользователей

Минимальная роль в иерархии – служащий (*E*). Иерархия ролей разработчиков проектов имеет максимальную роль – директор (*DIR*), и минимальную роль – инженер (*ED*). В организации выполняются работы по двум проектам. В каждом проекте заданы максимальная роль – руководитель проекта (*PL*1, *PL*2 соответственно), минимальная роль – инженер проекта (*E*1, *E*2 соответственно) и несравнимые между собой роли – инженер по производству (*PE*1, *PE*2 соответственно) и инженер по контролю (*QE*1, *QE*2 соответственно).

Иерархия административных ролей состоит из четырех ролей с максимальной ролью – старший офицер безопасности (*SSO*).

В рассматриваемом примере административная роль *PSO*1 позволяет включать во множества авторизованных ролей пользователя роли *PE*1, *QE*1, *E*1. При этом для того, чтобы любая из перечисленных ролей могла быть включена во множество авторизованных ролей пользователя, он уже должен обладать ролью *ED*.

Для роли $x \in R$ обозначим:

$x: U \rightarrow \{false, true\}$ – булева функция такая, что по определению для пользователя $u \in U$ справедливо равенство $x(u) = true$ тогда и только тогда, когда $x \in UA(u)$.

Определение 9.2. Предварительным условием для роли $r \in R$ называется булева функция c_r: $U \rightarrow \{false, true\}$ такая, что по определению для пользователя $u \in U$ справедливо равенство $c_r(u) = c_r(x_1(u), \ldots, x_k(u))$, где $u \in U$, $x_1, \ldots, x_k \in R$ и $c_r(y_1, \ldots, y_k)$ – булева функция от k булевых переменных. При этом роль $r \in R$ может быть включена во множество авторизованных ролей пользователя $u \in U$, если $c_r(u) = true$.

Обозначим через *CR* все предварительные условия для ролей из *R*.

Определение 9.3. Для администрирования множеств авторизованных ролей пользователей на множестве административных ролей задаются функции:

- $can\text{-}assign_r$: $AR \rightarrow CR \times 2^R$ – функция, задающая для каждой административной роли множество ролей, которые могут быть включены во множе-

ство авторизованных ролей пользователя с использованием данной административной роли при выполнении заданных предварительных условий;

- *can-revoke*$_r$: $AR \rightarrow 2^R$ – функция, задающая для каждой административной роли множество ролей, которые могут быть исключены из множества авторизованных ролей пользователя с использованием данной административной роли.

Как правило, множество ролей, являющееся значением функций *can-assign*$_r$ или *can-revoke*$_r$, задается интервалом ролей одного из четырех видов:

- $[x, y] = \{x' \in R$, где $x \le x'$ и $x' \le y\}$;
- $(x, y] = \{x' \in R$, где $x < x'$ и $x' \le y\}$;
- $[x, y) = \{x' \in R$, где $x \le x'$ и $x' < y\}$;
- $(x, y) = \{x' \in R$, где $x < x'$ и $x' < y\}$,

где $x, y \in R$.

Пример 9.2. Рассмотрим иерархию ролей и иерархию административных ролей из примера 9.1. Зададим значения функций *can-assign*$_r$ (табл. 9.1) и *can-revoke*$_r$ (табл. 9.2) соответственно для административных ролей *PSO1*, *PSO2* и *DSO*.

Таблица 9.1

Значения функции *can-assign*$_r$

Административная роль	Предварительное условие	Множество ролей
*PSO*1	*ED*	[*E*1, *PL*1)
*PSO*2	*ED*	[*E*2, *PL*2)
DSO	*ED and* (*not PL*1)	[*PL*2, *PL*2]
DSO	*ED and* (*not PL*2)	[*PL*1, *PL*1]

Таблица 9.2

Значения функции *can-revoke*$_r$

Административная роль	Множество ролей
*PSO*1	[*E*1, *PL*1)
*PSO*2	[*E*2, *PL*2)
DSO	(*ED*, *DIR*)

Таким образом, административная роль *PSO*1 позволяет включить для пользователя, уже обладающего ролью *ED*, во множество его авторизованных ролей роли *E*1, *PE*1 и *QE*1. Административная роль *DSO* позволяет включить для пользователя, уже обладающего ролью *ED*, во множе-

ство его авторизованных ролей роль $PL1$, при этом данный пользователь не должен обладать ролью $PL2$.

Следует отметить, что в общем случае значения функций $can\text{-}assign_r$ и $can\text{-}revoke_r$ задаются независимо друг от друга, т. е. удаление роли из множества авторизованных ролей пользователя может быть выполнено независимо от того, каким образом эта роль была включена в это множество, и наоборот.

Кроме того, значения функции $can\text{-}revoke_r$ определяются в общем случае без учета иерархии ролей. Следовательно, может оказаться, что для некоторой административной роли разрешено удаление роли $r \in R$ из множества авторизованных ролей $UA(u)$ пользователя $u \in U$. Если при этом найдется роль $r' \in R$ такая, что $r \le r'$ и $r' \in UA(u)$, то удаление роли r из множества авторизованных ролей пользователя u не будет иметь эффекта, так как права доступа роли r являются подмножеством прав доступа роли r', авторизованным пользователем которой останется пользователь u.

Одним из путей решения данной проблемы может являться каскадное удаление. При каскадном удалении, если роль r удаляется из множества авторизованных ролей пользователя, то из этого множества должны быть удалены все роли r' такие, что $r \le r'$. Однако если для некоторой административной роли $ar \in AR$ выполняются условия $r \in can\text{-}revoke_r(ar)$, $r' \notin can\text{-}revoke_r(ar)$, то каскадное удаление будет невозможно. Таким образом, порядок задания значений функции $can\text{-}revoke_r$ и процедура удаления роли из множества авторизованных ролей пользователя требуют дополнительного уточнения.

9.3. АДМИНИСТРИРОВАНИЕ МНОЖЕСТВ ПРАВ ДОСТУПА, КОТОРЫМИ ОБЛАДАЮТ РОЛИ

При администрировании прав доступа ролей изменяются значения функции PA, для чего определяются специальные административные роли из множества AR. Подход к определению порядка администрирования множеств прав доступа ролей аналогичен использованному для администрирования множеств авторизованных ролей пользователя. При этом предварительные условия задаются для прав доступа.

Определение 9.4. Для администрирования множеств прав доступа, которыми обладают роли, на множестве административных ролей задаются:

- $can\text{-}assign_p$: $AR \to CR \times 2^R$ – функция, задающая для каждой административной роли множество ролей, для которых разрешено включать

права доступа во множества прав доступа с использованием данной административной роли при выполнении заданных предварительных условий;

- *can-revoke*$_p$: $AR \rightarrow 2^R$ – функция, задающая для каждой административной роли множество ролей, для которых разрешено удаление прав доступа из множеств прав доступа с использованием данной административной роли.

Пример 9.3. Рассмотрим иерархию ролей и иерархию административных ролей из примера 9.1. Зададим значения функций *can-assign*$_p$ (табл. 9.3) и *can-revoke*$_p$ (табл. 9.4) для административных ролей *PSO*1, *PSO*2 и *DSO*, соответственно.

Таблица 9.3

Значения функции *can-assign*$_p$

Административная роль	Предварительное условие	Множество ролей
DSO	*DIR*	[*PL*1, *PL*1]
DSO	*DIR*	[*PL*2, *PL*2]
*PSO*1	*PL*1 *and* (*not QE*1)	[*PE*1, *PE*1]
*PSO*1	*PL*1 *and* (*not PE*1)	[*QE*1, *QE*1]
*PSO*2	*PL*2 *and* (*not QE*2)	[*PE*2, *PE*2]
*PSO*2	*PL*2 *and* (*not PE*2)	[*QE*2, *QE*2]

Таблица 9.4

Значения функции *can-revoke*$_p$

Административная роль	Множество ролей
DSO	(*ED*, *DIR*)
*PSO*1	[*QE*1, *QE*1]
*PSO*1	[*PE*1, *PE*1]
*PSO*2	[*QE*2, *QE*2]
*PSO*2	[*PE*2, *PE*2]

Таким образом, административная роль *DSO* позволяет включить права доступа, входящие во множество прав доступа роли *DIR*, во множества прав доступа ролей *PL*1, *PL*2. Причем данные права доступа могут быть включены во множества прав доступа ролей, находящихся ниже *PL*1, *PL*2 по иерархии. Административная роль *PSO*1 позволяет включить права доступа, входящие во множество прав доступа роли *PL*1, во множества

прав доступа ролей *PE*1 и *QE*1, но только в каждую по отдельности. Административная роль *DSO* позволяет удалять права доступа из множества прав доступа ролей, находящихся в иерархии между ролями *ED* и *DIR*, а административная роль *PSO*1 позволяет удалять права доступа из множества прав доступа ролей *PE*1 и *QE*1.

В общем случае значения функции $can\text{-}revoke_p$ задаются без учета иерархии ролей. В связи с этим необходимо решать проблему удаления прав доступа ролей, аналогичную рассмотренной ранее проблеме удаления ролей из множеств авторизованных ролей пользователей.

9.4. АДМИНИСТРИРОВАНИЕ ИЕРАРХИИ РОЛЕЙ

Определение правил администрирования, позволяющих изменять иерархию ролей, является самой сложной задачей, рассматриваемой в модели администрирования ролевого управления доступом. Для решения данной задачи используются подходы, реализованные при определении правил администрирования множеств авторизованных ролей пользователей и прав доступа ролей. Для чего задаются три иерархии, элементами которых являются, соответственно:

- возможности – множества прав доступа и других возможностей;
- группы – множества пользователей и других групп;
- объединения – множества пользователей, прав доступа, групп, возможностей и других объединений.

Иерархия объединений является наиболее общей и может включать в себя иерархии возможностей и групп. Определение возможностей и групп требуется для обеспечения соответствия правил администрирования ролей в модели и используемых на практике технологий обработки информации и создания административных структур организаций. Например, для выполнения своих функций пользователю может быть необходим некоторый набор прав доступа, причем отсутствие в этом наборе некоторого права доступа может сделать бессмысленным обладание имеющимися правами.

На основе иерархий возможностей, групп и объединений задается иерархия ролей, элементами которой являются роли-возможности, роли-группы, роли-объединения (*UP*-роли).

Определение 9.5. Роли-возможности – роли, которые обладают только заданными в соответствующей возможности правами доступа.

Определение 9.6. Роли-группы – роли, на которые могут быть авторизованы одновременно только все пользователи соответствующей группы.

Определение 9.7. Роли-объединения – роли, которые обладают возможностями, правами доступа и на которые могут быть авторизованы группы пользователей и отдельные пользователи.

Роли-объединения являются общим случаем ролей, рассматриваемых в модели администрирования ролевого управления доступом.

Определение 9.8. Роль-объединение r_1 в иерархии ролей превосходит роль-объединение r_2, если выполняются условия:

- возможности и права доступа роли r_2 являются подмножеством возможностей и прав доступа роли r_1,
- пользователи и группы пользователей, которые могут быть авторизованы на роль r_1, являются подмножеством множества пользователей и групп пользователей, которые могут быть авторизованы на роль r_2.

Определение 9.9. Для администрирования возможностей и групп пользователей на множестве административных ролей задаются функции:

- $can\text{-}assign_a$: $AR \to CR \times 2^{UPR}$ – функция, задающая для каждой административной роли множество ролей-объединений, во множество прав доступа которых разрешено включать возможности с использованием данной административной роли при выполнении заданных предварительных условий;
- $can\text{-}revoke_a$: $AR \to 2^{UPR}$ – функция, задающая для каждой административной роли множество ролей-объединений, из множества прав доступа которых разрешено удаление возможностей с использованием данной административной роли;
- $can\text{-}assign_g$: $AR \to CR \times 2^{UPR}$ – функция, задающая для каждой административной роли множество ролей-объединений, которые разрешено включать во множество авторизованных ролей групп пользователей с использованием данной административной роли при выполнении заданных предварительных условий;
- $can\text{-}revoke_g$: $AR \to 2^{UPR}$ – функция, задающая для каждой административной роли множество ролей-объединений, которые разрешено удалять из множество авторизованных ролей групп пользователей с использованием данной административной роли.

При этом во множество значений заданных функций входит множество 2^{UPR}, где UPR – множество ролей объединений, что обеспечивает общность способов определения правил администрирования для всех используемых иерархий.

Необходимость определения иерархий возможностей и групп обусловлена также тем, что построенные на их основе правила администрирования отличны друг от друга. Предварительные условия, заданные в функции $can\text{-}assign_a$, используется аналогично предварительным условиям, заданным в функции $can\text{-}assign_p$, а предварительные условия, заданные

в функции *can-assign*$_g$, используется аналогично предварительным условиям, заданным в функции *can-assign*$_r$. Например, для того, чтобы во множество прав доступа роли-объединения была включена возможность, данная возможность должна входить во множества прав доступа ролей, указанных в предварительном условии функции *can-assign*$_a$. А для того, чтобы роль-объединение была включена во множество авторизованных ролей группы пользователей, пользователи данной группы должны уже обладать ролями в соответствии с предварительным условием функции *can-assign*$_g$.

Включение возможности во множество прав доступа роли-объединения означает, что соответствующая роль-возможность в иерархии ролей станет непосредственно «ниже» роли-объединения. Наоборот, включение роли-объединения во множество авторизованных ролей группы пользователей означает, что соответствующая роль-группа в иерархии ролей станет непосредственно «выше» роли-объединения.

Определение 9.10. Для администрирования иерархии ролей (добавления и удаления ролей, включения или удаления отношений иерархии) на множестве административных ролей задается функция:

- *can-modify*: $AR \rightarrow 2^{UPR}$ – определяющая для каждой административной роли интервал ролей (исключая границы интервала), на котором возможно изменение иерархии ролей с использованием данной административной роли.

Пример 9.4. Рассмотрим иерархию ролей и иерархию административных ролей из примера 9.1. Определим значения функции *can-modify* (табл. 9.5) для административных ролей *PSO*1 и *DSO*.

Таблица 9.5

Значения функции *can-modify*

Административная роль	Множество ролей
*PSO*1	(*E*1, *PL*1)
DSO	(*ED*, *DIR*)

Административные роли *PSO*1и *DSO* позволяют изменять иерархию ролей на интервалах, соответственно, (*E*1, *PL*1) и (*ED*, *DIR*).

Описанные способы задания правил администрирования иерархии ролей в общем случае требуют уточнения. Так, например, возможно возникновение следующей ситуации. Пользователь с административной ролью *DSO* удаляет из иерархии роль *PL*1. В то же время роль *PL*1 входит в предварительные условия и участвует в определении интервалов значений функций *can-assign*$_r$ и *can-revoke*$_r$. Следовательно, для административ-

ной роли *DSO* должно быть запрещено удаление ролей, участвующих в задании значений и предварительных условий функций администрирования.

Кроме того, возможна иная ситуация.

Пример 9.5. Возьмем иерархию ролей и иерархию административных ролей согласно примеру 9.1 и функцию *can-modify*, значения которой заданы в соответствии с табл. 9.5 (рис. 9.4).

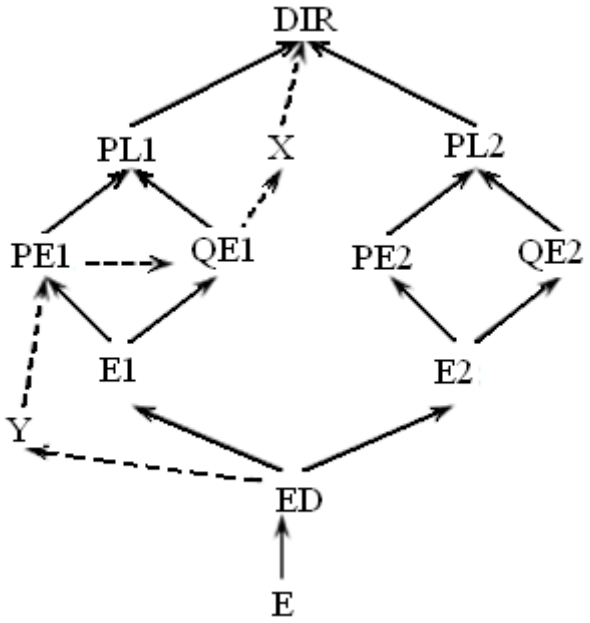

Рис. 9.4. Пример модификации иерархии ролей

Предположим, что пользователь с административной ролью *DSO* добавил в иерархию ролей роли *X* и *Y*. Так как пользователь с административной ролью *PSO*1 может изменять иерархию ролей на интервале ролей (*E*1, *PL*1), пусть он определит роль *QE*1, как старшую над ролью *PE*1. Следовательно, административная роль *PSO*1 позволяет задать роль *X* старшей над ролью *Y*, несмотря на то, что эти роли не входят в определенный для *PSO*1 интервал (*E*1, *PL*1).

Возможна реализация нескольких подходов по определению порядка администрирования иерархии ролей в данной ситуации. Во-первых, можно запретить пользователю с административной ролью *DSO* добавлять роли *X* и *Y*, так как это нарушает целостность интервала (*E*1, *PL*1), на котором разрешено администрирование для роли *PSO*1. Во-вторых, можно запретить пользователю с административной ролью *PSO*1 определять роль *QE*1 старшей над ролью *PE*1. В-третьих, можно разрешить выполнять все запрашиваемые действия с использованием административных ролей *DSO* и *PSO*1.

9.5. МОДЕЛЬ МАНДАТНОГО РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ

Ролевое управление доступом является развитием дискреционного управления доступом, в то же время оно является достаточно гибким и, используя механизм ролей, позволяет реализовать требования мандатной политики безопасности, ориентированной на защиту конфиденциальности информации.

Рассмотрим подход, реализующий мандатное управление доступом на основе базовой модели ролевого управления доступом.

Используем следующие обозначения:

U – множество пользователей (субъектов);

O – множество объектов;

$(L, \leq)$ – решетка уровней конфиденциальности;

$c{:}U \to L$ – функция уровней доступа пользователей;

$c{:}\ O \to L$ – функция уровней конфиденциальности объектов;

$A = \{read, write\}$ – виды доступа.

Будем различать два вида мандатного управления доступом: либеральный и строгий (согласно модели Белла – Лападулы).

Определение 9.11. Доступ $(s, (o, r)) \in S \times P$ является безопасным для либерального мандатного управления доступом, если выполняется одно из условий:

- $r = read$ и $c(user(s)) \geq c(o)$ (*ss*-свойство);
- $r = write$ и, если существует доступ $(s, (o', read)) \in S \times P$, то $c(o) \geq c(o')$ (либеральное *-свойство).

Определение 9.12. Доступ $(s, o, r) \in S \times P$ является безопасным для строгого мандатного управления доступом, если выполняется одной из условий:

- $r = readис(user(s)) \geq c(o)$ (*ss*-свойство);
- $r = write$ и, если существует доступ $(s, (o', read)) \in S \times P$, то $c(o) = c(o')$ (строгое *-свойство).

Построим систему ролевого управления доступом. Пусть

$R = \{x\_read \mid x \in L\} \cup \{x\_write \mid x \in L\}$ – множество ролей;

$P = \{(o, read) \mid o \in O\} \cup \{(o, write) \mid o \in O\}$ – множество прав доступа.

Зададим на множестве ролей R иерархию, при этом иерархии ролей на множествах $\{x\_read \mid x \in L\}$ и $\{x\_write \mid x \in L\}$ будут независимы.

Определение 9.13. Иерархией на множестве ролей R в соответствии с требованиями либерального мандатного управления доступом называется

отношение частичного порядка «≤», где для ролей r, $r' \in R$ справедливо неравенство $r \le r'$, если выполняется одно из условий:

- $r = x\_read$, $r' = x'\_read$ и $x \le x'$;
- $r = x\_write$, $r' = x'\_write$ и $x' \le x$.

Определение 9.14. Иерархией на множестве ролей R в соответствии с требованиями строгого мандатного управления доступом называется отношение частичного порядка «≤», где для ролей r, $r' \in R$ справедливо неравенство $r \le r'$, если выполняется одно из условий:

- $r = x\_read$, $r' = x'\_read$ и $x \le x'$;
- $r = x\_write$, $r' = x'\_write$ и $x = x'$ (каждая роль вида x_write сравнима только сама с собой).

Определение 9.15. Модель ролевого управления доступом соответствует требованиям либерального мандатного управления доступом, если иерархия на множестве ролей R соответствует требованиям определения 9.14 и выполняются ограничения:

- ограничение функции UA – для каждого пользователя $u \in U$ роль $x\_read = \oplus(UA(u) \cap \{y\_read \mid y \in L\}) \in UA(u)$ (здесь $x = c(u)$) и $\{y\_write \mid y \in L\} \subset UA(u)$;
- ограничение функции $roles$ – для каждой сессии $s \in S$ справедливо равенство $roles(s) = \{y\_read \mid y \in L, y \le x\} \cup \{x\_write\}$;
- ограничение функции PA – должно выполняться:

– для каждого $x \in L$ доступ $(o, read) \in PA(x\_read)$ тогда и только тогда, когда доступ $(o, write) \in PA(x\_write)$;

– для каждого доступа $(o, read) \in P$ существует единственная роль x_read: $(o, read) \in PA(x\_read)$ (здесь $x = c(o)$).

Определение 9.16. Модель ролевого управления доступом соответствует требованиям строгого мандатного управления доступом, если иерархия на множестве ролей R соответствует требованиям определения 9.15 и выполняются ограничения:

- ограничение функции UA – для каждого пользователя $u \in U$ роль $x\_read = \oplus(UA(u) \cap \{y\_read \mid y \in L\}) \in UA(u)$ (здесь $x = c(u)$) и $\{y\_write \mid y \in L\} \subset UA(u)$;
- ограничение функции $roles$ – для каждой сессии $s \in S$ справедливо равенство $roles(s) = \{x\_read, x\_write\}$;
- ограничение функции PA – должно выполняться:

– для каждого $x \in L$ доступ $(o, read) \in PA(x\_read)$ тогда и только тогда, когда доступ $(o, write) \in PA(x\_write)$;

– для каждого доступа $(o, read) \in P$ существует единственная роль x_read: $(o, read) \in PA(x\_read)$ (здесь $x = c(o)$).

Таким образом, требования соответствия либеральному и строгому мандатному управлению доступом для моделей ролевого управления доступом совпадают во всем, кроме требований к соответствующей иерархии ролей и ограничениям на функцию *roles*.

В рамках модели мандатного ролевого управления доступом дадим определение информационного потока.

Определение 9.17. Будем считать, что существует информационный поток от объекта $o \in O$ к объекту $o' \in O$ тогда и только тогда, когда существуют роли r, $r' \in R$, сессия $s \in S$ такие, что $(o, read) \in PA(r)$, $(o', write) \in PA(r')$ и $r, r' \in roles(s)$.

Обоснуем, что в модели ролевого управления доступом, соответствующей требованиям либерального или строгого мандатного управления доступом, невозможна реализация запрещенных информационных потоков от объектов с высоким уровнем конфиденциальности к объектам с низким уровнем диссертации.

Теорема 9.1. Если модель ролевого управления доступом соответствует требованиям либерального или строгого мандатного управления доступом, то в ней для любых объектов o, $o' \in O$ таких, что $c(o) > c(o')$, невозможно возникновение информационного потока от o к o'.

Доказательство. Докажем от противного. Пусть существуют объекты o, $o' \in O$ такие, что $c(o) > c(o')$, и возможно возникновение информационного потока от o к o'. По определению 9.17 существуют роли r, $r' \in R$ и сессия $s \in S$, такие что $(o, read) \in PA(r)$, $(o', write) \in PA(r')$ и $r, r' \in roles(s)$. Следовательно, выполняется одно из условий:

- требования либерального мандатного управления доступом и по определению 9.16 условия $r = c(o)\_read$, $r' = c(o')\_write$ и $c(o) \le c(o')$;
- требования строгого мандатного управления доступом и по определению 9.17 условия $r = c(o)\_read$, $r' = c(o')\_write$ и $c(o) = c(o')$.

Это противоречие. Теорема доказана.

Модель мандатного ролевого управления доступом в первую очередь ориентирована на обеспечение защиты конфиденциальности информации. В то же время возможно доопределение свойств модели с целью анализа систем защиты целостности и конфиденциальности информации.

Используем обозначения модели ролевого управления доступом защиты от угрозы конфиденциальности информации. Также используем обозначения:

$(LI, \le)$ – решетка уровней целостности информации;

$ci: U \to LI$ – функция уровня целостности пользователя;

$ci: O \to LI$ – функция уровня целостности объекта.

По аналогии с моделью ролевого управления доступом защиты от угрозы конфиденциальности информации, будем различать два вида мандатного контроля целостности информации: либеральный и строгий.

Определение 9.18. Доступ $(s, (o, r)) \in S \times P$ является безопасным для либерального мандатного контроля целостности, если выполняется одно из условий:

- $r = read$ и $ci(user(s)) \le ci(o)$;
- $r = write$ и, если существует доступ $(s, (o', read)) \in S \times P$, то $ci(o) \le ci(o')$.

Определение 9.19. Доступ $(s, (o, r)) \in S \times P$ является безопасным для строгого мандатного контроля целостности, если выполняется одно из условий:

- $r = read$ и $ci(user(s)) \le ci(o)$;
- $r = write$ и, если существует доступ $(s, (o', read)) \in S \times P$, то $ci(o) = ci(o')$.

Рассмотрим иерархию ролей на множестве $RI = \{xi\_read \mid xi \in LI\} \cup \{xi\_write \mid xi \in LI\}$. Иерархии ролей на множествах $\{xi\_read \mid xi \in LI\}$ и $\{xi\_write \mid xi \in LI\}$ независимы.

Определение 9.20. Иерархией на множестве ролей RI в соответствии с требованиями либерального мандатного контроля целостности называется отношение частичного порядка «≤», где для ролей $r, r' \in R$ справедливо неравенство $r \le r'$, если выполняется одно из условий:

- $r = xi\_read$, $r' = xi'\_read$ и $xi' \le xi$;
- $r = xi\_write$, $r' = xi'\_write$ и $xi \le xi'$.

Определение 9.21. Иерархией на множестве ролей RI в соответствии с требованиями строгого мандатного контроля целостности называется отношение частичного порядка «≤», где для ролей $r, r' \in R$ справедливо неравенство $r \le r'$, если выполняется одно из условий:

- $r = xi\_read$, $r' = xi'\_read$ и $xi' \le xi$;
- $r = xi\_write$, $r' = xi'\_write$ и $xi = xi'$ (каждая роль вида xi_write сравнима только сама с собой).

Определение 9.22.1. Модель ролевого управления доступом соответствует требованиям либерального мандатного контроля целостности, если иерархия на множестве ролей RI соответствует требованиям определения 9.20 и выполняются ограничения:

- ограничение функции UA – для каждого пользователя $u \in U$ роль $xi\_read = \oplus(UA(u) \cap \{yi\_read \mid yi \in LI\}) \in UA(u)$ (здесь $xi = ci(u)$) и $\{yi\_write \mid yi \in LI\} \subset UA(u)$;
- ограничение функции $roles$ – для каждой сессии $s \in S$ справедливо равенство $roles(s) = \{yi\_read \mid yi \in LI, yi \ge xi\} \cup \{xi\_write\}$;

- ограничение функции PA – должно выполняться:

– для каждого $xi \in LI$ доступ $(o, read) \in PA(xi\_read)$ тогда и только тогда, когда доступ $(o, write) \in PA(xi\_write)$;

– для каждого доступа $(o, read) \in P$ существует единственная роль xi_read: $(o, read) \in PA(xi\_read)$ (здесь $xi = ci(o)$).

Определение 9.22.2. Модель ролевого управления доступом соответствует требованиям строгого мандатного контроля целостности, если иерархия на множестве ролей RI соответствует требованиям определения 9.21 и выполняются ограничения:

- ограничение функции UA – для каждого пользователя $u \in U$ роль $xi\_read = \oplus(UA(u) \cap \{yi\_read \mid yi \in LI\}) \in UA(u)$ (здесь $xi = ci(u)$) и $\{yi\_write \mid yi \in LI\} \subset UA(u)$;
- ограничение функции $roles$ – для каждой сессии $s \in S$ справедливо равенство $roles(s) = \{xi\_read, xi\_write\}$;
- ограничение функции PA – должно выполняться:

– для каждого $xi \in LI$ доступ $(o, read) \in PA(xi\_read)$ тогда и только тогда, когда доступ $(o, write) \in PA(xi\_write)$;

– для каждого доступа $(o, read) \in P$ существует единственная роль xi_read: $(o, read) \in PA(xi\_read)$ (здесь $xi = ci(o)$).

Теорема 9.2. Если модель ролевого управления доступом соответствует требованиям либерального или строгого мандатного контроля целостности, то в ней для любых объектов o, $o' \in O$ таких, что $ci(o) < ci(o')$, невозможно возникновение информационных потоков от o к o'.

Доказательство. Доказательство теоремы осуществляется аналогично доказательству теоремы 9.1.

Построим систему мандатного ролевого управления доступом, ориентированную на защиту конфиденциальности и целостности информации. Пусть $R = (\{x\_read \mid x \in L\} \times \{xi\_read \mid xi \in LI\}) \cup (\{x\_write \mid x \in L\}) \times \{xi\_write \mid xi \in LI\})$ – множество ролей.

Зададим на множестве ролей R иерархию, при этом возможно произвольное сочетание иерархий ролей либерального или строгого мандатного управления доступом с либеральным или строгим контролем целостности. Иерархии ролей на множествах $\{x\_read \mid x \in L\} \times \{xi\_read \mid xi \in LI\}$ и $\{x\_write \mid x \in L\}) \times \{xi\_write \mid xi \in LI\}$ независимы.

В качестве примера рассмотрим требования либерального мандатного управления доступом и контроля целостности.

Определение 9.23. Иерархией на множестве ролей R в соответствии с требованиями либерального мандатного управления доступом и контроля целостности называется отношение частичного порядка «≤», где для $r = (rc, ri)$, $r' = (rc', ri') \in R$, при этом ролей $rc, rc' \in \{x\_read \mid x \in L\} \cup \{x\_write$

$| x \in L\})$, $ri, ri' \in \{xi\_read \mid xi \in LI\} \cup \{xi\_write \mid xi \in LI\}\}$, справедливо неравенство $r \le r'$, если $rc \le rc'$ в иерархии ролей либерального мандатного управления доступом (в соответствии с определением 9.13) и $ri \le ri'$ в иерархии ролей либерального мандатного контроля целостности (в соответствии с определением 9.20).

Определение 9.24. Модель ролевого управления доступом соответствует требованиям либерального мандатного управления доступом и контроля целостности, если иерархия на множестве ролей R соответствует требованиям определения 9.23 и выполняются ограничения:

- ограничение функции UA – для каждого пользователя $u \in U$ роль $(x\_read, xi\_read) = \oplus(UA(u) \cap (\{y\_read \mid y \in L\} \times \{yi\_read \mid yi \in LI\})) \in UA(u)$ (здесь $x = c(u)$, $xi = ci(u)$) и $\{y\_write \mid y \in L\}) \times \{yi\_write \mid yi \in LI\} \subset UA(u)$;
- ограничение функции $roles$ – для каждой сессии $s \in S$ справедливо равенство $roles(s) = \{(y\_read, yi\_read) \mid y \in L, yi \in LI, y \le x, yi \ge xi\} \cup \{(x\_write, xi\_write)\}$;
- ограничение функции PA – должно выполняться:

– для каждых $x \in L$, $xi \in LI$ доступ $(o, read) \in PA((x\_read, xi\_read))$ тогда и только тогда, когда доступ $(o, write) \in PA((x\_write, xi\_write))$;

– для каждого доступа $(o, read) \in P$ существует единственная роль (x_read, xi_read): $(o, read) \in PA((x\_read, xi\_read))$ (здесь $x = c(o)$, $xi = ci(o)$).

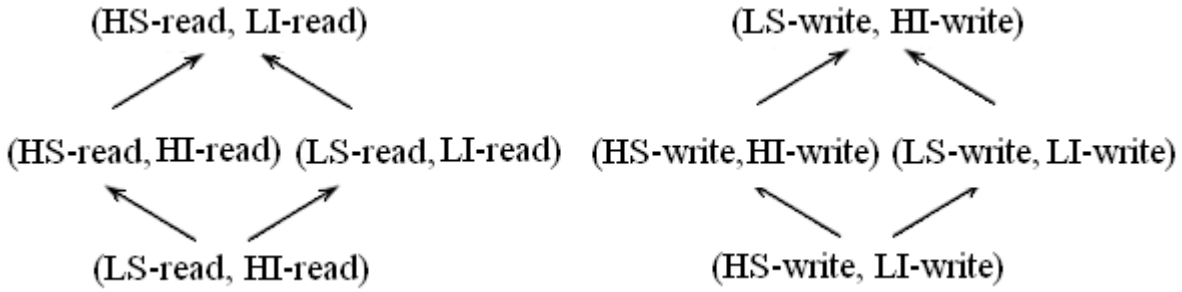

Рис. 9.5. Либеральное мандатное управление доступом и мандатный контроль целостности

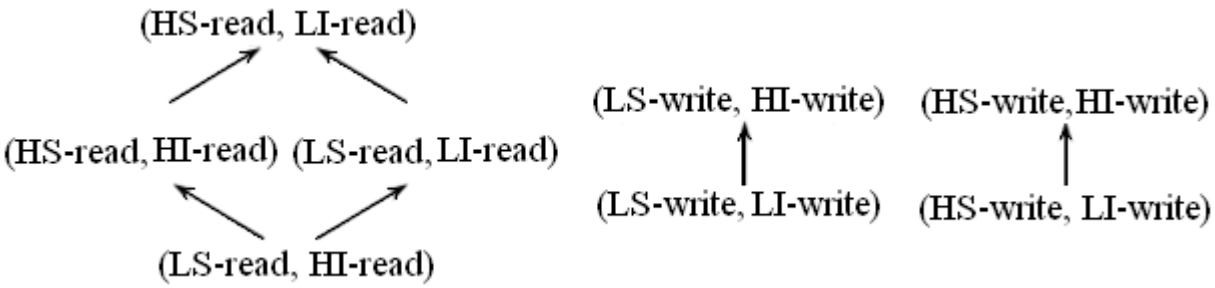

Рис. 9.6. Строгое мандатное управление доступом и либеральный мандатный доступ

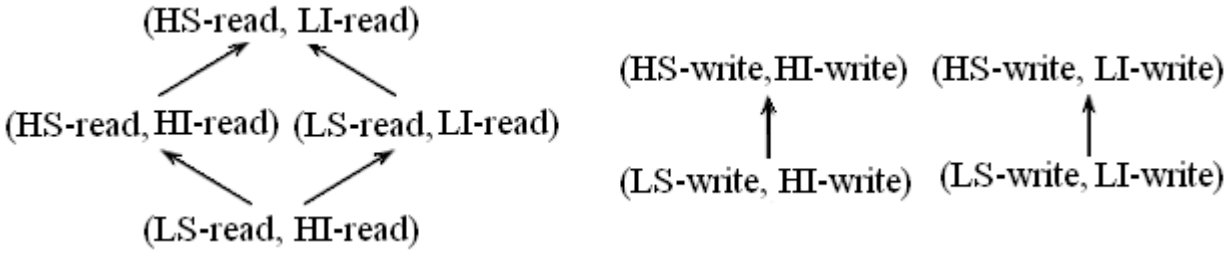

Рис.9.7. Либеральное мандатное управление доступом
и строгий мандатный контроль целостности

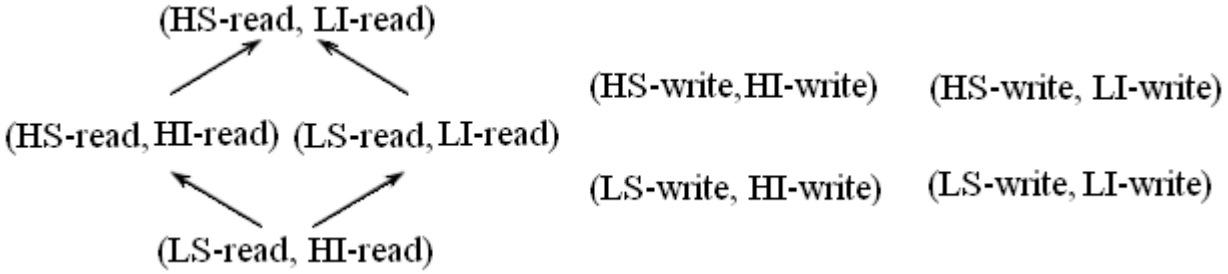

Рис.9.8. Строгое мандатное управление доступом
и мандатный контроль целостности

Для решеток уровней конфиденциальности $(L, \leq) = \{LS, HS\}$ и уровней целостности $(LI, \leq) = \{LI, HI\}$ на рис. 9.5–9.8 представлены иерархии ролей для четырех возможных сочетаний иерархий ролей либерального или строгого мандатного управления доступом с либеральным или строгим мандатным контролем целостности.

Контрольные вопросы и задания

1. Является ли решеткой иерархия ролей, соответствующая требованиям либерального или строгого мандатного управления доступом в модели мандатного ролевого управления доступом?
2. Либеральное или строгое мандатное управление доступом модели рассматривается в классической модели Белла – Лападулы?
3. Каким образом в определениях либерального или строгого мандатного управления доступом модели ролевого управления доступом реализовано *ss*-свойство классической модели Белла – Лападулы?
4. Каким образом в определениях либерального или строгого мандатного управления доступом модели ролевого управления доступом реализовано *-свойство классической модели Белла – Лападулы?
5. Построить графы иерархии ролей для модели мандатного ролевого управления доступом для решеток уровней конфиденциальности $(L, \leq) = \{LS, MS, HS\}$ и уровней целостности $(LI, \leq) = \{LI, MI, HI\}$.

10. СУБЪЕКТНО-ОРИЕНТИРОВАННАЯ МОДЕЛЬ ИЗОЛИРОВАННОЙ ПРОГРАММНОЙ СРЕДЫ

Как правило, модели, ориентированные на реализацию политики безопасности, не учитывают возможности субъектов по изменению конфигурации или параметров функционирования КС, которые могут привести к изменению ее свойств и, как предельный случай, к полной неприменимости той или иной модели к описанию отношений «субъект-объект» в измененной системе.

Рассмотрим на основе субъектно-ориентированную модель ИПС, в которой основное внимание уделяется определению порядка безопасного взаимодействия субъектов системы, описанию и обоснованию необходимых условий реализации в системе ИПС. Данная модель основана на дискреционном управлении доступом.

Определение 10.1. Объект o в момент времени t ассоциирован с субъектом s, если состояние объекта o повлияло на состояние субъекта s в следующий момент времени $t + 1$ (т. е. субъект s использует информацию, содержащуюся в объекте o).

Субъект в общем случае реализует некоторое отображение множества ассоциированных объектов в момент времени t на множество ассоциированных объектов в момент времени $t + 1$. В связи с этим можно выделить ассоциированные объекты, изменение которых изменяет вид отображения ассоциированных объектов (объекты, содержащие, как правило, код программы) – функционально ассоциированные, и ассоциированные объекты-данные (являющиеся аргументом операции, но не изменяющие вида отображения). Далее под ассоциированными объектами понимаются функционально ассоциированные объекты, в иных случаях делаются уточнения.

Определение 10.2. Объект o называется источником для субъекта s', если существует субъект s, в результате воздействия которого на объект o в системе возникает субъект s'. Субъект s, порождающий новый субъект из объекта o, в свою очередь называется активизирующим субъектом для субъекта s', которого назовем порожденным субъектом.

Используем обозначения:

$[s]_t$ – множество объектов, ассоциированных с субъектом s в момент времени t;

$Stream(s, o) \rightarrow o'$ – поток информации от объекта o к объекту o';

Create(*s*, *o*) → *s′* – операция порождения субъектов (из объекта *o* порожден субъект *s′* при активизирующем воздействии субъекта *s*).

Очевидно, что операция порождения субъектов зависит как от свойств активизирующего субъекта, так и от содержания объекта-источника.

В момент порождения субъекта *s′* из объекта *o* он является ассоциированным объектом для субъекта *s′*.

В определении подчеркнуто, что поток информации рассматривается не между субъектом и объектом, а между объектами, например, объектом и ассоциированными объектами субъекта (либо между двумя объектами). Активная роль субъекта выражается в реализации данного потока (это означает, что операция порождения потока локализована в субъекте и отображается состоянием его функционально ассоциированных объектов). Отметим, что операция *Stream* может создавать новый объект или уничтожать его.

Отношение «между объектами существует информационный поток» является транзитивным (относительно пары субъектов), т. е. если существует *Stream*(*s*, *o*) → *o′* и существует *Stream*(*s′*, *o′*) → *o″*, то существует и *Stream*((*s*, *s′*), *o*) → *o″*.

Выделим все множество информационных потоков P_f и в соответствии с априорно заданной политикой безопасности разобьем его на два непересекающихся подмножества N_f и L_f, где $P_f = N_f \cup L_f$, $N_f \cap L_f = \varnothing$, N_f – подмножество запрещенных информационных потоков, характеризующее несанкционированный доступ, L_f – подмножество информационных потоков, характеризующих легальный доступ.

В рассматриваемой субъектно-ориентированной модели не производится уточнений известных моделей политик безопасности, но формулируются условия корректного существования элементов КС, обеспечивающих реализацию той или иной политики безопасности. Поскольку критерий разбиения на множества N_f и L_f не связан со следующими далее утверждениями (постулируется лишь наличие субъекта, реализующего фильтрацию потоков), то можно говорить об инвариантности субъектно-ориентированной модели относительно любой принятой в КС политики безопасности (не противоречащей условиям утверждений).

Для разделения всего множества потоков в КС на подмножества N_f и L_f необходимо существование активной компоненты (субъекта), который:

- активизировался бы при возникновении любого потока;
- производил бы фильтрацию потоков в соответствии с принадлежностью к множествам N_f или L_f.

Определение 10.3. Монитор обращений (МО) – субъект, активизирующийся при возникновении любого информационного потока между объектами КС.

Теперь сформулируем понятие монитора безопасности (в литературе также применяется понятие монитора ссылок). Это понятие связано с упоминаемой выше задачей фильтрации потоков. Поскольку целью является обеспечение безопасности КС, то и целевая функция монитора – фильтрация с целью обеспечения безопасности.

Определение 10.4. Монитор безопасности объектов (МБО) – МО, который разрешает потоки, принадлежащие только множеству легального доступа L_f. Разрешение потока в данном случае понимается как выполнение операции над объектом-получателем потока, а запрещение – как невыполнение (т. е. неизменность объекта-получателя потока).

Монитор безопасности объектов фактически является механизмом реализации политики безопасности в КС.

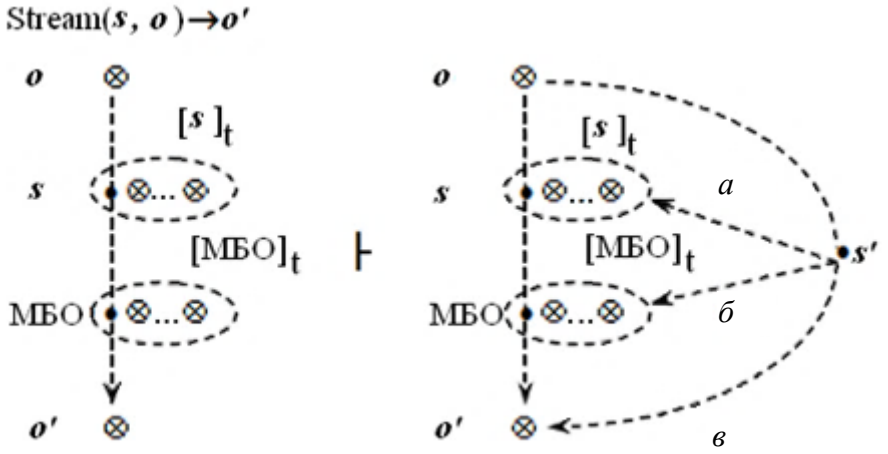

Рис. 10.1. Возможные пути обхода политики безопасности нарушителем *s′*: *а* – изменение функциональности субъекта *s*; *б* – изменение функциональности МБО; *в* – реализация информационного потока в обход МБО

Представляется очевидным, что при изменении функционально ассоциированных с МБО объектов могут измениться и свойства самого МБО, заключающиеся в фильтрации потоков, и, как следствие, могут возникнуть потоки, принадлежащие множеству N_f (рис. 10.1).

Введем в связи с этим понятие корректности субъектов.

Определение 10.5. Субъекты s и s' называются корректными относительно друг друга, если в любой момент времени отсутствует поток (изменяющий состояние объекта) между любыми объектами o и o', ассоциированными, соответственно, с субъектами s и s'. Таким образом, выполняется условие: для $t \geq 0$, $o \in [s]_t$, $o' \in [s']_t$ не существует субъекта s'' такого, что или $Stream(s'', o) \rightarrow o'$ или $Stream(s'', o') \rightarrow o$.

Смысл понятия корректности можно пояснить на следующем примере: программы, существующие в едином пространстве ОС, не должны иметь функциональных возможностей изменения «чужого» вектора кода и состояния переменных.

Определение 10.6. Субъекты s и s' называются абсолютно корректными относительно друг друга, если множества ассоциированных объектов указанных субъектов не имеют пересечения. Таким образом, выполняется условие: для $t \geq 0$, $[s]_t \cap [s']_t = \varnothing$.

Абсолютная корректность легко достижима в случае виртуального адресного пространства.

Определение 10.7. Монитор порождения субъектов (МПС) – субъект, активизирующийся при любом порождении субъектов.

По аналогии с переходом от МО к МБО введем понятие монитора безопасности субъектов.

Определение 10.8. Монитор безопасности субъектов (МБС) – МПС, который разрешает порождение субъектов только для фиксированного множества пар активизирующих субъектов и объектов-источников.

Определение 10.9. КС называется замкнутой по порождению субъектов (обладает замкнутой программной средой), если в ней действует МБС.

Воздействие МБС выделяет во всем множестве субъектов КС подмножество разрешенных субъектов.

Замкнутости КС по порождению субъектов недостаточно для описания свойств системы в части защищенности, поскольку необходимо обеспечить корректность порождаемых МБС субъектов относительно его самого и МБО. Механизм замкнутой программной среды сокращает множество возможных субъектов до некоторого множества фиксированной мощности, но при этом допускает существование некорректных субъектов, включенных в замкнутую среду.

Сформулируем определение ИПС.

Определение 10.10. Программная среда называется изолированной (абсолютно изолированной), если она является замкнутой по порождению субъектов (в ней действует МБС) и субъекты из порождаемого множества корректны (абсолютно корректны) относительно друг друга, МБС и МБО.

Любое подмножество субъектов ИПС, включающее МБС, также составляет ИПС. Дополнение ИПС (абсолютной ИПС) субъектом, корректным (абсолютно корректным) относительно любого из числа входящих в ИПС (абсолютную ИПС), оставляет ее изолированной (абсолютно изолированной).

Лемма 10.1. МБО в абсолютной ИПС разрешает порождение потоков только из множества L_f.

Доказательство. Из определения абсолютной ИПС следует возможность существования в программной среде только конечного множества

субъектов, которые в свою очередь корректны относительно МБС и МБО. Следовательно, ассоциированные объекты МБО могут изменяться только самим МБО, и в КС реализуются только потоки, принадлежащие множеству L_f. Лемма доказана.

При рассмотрении операции порождения субъекта возникает важная задача, связанная с тем, что в реальных КС одинаково поименованные объекты в различные моменты времени могут иметь различное размещение (в разных каталогах).

Определение 10.11. В момент времени t объекты o и o' тождественны, если они совпадают как слова, записанные на одном языке.

Определение 10.12. В момент времени t субъекты s и s' тождественны, если попарно тождественны все ассоциированные с ними объекты.

Порожденные субъекты тождественны, если тождественны порождающие субъекты и объекты-источники. Верность данного замечания следует из тождества функционально ассоциированных объектов в порождающих субъектах, которые отвечают за порождение нового субъекта, а также из тождества аргументов (ассоциированных объектов-данных), которые соответствуют объектам-источникам.

Предположим, что зафиксировано состояние объекта o в некоторый момент времени t_0. Будем обозначать состояние объекта o в момент времени t как $o[t]$.

Определение 10.13 Операция порождения субъекта $create(s, o) \rightarrow s'$ называется порождением с контролем неизменности объекта-источника, если для любого момента времени $t > t_0$, в который активизирована операция порождения объекта «*Create*», порождение субъекта s' возможно только при тождественности объектов $o[t_0]$ и $o[t]$.

В условиях определения 10.13 порожденные субъекты $s'[t_1]$ и $s'[t_2]$ тождественны, если $t_1 > t_0$ и $t_2 > t_0$. При $t_1 = t_2$ порождается один и тот же субъект.

Лемма 10.2 (базовая теорема изолированной программной среды, ИПС). Если с момента времени t_0 в абсолютной ИПС действует только порождение субъектов с контролем неизменности объекта-источника, и все порождаемые субъекты абсолютно корректны относительно друг друга и существующих субъектов (в том числе, МБО и МБС), то в любой момент времени $t > t_0$ программная среда также остается абсолютной ИПС.

Доказательство. По условию теоремы в программной среде возможно существование потоков, изменяющих состояние объектов, не ассоциированных в этот момент времени с каким-либо субъектом. Если объект с измененным состоянием не является источником для порождения субъекта, то множество субъектов ИПС не расширяемо, в противном случае

(измененный объект является источником для порождения субъекта) по условиям теоремы (порождение субъекта с контролем) порождение субъекта невозможно. Следовательно, мощность множества субъектов не может превышать той, которая была зафиксирована до изменения состояния любого объекта. Таким образом, в любой момент времени $t > t_0$ программная среда остается абсолютной ИПС. Лемма доказана.

Можно сформулировать методологию проектирования (разработки) гарантированно защищенных КС. Сущность данной методологии состоит в том, что при проектировании защитных механизмов КС необходимо опираться на совокупность приведенных выше достаточных условий, которые должны быть реализованы для субъектов, что гарантирует защитные свойства, определенные при реализации МБО в КС (т. е. гарантированное выполнение заданной МБО политики безопасности).

Рассмотренная концепция ИПС является расширением классического подхода к реализации ядра безопасности. Обычно модель функционирования ядра безопасности изображается в виде следующей схемы (рис. 10.2). Объект управления (ОУ) содержит в себе информацию о разрешенных или запрещенных информационных потоках.

Рис. 10.2. Классическая схема ядра безопасности

Для учета влияния субъектов в КС на систему защиты целесообразно рассматривать расширенную схему взаимодействия элементов системы. При этом должна быть особо подчеркнута роль МБС при порождении субъектов из объектов (рис. 10.3).

Взаимодействие субъектов и объектов при порождении потоков уточнено введением ассоциированных с субъектом объектов. $ОУ_{МБО}$ содержит информацию о разрешенных значениях отображения «*Stream*» (об элементах множества L_f или N_f) и ассоциирован (ассоциированный объект-данные) с МБО, и $ОУ_{МБС}$ – содержит информацию о значениях отображения «*Create*» (элементы множества разрешенных объектов-источников) и ассоциирован с МБС.

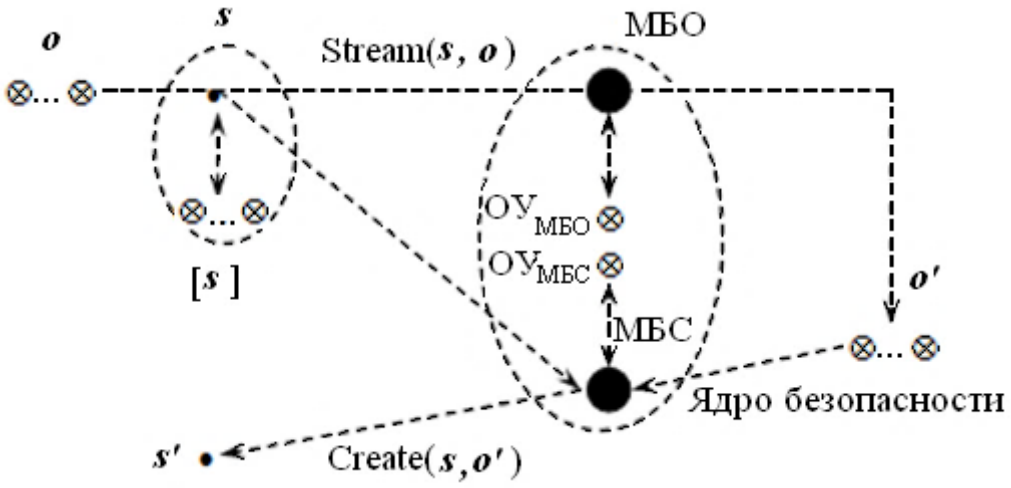

Рис. 10.3. Ядро безопасности
с учетом контроля порождения субъектов

Опираясь на лемму 10.2, опишем метод субъектно-объектного взаимодействия в рамках ИПС для наиболее распространенной архитектуры КС. Для создания гарантированно защищенной КС (в которой выполняется заданная политика безопасности) достаточно:

1) убедиться в попарной корректности субъектов, замыкаемых в ИПС, и убедиться в корректности любого субъекта относительно МБО и МБС;

2) спроектировать и реализовать программно или программно-аппаратно МБС так, чтобы:

- для любого субъекта и любого объекта производился контроль порождения субъектов, т. е. чтобы реализация МБС соответствовала его определению;
- порождение любого субъекта происходило с контролем неизменности объекта-источника.

3) реализовать МБО в рамках априорно сформулированной политики безопасности.

Необходимо обратить внимание на следующее. $ОУ_{МБС}$, который является ассоциированным объектом МБС, играет решающую роль в проектировании ИПС. При возможности изменения состояния $ОУ_{МБС}$, с одной стороны, возможно «размыкание» программной среды, т. е. добавление к множеству разрешенных субъектов дополнительных, реализующих функции нарушителя. С другой стороны, процесс управления безопасностью подразумевает возможность изменения объекта управления. Возможность изменения объекта управления (реализация потока *stream* (*субъект управления*, *ассоциированный объект субъекта управления*) $\rightarrow ОУ_{МБС}$) должна присутствовать для выделенных субъектов (возможно с дополнительным условием активизации этого субъекта выделенным пользователем или пользователями).

Важную роль при проектировании ИПС играет свойство большинства КС, заключающееся в поэтапной активизации субъектов из объектов различного уровня представления информации.

В общем случае можно говорить о рекурсивной структуре объектов некоторого уровня, вмещающей объекты предыдущего уровня. С учетом иерархической структуры представления объектов можно говорить о том, что в начальные этапы активизации КС декомпозиция на субъекты и объекты динамически изменяется. Следовательно, основная теорема ИПС может быть применима только на отдельных интервалах времени, когда уровень представления объектов постоянен и декомпозиция фиксирована. Можно утверждать, что ИПС, действующую от момента активизации до момента окончания работы КС, невозможно сформировать в начальный момент активизации.

Пусть в КС выделяется конечное число уровней представления объектов, и R – максимальный уровень представления объекта.

С точки зрения выполнения условий леммы 10.2 имело бы смысл говорить о некотором «стационарном» состоянии КС, когда в отображениях «*Stream*» и «*Create*» участвуют только объекты уровня R. Тогда реализация МБС может быть значительно упрощена (в том смысле, что все аргументы-объекты операции *create* имеют уровень R).

Практическая реализация всех КС позволяет выделить две фазы их работы: активизация субъектов с ростом уровня представления объектов (фаза загрузки или начальная фаза) и фаза стационарного состояния (когда уровень представления объектов не увеличивается). Тогда реализация ИПС может состоять из двух этапов:

1. Предопределенное выполнение начальной фазы, включающее в себя момент активизации МБС и МБО (ступенчатая загрузка).
2. Работа в стационарной фазе в режиме ИПС с контролем неизменности объектов-источников.

Контрольные вопросы и задания

1. Почему в основе субъектно-ориентированной модели изолированной программной среды использовано дискреционное управление доступом?
2. Каковы основные рассматриваемые в субъектно-ориентированной модели изолированной программной среды способы реализации нарушителем запрещенных информационных потоков?
3. Почему для реализации ИПС необходимо требовать наличия в КС контроля порождения субъектов?
4. В чем отличие замкнутой программной среды от изолированной программной среды?
5. Как реализуется ступенчатая загрузка в современных ОС?

11. ОБЕСПЕЧЕНИЕ ЦЕЛОСТНОСТИ

11.1. КРИПТОГРАФИЧЕСКИЕ ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

По аналогии с моделями контроля доступа используются теоретические модели контроля целостности. В качестве примера рассмотрим модель, предложенную Кларком и Вилсоном. Другая, более известная, модель контроля целостности – это модель Биба, которую можно охарактеризовать как мандатную модель Белла – Лападулы, примененную для контроля целостности.

Модель Кларка – Вилсона появилась в результате проведенного ими анализа реально применяемых методов обеспечения целостности документооборота в коммерческих компаниях. В отличие от моделей Биба и Белла – Лападулы данная модель изначально ориентирована на нужды коммерческих заказчиков.

Основные понятия рассматриваемой модели – это корректность транзакций и разграничение функциональных обязанностей. Модель задает правила функционирования компьютерной системы и определяет две категории объектов данных и два класса операций над ними.

Все содержащиеся в системе данные подразделяются на контролируемые и неконтролируемые элементы данных (*constrained data items*, или *CDI*, и *unconstrained data items*, или *UDI*, соответственно). Целостность первых обеспечивается моделью Кларка – Вилсона. Последние содержат информацию, целостность которой в рамках данной модели не контролируется (этим и объясняется выбор терминологии).

Далее модель вводит два класса операций над элементами данных: процедуры контроля целостности (*integrity verification procedures, IVP*) и процедуры преобразования (*transformation procedures, TP*). Первые из них обеспечивают проверку целостности контролируемых элементов данных (*CDI*), вторые изменяют состав множества всех *CDI* (например, преобразуя элементы *UDI* в *CDI*).

Наконец, модель содержит девять правил, определяющих взаимоотношения элементов данных и процедур в процессе функционирования системы.

Правило С1. Множество всех процедур контроля целостности (*IVP*) должно содержать процедуры для контроля целостности любого элемента данных из множества всех *CDI*.

Правило С2. Все процедуры преобразования (*TP*) должны быть реализованы корректно в том смысле, что не должны нарушать целостность обрабатываемых ими *CDI*. Кроме того, с каждой процедурой преобразования должен быть связан список элементов *CDI*, которые допустимо обрабатывать данной процедурой. Такая связь устанавливается администратором безопасности.

Правило Е1. Система должна контролировать допустимость применения ТР к элементам *CDI* в соответствии со списками, указанными в правиле С2.

Правило Е2. Система должна поддерживать список разрешенных конкретным пользователям процедур преобразования с указанием допустимого для каждой *TP* и данного пользователя набора обрабатываемых элементов *CDI*.

Правило С3. Список, определенный правилом С2, должен отвечать требованию о разграничении функциональных обязанностей.

Правило Е3. Система должна аутентифицировать всех пользователей, пытающихся выполнить какую-либо процедуру преобразования.

Правило С4. Каждая *TP* должна записывать в журнал регистрации информацию, достаточную для восстановления полной картины каждого применения этой *TP*. Журнал регистрации – это специальный элемент *CDI*, предназначенный только для добавления в него информации.

Правило С5. Любая *TP*, которая обрабатывает элемент *UDI*, должна выполнять только корректные преобразования этого элемента, в результате которых *UDI* превращается в *CDI*.

Правило Е4. Только специально уполномоченное лицо может изменять списки, определенные в правилах С2 и Е2. Это лицо не имеет права выполнять какие-либо действия, если оно уполномочено изменять регламентирующие эти действия списки.

Роль каждого из девяти правил модели Кларка – Вилсона в обеспечении целостности информации можно проиллюстрировать, показав, каким из теоретических принципов политики контроля целостности отвечает данное правило. Ниже приведены первые шесть из сформулированных принципов:

1. Корректность транзакций.
2. Аутентификация пользователей.
3. Минимизация привилегий.
4. Разграничение функциональных обязанностей.
5. Аудит произошедших событий.
6. Объективный контроль.

Соответствие правил модели Кларка – Вилсона задано таблицей:

Правило	*Принцип*
C1	1, 6
C2	1
E1	3, 4
E2	1, 2, 3, 4
C3	4
E3	2
C4	5
C5	1
E5	4

Как видно из таблицы, принципы 1 (корректность транзакций) и 4 (разграничение функциональных обязанностей) реализуются большинством правил, что соответствует основной идее модели.

11.2. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Шифрование – это способ преобразования, применяемый для хранения важной информации в ненадежных источниках или передачи ее по незащищенным каналам связи. Согласно ГОСТ 28147–89, шифрование – процесс зашифрования или расшифрования.

Ключ шифрования – это секретная информация, используемая криптографическим алгоритмом при шифровании (расшифровании) сообщений. При использовании одного и того же алгоритма результат шифрования зависит от ключа. Длина ключа является основной характеристикой криптостойкости и измеряется в битах.

В зависимости от структуры используемых ключей методы шифрования подразделяются:

- на симметричное шифрование: для зашифрования и расшифрования используется один и тот же (секретный) ключ;
- асимметричное шифрование: для зашифрования используется один ключ (открытый), а для расшифрования – другой (закрытый, секретный). Данный вид шифрования также называют шифрованием с открытым ключом.

Различные виды шифрования обладают различной криптостойкостью.

11.3. КРИПТОГРАФИЯ С СЕКРЕТНЫМ КЛЮЧОМ

Симметричное шифрование – это способ шифрования, в котором для зашифрования и расшифрования применяется один и тот же криптографический ключ. Ключ алгоритма должен сохраняться в секрете, как отправителем, так и получателем сообщения. Ключ алгоритма выбирается сторонами до начала обмена сообщениями.

Симметричные шифры бывают следующих видов:

- блочные шифры – обрабатывают информацию блоками определенной длины (64, 128 бит и т. д.), применяя к блоку ключ в установленном порядке, как правило, несколькими циклами перемешивания и подстановки, называемыми раундами. Результатом повторения раундов является лавинный эффект – нарастающая потеря соответствия битов между блоками открытых и зашифрованных данных.
- поточные шифры, в которых шифрование проводится над каждым битом либо байтом исходного (открытого) текста с использованием гаммирования. Поточный шифр может быть легко создан на основе блочного, запущенного в специальном режиме, например, ГОСТ 28147–89 в режиме гаммирования. Гаммирование – это процесс «наложения» определенной ключевой последовательности («гаммы») на открытый текст. Например, это может быть операция «исключающего ИЛИ». При расшифровании операция проводится повторно с тем же ключом, в результате чего получается исходный текст.

Существует множество алгоритмов симметричных шифров, существенными параметрами которых являются: стойкость, длина ключа, число раундов, длина обрабатываемого блока, сложность аппаратной (программной) реализации, сложность преобразования.

Примерами симметричных алгоритмов служат AES (стандарт шифрования данных в США), ГОСТ 28147–89 (советский и российский стандарт шифрования, также является стандартом СНГ) и другие.

Можно провести сравнение симметричного шифрования с асимметричным по следующим параметрам:

Достоинства:

- скорость (на 3 порядка выше по данным Applied Cryptography);
- простота реализации (за счет более простых операций);
- меньшая требуемая длина ключа при сопоставимой стойкости.

Недостатки:

- сложность управления ключами в большой сети;
- сложность обмена ключами.

Для компенсации недостатков симметричного шифрования в настоящее время широко применяется комбинированная (гибридная) крипто-

графическая схема, где с помощью асимметричного шифрования передается *сеансовый ключ*, используемый обеими сторонами для дальнейшего обмена данными с помощью симметричного шифрования.

Существенным недостатком симметричных шифров является невозможность их использования для электронной подписи, так как ключ известен каждой стороне.

11.4. КРИПТОГРАФИЯ С ОТКРЫТЫМ (ОБЩИМ) КЛЮЧОМ

Асимметричное шифрование – это шифрование, при котором открытый ключ передается по открытому (т.е. незащищенному, доступному для наблюдения) каналу, и используется для шифрования посылаемого сообщения. Для расшифрования полученного сообщения используется другой, секретный ключ. Криптографические системы с открытым ключом в настоящее время широко применяются в различных сетевых протоколах, в частности, в протоколах TLS и его предшественнике SSL (лежащих в основе HTTPS), в SSH. Также используется в протоколах PGP, S/MIME.

Пусть K – пространство ключей, а e и d – ключи шифрования и расшифрования соответственно. Введем функцию шифрования E_e для произвольного ключа $e \in K$. Этой функцией зашифруем сообщение m и получим шифротекст c: $E_e(m) = c$. Здесь $c \in C$, где C – пространство криптограмм, а $m \in M$, где M – пространство сообщений. Теперь введем функцию расшифрования D_d, с помощью которой можно найти сообщение m, зная шифротекст c: $D_d(c) = m$.

Рассмотрим множество ключей $\{E_e: e \in K\}$ шифрования, и пусть $\{D_d: d \in K\}$ – соответствующее множество ключей для расшифрования. Рассмотрим пару (E, D) и предположим, что каждая пара имеет следующее свойство: зная E_e, неосуществимо решить уравнение $E_e(m) = c$ относительно m, т. е. для данного произвольно выбранного шифротекста $c \in C$ неосуществимо найти сообщение $m \in M$. Это значит, что по данному e невозможно определить соответствующий ключ расшифрования d.

Учитывая сделанные выше предположения, рассмотрим передачу информации от некоторого отправителя A получателю B (A и B здесь могут быть как физическими лицами, так и организациями и т. д.):

1. A выбирает пару (e, d) и посылает ключ шифрования e (открытый ключ) B по открытому каналу, а ключ расшифрования d (закрытый ключ) защищен и секретен (он не должен передаваться по открытому каналу).

2. Чтобы послать сообщение m для A, B применяет функцию шифрования, определенную открытым ключом e: $E_e(m) = c$, c – полученный шифротекст. В качестве сообщения m может, в т. ч. передаваться *сеансовый ключ*, используемый в дальнейшем обеими сторонами для совместного обмена данными.

3. A расшифровывает шифротекст c, применяя обратное преобразование D_d, однозначно определенное значением d.

Идея криптографии с открытым ключом тесно связана с идеей односторонних функций. Односторонняя функция (англ. one-wayfunction) – это функция, которая эффективно вычисляется за полиномиальное время на детерминированной машине Тьюринга, но не существует полиномиальной вероятностной машины Тьюринга, которая обращает эту функцию с более чем экспоненциально малой вероятностью. Следовательно, зная x, легко вычислить $f(x)$, напротив, зная $f(x)$, неосуществимо получить x.

Но сама односторонняя функция бесполезна в применении: ею можно зашифровать сообщение, но расшифровать нельзя. Поэтому криптография с открытым ключом использует односторонние функции с лазейкой. В математических терминах, если f – функция с лазейкой, то существует некоторая секретная информация y, такая, что для заданных $f(x)$ и y легко вычислить x. Рассмотрим навесной замок с ключом. Перевести замок с открытого на закрытый без использования ключа тривиально, вставив дужку в замок. Легкое открытие замка требует использования ключа. Здесь ключ – лазейка, а замок – функция с лазейкой.

Начало асимметричным шифрам было положено в 1976 г. в работе У. Диффи и М. Хеллмана «Новые направления в современной криптографии». Они предложили систему обмена общим секретным ключом на основе проблемы дискретного логарифма. Вообще, в основу известных асимметричных криптосистем кладется одна из сложных математических проблем, которая позволяет строить односторонние функции и функции-ловушки. Например, криптосистема Ривеста – Шамира – Адлемана (RSA) использует проблему факторизации больших чисел, а криптосистемы Меркля – Хеллмана и Хора – Ривеста опираются на так называемую задачу об укладке рюкзака.

Ниже приведены принципы построения криптосистем с открытым ключом.

1. Вначале необходимо выбрать трудную задачу P. Она должна решаться сложно в смысле теории: нет алгоритма, с помощью которого можно было бы перебрать все варианты решения задачи P за полиномиальное время относительно порядка входных данных.

2. Можно выделить легкую подзадачу P' из P. Она должна решаться за полиномиальное время, лучше, если за линейное.

3. «Перетасовываем и взбалтываем» P', чтобы получить задачу P, совершенно не похожую на первоначальную. Задача P, по крайней мере, должна выглядеть как оригинальная труднорешаемая задача P.

4. P открывается с описанием, как она может быть использована в роли ключа зашифрования. Как из P получить P', держится в секрете как секретная лазейка.

5. Криптосистема организована так, что алгоритмы расшифрования для легального пользователя и криптоаналитика существенно различны. В то время как первый решает задачу P, второй использует секретную лазейку и решает задачу P'.

Казалось бы, что криптосистема с открытым ключом – идеальная система, не требующая безопасного канала для передачи ключа шифрования. Это подразумевало бы, что два легальных пользователя могли бы общаться по открытому каналу, не встречаясь, чтобы обменяться ключами. К сожалению это не так. Рис. 11.1 иллюстрирует, как нарушитель может захватить систему (расшифровать сообщение, предназначенное стороне A) без взламывания системы шифрования.

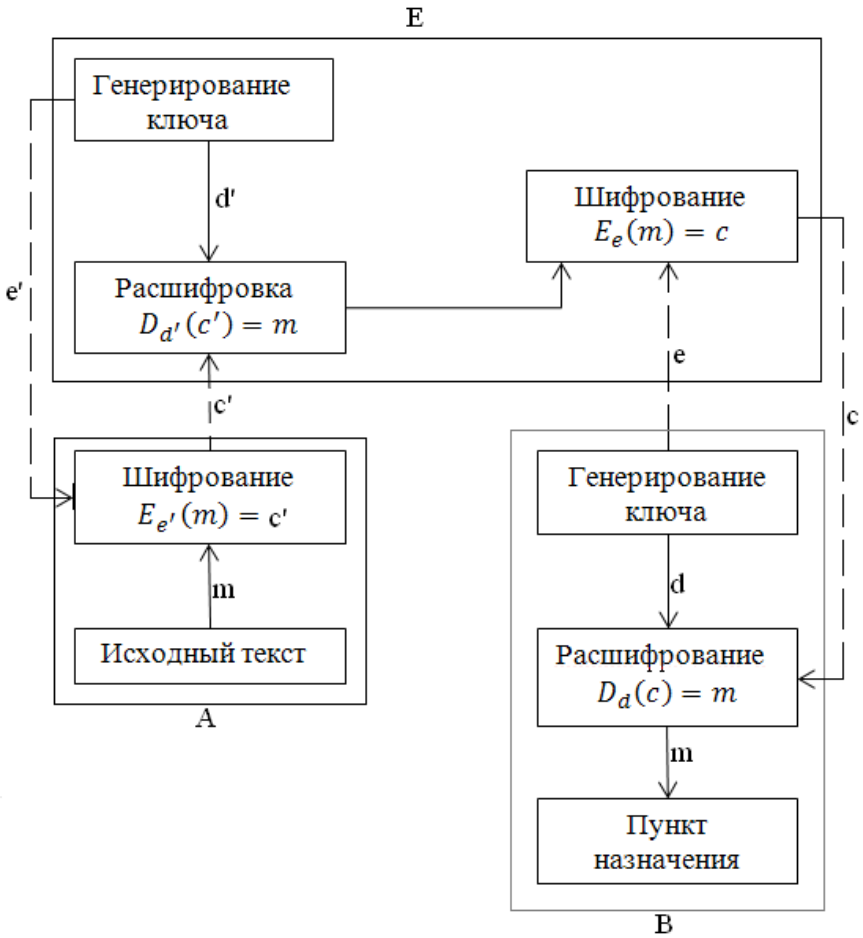

Рис. 11.1. Схема дешифрования сообщения

В этой модели нарушитель перехватывает открытый ключ e, посланный стороной A для стороны B. Затем создает пару ключей e' и d', «маскируется» под A, посылая B открытый ключ e', который, как думает B, открытый ключ, посланный ей A. Нарушитель перехватывает зашифрованные сообщения от B к A, расшифровывает их с помощью секретного ключа d', заново зашифровывает открытым ключом e стороны A и отправляет сообщение к A. Таким образом, никто из участников не догадывается, что есть третье лицо, которое может как просто перехватить сообщение m, так и подменить его на ложное сообщение m'. Такой тип атаки называется «человек посередине». Ее существование подчеркивает необходимость аутентификации открытых ключей. Для этого обычно используют сертификаты.

Также криптосистему с открытым ключом (RSA) можно было бы взломать с помощью атаки «грубой силы». К примеру, с помощью алгоритма Шора при использовании достаточно мощного квантового компьютера (в настоящее время не существует). Однако для создания криптосистемы с открытым ключом может быть выбран другой криптографический алгоритм, для которого использование алгоритма Шора неэффективно. В криптосистеме с открытым ключом используют односторонние функции. Сложность вычислений таких функций не является линейной от количества битов ключа, а возрастает быстрее, чем ключ. Поэтому ключ должен быть достаточно большим, чтобы сделать лобовую атаку невозможной, что значительно замедляет процесс шифрования.

Еще одна форма атаки – вычисление закрытого ключа по известному открытому ключу (рис. 11.2). Криптоаналитик знает алгоритм шифрования E_e и по нему пытается найти D_d.

Этот процесс упрощается, если криптоаналитик перехватил несколько шифрованных текстов c, посланных A для B, если криптосистема с открытым ключом основана на проблеме факторизации больших чисел. К примеру, RSA использует в качестве части открытого ключа большое бипростое число n. Сложность взлома такого алгоритма состоит в трудности разложения числа n на множители.

Для многих методов асимметричного шифрования криптостойкость, полученная в результате криптоанализа, существенно отличается от величин, заявляемых разработчиками алгоритмов на основании теоретических оценок. Поэтому во многих странах вопрос применения алгоритмов шифрования данных находится в поле законодательного регулирования. В России к использованию в государственных и коммерческих организациях разрешены только те программные средства шифрования данных, которые прошли государственную сертификацию в органах ФСБ.

Рис. 11.2. Вариант атаки на криптосистему с открытым ключом

Преимущества асимметричного шифрования перед симметричными шифрами состоят:

- в отсутствии необходимости предварительной передачи секретного ключа по защищенному каналу;
- простоте реализации распределения ключей;
- отсутствии необходимости обновлять ключ после каждой передачи, так как в асимметричных криптосистемах пару (*E, D*) можно не менять значительное время;
- в том, что число ключей в криптосистеме с открытыми ключами значительно меньше, чем в симметричной.

Недостатки асимметричного шифрования перед симметричными шифрами состоят в том, что хотя сообщения надежно шифруются, получатель и отправитель «засвечиваются» самим фактом пересылки шифрованного сообщения;

- несимметричные алгоритмы используют более длинные ключи, чем симметричные;
- процесс шифрования-расшифрования с использованием пары ключей проходит на два-три порядка медленнее, чем шифрование-расшифрование того же текста симметричным алгоритмом;
- в чистом виде асимметричные криптосистемы требуют существенно больших вычислительных ресурсов, поэтому на практике используются только для обмена секретным ключом.

11.5. ХЕШИРОВАНИЕ

Хеширование (англ. *hashing*) – это преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины. Такие преобразования также называются хеш-функциями или функциями свертки, а их результаты называют хешем или хеш-кодом. Простейшим примером хеш-функции может служить контрольная сумма или CRC.

В общем случае однозначного соответствия между исходными данными и хеш-кодом нет. Поэтому существует множество массивов данных, дающих одинаковые хеш-коды – так называемые коллизии. Вероятность возникновения коллизий играет немаловажную роль в оценке качества хеш-функций.

Среди множества существующих хеш-функций принято выделять криптографически стойкие, применяемые в криптографии. Криптостойкая хеш-функция прежде всего должна обладать стойкостью к коллизиям двух типов:

- первого рода: для заданного сообщения невыполнимо подобрать другое сообщение, имеющее такой же хеш (необратимость хеш-функции).
- второго рода: невыполнимо найти пару сообщений, имеющих одинаковый хеш.

Простейшим (хотя и не всегда приемлемым) способом усложнения поиска коллизий является увеличение разрядности хэша, например, путем параллельного использования двух или более различных хеш-функций.

Для криптографических хеш-функций также важно, чтобы при малейшем изменении аргумента значение функции сильно изменялось. В частности, значение хеша не должно давать утечки информации даже об отдельных битах аргумента. Это требование является залогом криптостойкости алгоритмов шифрования, хеширующих пользовательский пароль для получения ключа.

11.6. ЭЛЕКТРОННАЯ ПОДПИСЬ

Электронная подпись (ЭП) или электронная цифровая подпись – это реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной цифровой подписи и позволяющий идентифицировать

владельца ключа подписи, установить отсутствие искажения информации в электронном документе, а также обеспечить невозможность отказа владельца ключаот подписи.

Схема электронной подписи обычно включает в себя:

- алгоритм генерации ключевых пар пользователя;
- функцию вычисления подписи;
- функцию проверки подписи.

Функция вычисления подписи на основе документа и секретного ключа пользователя вычисляет собственно подпись. В зависимости от алгоритма функция вычисления подписи может быть детерминированной или вероятностной. Детерминированные функции всегда вычисляют одинаковую подпись по одинаковым входным данным. Вероятностные функции вносят в подпись элемент случайности, что усиливает криптостойкость алгоритмов ЭП. Однако для вероятностных схем необходим надежный источник случайности (либо аппаратный генератор шума, либо криптографически надежный генератор псевдослучайных бит), что усложняет реализацию.

В настоящее время детерминированные схемы практически не используются. Даже в изначально детерминированные алгоритмы сейчас внесены модификации, превращающие их в вероятностные.

Функция проверки подписи проверяет, соответствует ли данная подпись данному документу и открытому ключу пользователя. Открытый ключ пользователя доступен всем, так что любой может проверить подпись под данным документом.

Поскольку подписываемые документы – переменной (и достаточно большой) длины, в схемах ЭП подпись ставится не на сам документ, а на его хеш. Для вычисления хеша используются криптографические хеш-функции, что гарантирует выявление изменений документа при проверке подписи. Хэш-функции не являются частью алгоритма ЭП, поэтому в схеме может быть использована любая надежная хэш-функция.

Алгоритмы ЭП делятся на два больших класса: обычные цифровые подписи и цифровые подписи с восстановлением документа. Обычные цифровые подписи необходимо пристыковывать к подписываемому документу. К этому классу относятся, например, алгоритмы, основанные на эллиптических кривых (ECDSA, ГОСТ Р 34.10–2001, ДСТУ 4145–2002). Цифровые подписи с восстановлением документа содержат в себе подписываемый документ: в процессе проверки подписи автоматически вычисляется и тело документа. К этому классу относится алгоритм RSA.

Цифровая подпись обеспечивает:

- удостоверение источника документа:в зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесенные изменения», «метка времени» и т. д.;

- защиту от изменений документа: при любом случайном или преднамеренном изменении документа (или подписи) изменится хеш, следовательно, подпись станет недействительной;
- невозможность отказа от авторства: так как создать корректную подпись можно лишь, зная закрытый ключ, а он известен только владельцу, то владелец не может отказаться от своей подписи под документом;
- предприятиям и коммерческим организациям сдачу финансовой отчетности в государственные учреждения в электронном виде;
- организацию юридически значимого электронного документооборота.

Возможны следующие угрозы электронной подписи:

- злоумышленник может попытаться подделать подпись для выбранного им документа;
- злоумышленник может попытаться найти документ по данной подписи. Однако в подавляющем большинстве случаев такой документ может быть только один.

11.7. СЕРТИФИКАТЫ

Обычно при использовании электронной подписи открытый ключ проверки ЭП передается упакованным в специальный контейнер, в котором, помимо ключа, находится также информация о том, кому этот ключ выдан, кем, для каких целей, какой алгоритм генерации использовался, до какого срока он может применяться и т. п. Вся эта информация математически связана с ключом и изменена быть не может. Такой контейнер называется сертификатом.

Существуют стандарты на сертификаты, принятые ITU (Международный союз телекоммуникаций). В настоящее время действующий стандарт – X.509. Наиболее распространенные варианты сертификатов – X.509v1 и X.509v3.

Не составляет технических проблем поместить сертификат во внешнее аппаратной устройство. Большинство современных устройств доступа – это фактически сертификат, помещенный внутрь микросхемы. Примеры таких устройств – E-Tokens, смарт-карты и т. п.

В каждом органе сертификации существует специальный список отозванных (недействительных) сертификатов (*certificate revocationlist*). В нем тот, кто пользуется сертификатами этого центра сертификации (*certificate authority*), может проверить валидность того или иного сертификата.

Контрольные вопросы

1. Что такое хеширование?
2. Чем архитектура электронной цифровой подписи отличается от архитектуры шифрования с открытым ключом?
3. Что такое сертификат?
4. В чем преимущества асимметричного шифрования перед симметричным?
5. В чем различие между открытым и закрытым ключом в алгоритмах асимметричного шифрования?

12. ОПРЕДЕЛЕНИЕ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ

12.1. ЭКСПЕРИМЕНТАЛЬНЫЙ ПОДХОД

Экспериментальный подход заключается в проведении атаки и проверки, осуществилась атака или нет. Приведем некоторые определения.

Атака – реализация угрозы безопасности. Атаки не описаны ни в моделях безопасности, ни в стандартах. Невозможно перечислить все способы и реализации атак.

Уязвимость – слабое место в информационной системе или ошибка в программном коде, которая может привести к нарушению безопасности, путем осуществления то или иной угрозы. Уязвимость позволяет реализовать угрозу без знаний о средствах защиты реальной системы. Процесс появления новых уязвимостей бесконечен.

Защита реальных систем не может быть безупречной. Чтобы найти уязвимость, необязательно знать детали реализации атакуемой системы.

Эксплойт – термин, служащий для обозначения фрагмента программного кода, который использует возможности предоставляемой ошибки, уязвимости, и ведет к повышению привилегий, выполнению произвольных команд, или отказу в обслуживании компьютерной системы.

Безопасность (на практике) – успешное противостояние атакам. Система безопасна, если не содержит уязвимостей.

Система обычно детерминирована, т. е. все доступные варианты атаки определяются текущим состоянием. Поэтому некоторые атаки можно пробовать гипотетически.

Для реализации большинства атак используют уязвимости, но бывают атаки и без уязвимостей. Например перебор (bruteforce) – универсальная атака, возможная без каких либо знаний о системе. Универсальные атаки имеют меньше шансов на успех. Поэтому мы можем заменить попытки осуществления атак на проверку свойств нашей системы и свойств систем, для которых данные атаки проходят. Типичная уязвимость – уязвимость, использующая переполнение буфера.

Способы борьбы с переполнением включают запрет выполнения стека (Intel); сохранение в стеке случайного количества параметров (но при этом нельзя узнать, где находится точка возврата); установку границ буфера с помощью некоторого случайного числа.

Уязвимости появляются вследствие ошибок проектирования, реализации и конфигурирования. При этом не все ошибки могут приводить к уязвимостям.

Уязвимости могут быть классифицированы, например, следующим образом:

- по этапам разработки и эксплуатации: проектирования, реализации, конфигурирования;
- по расположению: в стеке, в куче и пр.

Очень важно, где допущена ошибка: в средствах зашиты или в каком-либо другом компоненте системы. Любая ошибка в системе защиты – уязвимость. Типичная уязвимость в средствах защиты приводит к возможности реализации троянского коня. Уязвимости не в средствах защиты могут привести к реализации некоторой угрозы.

12.2. МОДЕЛЬ ПРОТИВОСТОЯНИЯ УГРОЗ И СРЕДСТВ ЗАЩИТЫ

Уязвимость возникает по двум причинам: ошибки в средствах защиты; или отсутствие информации об угрозе или предположение о том, что данной угрозы не будет в системе. Нарушитель может использовать угрозы, которые не были учтены при проектировании средств защиты. Иначе говоря, нарушитель должен найти ошибки в обоснованиях угроз, от которых защищают средства защиты.

Непосредственные ошибки в реализации средства защиты являются ошибками первого рода, а неправильные утверждения о функциях средств защиты – ошибками второго рода.

Как обеспечить и как оценивать безопасность?

Оценивать безопасность можно по количеству уязвимостей.

Борьба с ошибками в средствах защиты заключается: в возможном уменьшении объема кода средств защиты, усилении тестирования, а также сертификации. Для сокращения уязвимостей в программном обеспечении (ПО) необходимо сокращать функциональные возможности ПО, контролировать их с помощью средств защиты или делать так, чтобы на них нельзя было воздействовать (рис. 12.1).

Можно сказать, что уязвимость – это ошибка, приводящая к нарушению функционального баланса между средствами защиты и другим программным обеспечением.

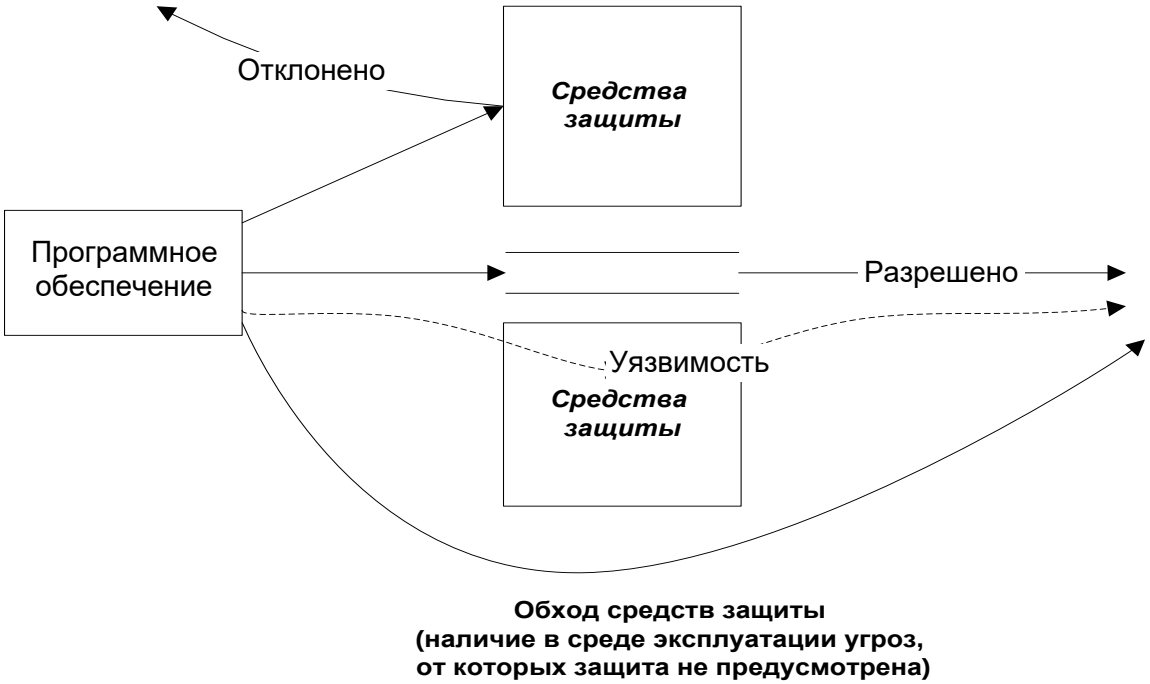

Рис. 12.1 Противостояние угроз и средств защиты

Оценка безопасности системы заключается в поиске новых и проверке известных уязвимостей.

Обеспечение безопасности системы состоит: в контроле и своевременном исправлении ошибок; минимизации объема кода средств защиты; контроле над тем, чтобы не было ПО, которое не контролировалось бы средствами защиты; уменьшении привилегий программ.

Предусловия – логический предикат, который истинен до выполнения процедуры. Постусловия – логический предикат, который истинен после выполнения процедуры. Для доказательства корректности работы программ строятся формальные логические выводы на основе предусловий и постусловий.

12.3. ПРИМЕР УЯЗВИМОСТИ: УЯЗВИМОСТЬ В ДРАЙВЕРАХ

Практически драйвер может прочитать файл, который пользователь прочитать не может. При проектировании ОС Windows эти угрозы считались несущественными, так как в компании Microsoft считали, что разрабатывать драйвера будут либо сотрудники Microsoft, либо производители

устройств. Проблема заключается в том, что в драйвере может оказаться ошибка, которая позволит нарушителю заменить код драйвера, или выполнить произвольный код.

Решение состоит в запрете модификации компонентов. Верификацию того, что драйвер не несет в себе вредоносных функций, часто заменяют на проверку цифровой подписи.

Это возможно осуществить только до некоторого уровня, так как драйвера необходимо устанавливать, или обновлять.

Лучше не выдвигать предположений о корректной работе драйвера и включать драйверы в комплекс средств защиты. При включении драйвера в средства защиты ошибки первого рода становятся ошибками второго рода.

Корректность работы средств защиты зависит от выбранной доверенной среды. К уязвимостям в средствах защиты может привести любая ошибка. Данный подход представлен на рис. 12.2.

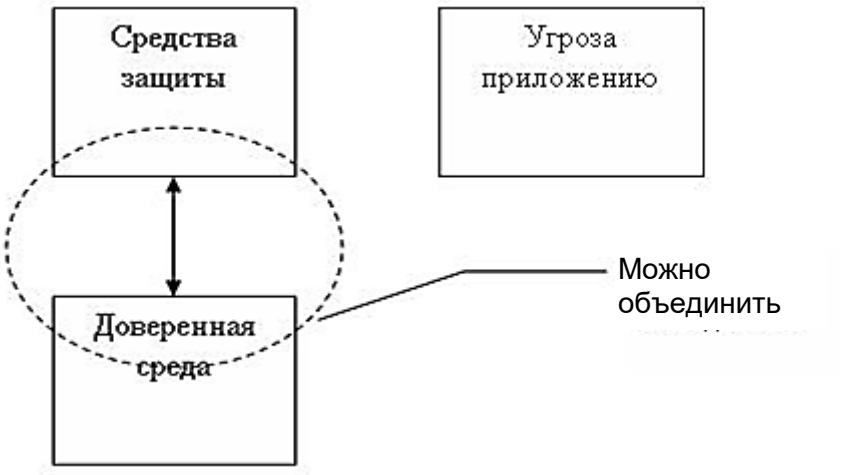

Рис. 12.2. Экспериментальный подход

Рис. 12.3. Модификация экспериментального подхода

Следовательно, во избежание ошибок необходимо минимизировать средства защиты, в том числе, максимально перенести функциональность из драйвера в приложение (рис. 12.3).

12.4. ДОСТОИНСТВА И НЕДОСТАТКИ ЭКСПЕРИМЕНТАЛЬНОГО ПОДХОДА

Как было указано выше, на практике безопасность системы проще всего оценивать по наличию уязвимостей. Иными словами, чем больше уязвимостей, тем менее система безопасна. Можно проверять наличие существующих уязвимостей, так как поиск новых уязвимостей является дорогостоящим. Подход, основанный на оценке уязвимостей, с точки зрения практики является эффективным, так как позволяет оценивать безопасность здесь и сейчас. Однако необходимо понимать, что свойства продукта при этом не исследуются, вместо этого проверяется наличие уязвимостей. Кроме того, так как мы не знаем всех уязвимостей, существование некоторых уязвимостей не может быть обнаружено, следовательно, оценка безопасности продукта в данном случае является неполной.

Таким образом, экспериментальный подход хорошо работает, только если существуют сведения об актуальных уязвимостях. При этом оценка безопасности должна быть постоянной, в отличие от теоретических моделей.

Наличие или отсутствие уязвимостей ничего не говорит о функциональных возможностях продукта. В этом подходе распространяется информация об уязвимостях, следовательно, возможно использование этой информации для деструктивных действий. При этом неясно, какому стандарту удовлетворяет продукт, какая модель в нем заложена.

Контрольные вопросы

1. В чем заключается экспериментальный подход к определению безопасности информационных систем?
2. Каковы преимущества и недостатки экспериментального подхода?
3. Каковы причины возникновения уязвимостей?
4. Что такое атака?
5. Что такое эксплойт?

13. ОЦЕНКА РИСКОВ

13.1. ПОНЯТИЕ ОЦЕНКИ РИСКОВ

Оценка рисков нарушения безопасности информационных систем является неотъемлемой частью процесса обеспечения информационной безопасности. Информационные системы зачастую являются важнейшим звеном в цепи процессов функционирования коммерческих организаций, и нарушения информационной безопасности зачастую приводят к серьезным потерям.

Риск – вероятность того, что угроза через уязвимость (уязвимости) ресурсов приведет к потере или повреждению ресурсов.

Угроза – любое событие, действующая сила или стечение обстоятельств, способное оказывать негативное воздействие на систему.

Уязвимость – недостаток (слабая сторона) ресурса или группы ресурсов, которая может быть использована для осуществления угрозы.

Информационный риск – вероятность причинения ущерба организации в результате нарушения конфиденциальности, целостности или доступности информации.

Анализ рисков (*riskanalysis*) – процесс идентификации рисков безопасности, определения их величины и источников.

Определение риска (*riskevaluation*) – процесс сопоставления предполагаемого риска с заданными критериями оценки риска для определения значимости риска.

Оценка рисков (*riskassessment*) – процесс анализа и определения рисков.

Управление рисками (*riskmanagement*) – процесс сопоставления идентифицированных и оцененных рисков, выгод/потерь при применении (неприменении) мер безопасности и выработка политики безопасности и стратегии дальнейшего обеспечения безопасности, отвечающего задачам организации.

Управление рисками – процесс идентификации и оценки рисков, а также принятия мер для снижения уровня риска до приемлемого уровня.

Меры безопасности – деятельность, процедура или механизм, снижающий степень риска.

Остаточный риск – риск, сохранившийся после применения мер безопасности.

Оценка рисков как этап обеспечения безопасности получила распространение с развитием опасных производств химической и атомной промышленности в середине прошлого века. Однако «классические» методы оценки рисков (FMEA, FTA, HAZOP) плохо применимы к информационным системам, в силу того, что современные информационные системы, помимо прочего:

- превосходят по масштабам и сложности любые технологические процессы;
- зачастую содержат компоненты, удаленные на значительные расстояния.
- подвержены широкому спектру информационных воздействий, характеризующихся высокой скоростью, интенсивностью и трудностью определения вредоносного воздействия среди прочих.

Принципиально другой подход к оценке рисков – экономический. Данный подход направлен на денежную оценку возможных потерь ресурсов. Но и в экономическом ключе информационные риски трудно оценивать в комплексе в силу нематериальной ценности целого ряда составляющих информационной системы, возможно рассмотрение лишь отдельных аспектов (например анализ затрат и выгод).

Данные обстоятельства привели к появлению ряда специализированных методик оценки рисков информационной безопасности. Существующие методики направлены на адаптацию существующих методов оценки рисков к задачам информационной безопасности и не ограничиваются собственно оценкой рисков, рассматривая весь процесс управления рисками.

Начальный этап *процесса управления рисками* направлен на определение границ исследования с целью минимизации излишних действий (установление границ исследования).

Анализ рисков, как будет рассмотрено в дальнейшем, включает в себя определение ресурсов, угроз, уязвимостей и мер безопасности (анализ рисков).

Определение рисков направлено на получение значения степени подверженности ресурса риску (определение рисков).

На данном этапе определяются меры безопасности, направленные на снижение риска (выбор мер безопасности).

На данном этапе рассматривается вопрос о приемлемости риска после применения мер безопасности (принятие риска).

Политика безопасности, полученная в результате данного этапа, содержит детальное описание, и обоснование необходимых мер безопасности (составление политики безопасности).

На финальном этапе происходит составление итогового заключения по проведенному исследованию. Результатом является характеристика

исследованной системы, и рекомендации по обеспечению безопасности системы на основе полученных данных (составление плана безопасности).

Под ресурсом понимается материальное или нематериальное имущество, которому организация-владелец присваивает финансовую ценность и обеспечивает его безопасность – все, что имеет ценность для организации, эксплуатирующей информационную систему: не только сами данные (информация) и элементы информационной системы, напрямую подверженные информационным угрозам, но и персонал, и активы организации, в том числе и активы нематериальные, такие как имидж и доверие заинтересованных сторон.

Результатом данного этапа является перечисление всевозможных ресурсов системы (идентификация ресурсов).

После идентификации и перечисления ресурсы подлежат определению и оценке. Оценка ресурса зависит от применяемого метода оценки рисков. Связанные между собой ресурсы оцениваются в зависимости от степени их связи и ценности остальных ресурсов.

В результате данного этапа должны быть перечислены все возможные ресурсы и значения их ценности по ее составляющим (определение или классификация ресурсов и зависимостей между ними).

Процесс определения и классификации угроз включает в себя этапы определения источника и объекта угрозы, а также возможности и вероятности осуществления угрозы. При этом должны приниматься в расчет природа угрозы, и дополнительные факторы, различные для различных типов угроз (географические для природных, социальные для злонамеренных и т. п.).

В результате данного этапа должен быть представлен список возможных угроз, ресурсов, им подверженных, и вероятность осуществления угроз по отношению к ресурсу (определение угроз).

Определение уязвимостей включает в себя обнаружение и классификацию слабостей в защите всех прямых и косвенных элементов информационной системы с учетом тесной связи самих защищаемых объектов и систем защиты.

Результат – возможность реализации угрозы для конкретного ресурса через конкретную уязвимость (определение уязвимостей).

На данном этапе производится определение существующих или планируемых мер безопасности, и проверка их на совместимость. Результат – список всех мер безопасности, их осуществления и текущего состояния (определение мер безопасности).

Цель следующего этапа (определение рисков) – получить оценку значения риска, которому подвергается информационная система и ее ресурсы, для дальнейшего использования этой информации для принятия мер безопасности.

Для сбора сведений об информационной системе на разных этапах процесса оценки рисков могут использоваться следующие методы:

- Вопросники. Они могут касаться, прежде всего, административных и процедурных регуляторов безопасности, существующих или планируемых. Вопросники распространяются среди административного и технического персонала, проектирующего и/или обслуживающего систему.
- Интервью. Беседы проводятся с административным и техническим персоналом и концентрируются на темах эксплуатации и управления.
- Просмотр документации. Политика безопасности, нормативные документы, техническая документация, предыдущие отчеты по оценке рисков, оценка критичности ресурсов, результаты аудита и тестирования, планы безопасности и т.п. – ценный источник информации о существующих и планируемых мерах безопасности, о степени критичности и чувствительности систем и данных.
- Применение инструментов автоматического сканирования. Программные и аппаратные инструменты автоматического обнаружения и анализа защищенности, позволяют эффективно собирать системную информацию, строить карту информационной системы, получать профили отдельных узлов системы и подсистем.

13.2. ОПРЕДЕЛЕНИЕ УЯЗВИМОСТЕЙ И ОЦЕНКА РИСКОВ

Идентификация уязвимостей может выполняться с помощью средств анализа защищенности и путем анализа общедоступных источников соответствующей информации. В специфических случаях требуется привлечение специальных знаний об особенностях системы и ее конфигурации.

Определение уязвимостей включает нахождение слабых сторон окружающей среды, самой организации, технических и административных процедур, персонала, руководства, аппаратного, программного обеспечения и линий связи информационной системы которые могут быть использованы источником угрозы для причинения вреда ресурсам и организации, ими владеющей.

Однако присутствие уязвимости само по себе не является источником опасности, так как для нанесения ущерба системе должна существовать угроза, использующая данную уязвимость. Однако уязвимости, которым не соответствует ни одной известной угрозы, также должны контролироваться.

При определении уязвимостей применяется составление деревьев (графов) атак. Граф (дерево) атаки – это направленный граф, вершины которого соответствуют стадиям атаки, а ребра – переходам между стадиями; с каждым ребром связывается время, требующееся на успешное проведение очередной стадии атаки либо затраты, необходимые злоумышленнику для преодоления очередной ступени защиты.

Категоризация уязвимостей каждого элемента системы по типу атаки необходимо для того, чтобы связать с каждым ребром графа атаки набор уязвимостей, делающих переход по данному ребру возможным. Дерево атак может быть использовано и для количественного анализа системы, если связать с его узлами/ребрами/поддеревьями определенные количественные показатели.

В оценке рисков ключевую роль играет собственно анализ рисков как процесс сопоставления найденных компонентов риска с целью нахождения практически применимых характеристик риска. Анализ конкретного риска для данного ресурса или группы ресурсов может сводиться к определению в некоторой форме степени подверженности ресурса риску либо качественной (количественной) характеристике возможного ущерба.

Анализ рисков в целом оперирует значениями:

- ценность ресурса (*AssetValue, AV*);
- вероятность осуществления угрозы (*Likelihood, L*);
- величина воздействия на ресурс (*Impact, I*);
- показатель уязвимости (*Vulnerability, V*);
- величина угрозы (*Threat, T*);
- меры безопасности (*Safeguards, S*).

Качественный анализ рисков ставит задачу получения некоторого сравнительного значения величины подверженности риску данного элемента системы среди прочих, что позволило бы в дальнейшем выработать оптимальную стратегию минимизации риска.

Входные данные для качественного анализа определяются при сопоставлении необходимого параметра используемой шкале оценки. На практике лица, ответственные за обеспечение функционирования элемента системы или ресурса на различных уровнях выбирают наиболее точно описывающее данный элемент/ресурс значение параметра.

Результат качественного анализа рисков – численное значение – приоритет (относительная величина) риска имеет смысл лишь при сопоставлении его по качественной шкале аналогичным параметрам для других частей системы. Так, например, величина риска может составлять от 0 до 100 баллов, где значение от 0 до 10 баллов принимается «низким», от 10 до 25 – «средним», а от 25 до 100 – «высоким», и элементы

с высоким риском признаются приоритетными при принятии мер по снижению риска.

Для расчетов при качественном анализе широко применяются матрицы рисков. Матрица рисков представляет собой таблицу для функции двух аргументов, где координатные оси соответствуют входным параметрам, а границы ячеек – используемым шкалам значений входных параметров. Результирующее значение для каждой ячейки оценивается по заданной шкале.

В матрицах рисков аналогичным способом могу вычисляться и другие показатели – критичность и т. д.

Количественный анализ рисков ставит задачу количественной, а именно, финансовой оценки защищаемых ресурсов и их возможных потерь от негативного воздействия на систему.

Основным фактором количественного анализа рисков является ценность ресурса (*AV*). Под ценностью ресурса понимается его финансовая ценность. В случае невозможности прямого определения финансовой ценности ресурса (например, нематериального – информация, сервисы, услуги), рассчитывается его косвенная стоимость, с учетом специфических характеристик ресурса, таких как конфиденциальность (*confidentiality, C*), доступность (*availability, A*), целостность (*integrity, I*), подотчетность (*accountability*, A_c), возможность проведения аудита (*auditability*, A_u).

Следующим фактором является фактор подверженности угрозе (*Exposure Factor, EF*). Это – часть ресурса, подверженная повреждению конкретным риском.

Далее определяется ожидаемая единичная потеря (*Single Loss Expectancy, SLE*), которая представляет ожидаемые денежные потери в каждом случае возникновения риска:

$$SLE = AV * EF.$$

Затем следует вероятность появления риска за год (*Annualized Rate of Occurrence, ARO*).

Далее рассчитываются ожидаемые потери за год (*Annualized Loss Expectancy, ALE*):

$$ALE = SLE * ARO.$$

Количественный анализ оперирует понятиями, входящими также в сферу качественного анализа, такими, как фактор подверженности угрозе.

13.3. МЕТОДЫ С ИСПОЛЬЗОВАНИЕМ ДЕРЕВЬЕВ

Анализ дерева отказов (*fault tree analysis, FTA*) производится с помощью построения дерева отказов на основе полученной информации о системе.

Дерево отказов – это представление в виде направленного графа последовательности всевозможных взаимосвязанных событий, осуществление которых может привести к нарушению функционирования системы. В корне дерева находится результирующее событие – отказ системы, в листьях – исходные события. Взаимосвязь между событиями, приводящая (не приводящая) к переходу на следующий уровень реализуется логическими функциями (в основном И/ИЛИ/исключающее ИЛИ и т. п.).

С ветвями дерева отказов могут быть связаны качественные характеристики – например, приоритет угрозы, развивающейся по данному направлению, или тяжесть ее последствий, или вероятность ее дальнейшего распространения.

Метод анализа дерева отказов универсален и позволяет при наличии подробной информации о системе получить наглядную картину уязвимости системы.

Анализ дерева событий (ETA) подразумевает построение дерева событий на основе анализа всех возможных событий, происходящих в системе. В отличие от деревьев отказов деревья событий используют индуктивную логику.

Деревья отказов могут быть использованы для анализа систем, в которых все или некоторые компоненты находятся в непрерывном функционировании. Инициирующее событие (стартовая точка для дерева событий) нарушает нормальную работу системы, и дальнейшие события развиваются по нескольким возможным сценариям, отраженным в дереве, зависящим от успехов/неудач в работе компонентов системы.

13.4. МЕРЫ БЕЗОПАСНОСТИ

Для успешного снижения уровня риска необходимы корректные, обоснованные и рациональные меры безопасности. Существующие и планируемые меры безопасности должны приниматься в расчет наряду со структурой системы обеспечения безопасности и ограничениями временного, финансового и прочего характера.

Показательные характеристики рисков, определенные на предыдущих этапах, должны служить основой для определения мер безопасности, направленных на защиту системы.

Полученные данные о степенях риска и уязвимостях системы указывают возможные направления защиты системы от нежелательного воздействия.

Меры безопасности могут быть применены к окружающей среде, персоналу, руководству, аппаратному/программному обеспечению, линиям связи.

Принципиальные меры по снижению риска позволяют избежать риска, передать риск (страхование), уменьшить степень угрозы, минимизировать уязвимости, снизить возможные последствия, вовремя обнаружить и отразить нежелательные воздействия.

Структура информационной безопасности системы описывает, каким образом требования безопасности должны быть удовлетворены для информационной системы на уровне ее архитектуры. Данный этап особенно важен в случае разработки новой системы, или внесения значительных изменений в существующую.

Основная трудность в выборе мер безопасности – достичь компромисса между интенсивностью мер безопасности, их гибкостью, ценой и трудностью внедрения. Задача данного этапа состоит в исследовании всевозможных ограничений, возникающих в зависимости от ситуации, таких как временные, финансовые, технические, социологические, природные и законодательные ограничения.

На практике невозможно создать абсолютно защищенную систему, в силу чего после применения мер безопасности и оценки снижения рисков сохранятся так называемые остаточные риски. Эти риски подразделяются на приемлемые и неприемлемые для организации, что устанавливается на основе рассмотрения влияния исследуемых рисков на функционирование организации. Неприемлемые риски не могут быть приняты без дальнейшего рассмотрения, которое либо установит дополнительные меры безопасности, либо переведет риски в категорию приемлемых по иным соображениям (малая вероятность, нерентабельность предотвращения и т. д.).

Задача методики оценки рисков – предоставить детальные инструкции для практического применения существующих стандартов в области информационной безопасности. Методика оценки рисков ставит задачу обозначить алгоритм проведения необходимых процедур, классификацию и способы представления различных типов исходных данных, промежуточных и итоговых результатов, и определить метод вычисления необходимых параметров.

На сегодняшний день разработаны методики оценки рисков информационной безопасности, как правило, снабженные программными комплексами для проведения необходимых вычислений. Основная тенденция развития подобных систем – стремление к обеспечению максимальной гибкости при минимизации временных и финансовых затрат на проведение анализа рисков.

Контрольные вопросы

1. Что такое риск?
2. Что такое угроза?
3. Что такое остаточный риск?
4. Как можно качественно оценить риск?
5. Какие существуют методы количественной оценки рисков?

14. ВЕРИФИКАЦИЯ ЗАЩИТЫ

14.1. ДОКАЗАТЕЛЬНАЯ БАЗА НАДЕЖНОСТИ РЕАЛИЗАЦИИ ПОЛИТИКИ БЕЗОПАСНОСТИ

После того, как ПБ сформулирована и принята, способы ее реализации представляются определенным набором методов, правил и механизмов, интегрированных в модели и сценарии, которые призваны гарантировать соблюдение политики. Такой набор реализуется в виде аппаратного и программного комплекса – самостоятельной подсистемы или продукта информационных технологий. Любой из подобных комплексов требует оценки его возможностей в полном объеме реализовать все положения ПБ.

Система защиты адекватна политике безопасности, если она надежно реализует задаваемые этой политикой правила. Система защиты считается неприемлемой в случае, когда она ненадежно поддерживает политику безопасности.

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.

Гарантированность как мера уверенности в том, что инструментальные средства защиты обеспечивают выполнение принятой ПБ, главным образом подтверждается на архитектурном и технологическом уровнях их реализации.

С позиции архитектурного уровня особая роль принадлежит защите ядра ОС, четкой систематизации привилегий для выполнения аппаратных и системных функций, их минимизации для отдельных компонентов, сегментации адресного пространства процессов и т. п.

С точки зрения гарантированности технологической можно выделить следующие подходы, требующие для своего осуществления математических моделей, алгоритмического и программного обеспечения:

- тестирование;
- верификация на основе формальных методов доказательства;
- математические модели гарантированно защищенных систем.

Верификация в автоматизированном режиме на основе формальных методов доказательства – эффективный подход к обоснованию гарантированной защищенности компьютерных систем на всех этапах их жизненно-

го цикла. Необходим поиск подходов к созданию подсистем, автоматизирующих процессы верификации безопасности отдельных функционально замкнутых компонентов, описанию и практической реализации моделей их взаимодействия в составе больших распределенных систем. Теоретические основы для решения подобных задач, подходы к формализации программ и потенциальных средств разрушающего воздействия, методы анализа и инструментальные средства на этом направлении только создаются.

Если модель ПБ удается достаточно подробно и четко формализовать математически, то высока вероятность получить аналитически или путем строгих логических рассуждений доказательство гарантии выполнения объектом защиты ПБ. Для относительно простых систем и перечисленных выше моделей ПБ в условиях достаточно «жестких» дополнительных ограничений получить такие оценки можно. Их удается получить, например, в рамках модели «take-grant», описывающей с помощью теории графов способы распространения прав доступа в системах с дискреционной ПБ, или теоретико-множественных моделей Белла – Лападулы, а также ряда других.

Однако практические системы, подлежащие защите, являются более сложными и, как правило, распределенными.

В отличие от монолитных вычислительных систем, имеющих заданный периметр и спецификацию, распределенные системы не имеют ограниченного периметра, а число их компонентов может меняться. Между компонентами в сетях существуют каналы, которые могут проходить по незащищенной или враждебной территории. Этими чертами различия между монолитной и распределенными вычислительными 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. Каждый компонент 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 компонентов в условиях рассматриваемого окружения и организации защищенных шлюзов для взаимодействия компонентов. 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 обязатель-

ство 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 – Network Trusted Computing Base*). При этом средства поддержки безопасности в сети вовсе не должны составлять полный комплекс (удовлетворяющий стандарту ОК) механизмов защиты в каждом отдельном 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 компонентов *NTCB*, поддерживающей политику в сети, а также задача оценки защитных функций компонентов, из которых *NTCB* возможно синтезировать. Для анализа и синтеза таких систем, с одной стороны, применяется подход, который 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 компонентов без переоценки всего в целом становится принципиальной.

14.2. СИНТЕЗ И ДЕКОМПОЗИЦИЯ ЗАЩИТЫ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Рассмотрим сеть, компоненты которой связаны каналами связи, а каждый из компонентов 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 компонентов 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 компонентов защиты.

В связи с этим предположим, что *TCB* компоненты в каждом локальном случае поддерживают функции монитора обращения. Единая политика безопасности в сети не означает, что все *TCB* компоненты поддерживают одну и ту же политику и соответствуют требованиям одного класса. Например, один компонент может быть классифицирован по классу С2 (хотя в нем тоже требуется дополнительно наличие монитора обращения), а другой – по классу В3. При этом оба компонента поддерживают единые дискреционную и мандатную политики, хотя первый компонент – одноуровневый (соответствует одному классу обрабатываемой информации), а второй – 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 14.1. Пусть выполнены следующие условия.

1. Каждый субъект сети существует только в одном компоненте на протяжении всего жизненного цикла.
2. Каждый субъект может иметь доступ только к объектам своего компонента.
3. Каждый компонент содержит отнесенный к этому компоненту монитор обращений, который рассматривает только обращения субъектов этого компонента к объектам этого компонента.
4. Все каналы, связывающие компоненты, не компрометируют безопасность информации, в них проходящей.

Тогда совокупность мониторов обращения компонентов является монитором обращения в сети.

Доказательство. Этот результат потребует дополнительного уточнения некоторых основных понятий. Рассмотрим сеть из компонентов, 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, т. е. 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 счита-

ем 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 компонентов. Тогда из вышесказанного следует, что нет объектов вне локальных компонентов, а дистанционный доступ происходит за счет создания нового объекта, который полностью контролируется локальным *TCB* компонента. Покажем, что формальное объединение мониторов обращения компонентов – монитор обращения сети (М).

Каждое обращение субъекта к объекту в системе происходит через М. В самом деле, каждый субъект существует только в одном компоненте по условию 1 и может обращаться к объектам только своего компонента по условию 2. Поэтому все обращения в системе ограничены рамками компонентов. В этом случае каждое обращение обрабатывается монитором обращения соответствующего компонента согласно определению монитора обращения.

Монитор обращения каждого компонента согласно определению гарантированно защищен. Поэтому объединение их гарантированно защищено.

Функционирование всех мониторов обращения компонентов полностью тестируется (т.е. существует полная система тестов). Тогда совокупность тестов компонентов полностью тестирует М.

Теорема доказана.

Теперь рассмотрим вопрос о синтезе единой вычислительной системы из компонентов 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 *TCB* 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 *TCB*, 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 *TCB* 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.

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

Использованный в теореме безопасный канал связи должен удовлетворять следующим требованиям:

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.

14.3. АНАЛИЗ КОМПОНЕНТОВ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

Анализ и оценка защиты распределенных систем, как единого целого, предполагает анализ частей, а затем построение оценки защищенности всей системы в целом. Анализ компонентов и синтез единой оценки защищенности всей системы необходим также при модернизации системы, при замене старых компонентов 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 компонентов можно было построить оценку защищенности системы в целом?
2. Какими критериями надо пользоваться при анализе компонентов и как из результатов для компонентов синтезировать общую оценку?

В случае когда декомпозиция происходит так, что в каждом компоненте реализован монитор обращения, то, при выполнении условий теоремы предыдущего параграфа, во всей системе есть монитор обращения. Тогда гипотетическое объединение 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 так, что в любом компоненте реализован монитор обращений, позволяет рассредоточить функции *NTCB* по различным компонентам.

В «Красной книге» допускается, что *TCB* 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 1 и выполнять некоторый набор из перечисленных функций (всего имеется 16 вариантов таких наборов). При наличии этих свойств подсистема может быть компонентом распределенной сети и входить в *NTCB*.

Примеры включения таких подсистем.

Пример 14.1 (рис. 14.1). Пусть дан М – компонент (т. е. подсистема, единственной функцией которой является поддержка мандатного контроля доступа).

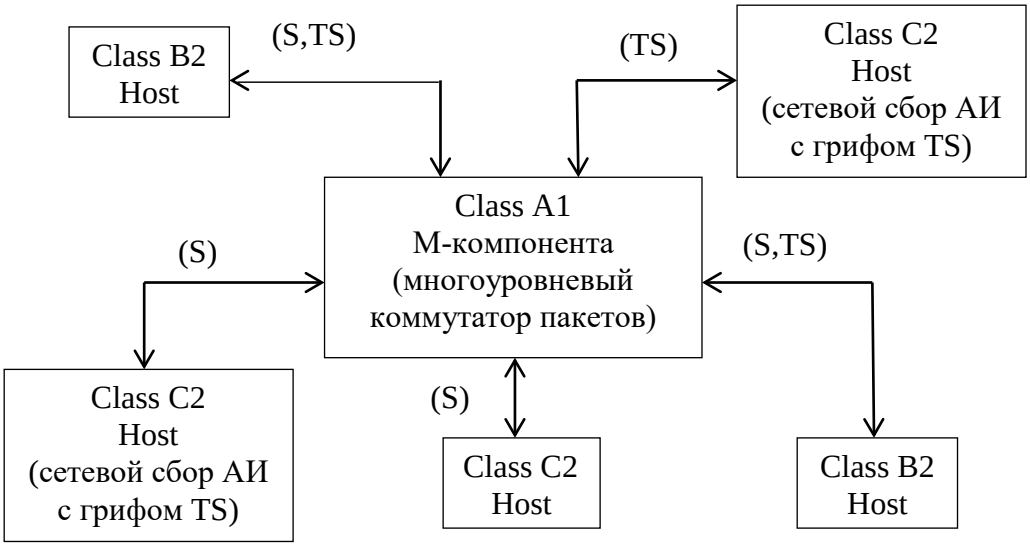

Рис. 14.1. Взаимосвязь компонента М
с другими компонентами системы

Пусть также эта подсистема обладает монитором обращений, оценена как самостоятельная система по классу A1. Тогда ее можно включить как компонент в гарантированно защищенную распределенную систему обработки информации, например, для выполнения функций многоуровневой коммутации пакетов. Это можно проиллюстрировать схемой, взятой из «Красной книги».

На приведенной схеме показана взаимосвязь M – компонента с другими компонентами и, в частности, с подсистемами *TCB*. Минимальное взаимодействие необходимо с системой аудита.

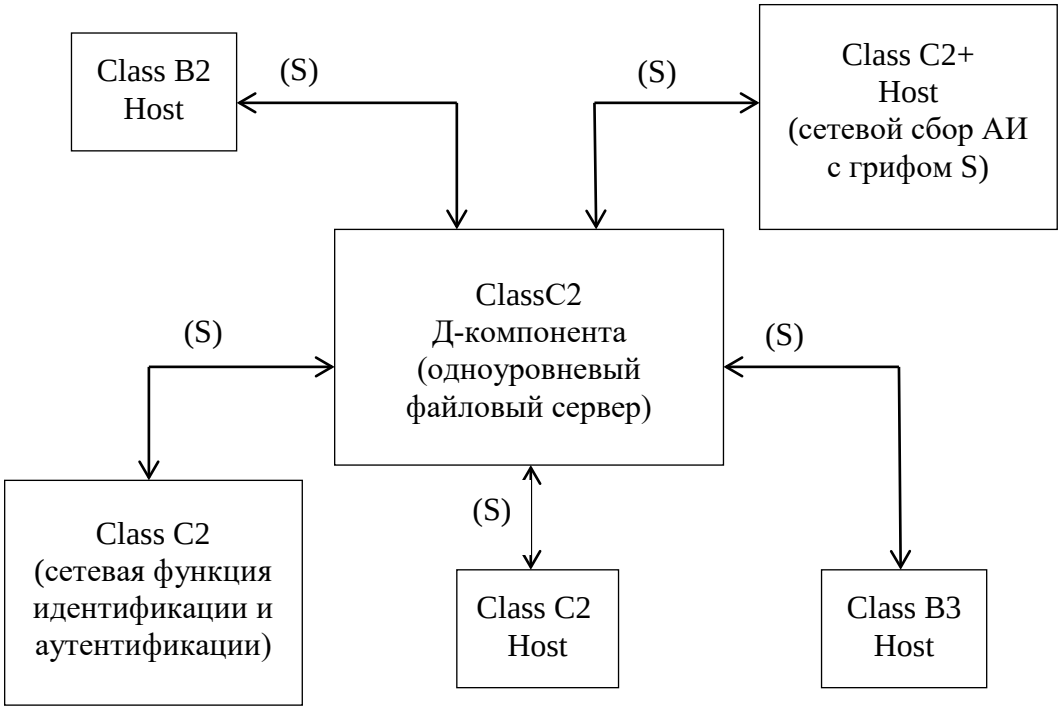

Рис. 14.2. Взаимосвязь компонента Д
с другими компонентами системы

Пример 14.2 (рис. 14.2). Данный пример из «Красной книги» показывает использование Д-компонента в качестве одноуровневого файлового сервера сети. В этом примере обозначение C2+ показывает, что система может быть оценена по классу C2, но иметь дополнительные функции, которые присущи дискреционной политике и аудиту в классе B3 и выше. (Дополнительно требуется выполнение функции блокировки при превышении числа опасных событий выше порога, матрица запрещенных доступов и т. д.).

Контрольные вопросы

1. Что такое «гарантированная защита»?
2. Какие существуют подходы к анализу и оценке защищенности распределенной системы?
3. При каких условиях совокупность мониторов обращения компонентов является монитором обращения в сети?
4. В чем заключается безопасность связи?
5. Что такое имитозащита?

15. ПРЕДСТАВЛЕНИЕ ПОЛИТИКИ БЕЗОПАСНОСТИ

В соответствии с определением политики безопасности (ПБ) в основе ее формального описания должны лежать спецификации, декларирующие правила доступа субъектов системы к объектам и образующие это формальное описание. Поэтому средства формального описания политики безопасности должны обладать достаточной выразительной силой, т. е. по сути являться языковыми. На основании такого языка необходимо иметь возможность описывать достаточно широкий круг политик безопасности.

Язык описания политики безопасности в зависимости от выбранных в политике безопасности правил разрешения доступа должен быть способен специфицировать закрытые политики безопасности:

- при описании политик безопасности все разрешенные виды доступа должны быть определены (все неопределенные виды доступа считаются запрещенными);

открытые политики безопасности:

- при описании политик безопасности все запрещенные виды доступа должны быть определены (все неопределенные виды доступа считаются разрешенными);

гибридные политики безопасности:

- при описании политик безопасности должны быть определены все запрещенные и разрешенные виды доступа;

гибридные политики безопасности с разрешенными противоречиями.

Язык описания политики безопасности должен определять корректные спецификации. Спецификация политики безопасности является корректной, если она непротиворечива и полна.

Под непротиворечивостью спецификации понимается наличие одного решающего правила для каждого доступа субъекта системы к объекту (доступ не может быть одновременно разрешен и запрещен).

Под полнотой спецификации понимается существование правила для каждого доступа субъекта системы к объекту, запрещающего или разрешающего этот доступ. Если для некоторого доступа не определена авторизация, должно присутствовать разрешающее правило-умолчание.

Существует несколько методов описания политик безопасности, в том числе аналитические, графовые, объектные и логические.

Аналитические методы базируется на описании ПБ в терминах математических формул и дальнейшем построении математического дока-

зательства безопасности. Для описания ПБ определяются множества субъектов, объектов и операции над ними. Далее алгоритм анализа безопасности состоит в следующем:

1. В терминах теории множеств определяется система и ее состояния.
2. Задаются функции переходов системы из состояния в состояние.
3. Определяются критерии безопасности системы.
4. Проводится формальное доказательство безопасности системы, т. е. соответствия системы выбранному критерию безопасности.

Методы подкреплены сильной математической базой, поэтому являются безупречными в плане доказательства теорем безопасности. Однако методы, основанные на формулах, трудно применимы для оценки реальных систем, так как сложно сопоставить объекты и свойства реального мира элементам математических формул и их свойствам. Аналитические методы требуют тщательного анализа объекта оценки.

Рассмотрим пример применения АМ для описания ПБ системы, построенной на базе дискреционной модели. Представление системы $\Sigma(Q, R, C)$ состоит из элементов:

- конечные исходные наборы субъектов $S_0 = \{s_1, \ldots, s_n\}$ и объектов $O_0 = \{o_1, \ldots, o_m\}$, где $S_0 \subseteq O_0$;
- конечный набор прав доступа $R = \{r_1, \ldots, r_n\}$;
- исходная матрица доступа $M0$ с правами доступа субъектов к объектам;
- конечный набор команд $C = \{\alpha i(x_1, \ldots, x_k)\}$, каждая из которых состоит из условий выполнения и интерпретации в терминах элементарных операций:

command α (*x*1, …, *xk*)
if *r*1 in *M*[*xs*1, *xo*1] and
*r*2 in *M*[*xs*2, *xo*2] and
…
rm in M[*xsm*, *xom*]
then *op*1, *op*2, …, *opn*,

где α – имя команды; *xi* – параметр команды, являющийся идентификатором субъектов и объектов; *si* и *oi* – индексы субъектов и объектов в диапазоне от 1 до *k*; *opi*– элементарная операция.

Поведение системы во времени моделируется с помощью последовательности состояний {*Qi*}, в которой каждое последующее состояние является результатом применения некоторой команды из множества *C* к предыдущему: $Q_{n+1} = C_n(Q_n)$. Пространство состояний – декартово произведение $O \times S \times R$. Состояние *Q* – кортеж (*S*, *O*, *M*), где *M* – матрица прав

доступа. Ячейка $M[s, o]$ содержит набор прав субъекта s к объекту o, принадлежащих множеству прав доступа R. Начальное состояние $Q_0 = (S_0, O_0, M_0)$ является безопасным относительно права r, если не существует применимой к Q_0 последовательности команд, в результате которой r будет занесено в ячейку матрицы M, в которой оно отсутствовало в состоянии Q_0.

Графовые методы моделирования и анализа безопасности основаны на представлении системы и отношений в ней в виде графа. Вершины графа моделируют сущности системы (субъекты и объекты), а дуги показывают отношения между ними. Каждая дуга помечается некоторой меткой-атрибутом, показывающей набор прав, типов доступа, операций одной сущности системы над другой и т. п. Для обозначения отношений между сущностями могут быть задействованы вершины графа: они помечаются как типы или роли сущностей.

Анализ безопасных переходов системы производится посредством задания перехода графа в новое состояние после выполнения операций над сущностями. При этом под состоянием графа понимается совокупность вершин, дуг и их атрибутов. Оценка выполнения правил ПБ в некотором состоянии системы проводится как сопоставление графа, соответствующего текущему состоянию, с графом, соответствующим небезопасному состоянию (сопоставление с шаблоном).

Показательным примером графового метода является визуальный язык объектных ограничений *LaSCO* (*Language for Security Constraintson Objects*).

Политика безопасности моделируется с помощью задания объектов – сущностей системы, имеющих определенные состояния, и событий – доступов или взаимодействий (сигналов) между двумя объектами в определенные моменты времени. Состояние системы представляется как снимок системы в некоторый момент времени и описывается множеством объектов и множеством предстоящих событий. События имеют неизменяемые ассоциированные параметры. Каждый объект имеет фиксированный набор меток, описывающих состояние объекта с помощью пар «атрибут – значение». Значения атрибутов могут меняться во времени (за исключением уникального параметра идентификации объекта). События и изменения атрибутов объектов происходят в дискретные моменты времени. Момент состояния системы отражается в событиях параметром *time*.

Объектные методы базируются на теории объектно-ориентированного проектирования, где в качестве объектов выступают правила ПБ, группы правил, роли. Каждый объект принадлежит к определенному типу – классу, – который занимает свое место в иерархии типов. Сущности системы (субъекты и объекты) также подразделяются на типы, называемые доменами.

Представление и анализ ПБ проводится как объектная декомпозиция системы и оценка безопасных/небезопасных состояний. Анализ системы проводится по следующей схеме:

1. Задание типов и объектов данных типов для сущностей системы.
2. Связывание объектов в иерархию.
3. Определение методов доступа к объектам, т. е. правил доступа.

Декомпозиция ИС представляется совокупностью автономных действующих сущностей, которые взаимодействуют друг с другом, чтобы обеспечить поведение системы. Таким образом, каждый объект обладает своим собственным поведением, и каждый из них моделирует некоторую сущность реальной системы. С этой точки зрения объект является вполне осязаемой вещью, которая демонстрирует вполне определенное поведение. Оценка выполнения ПБ проводится путем сопоставления текущего и критического состояний системы объектов.

Вариантом объектного подхода служит язык моделирования *Ponder*. Правила авторизации, введенные в языке и определяющие, какие действия домена субъекта могут совершаться над множеством объектов, по существу являются правилами разграничения доступа. Правило позитивной авторизации определяет, какие действия разрешено осуществлять субъекту над объектом, негативной авторизации – какие действия запрещены.

Синтаксис правила авторизации в языке *Ponder*:

inst(***auth+*** | ***auth-***)имя_правила "{"
subject домен**;**
targetдомен**;**
actionсписок_допустимых_операций**;**
[***when***ограничения**;**] "}"

Такие слова как ***inst***, ***auth+***, ***action*** выполняют роль служебных слов языка *Ponder*:

- ***inst*** – объявление правила;
- ***auth+*** – позитивная (действия, указанные как значения **action**);
- ***auth-*** – негативная авторизация;
- ***subject*** – сущность или множество сущностей, для которых задается авторизация;
- ***target*** – сущность или множество сущностей, над которыми применяется правило;
- ***action*** – действия, на которые авторизован ***subject*** над ***target***;
- ***when*** – ограничения, в рамках которых действует авторизация.

Альтернативные выражения заключены в круглые скобки "()", разделенные символом '|'. Необязательные элементы описываются в квадратных

скобках "[]". Повторения описываются с помощью фигурных скобок "{}". Ограничения являются необязательными во всех типах правил и могут применяться для ограничения применимости правил, основанных на времени или значениях атрибутов сущностей, к которым относится правило.

Язык *Ponder* позволяет задавать правила (рис. 15.1) делегирования прав, правила обязательства, выполняющиеся автоматически при наступлении определенных событий в системе, правила группирования и т. п. Поскольку язык сильно абстрагирован от формальной теории, возникают сложности при доказательстве непротиворечивости выводимых правил. Однако строгая иерархия позволяет структурировать их необходимым образом и способствует лучшему пониманию моделируемой ПБ.

С помощью языка *Ponder* правила чтения и записи в ИС, использующей дискреционную модель безопасности, можно смоделировать следующим образом:

```
inst auth+ read {                    instauth+ write {
subject s=Subjects;                  subject s=Subjects;
targeto=Objects;                     targeto=Objects;
actionread();                        actionread();
whenin(r, rights(s,o));}             whenin(w,rights(s,o));}
```

Субъекты – сущности типа *Subjects* – могут производить операции чтения и записи над объектами – сущностями типа *Objects* – только в том случае, если они обладают соответствующими правами доступа (rights(s,o)), проставленными в соответствующей ячейке матрицы доступа.

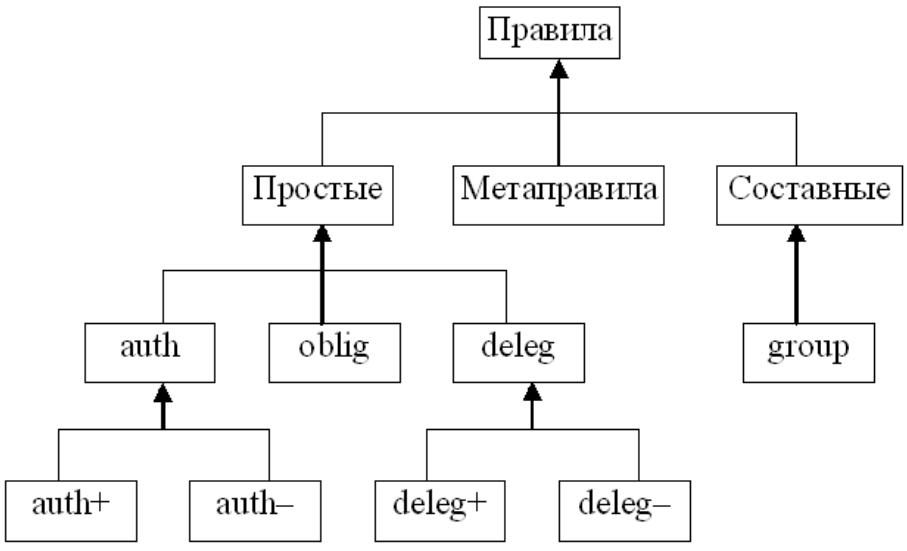

Рис. 15.1. Иерархия правил в языке *Ponder*

Делегирование полномочий моделируется правилом ***deleg***:

instdeleg+ (read, write) giverights{
grantee Subjects**;**
subject s=Subjects**;**
target o=Subjects**;**
whenin(ga,s.rights(o))}

Правило показывает, что субъект может передать права доступа на операции чтения и записи только, если они есть у него самого, и он обладает правом передачи полномочий (право *ga*, grantaccess).

Способ моделирования и анализа ПБ посредством логических методов основан на описании системы и правил ее поведения в виде логических структур (фактов) и задания на их основе правил логического вывода. Правила логического вывода могут порождать новые правила и факты. Фактами описывают сущности системы и их характеристики. Текущее состояние базы знаний (набор фактов) моделирует состояние системы, а правила вывода представляют собой механизм перехода системы из одного состояния в другое под действием запросов. Ограничения, позволяющие судить о невыполнении правил ПБ, задаются либо в виде правил, описывающих критическое состояние, либо в виде фактов.

Примером применения логического метода служит достаточно простой язык авторизаций (ЯА) для описания субъектов, объектов, типов объектов, а также групповой иерархии субъектов и взаимодействий сущностей.

Моделирование ПБ проводится в терминах логики предикатов. Вводятся множества *S, O* и *A* – множества субъектов $S = U \cup G$ – множество субъектов (*U* – множество пользователей, *G* – множество групп пользователей), объектов и действий субъектов соответственно. Система *DS* представляется как кортеж множеств (*O, T, S, A*), где *T* – множество типов объектов. Термом авторизации называется совокупность (*s, o, a*), где ее элементы – константы или переменные, определенные на множествах *S, O* и *A*. Политика безопасности – набор термов авторизации. Термы (*s, o, a*) в политике *P* устанавливают доступ, разрешенный политикой. В ПБ формулируются условия выполнения термов и правила логического вывода новых термов.

Язык авторизаций состоит из ограничений, которые задаются как факты с аргументами из множеств *S, O, A,* и логических правил, построенных на основе термов авторизации вида $(s, o, a) \leftarrow L1 \wedge \ldots \wedge Ln$, где *Li* – терм авторизации. Основу ЯА составляет фиксированный набор предикатов (табл. 15.1.) вместе с их отрицаниями.

Таблица 15.1

Предикаты языка авторизаций

Предикат	Назначение
cando	Разрешение доступа субъекта к объекту
dercando	Аналогичен cando с учетом логического вывода
do	Разрешение доступа конкретного субъекта к конкретному объекту
done	Разрешение, если субъект ранее уже выполнял указанное действие над объектом
error	Ошибка авторизации
dirin, in	Прямое и косвенное включение субъектов в группы
typeof	Тип объекта
owner	Принадлежность конкретного объекта определенному субъекту

Посредством ЯА возможно создать логическую программу авторизации, которая проводит сопоставление состояния системы с требованиями ПБ.

Однако известные решения на базе ЛМ, в том числе и ЯА, не обладают универсальностью, необходимой для моделирования и анализа ПБ в реальных ИС, поскольку имеют ряд ограничений:

1. Сложности при моделировании некоторых дискреционных моделей. Так, например, модель, основанная на ролевом управлении доступом, требует допущения вида "группа ≡ роль", что не соответствует действительности и вносит затруднения в процесс анализа ролевой политики.

2. Отсутствие математических операций сравнения (например, >, <, =, ≥, ≤, ≠) применительно к описанию отношений «субъект-объект», что усложняет процесс моделирования политик мандатного управления доступом. Так, например, правило «нет чтения вверх», или простое свойство безопасности модели Белла – Лападулы, содержит сравнение уровней безопасности. Отсутствие сравнений делает невозможным описание такого рода правил.

3. Характеристика субъекта посредством одного атрибута "группа", а объекта – при помощи атрибута "тип". На практике сущность может ассоциироваться с несколькими атрибутами. Так, необходимость применения меток в мандатных моделях требует введения атрибута "уровень безопасности" и соответствующего расширения языкового аппарата. Каждая сущность должна содержать список атрибутов, характеризующих ее в терминах безопасности.

4. Отсутствие средств отладки и протоколирования, что ставит под сомнение возможность детального анализа ПБ и отслеживания процесса выполнения правил авторизации.

Ограничения известных решений, основанных на базе логического подхода, приводят к отсутствию их наглядности и полноты, но нисколько не умаляют достоинства самого логического метода.

Аналитические методы имеют в своей основе хорошо проработанный аппарат представления в виде математических функций и элементов матлогики. АМ наиболее точны и строги с точки зрения теории, но они лишены наглядности и требуют специальной подготовки специалистов.

Графовые методы обладают свойствами гибкости при моделировании произвольной ПБ и наилучшей наглядности при моделировании состояния системы. ГМ являются очень выразительными и простыми для понимания и анализа ПБ системы. Методы базируются на хорошо разработанных математических теориях: теории графов и топологии. К недостаткам ГМ следует отнести статичность моделирования, т. е. демонстрация определенных состояний системы без указания взаимосвязи следующих состояний с предыдущим, и то, что в графе операции определены над сущностями системы, а не над правилами, которые задают ПБ. По этой причине метод графов становится трудно применимым при оценке комбинированных ПБ.

Объектные методы позволяют моделировать защищенность крупных систем, поскольку правила применяются к группам сущностей (т. е. к объектам указанного типа). Принципы инкапсуляция и наследования, известные из объектно-ориентированного проектирования, способствуют универсальности и расширяемости представления. ОМ обобщают методы графов с учетом того, что взаимосвязи моделируются не между вершинами-сущностями, а между объектами заданных классов. Метод объектов можно рассматривать, как модификацию метода графов, дополненного понятиями класса, объекта, методов и инкапсуляции. ОМ приспособлены к моделированию сложных иерархических систем. К недостаткам метода следует отнести сложность объектной декомпозиции ИС. Неподготовленный человек, будучи не в состоянии полностью воссоздать сложный объект, абстрагируется и просто игнорирует не слишком важные с его точки зрения детали и, таким образом, имеет дело с неоправданно идеализированной моделью, отвлеченной от реальной функции защиты. Поэтому при применении ОП всегда необходимы поиск должного уровня абстрагирования.

Метод логического программирования позволяет добиться простоты реализации и проведения автоматического анализа ПБ, поскольку сама программа оценки защищенности может быть задана в терминах логики предикатов. Формальная база такого метода, а именно математическая логика, позволяет получать спецификацию системы и применять аппарат логического вывода при моделировании и анализе ПБ.

Контрольные вопросы и задания

1. Какие типы политик безопасности существуют?
2. Приведите пример применения аналитических методов для описания системы.
3. Каким образом проводится анализ системы при использовании объектных методов?
4. Какая иерархия правил существует в языке Ponder?
5. В чем заключаются преимущества логических методов моделирования?

16. УПРАВЛЕНИЕ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТЬЮ

16.1. ПОНЯТИЕ УПРАВЛЕНИЯ БЕЗОПАСНОСТЬЮ

Информация является одним из самых главных деловых ресурсов, который обеспечивает организации добавочную стоимость, и вследствие этого нуждается в защите. Слабые места в защите информации могут привести к финансовым потерям, и нанести ущерб коммерческим операциям. Поэтому в наше время вопрос разработки системы управления информационной безопасностью и ее внедрения в организации является концептуальным.

При построении системы защиты компании должны учитывать специфику бизнеса, а не ориентироваться на достижение всех свойств информационных ресурсов. Например, для банковского сектора ключевой задачей в области информационной безопасности станет обеспечение целостности финансовой информации; для операторов связи – доступности информационных ресурсов, начиная с адекватной пропускной способности каналов и заканчивая доступностью коммерческих сервисов; для государственных компаний, в свою очередь, важна конфиденциальность информации. Конечно, это не значит, что банки не заинтересованы в доступности информации или в госсекторе нет необходимости иметь целостные данные, отнюдь. Просто начинать внедрение системы безопасности надо с критически важных ее аспектов, и тогда при правильном подходе к построению архитектуры в конечном счете можно действительно получить надежную систему управления информационной безопасностью (СУИБ).

Для грамотного построения СУИБ уже сформированы готовые и отработанные стандарты. На сегодняшний день существует две большие группы международных стандартов для систем управления информационной безопасностью: ИСО/МЭК 27000 и ИСО/МЭК 13335.

Семейство 27000 включает в себя международные стандарты, определяющие требования к системам менеджмента защиты информации, управление рисками, метрики и измерения, а также руководство по внедрению.

Для этого семейства стандартов используется последовательная схема нумерации, начиная с 27000 и далее:

- ISO27000 Определения и основные принципы. Унификация со стандартами COBITи ITIL.

- ISO/IEC 27001:2005 Информационные технологии. Методы обеспечения безопасности. Системы менеджмента защиты информации. Требования (BS 7799-2:2005).
- ISO/IEC 27002:2005 Информационные технологии. Методы обеспечения безопасности. Практические правила управления информационной безопасностью (ранее ISO/IEC 17799:2005).
- ISO27003Руководство по внедрению системы управления информационной безопасностью.
- ISO 27004 Измерение эффективности системы управления информационной безопасностью.
- ISO 27005Измерение эффективности системы управления информационной безопасностью. Управление рисками информационной безопасности (на основе BS 7799-3:2006).
- ISO/IEC 27006:2007 Информационные технологии. Методы обеспечения безопасности. Требования к органам аудита и сертификации систем управления информационной безопасностью.
- ISO 27007Руководство для аудитора СУИБ.
- ISO 270011Руководство по управлению информационной безопасностью для телекоммуникаций.

Следующие 4 стандарта ИСО/МЭК 13335также предназначены для обеспечения безопасности информационных технологий:

- ISO13335-1:2004Информационные технологии. Методы и средства обеспечения безопасности. Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий.
- ISO13335-3:1998Информационные технологии. Методы и средства обеспечения безопасности. Методы менеджмента безопасности информационных технологий.
- ISO13335-4:2000Информационные технологии. Методы и средства обеспечения безопасности. Выбор защитных мер.
- ISO13335-5:2001Информационные технологии. Методы и средства обеспечения безопасности. Руководство по менеджменту безопасности сети.

Также неотъемлемой частью СУИБ является управление инцидентами ИБ. Этому вопросу посвящен нормативный документ ISO/IECTR 18044:2004 "Информационная технология. Методы и средства обеспечения безопасности. Менеджмент инцидентов информационной безопасности". Данный документ описывает инфраструктуру управления инцидентами в рамках циклической модели PDCA. Даются подробные спецификации для стадий планирования, эксплуатации, анализа и улучшения процесса. Рассматриваются вопросы обеспечения нормативно-распорядительной документацией, ресурсами, даются подробные рекомендации по необходимым процедурам.

16.2. ISO/IEC 27001 «СИСТЕМЫ МЕНЕДЖМЕНТА ЗАЩИТЫ ИНФОРМАЦИИ. ТРЕБОВАНИЯ»

Международный стандарт ISO/IEC 27001:2005 разработан Международной организацией по стандартизации (ISO) и Международной электротехнической комиссией (IEC) на основе британского стандарта BS 7799. Он был подготовлен для того, чтобы предоставить модель для создания, внедрения, эксплуатации, постоянного контроля, анализа, поддержания в рабочем состоянии и улучшения системы менеджмента защиты информации (СМЗИ). Рекомендуется, чтобы принятие СМЗИ было стратегическим решением для организации. Требования данного стандарта имеют общий характер и могут быть использованы широким кругом организаций – малых, средних и больших – коммерческих и индустриальных секторов рынка: финансовом и страховом, в сфере телекоммуникаций, коммунальных услуг, в секторах розничной торговли и производства, различных отраслях сервиса, транспортной сфере, органах власти и многих других (рис. 16.1).

Стандарт ISO 27001 определяет информационную безопасность как: сохранение конфиденциальности, целостности и доступности информации; кроме того, могут быть включены и другие свойства, такие как подлинность, невозможность отказа от авторства, достоверность.

Конфиденциальность – обеспечение доступности информации только для тех, кто имеет соответствующие полномочия (авторизированные пользователи).

Целостность – обеспечение точности и полноты информации, а также методов ее обработки.

Доступность – обеспечение доступа к информации авторизированным пользователям.

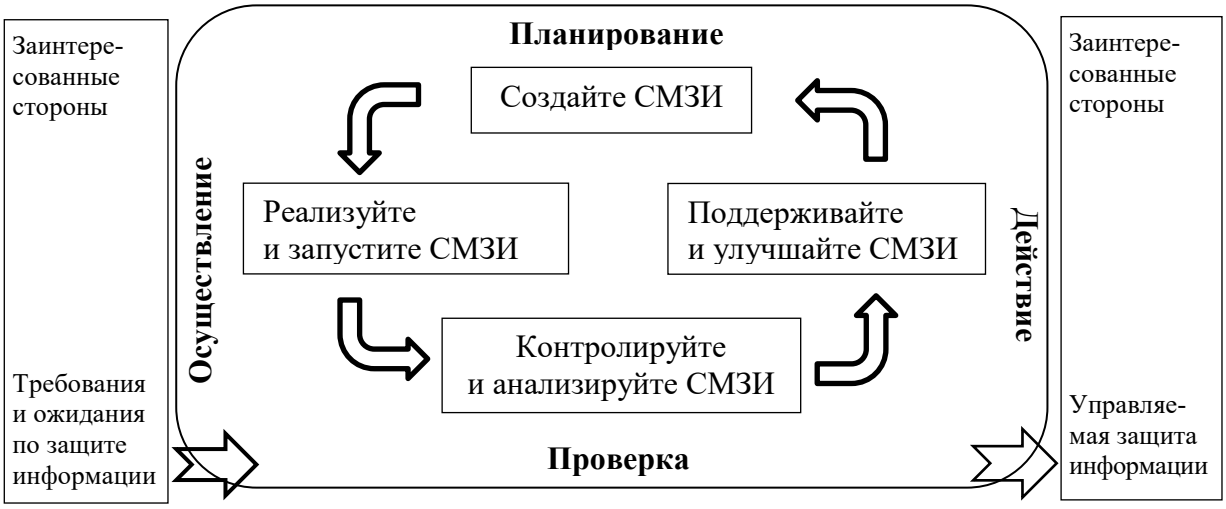

Рис. 16.1. Модель PDCA, примененная к процессам СМЗИ

ISO/IEC 27001:2005 представляет собой перечень требований к системе менеджмента информационной безопасности, обязательных для сертификации. В соответствии с этим стандартом, любая СМЗИ должна основываться на PDCA модели.

На рис. 16.1 показано, как на вход СМЗИ поступают требования защиты информации и ожидания заинтересованных сторон и посредством необходимых действий и процессов СМЗИ выдает результаты по защите информации, которые удовлетворяют этим требованиям и ожиданиям.

16.3. ISO/IEC 13335-1 «КОНЦЕПЦИЯ И МОДЕЛИ МЕНЕДЖМЕНТА БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ»

Настоящий стандарт представляет собой руководство по управлению безопасностью информационных и телекоммуникационных технологий (ИТТ), устанавливает концепцию и модели, лежащие в основе базового понимания безопасности ИТТ, и раскрывает общие вопросы управления, которые важны для успешного планирования, реализации и поддержки безопасности ИТТ.

Целью является формирование общих понятий и моделей управления безопасностью ИТТ. Приведенные в нем положения носят общий характер и применимы к различным методам управления и организациям.

Часть документа посвящена терминам и определениям, далее определяются концепции безопасности и взаимосвязи. Для создания эффективной программы безопасности ИТТ фундаментальными являются нижеперечисленные принципы безопасности:

- менеджмент риска – активы должны быть защищены путем принятия соответствующих мер. Защитные меры должны выбираться и применяться на основании соответствующей методологии управления рисками, которая, исходя из активов организации, угроз, уязвимостей и различных воздействий угроз, устанавливает допустимые риски и учитывает существующие ограничения;
- обязательства – важны обязательства организации в области безопасности ИТТ и в управлении рисками. Для формирования обязательств следует разъяснить преимущества от реализации безопасности ИТТ;
- служебные обязанности и ответственность – руководство организации несет ответственность за обеспечение безопасности активов. Служебные обязанности и ответственность, связанные с безопасностью ИТТ, должны быть определены и доведены до сведения персонала;

• цели, стратегии и политика – управление рисками, связанными с безопасностью ИТТ, должно осуществляться с учетом целей, стратегий и политики организации;

• управление жизненным циклом – управление безопасностью ИТТ должно быть непрерывным в течение всего их жизненного цикла.

С позиций фундаментальных принципов безопасности выделяются основные компонентов безопасности, вовлеченных в процесс управления безопасностью:

• Активы – все, что имеет ценность для организации.

• Угрозы – потенциальная причина инцидента, который может нанести ущерб системе или организации.

• Уязвимости – слабость одного или нескольких активов, которая может быть использована одной или несколькими угрозам.

• Воздействие – это результат инцидента информационной безопасности, вызванного угрозой и нанесшего ущерб ее активу.

• Риск – это способность конкретной угрозы использовать уязвимости одного или нескольких видов активов для нанесения ущерба организации. Следует учитывать, что риск никогда не устраняется полностью. Остаточный риск – это риск, остающийся после его обработки.

• Защитные меры – это действия, процедуры и механизмы, способные обеспечить безопасность от возникновения угрозы, уменьшить уязвимость, ограничить воздействие инцидента в системе безопасности, обнаружить инциденты и облегчить восстановление активов.

• Ограничения. Обычно ограничения устанавливает или признает руководство организации, а также определяет среда, в которой действует организация.

16.4. ISO/IEC 13335-3 «МЕТОДЫ МЕНЕДЖМЕНТА БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Настоящий стандарт устанавливает методы менеджмента безопасности информационных технологий. В основе этих методов лежат общие принципы, установленные в ИСО/МЭК 13335-1. Стандарт будет полезен при внедрении мероприятий по обеспечению безопасности информационных технологий.

Существуют четыре основных варианта стратегии анализа риска организации:

• использование базового подхода (с низкой степенью риска) для всех систем информационных технологий, независимо от уровня риска,

которому подвергаются системы, принятие того, что уровень обеспечения безопасности не всегда может оказаться достаточным. Подразумевается выбор стандартных защитных мер безопасности. Перечень рекомендуемых стандартных защитных мер приведен в документах по базовой безопасности;

- использование неформального подхода к проведению анализа риска, обращая особое внимание на системы информационных технологий, которые, как представляется, подвергаются наибольшему риску. Этот вариант подхода предусматривает проведение неформального анализа риска, основанного на практическом опыте конкретного эксперта. Неформальный подход предполагает использование знаний и практического опыта специалистов, а не структурных методов;
- проведение детального анализа риска с использованием формального подхода ко всем системам информационных технологий. Данный вариант подхода предполагает проведение детального анализа риска с получением результатов для всех систем информационных технологий, действующих в организации. Детальный анализ риска включает в себя подробную идентификацию и оценку активов, оценку возможных угроз, которым могут подвергнуться эти активы, а также оценку уровня их уязвимости;
- использование комбинированного подхода, в котором сначала проводится анализ риска «высокого уровня» с тем, чтобы определить, какие из систем информационных технологий подвержены высокому уровню риска и какие имеют критическое значение для ведения деловых операций, с последующим проведением детального анализа риска для выделенных систем, а для всех остальных – ограничиваются применением базового подхода к проблемам обеспечения безопасности.

Для выделения системы с высоким уровнем риска необходимо провести анализ уровня риска и выбрать варианты стратегии обеспечения безопасности информационных технологий. Затем выделенные системы рассматриваются с использованием метода детального анализа риска, а для остальных систем может применяться базовый подход (с принятием базового уровня риска).

16.5. ISO 27002 «ПРАКТИЧЕСКИЕ ПРАВИЛА УПРАВЛЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТЬЮ»

ISO/IEC 27002 – стандарт информационной безопасности, опубликованный организациями ISO и IEC. До 2007 г. данный стандарт назывался ISO/IEC 17799. Стандарт разработан в 2005 г. на основе версии ISO 17799,

опубликованной в 2000 г., которая являлась полной копией Британского стандарта BS 7799-1:1999.

Стандарт предоставляет лучшие практические советы по менеджменту информационной безопасности для тех, кто отвечает за создание, реализацию или обслуживание систем менеджмента информационной безопасности. Информационная безопасность определяется стандартом как «сохранение конфиденциальности (уверенности в том, что информация доступна только тем, кто уполномочен иметь такой доступ), целостности (гарантии точности и полноты информации и методов ее обработки) и доступности (гарантии в том, что уполномоченные пользователи имеют доступ к информации и связанным ресурсам)». Для определения требований к информационной безопасности организация должна учитывать три фактора:

- оценка рисков организации: посредством оценки рисков происходит выявление угроз активам организации, оценка уязвимости соответствующих активов и вероятности возникновения угроз, а также оценка возможных последствий;
- юридические, законодательные, регулирующие и договорные требования, которым должны удовлетворять организация, ее торговые партнеры, подрядчики и поставщики услуг;
- специфический набор принципов, целей и требований, разработанных организацией в отношении обработки информации.

После того, как определены требования к информационной безопасности, следует выбрать и внедрить такие мероприятия по управлению информационной безопасностью, которые обеспечат снижение рисков до приемлемого уровня. Выбор мероприятий по управлению информационной безопасностью должен основываться на соотношении стоимости их реализации к эффекту от снижения рисков и возможным убыткам в случае нарушения безопасности.

16.6. ISO 18044 «МЕНЕДЖМЕНТ ИНЦИДЕНТОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ»

Настоящий стандарт устанавливает рекомендации по менеджменту инцидентов информационной безопасности для руководителей подразделения по информационной безопасности, информационных систем, сервисов и сетей. Под инцидентом информационной безопасности понимают событие, являющееся следствием одного или нескольких нежелательных, или неожиданных событий ИБ, имеющих значительную вероятность компрометации бизнес-операции и создания угрозы ИБ.

Настоящий стандарт состоит из 11 разделов и построен следующим образом. В разделе 1 приводится область применения, в разделе 2 – перечень ссылок, в разделе 3 – термины и определения, которые в основном заимствованные из ISO/IEC 13335-1 и ISO/IEC 17799. В разделе 4 представлены основы менеджмента инцидентов информационной безопасности. В соответствии с этим стандартом любая система менеджмент инцидентов ИБ состоит из четырех отдельных процессов: планирование и подготовка, использование, пересмотр (анализ), улучшение.

16.7. РАЗРАБОТКА СУИБ И ТРЕБОВАНИЯ К СУИБ

Система управления информационной безопасностью (СУИБ) является частью общей системы, основанной на подходе анализа рисков, с целью создания, внедрения, эксплуатации, постоянного контроля, анализа, поддержки в рабочем состоянии и улучшения средств защиты информации. На основании нормативных документов определим, каким задачам, требованиям, характеристикам должна удовлетворять СУИБ с точки зрения стандартов.

На первом шаге в процессе управления безопасностью информационных технологий необходимо определить, какой общий уровень риска является приемлемым для данной организации. Правильно выбранный уровень приемлемого риска и, соответственно, допустимый уровень безопасности являются ключевыми моментами успешного управления безопасностью. Допустимый общий уровень безопасности определяется целями, которые ставит перед собой организация при создании системы обеспечения безопасности информационных технологий. Для того чтобы оценить и сформулировать такие цели, необходимо изучить имеющиеся активы и определить, насколько ценными они являются для данной организации. Главной целью построения СУИБ является оценка и удержание уровня риска в приемлемых значениях.

В качестве управляемых объектов выступают риски организации, а в качестве органов управления – защитные меры.

СУИБ должна обеспечивать выбор адекватных средств управления безопасностью, которые защищают информационные активы от возможных угроз, как случайных, так и преднамеренных. Необходимо определить источники таких угроз и оценить вероятность их реализации. Важно не упустить из виду ни одной возможной угрозы, так как в результате возможно нарушение функционирования или появление уязвимостей системы.

Полезным может быть использование перечня наиболее часто встречающихся угроз (примеры типичных видов угроз приведены в приложении С стандарта ISO 13335-3). Также полезно использование каталогов угроз (наиболее соответствующих нуждам конкретной организации или виду ее деловой деятельности), так как ни один перечень не может быть достаточно полным. Ниже приведены некоторые наиболее часто встречающиеся варианты угроз:

- ошибки и упущения;
- мошенничество и кража;
- случаи вредительства со стороны персонала;
- ухудшение состояния материальной части и инфраструктуры;
- программное обеспечение хакеров, например, имитация действий законного пользователя;
- программное обеспечение, нарушающее нормальную работу системы;
- промышленный шпионаж.

16.8. ПОЛИТИКА БЕЗОПАСНОСТИ

Должна быть сформирована политика безопасности и затем необходимо проводить ее в соответствии с направленностью деятельности организации, состоянием обеспечения безопасности, а также с учетом положений законодательства и нормативных документов в области обеспечения безопасности. Как минимум, политика должна включать следующее:

- определение информационной безопасности, ее общих целей и сферы действия, а также раскрытие значимости безопасности как инструмента, обеспечивающего возможность совместного использования информации;
- анализ рисков и изложение целей и принципов информационной безопасности, сформулированных руководством;
- безопасность аппаратно-программного обеспечения:

– идентификация и аутентификация;
– контроль доступа;
– журнал учета использования ресурсов и аудит;

- разделы, связанные с:

– безопасностью связи, например, криптографическая аутентификация и аутентификация сообщений;
– физической безопасностью;
– безопасностью документов и носителей информации;
– безопасностью персонала;

• определение общих и конкретных обязанностей сотрудников в рамках управления информационной безопасностью, включая информирование об инцидентах нарушения информационной безопасности;

• ссылки на документы, дополняющие политику информационной безопасности, например, более детальные политики и процедуры безопасности для конкретных информационных систем, а также правила безопасности, которым должны следовать пользователи.

СУИБ должна строиться по принципу модели PDCA. Она должны постоянно контролироваться и анализироваться, поддерживаться в рабочем состоянии и иметь возможность улучшения. Стандарт ISO 27001 выдвигает требования по каждому этапу жизненного цикла модели PDCA. Система должны обеспечивать:

- анализ рисков;
- выработку управляющего воздействия;
- проведение внутренних аудитов;
- контроль конфигурации и управление изменениями;
- управление инцидентами безопасности информации.

При проведении анализа рисков необходимо выбрать подход к анализу (требования к анализу рисков). В стандарте ISO 13335-3 выделяется четыре основных варианта анализа рисков.

Анализ рисков должен включать в себя следующее:

• установление границ рассмотрения – какие из ресурсов должны быть учтены при анализе рисков;

• идентификацию ресурсов;

• оценку активов. Оценку нужно проводить в соответствии со стандартом ISO 13335-3, где описываются два подхода к оценке. Первый заключается в инвентаризации всех активов и согласование масштабов оценки. Оценка может быть количественной (конкретная стоимость) или качественной (шкале в диапазоне от "очень низкой" до "очень высокой" цены). Другой подход к оценке активов основывается на затратах, понесенных по причине утраты конфиденциальности, целостности или доступности вследствие происшедшего инцидента;

• оценку угроз. После идентификации источника угроз (кто и что является причиной угрозы) и объекта угрозы (какой из элементов системы может подвергнуться воздействию угрозы) необходимо оценить вероятность реализации угрозы;

• оценку уязвимостей. Необходимо оценить, насколько велика степень уязвимости или насколько легко ее можно использовать. Степень уязвимости нужно оценивать по отношению к каждой угрозе, которая может использовать эту уязвимость в конкретной ситуации;

- оценку рисков. Метод оценки должен быть наиболее удобным для данной организации и внушающим доверие, результаты которого можно было бы воспроизвести.

СУИБ должна снижать уровень риска до приемлемого уровня. Это делается с помощью средств управления (требования к управляющему воздействию). Под средствами управления подразумеваются защитные меры. Они должны удовлетворить требованиям, выявленным на этапах оценки и обработки рисков. Технические меры защиты должны выбираться в соответствии с степенью риска для обеспечения функциональной пригодности и надежной системы безопасности. Функциональная пригодность системы должна включать в себя, как минимум:

- проведение идентификации и аутентификации пользователя;
- выполнение требований логического контроля допуска;
- обеспечение ведения контрольного журнала и регистрацию происходящих в системе безопасности событий;
- определение подлинности сообщений;
- шифрование информации.

Система безопасности должна обеспечивать необходимый уровень надежности при осуществлении функций безопасности и подвергаться различным проверкам и тестированиям безопасности, обеспечивающих подтверждение этого уровня.

Средства управления должны выбираться в соответствии с Приложением А стандарта ISO 27001 (они составлены из разделов 5–15 ISO/IEC 17799:2005).

Необходимо проводить оценку приемлемости рисков. После выбора защитных мер и идентификации снижения уровня риска в результате применения защитных мер всегда будут иметь место остаточные риски, поскольку система не может быть абсолютно безопасной. Эти остаточные риски должны оцениваться организацией как приемлемые или неприемлемые.

Аудиты должны быть спланирована с учетом статуса и важности процессов и областей, которые нужно проверять, а также результатов предыдущих аудитов. Должны быть определены критерии, область приложения, частота и методы аудита. Выбор аудиторов и проведение аудитов должны гарантировать объективность и беспристрастность процесса аудита. Аудиторы не должны проверять свою собственную работу. Аудиты СУИБ должны проверяться через запланированные интервалы времени.

Информационные системы и окружающая среда, в которой они функционируют, постоянно изменяются. Изменения информационных систем есть результат появления новых защитных мер и услуг или обнаружения новых угроз и уязвимостей. Данные изменения могут также привести к новым угрозам и образованию новых уязвимостей поэтому, когда

планируются или происходят изменения в информационной системе, важно определить, как это повлияет (если повлияет) на информационную безопасность системы в целом.

Таким образом, контроль конфигурации и управление изменениями должны включать следующее (требования к контролю конфигурации и управлению изменениями):

- обнаружение ошибок в результатах обработки;
- выявление предпринимаемых и успешных нарушений защиты и инцидентов;
- обнаружение событий в системе защиты информации;
- определение результативность мер защиты при нарушениях;
- измерение результативность средств управления для того, чтобы проверить, что требования защиты были удовлетворены;
- анализ оценки рисков через запланированные интервалы;
- анализ остаточных рисков;
- внедрение выявленных улучшений СУИБ;
- осуществление надлежащих корректирующих и предупреждающих действий.

СУИБ должна содержать подсистему управления инцидентами, которая собирает информацию по инцидентам, связанным с нарушением безопасности, и проводит ее анализ. Эта информация необходима для поддержки процесса анализа.

Подсистема обработки инцидентов должна включать:

- подготовку, которая должна содержать предупреждающие меры со стороны руководства организации, рекомендации и процедуры по обработке инцидентов (включая сохранение доказательных данных, обслуживание файлов регистрации данных по событиям и связь с общественностью), необходимые документы, планы обеспечения бесперебойной работы информационной системы;
- уведомление, которое должно содержать процедуры, средства и обязанности по регистрации информации об инцидентах;
- оценку, которая должна содержать процедуры и обязанности по расследованию инцидентов и определению уровня их опасности;
- управление, которое должно содержать процедуры и обязанности по обработке инцидентов, снижению ущерба от них и уведомлению вышестоящего руководства;
- восстановление, которое должно содержать процедуры и обязанности по восстановлению нормальной работы;
- анализ, который должен содержать процедуры и обязанности при выполнении действий после инцидента, включая расследование юридических аспектов инцидента и анализ тенденций.

Документация должна включать записи о решениях руководства, обеспечивать прослеживаемость действий до решений руководства и политики, а также обеспечивать воспроизводимость результатов.

Документация СУИБ должна включать следующее:

- процедуры и средства управления в поддержку СУИБ;
- описание методологии оценки рисков;
- отчет об оценке рисков;
- план обработки рисков;
- документированные процедуры, необходимые организации для того, чтобы гарантировать результативное планирование, работу и управление ее процессами защиты информации, а также для того, чтобы описать, как измерять результативность средств управления;
- записи, в которых должны быть отражены показатели процесса создания СУИБ, а также все эпизоды значительных инцидентов в системе безопасности.

Существует достаточно большое количество различных нормативных документов для построения СУИБ, которые дополняют друг друга. В документах прописываются этапы разработки, построения, эксплуатирования и модернизации системы. Для каждого этапа раскрываются общие методы, средства, способы, которые должны быть использованы для построения СУИБ. Важно, чтобы система могла активно реагировать на внутренние и внешние воздействия, принимать адекватные меры для борьбы с неблагоприятными воздействиями, а также могла гибко изменяться в зависимости от этих воздействий. Главная задача СУИБ состоит в снижении и удержании рисков на нужном уровне. С точки зрения систем управления управляющим параметром здесь являются риски, а регулирующим органом – применяемые средства защиты.

Контрольные вопросы

1. Каким стандартам необходимо следовать при построении СУИБ?
2. Что регламентируется в стандарте ISO 27002?
3. Что регламентируется в стандарте ISO 18044?
4. Что должна обеспечивать СУИБ?
5. Какова главная задача СУИБ?

ЗАКЛЮЧЕНИЕ

В заключение рассмотрения моделей безопасности компьютерных систем (МБКС) необходимо отметить, что на настоящий момент теория компьютерной безопасности – одна из самых развивающихся естественных наук. Постоянно появляются новые перспективные направления исследований, а уже имеющиеся получают все более глубокую научную проработку.

К числу первоочередных направлений следует отнести, в частности, формализацию положений теории компьютерной безопасности; разработку моделей безопасности, более точно отражающих существующий уровень развития компьютерной техники и информационных технологий и более удобных для практического использования и анализа защищенности реальных информационных систем; разработку средств и методов противодействия угрозам информационной войны.

В то же время следует отметить, что содержание настоящего пособия, не претендуя на полный охват современной теории, представляет комплексную, методологически обоснованную концепцию изложения ее положений.

Описанные в данном пособии основы теории МБКС не несут в себе ответов на вопросы, как защищать конкретную систему. Вместе с тем они включают общую методологию, позволяющую определить, какими защитными свойствами должна обладать проектируемая система и до какой степени, а также как должна вестись ее разработка.

В настоящее время Россия входит в число лидеров по уровню информационной безопасности. Как сказал президент Российской Федерации В.В. Путин, выступая на Международном конгрессе по кибербезопасности (ICC): «Эффективное цифровое развитие может быть основано только на цифровой свободе: на свободе для бизнеса, общественных организаций, для граждан, на снятии барьеров, которые сдерживают, ограничивают прогресс. Но при этом всем нам нужно понимать и свою ответственность, и потенциальные риски, угрозы, вызовы в цифровой сфере». И далее: «Планируется повысить уровень подготовки российских специалистов по противодействию киберпреступности, а для этого активнее внедрять практико-ориентированные подходы, использовать передовой зарубежный и российский опыт». В Российской Федерации известными центрами компьютерной безопасности являются такие учреждения, как Федеральная служба по техническому и экспортному контролю (ФСТЭК), Институт

криптографии, связи и информатики Академии федеральной службы безопасности (ИКСИ) и Академия криптографии Российской Федерации (АК РФ). Роль таких центров, российских и зарубежных, в развитии отдельных теоретических и практических областей компьютерной безопасности выражена, в частности, в том, что они являются центрами создания новых технологий, определяя перспективные направления дальнейшего движения научной мысли.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Девянин, П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие / П. Н. Девянин. – 2-е изд. доп. – Горячая линия – Телеком, 2013. – 338 с.

2. Башлы, П. Н. Информационная безопасность и защита информации: учебник / П. Н. Башлы, А. В. Бабаш, Е. К. Баранова. – М.: РИОР, 2013. – 222 с.

3. Защита информации: учеб. пособие / А. П. Жук [и др.]. [Электронный ресурс]. Режим доступа: НБ СФУ: http://znanium.com/catalog/product/405000. – М.: РИОР: ИНФРА-М, 2017. – 392 с.

Учебное издание

Богульская Нина Александровна
Кучеров Михаил Михайлович

МОДЕЛИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

Учебное пособие

Редактор *И. Н. Байкина*
Компьютерная верстка *Д. Р. Мазай*

Изображение на обложке: https://portail-qualite.public.lu

Подписано в печать 24.05.2019. Печать плоская. Формат 60×84/16
Бумага офсетная. Усл.-печ. л. 12,8. Тираж 100 экз. Заказ № 6395

Библиотечно-издательский комплекс
Сибирского федерального университета
660041, Красноярск, пр. Свободный, 82а
Тел. (391) 206-26-16; http://bik.sfu-kras.ru
E-mail: publishing_house@sfu-kras.ru