Д. С. Кулябов

Учебно-методическое пособие по курсу «Защита информации в компьютерных сетях» Часть 1

Для студентов направлений
«Прикладная математика и информатика» и
«Математика. Компьютерные науки»

Москва
2004

В пособии излагаются основные понятия безопасности. Рассмотрены стандарты в области информационной безопасности, защита операционных систем и основы криптографии.

Оглавление

1. Стандарты информационной безопасности

1.1. Основные понятия и определения

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

Модель безопасности (Security Model) — формальное представление политики безопасности.

Дискреционное управление доступом (Discretionary Access Control) — управление доступом, осуществляемое на основании заданного администратором множества разрешенных отношений доступа.

Мандатное управление доступом (Mandatory Access Control) — управление доступом, основанное на совокупности правил предоставления доступа, определенных на множестве атрибутов безопасности субъектов и объектов.

Ядро безопасности (Trusted Computing Base, TCB) — совокупность аппаратных, программных и специальных компонент вычислительной системы (ВС), реализующих функции защиты и обеспечения безопасности.

Идентификация (Identification) — процесс распознавания сущностей путем присвоения им уникальных меток.

Адекватность (Assurance) — показатель реально обеспечиваемого уровня безопасности, отражающий степень эффективности и надежности реализованных средств защиты и их соответствия поставленным задачам.

Таксономия (Taxonomy) — наука о систематизации и классификации сложно организованных объектов и явлений, имеющих иерархическое строение. Таксономия основана на декомпозиции явлений и поэтапном уточнении свойств объектов.

Прямое взаимодействие (Trusted Path) — принцип организации информационного взаимодействия, гарантирующий, что передаваемая информация не подвергнется перехвату или искажению.

1.2. Роль стандартов информационной безопасности

Главной задачей стандартов безопасности является создание основы для взаимодействия между производителями, потребителями и экспертами по квалификации продуктов информационных технологий.

Потребители заинтересованы в методике, позволяющей обоснованно выбрать продукт, отвечающий их потребностям. Для этого им необходима шкала оценки безопасности. К тому же потребители нуждаются в инструменте, с помощью которого они могли бы формулировать свои требования

производителям. При этом потребителей интересуют лишь характеристики и свойства конечного продукта, а не методы его получения.

Производителям стандарты необходимы в качестве средства сравнения возможностей их продуктов. Кроме того стандарты необходимы для процедуры сертификации, которая является механизмом объективной оценки свойств продуктов. С этой точки зрения требования должны быть максимально конкретными и регламентировать необходимость применения тех или иных средств, механизмов, алгоритмов и т.д. Кроме того, требования не должны противоречить существующим парадигмам обработки информации, архитектуре вычислительных систем и технологиям создания информационных продуктов.

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, а производителям — получить объективную оценку возможностей своего продукта.

Наиболее значимыми стандартами информационной безопасности являются: «Критерии безопасности компьютерных систем министерства обороны США», руководящие документы Гостехкомиссии России, «Европейские критерии безопасности информационных технологий», «Федеральные критерии безопасности информационных технологий США», «Канадские критерии безопасности компьютерных систем» и «Единые критерии безопасности информационных технологий».

1.3. Критерии безопасности компьютерных систем министерства обороны США («Оранжевая книга»)

1.3.1. Цель разработки

«Оранжевая книга» была разработана министерством обороны США в 1983 году с целью определения требований безопасности, предъявляемых к программному, аппаратному и специальному программному обеспечению компьютерных систем и выработки соответствующей методологии и технологии анализа степени поддержки политики безопасности в компьютерных системах военного назначения.

Предложенные в этом документе концепции защиты и набор функциональных требований послужили основой для формирования всех появившихся в последствии стандартов безопасности.

1.3.2. Таксономия требований и критериев

В «Оранжевой книге» предложены три критерия безопасности:
— политика безопасности
— аудит

— корректность

В рамках этих критериев сформулированы шесть базовых требований безопасности:

— политика безопасности
— метки
— идентификация и аутентификация
— регистрация и учет
— контроль корректности функционирования средств защиты
— непрерывность защиты

1.3.3. Классы безопасности компьютерных систем

«Оранжевая книга» предусматривает четыре группы критериев, которые соответствуют различной степени защищенности: от минимальной (группа D) до формально доказанной (группа A). Каждая группа включает один или несколько классов, характеризующихся различными требованиями безопасности. Уровень безопасности возрастает при движении от группы D к группе A, а внутри группы — с возрастанием номера класса.

Группа D. Минимальная защита.

Класс D. Минимальная защита.

К этому классу относятся все системы, которые не удовлетворяют требованиям других классов.

Группа C. Дискреционная защита.

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

Класс $C1$. Дискреционная защита.

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

Класс $C2$. Управление доступом.

Системы этого класса осуществляют более избирательное управление доступом с помощью применения средств индивидуального контроля за действиями пользователей, регистрацией, учетом событий и выделением ресурсов.

Группа B. Мандатная защита.

Основные требования этой группы — нормативное управление доступом с использованием меток безопасности, поддержка модели и политики безопасности, а также наличие спецификаций на функции ядра безопасности. Для систем этой группы монитор взаимодействия должен контролировать все события в системе [1, 2].

Класс $B1$. Защита с применением меток безопасности.

Системы этого класса должны соответствовать всем требованиям, предъявляемым к системам класса $C2$, и, кроме того, долж-

ны поддерживать определенную нормальную модель безопасности, маркировку данных и нормативное управление доступом.При экспорте из системы информация должна подвергаться маркировке. Обнаруженные в процессе тестирования недостатки, должны быть устранены.

Класс *B*2. Структурированная защита.

Для соответствия классу *B*2 ядро безопасности должно поддерживать формально определенную и четко документированную модель безопасности, предусматривающее произвольное и нормативное управление доступом, которое распространяется на все субъекты. Кроме того, должен осуществляться контроль скрытых каналов утечки информации. В структуре ядра безопасности должны быть выделены элементы, критичные с точки зрения безопасности. Интерфейс ядра безопасности должен быть четко определен, а его архитектура и реализация должны быть выполнены с учетом возможности проведения тестовых испытаний. По сравнению с классом *B*1 должны быть усилены средства аутентификации. Управление безопасностью осуществляется администраторами системы. Должны быть предусмотрены средства управления конфигурацией.

Класс *B*3. Домены безопасности.

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

Группа *A*. Верифицированная защита.

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

Класс *A*1. Формальная верификация.

Системы класса *A*1 функционально эквивалентны системам класса *B*3, и к ним не предъявляется никаких дополнительных функциональных требований. В отличие от систем класса *B*3 в ходе разработки должны применяться формальные методы верификации, что позволяет с высокой уверенностью получить корректную реализацию функций защиты. Процесс доказательства адекватности реализации начинается на ранней стадии разработки с построения формальной модели политики безопасности

и спецификаций высокого уровня. Для обеспечения методов верификации системы класса *A*1 должны содержать более мощные средства управления конфигурацией и защищенную процедуру дистрибуции.

1.4. Европейские критерии безопасности информационных технологий

«Европейские критерии безопасности информационных технологий» рассматривают следующие задачи средств информационной безопасности:

— защита информации от несанкционированного доступа с целью обеспечения конфидециальности
— обеспечение целостности информации посредством защиты от ее несанкционированной модификации или уничтожения
— обеспечение работоспособности систем с помощью противодействия угрозам отказа в обслуживании

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

1.4.1. Функциональные критерии

В «Европейских критериях безопасности информационных технологий» средства, имеющие отношение к информационной безопасности, рассматриваются на трех уровнях детализации:

1. цели, которые преследует обеспечение безопасности;
2. спецификации функций защиты;
3. механизмы спецификаций функций защиты.

Требования спецификаций функций защиты:

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

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

— аутентификация
— управление доступом
— конфидециальность данных
— целостность данных
— невозможность отказаться от совершенных действий

1.4.2. Классы безопасности

В «Европейских критериях безопасности информационных технологий» классов безопасности десять, пять из которых ($F-C1$, $F-C2$, $F-B1$, $F-B2$, $F-B3$) соответствуют критериям «Оранжевой книги» с аналогичными обозначениями.

Рассмотрим остальные классы безопасности:

— Класс $F-IN$.
Предназначен для систем с высокими потребностями в обеспечении целостности, что типично для систем управления базами данных. Его описание основано на концепции ролей, соответствующих видам деятельности пользователей, и предоставлении доступа к определенным объектам только посредством доверенных процессов. Должны различаться следующие виды доступа: чтение, запись, добавление, удаление, создание, переименование и выполнение объектов.

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

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

— Класс $F-DC$.
Уделяет особое внимание требованиям к конфидециальности информации. Информация по каналам связи должна передаваться в зашифрованном виде. Ключи шифрования должны быть защищены от несанкционированного доступа.

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

Уровни адекватности: уровень $E0$ — минимальная адекватность. При

проверке адекватности анализируется весь жизненный цикл системы — от начальной фазы проектирования до эксплуатации и сопровождения; уровень $E1$ — анализируется лишь общая архитектура системы, а адекватность средств защиты подтверждается функциональным тестированием; уровень $E3$ — к анализу привлекаются исходные тексты программ и схемы аппаратного обеспечения; уровень $E6$ — требуется формальное описание функций безопасности, общей архитектуры, а так же политики безопасности.

Три уровня безопасности:

— базовый
  если средства защиты способны противостоять отдельным случайным атакам;
— средний
  если средства защиты способны противостоять злоумышленникам, обладающим ограниченными ресурсами и возможностями;
— высокий
  если есть уверенность, что средства защиты могут быть преодолены только злоумышленником с высокой квалификацией, набор возможностей и ресурсов которого выходит за рамки возможного.

1.5. Руководящий документы Гостехкомиссии России

1.5.1. Таксономия требований и критериев

Руководящий документы Гостехкомиссии России [3] предлагают две группы критериев безопасности — показатели защищенности средств вычислительной техники от несанкционированного доступа и критерии защищенности автоматизированных систем обработки данных.

1.5.2. Показатели защищенности средств вычислительной техники от несанкционированного доступа

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

Данные показатели содержат требования защищенности средств вычислительной техники от несанкционированного доступа к информации и применяются к общесистемным программным средствам и операционным системам (с учетом архитектуры ЭВМ). Конкретные перечни показателей определяют классы защищенности средств вычислительной техники

и описываются совокупностью требований. Совокупность всех средств защиты составляет комплекс средств защиты.

Установлено семь классов защищенности (самый низкий класс — седьмой, самый высокий — первый). Показатели защищенности и установленные требования к классам приведены в табл. 1.1.

1.5.3. Требования к защищенности автоматизированных систем

Данные требования являются составной частью критериев защищенности автоматизированных систем обработки информации от несанкционированного доступа. Требования сгруппированы вокруг реализующих их подсистем защиты:

- подсистема управления доступом
  - идентификация, проверка подлинности, контроль доступа
  - управление потоками информации
- подсистема регистрации и учета
  - регистрация и учет
  - учет носителей информации
  - очистка освобождаемых носителей информации
  - сигнализация попыток нарушения защиты
- криптографическая подсистема
  - шифрование конфидециальной информации
  - шифрование информации, принадлежащей различным субъектам доступа на разных ключах
  - использование сертифицированных криптографических средств
- подсистема обеспечения целостности
  - обеспечение целостности программных средств и обрабатываемой информации
  - физическая охрана средств вычислительной техники и носителей информации
  - наличие администратора защиты информации
  - периодическое тестирование средств защиты от несанкционированного доступа
  - наличие средств восстановления средств защиты от несанкционированного доступа
  - использование сертифицированных средств защиты

1.5.4. Классы защищенности автоматизированных систем

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

Таблица 1.1

Распределение показателей защищенности по классам средств вычислительной техники

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

Обозначения:
«-» — нет требований к данному классу;
«+» — новые или дополнительные требования;
«=» — требования совпадают с требованиями к средствам вычислительной техники предыдущего класса.

— наличие в автоматизированных системах информации различного уровня конфидециальности
— уровень полномочий пользователей автоматизированных систем на доступ к конфидециальной информации
— режим обработки данных в автоматизированных системах (коллективный и индивидуальный)

В пределах каждой группы соблюдается иерархия классов защищенности автоматизированных систем.

Третья группа включает автоматизированные системы, в которых работает один пользователь, допущенный ко всей информации, размещенных на носителях одного уровня конфидециальности.

Вторая группа включает автоматизированные системы, в которых пользователи имеют одинаковые полномочия доступа ко всей информации, обрабатываемой и / или хранимой в автоматизированных системах на носителях различного уровня конфидециальности.

Первая группа включает многопользовательские автоматизированные системы, в которых обрабатывается и / или храниться информация различных уровней конфидециальности. Не все пользователи имеют равные права доступа.

1.5.5. Недостатки руководящих документов Гостехкомиссии России

— Отсутствие требований к защите от угроз работоспособности.
— Ориентация на противодействие несанкционированному доступу.
— Отсутствие требований адекватности реализации политики безопасности.
— Понятие «политика безопасности» трактуется исключительно как поддержание режима секретности и отсутствие несанкционированного доступа.
— Средства защиты ориентируются исключительно на противодействие внешним угрозам, а к структуре самой системы и ее функционировании не предъявляется никаких требований.
— Ранжирование требований по классам защищенности максимально упрощено и сведено до определения наличия / отсутствия заданного набора механизмов защиты, что существенно снижает гибкость требований и возможность их практического применения.

1.6. Федеральные критерии безопасности информационных технологий США

«Федеральные критерии безопасности информационных технологий США» представляют собой основу для разработки и сертификации компонентов информационных технологий с точки зрения обеспечения безопасности.

Цели создания данного документа:

— Определение универсального и открытого для дальнейшего развития базового набора требований безопасности, предъявляемых к современным информационным технологиям.
— Совершенствование существующих требований и критериев безопасности.
— Приведение в соответствие принятых в разных странах требований и критериев безопасности информационных технологий.
— Нормативное закрепление основополагающих принципов информационной безопасности.

«Федеральные критерии безопасности информационных технологий США» охватывают практически полный спектр проблем, связанный с защитой и обеспечением безопасности, т.к. включают все аспекты обеспечения конфидециальности, целостности и работоспособности.

Данный документ содержит положения, относящиеся только к отдельным продуктам информационных технологий, т.е. положения касаются только механизмов защиты, встроенных непосредственно в эти продукты в виде соответствующих программных, аппаратных или специальных средств. Но, в конечном счете, безопасность продукта информационных технологий определяется совокупностью собственных средств обеспечения безопасности и внешних средств, являющихся частью среды эксплуатации.

Ключевым понятием концепции информационной безопасности «Федеральные критерии безопасности информационных технологий США» является понятие «профиль защиты» (Protection Profile) — нормативный документ, который регламентирует все аспекты безопасности продукта информационных технологий в виде требований к его проектированию, технологии разработки и квалифицированному анализу.

«Федеральные критерии безопасности информационных технологий США» представляют процесс разработки систем обработки информации в виде следующих основных этапов:
— разработка и анализ профиля защиты;
— разработка и квалифицированный анализ продуктов информационных технологий;
— компановка и сертификация системы обработки информации в целом.

«Федеральные критерии безопасности информационных технологий США» регламентируют только разработку и анализ профиля защиты, а процесс создания продуктов информационных технологий и компановка систем обработки информации остаются вне рамок этого стандарта.

1.6.1. Назначение и структура профиля защиты

Профиль защиты предназначен для определения и обоснования состава и содержания средств защиты, сертификации технологии разработки и регламентации процесса квалификационного анализа продукта информационных технологий.

Профиль защиты состоит из следующих разделов:
— описание

содержит классификационную информацию для идентификации профиля в специальной картотеке;
содержит характеристику основной проблемы или группы проблем обеспечения безопасности, решаемых с помощью применения данного профиля;
— обоснование
содержит описание среды эксплуатации, предполагаемых угроз безопасности и методов использования продуктов информационных технологий;
содержит подробный перечень задач по обеспечению безопасности, решаемых с помощью применения данного профиля;
— функциональный требования к продукту информационных технологий
содержит описание функциональных возможностей средств защиты продуктов информационных технологий;
определяет условия, в которых обеспечивается безопасность в виде перечня угроз, которым успешно противостоят предложенные средства защиты (угрозы, ежащие вне этого диапазона, должны быть устранены с помощью дополнительных, не входящих в состав продукта, средств обеспечения безопасности);
— требования к технологии разработки продукта информационных технологий
содержат требования как к самому процессу разработки продукта информационных технологий, так и к условиям, в которых она проводится, к используемым технологическим средствам, а также к документированию этого процесса;
— требования к процессу квалификационного анализа продукта информационных технологий
регламентируют порядок проведения квалификационного анализа в виде методики исследования и тестирования продукта информационных технологий;

1.6.2. Схема разработки профиля защиты

Разработка профиля защиты осуществляется в три этапа:

1. анализ среды применения продукта информационных технологий с точки зрения безопасности:
   — предполагаемые угрозы;
   — слабые места в механизмах защиты;
   — политика безопасности;
   — нормативные документы и стандарты;
2. выбор прототипа в картотеке профилей;
3. профиль защиты:
   — описание;
   — обоснование;
   — функциональные требования;
   — требования к технологии разработки;
   — требования к процессу квалификационного анализа.

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

1.6.3. Таксономия функциональных требований

Функциональные требования к ядру безопасности:

- реализация политики безопасности:
  - политика аудита
    основная задача — обеспечение возможности однозначной идентификации субъекта, ответственного за те или иные действия в системе:
    - идентификация и аутентификация
      позволяют установить однозначное соответствие между пользователями и представляющими их в ВС субъектами разграничения доступа, а также подтвердить подлинность этого соответствия;
    - регистрация в системе
      происходит создание субъекта взаимодействия, с идентификатором которого будут ассоциироваться все дальнейшие действия пользователя;
      осуществляется учет места, времени и других параметров подключения к системе и ее блокирование во время отсутствия пользователя;
    - обеспечение прямого взаимодействия с ядром безопасности гарантирует, что пользователь взаимодействует с компонентами ядра безопасности напрямую, т.е. нет перехвата и искажения информации;
    - регистрация и учет событий
      позволяет распознать потенциально опасные ситуации и сигнализировать о случаях нарушения безопасности;
  - политика управления доступом:
    - произвольное управление доступом
      позволяет осуществлять назначение прав доступа с точностью до идентифицируемых субъектов и объектов, а также поддерживаемых типов доступа;
      обеспечивает контроль за распределением прав доступа среди субъектов;
    - нормативное управление доступом
      основано на контроле информационных потоков между субъектами и объектами и их атрибутов безопасности, что позволяет регламентировать порядок использования информации в системе и противостоять атакам типа «троянского коня»;
    - контроль скрытых каналов

содержит технические и административные меры, направленные на ликвидацию скрытых каналов посредством минимизации объема совместно используемых ресурсов и введения активных «шумовых помех»;

- политика обеспечения работоспособности
  позволяет гарантировать доступность ресурсов и обеспечение отказоустойчивости:
  - контроль за распределением ресурсов
    осуществляется посредством введения ограничений на потребление ресурсов или приоритетной системы распределения ресурсов;
  - обеспечение отказоустойчивости
    противостоит угрозам работоспособности;
- управление безопасностью
  регламентирует следующие аспекты функционирования системы:
  - компоновка, установка, конфигурация и поддержка ядра безопасности;
  - администрирование атрибутов безопасности пользователей (идентификаторов, полномочий, доступных ресурсов и т.д.);
  - администрирование политики управления доступом;
  - управление потреблением ресурсов системы;
  - аудит действий пользователей;
- мониторинг взаимодействий
  регламентирует порядок взаимодействия между компонентами системы и прохождение информационных потоков через ядро безопасности;
- логическая защита ядра безопасности
  устанавливает порядок доступа к внутренним компонентам ядра безопасности (данным и программам);
- физическая защита ядра безопасности
  задает ограничения на физический доступ к компонентам ядра безопасности, а также допустимые физические параметры среды функционирования ВС;
- самоконтроль ядра безопасности
  определяет возможности обеспечения контроля корректности выполнения функций ядра безопасности и целостности программ и данных, входящих в ядро безопасности;
- инициализация и восстановление ядра безопасности
  устанавливает возможности ядра безопасности по контролю за процессом собственной инициализации и способности к самовосстановлению после сбоев;
- ограничение привилегий при работе с ядром безопасности
  устанавливает порядок назначения полномочий для работы с ядром безопасности;
- простота использования ядра безопасности
  обеспечивает удобство пользования возможностями ядра безопасности как для высококвалифицированных администраторов, ответственных за функционирование и безопасность системы, так и для

рядовых пользователей, а также для разработчиков прикладных программ, взаимодействующих с ядром безопасности.

1.6.4. Ранжирование функциональных требований

Для обоснования выбора тех или иных требований и их непротиворечивости с существующими стандартами в области безопасности продуктов информационных технологий, функциональные требования ранжируются по уровням с помощью следующих четырех критериев:

- широта сферы применения
  определяется множеством сущностей, к которому могут быть применены данные требования, а именно:
  - пользователи системы, субъекты и объекты доступа
  - функции ядра безопасности и интерфейс взаимодействия с ядром безопасности
  - аппаратные, программные и специальные компоненты ядра безопасности
  - множество параметров конфигурации ядра безопасности
- степень детализации требований
  определяется множеством атрибутов сущностей, к которым применяются данные требования
- функциональный состав средств защиты
  определяется множеством функций, включенных в ядро безопасности для реализации той или иной группы функциональных требований
- обеспечиваемый уровень безопасности
  определяется условиями, в которых функциональные компоненты ядра безопасности способны противостоять заданному множеству угроз, отказам и сбоям

Ранжирование всегда предполагает установление некоторого отношения порядка. Однако, независимое ранжирование функциональных требований по каждому из описанных критериев, хотя и дает некоторое представление о различиях между функциональными возможностями средств защиты, не позволяет установить четкую, линейную шкалу оценки уровня безопасности. Поэтому в «Федеральных критериях безопасности информационных технологий США» отсутствуют рекомендации как по выбору и применению тех и или иных функциональных требований, так и по определению их роли в системе обеспечения безопасности. Вместо жестких указаний этот документ содержит согласованный с предшествующими ему стандартами ранжированный перечень функциональных требований и предоставляет разработчикам профиля защиты возможность самостоятельно сделать выбор необходимых методов и средств обеспечения безопасности.

1.6.5. Требования к технологии разработки продукта информационных технологий

Основное назначение данных требований — обеспечение адекватности условий разработки функциональным требованиям профиля защиты, и установить ответственность разработчика за корректность реализации этих требований.

Таксономия требований к технологии разработки продукта информационных технологий:

- процесс разработки
  - определение множества функций ядра безопасности в соответствии с функциональными требованиями
  - реализация ядра безопасности
    - определение состава функциональных компонент ядра безопасности, реализующих функции защиты
    - определение интерфейса ядра безопасности
    - декомпозиция ядра безопасности на функциональные модели
    - структуризация ядра безопасности на домены безопасности
    - минимизация функций и структуры ядра безопасности
  - адекватность реализации ядра безопасности
  - тестирование и анализ ядра безопасности
    - тестирование функций ядра безопасности
    - анализ возможностей нарушения безопасности
    - анализ скрытых каналов
- среда разработки
  - инструментальные средства
  - средства управления процессом разработки
  - процедура дистрибуции
- документирование
  - документирование функций ядра безопасности
  - полная документация на продукт информационных технологий
  - документирование тестирования и анализа продукта информационных технологий
    - документирование процесса тестирования функций
    - документирование анализа возможностей нарушения безопасности
    - документирование скрытых каналов
  - документирование среды и процесса разработки
- сопровождение
  - пользовательская документация
  - руководство по администрированию системы безопасности
  - процедура обновления версий и направления ошибок
  - процедура инсталляции

1.6.6. Требования к процессу квалификационного анализа продукта информационных технологий

Данные требования призваны обеспечить надежность и корректность процесса квалификационного анализа продукта информационных технологий.

Квалификационный анализ разбивается на три этапа:

- анализ
  - анализ архитектуры
  - анализ реализации
- контроль
  - контроль среды разработки
  - контроль процесса сопровождения продукта информационных технологий
- тестирование
  - тестирование функций ядра безопасности производителем продукта информационных технологий
  - независимое тестирование функций ядра безопасности

1.7. Канадские критерии безопасности компьютерных систем

«Канадские критерии безопасности компьютерных систем» разрабатывались как основа для оценки эффективности средств обеспечения безопасности компьютерных систем.

Цели:

- разработка единой шкалы критериев оценки безопасности компьютерных систем, позволяющей сравнивать системы обработки конфидециальной информации по степени обеспечения безопасности
- создание основы для разработки спецификаций безопасных компьютерных систем, которая могла бы использоваться разработчиками при проектировании подобных систем в качестве руководства для определения состава функций средств защиты
- разработка унифицированного подхода и стандартных средств для описания характеристик безопасных компьютерных систем

1.7.1. Базовые концепции

Все компоненты системы, находящиеся под управлением ядра безопасности, называются объектами.

Состояния объектов:

- объект-пользователь:
  пользователь представляет собой физическое лицо, взаимодействующее с компьютерной системой.

— объект-процесс:
процесс (или активный объект), представляющий пользователя в компьютерной системе, — это программа, выполнение которой инициировано пользователем.
— пассивный объект:
объект представляет собой пассивный элемент, над которым выполняют действия пользователи и процессы.

Все взаимодействия объектов контролируются в соответствии с реализованной в компьютерной системе политикой безопасности.

Тег — совокупность атрибутов безопасности, ассоциированных с пользователем, процессом или объектом. В качестве тега пользователя, процесса или объекта могут выступать соответствующий уникальный идентификатор, метка безопасности или целостности, криптографический ключ, талбица прав доступа или другие атрибуты в соответствии с реализованной в компьютерной системе политикой безопасности.

1.7.2. Структура канадских критериев безопасности компьютерных систем

Функциональные критерии:
— критерии конфидециальности:
  — контроль скрытых каналов;
  — произвольное управление доступом;
  — нормативное управление доступом;
  — повторное использование объектов;
— критерии целостности:
  — домены целостности;
  — произвольное управление целостностью;
  — нормативное управление целостностью;
  — физическая целостность;
  — возможность осуществления отката;
  — разделение ролей;
  — самотестирование;
— критерии работоспособности:
  — контроль за распределением ресурсов;
  — устойчивость к отказам и сбоям;
  — живучесть;
  — восстановление;
— критерии аудита:
  — регистрация и учет событий в системе;
  — идентификация и аутентификация;
  — прямое взаимодействие с ядром безопасности.

Критерии адекватности:
— архитектура системы;
— среда разработки:
  — процесс разработки;
  — управление конфигурацией в процессе разработки;
— контроль процесса разработки:

Таблица 1.2

Идентификаторы уровней

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

- разработка спецификаций;
- разработка архитектуры;
- создание рабочего проекта;
- реализация;

— постановка и сопровождение;
— документация:
- руководство пользователя по безопасности;
- руководство администратора безопасности;

— тестирование безопасности.

1.8. Единые критерии безопасности информационных технологий

1.8.1. Основные понятия

Задачи защиты — выражает потребность потребителей продуктов информационных технологий в противостоянии заданному множеству угроз безопасности или в необходимости реализации политики безопасности.

Профиль защиты — специальный нормативный документ, представляющий собой совокупность задач защиты, функциональных требований, требований адекватности и их обоснования. Служит руководством для разработчика продукта информационных технологий при создании проекта защиты.

Проект защиты — специальный нормативный документ, представляющий собой совокупность задач защиты, функциональных требований, требований адекватности, общих спецификаций средств защиты и их обоснования. В ходе квалификационного анализа служит описанием продукта информационных технологий.

1.8.2. Профиль защиты

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

Структура профиля защиты:

— введение
содержит информацию, необходимую для поиска профиля защиты в библиотеке профилей;
- идентификатор профиля защиты
уникальное имя, пригодное для поиска профиля защиты среди подобных ему профилей обозначения ссылок на него;
- обзор содержания
содержит краткую аннотацию профиля защиты, на основании которой потребитель может сделать вывод о соответствии данного профиля его запросам;

— описание продукта информационных технологий
содержит краткую характеристику продукта информационных технологий, его функциональное назначение, принципы работы, методы использования и т.д.;
— среда эксплуатации
содержит описание среды функционирования продукта информационных технологий с точки зрения безопасности;
  — условия эксплуатации
  содержит исчерпывающую характеристику продукта среды эксплуатации продукта информационных технологий с точки зрения безопасности, в том числе ограничения на условия его применения;
  — угрозы безопасности
  описание угроз безопасности, действующих в среде эксплуатации, которым должна противостоять защита продукта информационных технологий (для каждой угрозы должен быть указан ее источник, метод и объект воздействия);
  — политика безопасности
  определяет и объясняет правила политики безопасности, которая должна быть реализована в продукте информационных технологий;
— задачи защиты
отражают потребности потребителей в противодействии указанным угрозам безопасности и/или реализации политики безопасности;
  — задачи защиты продукта информационных технологий
  отражают потребности пользователей в противодействии угрозам безопасности и/или реализации политики безопасности;
  — другие задачи защиты
  отражают необходимость участия средств защиты продукта информационных технологий в противодействии угрозам безопасности и/или реализации политики безопасности совместно с другими компонентами информационных технологий;
— требования безопасности
содержит требования безопасности, которым должен удовлетворять продукт информационных технологий для решения задач защиты;
  — функциональные требования
  содержит типовые требования, предусмотренные «Едиными критериями безопасности информационных технологий» (необходимо обеспечить такой уровень детализации требований, который позволяет продемонстрировать их соответствие задачам защиты);
  — требования адекватности
  содержит ссылки на типовые требования уровней адекватности «Единых критериев безопасности информационных технологий», но допускает и определение дополнительных требований адекватности;
  — требования к среде эксплуатации
  может содержать функциональные требования и требования адекватности, которым должны удовлетворять компоненты ин-

формационных технологий, составляющих среду эксплуатации продуктов информационных технологий;
- дополнительные сведения
  содержит любую дополнительную информацию, которая может быть полезна для проектирования, разработки квалификационного анализа и сертификации продукта информационных технологий;
- обоснование
  должно демонстрировать, что профиль защиты содержит полное и связное множество требований, и что удовлетворяющий им продукт информационных технологий будет эффективно противостоять угрозам безопасности среды эксплуатации;
  - обоснование задач защиты
    должно демонстрировать, что задачи защиты, предложенные в профиле, соответствуют параметрам эксплуатации, и их решение позволит противостоять угрозам безопасности и реализовать политику безопасности;
  - обоснование требований безопасности
    показывает, что требования безопасности позволяют эффективно решить задачи защиты.

1.8.3. Проект защиты

Проект защиты содержит требования и задачи защиты продукта информационных технологий, а так же описывает уровень функциональных возможностей реализованных в нем средств защиты, их обоснование и подтверждение степени их адекватности.

Структура проекта защиты:
- введение
  содержит информацию, необходимую для идентификации проекта защиты, определения назначения, а так же обзор его содержания;
  - идентификатор
    представляет собой уникальное имя проекта защиты, необходимое для поиска и идентификации проекта защиты и соответствующего ему продукта информационных технологий;
  - обзор содержания
    представляет собой достаточно подробную аннотацию проекта защиты, позволяющую потенциальным потребителям определить пригодность продукта информационных технологий для решения их задач;
  - заявка на соответствие «Единым критериям безопасности информационных технологий»
    содержит описание всех свойств продукта информационных технологий, подлежащих квалификационному анализу на основе «Единых критериев безопасности информационных технологий»;
- описание продукта информационных технологий;
- среда эксплуатации:
  - условия эксплуатации;

  - угрозы безопасности;
  - политика безопасности;
- задачи защиты:
  - задачи защиты продукта информационных технологий;
  - другие задачи защиты;
- требования безопасности
  содержат требования безопасности к продукту информационных технологий, которыми руководствовался производитель в ходе его разработки:
  - функциональные требования
    допускает использование помимо типовых требований «Единых критериев безопасности информационных технологий» других, специфичных для данного продукта и среды его эксплуатации;
  - требования адекватности
    могут включать уровни адекватности, не предусмотренные в «Единых критериях безопасности информационных технологий»;
  - требования к среде эксплуатации;
- общие спецификации продукта информационных технологий
  описывают механизмы осуществления задач защиты с помощью определения высокоуровневых спецификаций средств защиты в соответствии с предъявляемыми функциональными требованиями и требованиями адекватности:
  - спецификации функций защиты
    описывают функциональные возможности средств защиты продукта информационных технологий, заявленные его производителем как реализующие требования безопасности;
  - спецификации уровня адекватности
    определяют заявленный уровень адекватности защиты продукта информационных технологий и его соответствие требованиям адекватности в виде представления параметров технологии проектирования и создания продукта информационных технологий;
- заявка на соответствие профилю защиты
  содержит материалы, необходимые для подтверждения заявки:
  - ссылка на профиль защиты
    однозначно идентифицирует профиль защиты, на реализацию которого претендует проект защиты, с указанием случаев, в которых обеспечиваемый уровень защиты превосходит требования профиля;
  - соответствие профилю защиты
    определяет возможности продукта информационных технологий, которые реализуют задачи и требования, содержащиеся в профиле защиты;
  - усовершенствование профилz защиты
    отражают возможности продукта информационных технологий, которые выходят за рамки задач защиты и требований, установленных в профиле защиты;
- обоснование
  должно показывать, что проект защиты содержит полное и связное

множество требований, что реализующий его продукт информационных технологий будет эффективно противостоять угрозам безопасности, действующим в среде эксплуатации, и что общие спецификации функций защиты соответствуют требованиям безопасности:

— обоснование задач защиты
должно демонстрировать, что задачи защиты, предложенные в проекте защиты, соответствуют свойствам среды эксплуатации, и что их решение позволит эффективно противостоять угрозам безопасности и реализовать требуемую политику безопасности;
— обоснование требований безопасности
показывает, что выполнение этих требований позволяет решить задачи защиты;
— обоснование общих спецификаций продукта информационных технологий
должно демонстрировать, что средства защиты и методы обеспечения их адекватности соответствуют предъявляемым требованиям;
— обоснование соответствия профилю защиты
показывают, что требования проекта защиты поддерживают все требования профиля защиты.

1.8.4. Требования безопасности

Требования безопасности делятся на:

— функциональные требования
регламентируют функционирование обеспечивающих безопасность компонентов продукта информационных технологий и определяют возможности средств защиты;
— требования адекватности
представляет собой характеристику продукта информационных технологий, которая показывает, насколько эффективно обеспечивается заявленный уровень безопасности, а также степень корректности реализации средств защиты.

Таксономия функциональных требований

— классы:
  — название;
  — описание класса;
  — разделы:
    — название и обозначение используется для ссылок на раздел; каждый раздел имеет свое уникальное название и семисимвольный идентификатор, состоящий из префикса — трехбуквенного идентификатора класса, знака подчеркивания и трехбуквенного обозначения раздела;
    — описание разделы;
    — ранжирование требований;
    — управляемые параметры

в данном пункте могут быть перечислены параметры, путем настройки которых должно осуществляться управление средствами защиты, реализующими требования данного раздела;
- объекты регистрации и учета
  в данном пункте требований перечислены операции и события, которые должны являться объектами регистрации и учета;
- требования:
  - название
    уникальное название требования, которое используется для ссылок на него в профиле и проекте защиты;
  - содержание
    «Единые критерии безопасности информационных технологий» разрешают использовать требования только без изменений, что обеспечивает их стандартизацию;
  - сопряженные требования
    содержит список требований различных разделов и классов, выполнение которых рассматривается как необходимое предварительное условие для реализации данного требования.

Таксономия классов функциональных требований
- аудит:
  - автоматическое реагирование на попытки нарушения безопасности;
  - регистрация и учет событий;
  - анализ протокола аудита;
  - доступ к протоколу аудита;
  - отбор событий для регистрации и учета;
  - протокол аудита;
  - постобработка протокола аудита;
- подтверждение приема/передачи информации:
  - предотвращение отречения от факта передачи информации;
  - предотвращение отречения от факта получения информации;
- криптография:
  - управление ключами;
  - криптографические средства;
- защита информации:
  - политика управления доступом;
  - средства управления доступом;
  - аутентификация информации;
  - экспорт информации из системы;
  - политики управления информационными потоками;
  - средства управления информационными потоками;
  - импорт информации;
  - защита информации при передаче по внутренним каналам;
  - уничтожение остаточной информации;

  - откат;
  - контроль целостности информации в процессе хранения;
  - защита внутрисистемной передачи информации при использовании внешних каналов;
  - целостность внутрисистемной передачи информации при использовании внешних каналов;
- идентификация и аутентификация:
  - реакция на неудачные попытки аутентификации;
  - атрибуты безопасности пользователей;
  - аутентификационные параметры пользователей;
  - аутентификация пользователей;
  - идентификация пользователей;
  - соответствие пользователей и субъектов;
- управление безопасностью:
  - управление средствами защиты;
  - управление атрибутами безопасности;
  - управление параметрами и конфигурацией средств защиты;
  - отзыв атрибутов безопасности;
  - ограничение времени действия атрибутов безопасности;
  - административные роли;
- конфидециальность работы в системе:
  - анонимность пользователей;
  - использование псевдонимов;
  - анонимность сеансов работы с системой;
  - защита от мониторинга сеансов работы с системой;
- надежность средств защиты:
  - тестирование программно-аппаратной платформы;
  - защита от сбоев;
  - готовность средств защиты к обслуживанию удаленных клиентов;
  - конфидециальность передаваемой информации при работе с удаленными клиентами;
  - целостность передаваемой информации при работе с удаленными клиентами;
  - защита внутренних каналов информационного обмена между средствами защиты;
  - физическая защита;
  - безопасность восстановления после сбоев;
  - распознавание повторных передач информации и имитации событий;
  - мониторинг взаимодействий;
  - разделение доменов;
  - синхронизация;
  - время;
  - согласованность обмена информацией между средствами защиты;
  - репликация информации, используемой средствами защиты;
  - самотестирование средств защиты;
- контроль за использованием ресурсов:

    - отказоустойчивость;
    - распределение ресурсов на основе приоритетов;
    - квотирование ресурсов;
- контроль доступа к системе:
    - ограничение на использование атрибутов безопасности;
    - ограничения числа одновременных сеансов;
    - блокировка сеанса работы с системой;
    - объявления, предупреждения, приглашения и подсказки;
    - протокол сеансов работы с системой;
    - управление сеансами работы с системой;
- прямое взаимодействие:
    - прямое взаимодействие между средствами защиты;
    - прямое взаимодействие с пользователями.

Таксономия требований адекватности

- управление проектом:
    - средства управления проектом;
    - управление версиями;
    - конфигурация проекта;
- дистрибуция:
    - постановка;
    - установка, настройка, запуск;
- разработка:
    - общие функциональные спецификации;
    - архитектура защиты;
    - форма представления продукта на сертификацию;
    - структура средств защиты;
    - частные спецификации средств защиты;
    - соответствие описаний различного уровня;
    - политика безопасности;
- документация:
    - руководства администратора;
    - руководства пользователя;
- процесс разработки:
    - безопасность среды разработки;
    - исправление ошибок и устранение уязвимостей;
    - технология разработки;
    - средства разработки;
- тестирование:
    - полнота тестирования;
    - глубина тестирования;
    - методика тестирования;
    - независимое тестирование;
- анализ защиты:
    - анализ скрытых каналов;
    - анализ возможностей неправильного использования средств защиты;
    - анализ стойкости средств защиты;

— анализ продукта на наличие уязвимостей.

Ранжирование требований адекватности. Ранжирование требований адекватности представлено в виде упорядоченных списков. Критерии адекватности используются в ходе квалификационного анализа продукта информационных технологий для определения степени корректности реализации функциональных требований и назначения продукту информационных технологий соответствующего уровня адекватности.

Уровни адекватности:

1. функциональное тестирование
   предназначено для тех случаев, когда угрозам безопасности не придается большого значения, т.е. существует независимая гарантия того, что в состав продукта входят средства защиты персональной и/или подобной информации, реализованные в соответствии с указанными требованиями;
2. структурное тестирование
   используется, когда пользователи и разработчики согласны удовлетворится низкой или умеренной степенью независимого подтверждения адекватности обеспечиваемого уровня безопасности (рекомендуется применять для унаследованных систем, уже находящихся в эксплуатации);
3. методическое тестирование и проверка
   применяется, когда пользователям и разработчикам требуются умеренная степень независимого подтверждения свойств продукта информационных технологий, а также полное и последовательное исследование свойств продукта и контроль в процессе создания, но без проведения дорогостоящего обратного проектирования;
4. методическая разработка, тестирование и анализ
   применяется, когда пользователи и разработчики требуют умеренную или высокую степень независимого подтверждения адекватности защиты продукта информационных технологий и готовы нести определенные дополнительные технические затраты;
5. полуформальные методы разработки и тестирование
   применяется, когда пользователи и разработчики требуют высокую степень независимого подтверждения адекватности защиты продукта информационных технологий, а также строгого применения определенных технологий разработки продукта информационных технологий, но без чрезмерных затрат;
6. полуформальные методы верификации разработки и тестирование
   применяются при разработке защищенных продуктов, предназначенных для использования в ситуациях с высокой степенью риска, где ценность защищаемой информации оправдывает дополнительные затраты;
7. формальные методы верификации разработки и тестирование
   могут быть использованы для разработки защищенных продуктов, предназначенных для использования в ситуациях с исключительно высокой степенью риска, или там, где ценность защищаемых объектов оправдывает высокие дополнительные затраты.

Жесткость требований адекватности возрастает от первого уровня к седьмому.

1.9. Сопоставление стандартов

Универсальность стандарта определяется множеством типов ВС и областью информационных технологий, к которым могут быть корректно применены его приложения.

Гибкость стандарта — это возможность его применения к постоянно развивающимся информационным технологиям и время его «устаревания» (гибкость может быть достигнута исключительно через фундаментальность требований и критериев и их инвариантность по отношению к механизмам реализации и технологиям создания продукта информационных технологий).

Гарантированность определяется мощностью предусмотренных стандартом методов и средств подтверждения надежности результатов квалификационного анализа.

Реализуемость — это возможность адекватной реализации требований и критериев стандарта на практике, с учетом затрат на этот процесс.

Актуальность отражает соответствие требований и критериев стандарта постоянно развивающемуся множеству угроз безопасности и новейшим методам и средствам, используемым злоумышленниками.

Степень соответствия стандартов предложенным показателям определяется по следующей качественной шкале:

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

Таблица 1.3

Сопоставление стандартов

Стандарты безопасности	Показатели сопоставления стандартов информационной безопасности				
	универсальность	гибкость	гарантированность	реализуемость	актуальность
Оранжевая книга (1983 г.)	ограниченная	ограниченная	ограниченная	высокая (за исключением класса А)	умеренная
Европейские критерии (1986 г.)	умеренная	умеренная	умеренная	высокая	умеренная
Документы ГТК (1992 г.)	ограниченная	ограниченная	отсутствует	высокая	ограниченная
Федеральные критерии (1992 г.)	высокая	отличная	достаточная	высокая	высокая
Канадские критерии (1993 г.)	умеренная	достаточная	достаточная	достаточная	достаточная
Единые критерии (1999 г.)	превосходная	превосходная	превосходная	превосходная	превосходная

2. Понятие о моделях безопасности ОС

2.1. Необходимость наличия встроенных средств защиты на уровне операционной системы

Увеличение осведомленности в необходимости защиты вылилось в увеличение количества попыток интегрировать защитные механизмы в компьютерные системы. Впрочем, эти попытки страдают от неверного предположения, что адекватная степень защиты может быть достигнута на уровне приложений без использования определенных механизмов защиты операционной системы. В реальности, механизмы защиты операционной системы играют решающую роль в обеспечении безопасности на более высоких уровнях.

Необходимость наличия встроенных средств защиты на уровне операционной системы бесспорна [1, 2]; операционная система обеспечивает защиту механизмов прикладного уровня от неправильного использования, обхода или навязывания ложной информации. Если она не сможет удовлетворить этим требованиям, появятся уязвимости в масштабах всей системы. Потребность в защищенных операционных системах особенно важна в современных компьютерных средах. Значительное увеличение возможностей взаимодействия и совместного использования данных увеличило риск использования систем настолько, что даже осторожный и опытный пользователь, использующий однопользовательскую систему, больше не защищен от угрозы злоумышленного кода (программ). В связи с тем, что разница между кодом программы и данными исчезает, злоумышленный код может быть введен в систему, причем без осознанного решения пользователя установить исполняемый код, когда бы данные не вводились в систему. Например, злоумышленный код может быть введен как Java-апплет или при просмотре кажущихся безобидными данных, в действительности содержащих исполняемый код.

2.2. Формальные модели безопасности

2.2.1. Основные понятия

Политика безопасности — совокупность норм и правил, регламентирующих процесс обработки информации, выполнение которых обеспечивает защиту от определенного множества угроз и составляет необходимое (а иногда и достаточное) условие безопасности системы.

Модель политики безопасности — формальное выражение политики безопасности.

Только с помощью формальных моделей безопасности можно доказать безопасность системы, опираясь при этом на объективные и неопровержимые постулаты математической теории.

Назначение моделей безопасности состоит в том, что они позволяют обосновать жизнеспособность системы и определяют базовые принципы ее архитектуры и используемые при ее построении технологические решения.

Основная цель создания политики безопасности 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 [4].

2.2.2. Применение формальных моделей безопасности

Формальные модели безопасности позволяют решить ряд задач, возникающих в ходе проектирования, разработки и сертификации защищенных систем.

Производители защищенных информационных систем используют модели безопасности в следующих случаях:

- при составление 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 защищенных информационных систем используют формальные модели безопасности для формулирования своих требований к производителю в четко определенной и непротиворечивой форме и для оценки соответствия защищенных систем своим потребностям.

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

2.2.3. Базовые представления моделей безопасности

Все рассматриваемые модели безопасности основаны на следующих базовых представлениях.

1. Система является совокупностью взаимодействующих сущностей — субъектов и объектов. Безопасность обработки информации обеспечивается путем решения задачи управления доступом объектов к субъектам в соответствии с заданным набором правил и ограничений, образующих политику безопасности. Система безопасна, если субъекты не имеют возможности нарушить правила политики безопасности. Общим подходом для всех моделей является 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 определяет состояние системы. Каждое состояние системы является либо безопасным, либо небезопасным в соответствии с предложенным в модели критерием безопасности.
6. Основной элемент модели безопасности — это доказательство утверждения (теоремы) о том, что система, находящаяся в безопасном состоянии, не может перейти в небезопасное состояние при соблюдении всех установленных правил и ограничений.

Среди моделей политик безопасности можно выделить два основных класса: *дискреционные (произвольные)* и *мандатные (нормативные)*.

2.2.4. Мандатная и дискреционная модели

Приведем определения различных политик безопасности.

В TCSEC [5,4] приведено довольно узкое определение мандатной безопасности, которое тесно связано с политикой многоуровневой безопасности Министерства Обороны США. Оно стало общепринятым определением мандатной безопасности, но это определение является недостаточным, т.к. оно игнорирует такие важные качества, как непередаваемость и динамическое разделение обязанностей [6]. Это определение фактически сводит дискреционную политику безопасности к модели Харрисона-Руззо-Ульмана, а мандатную — к модели Белла-ЛаПадулы. В результате такие политики, как ролевая и DTE просто выпадают из этой классификации. Вместо этого будут употребляются более общие определения мандатной и дискреционной политик безопасности, данные в [7].

Таким образом:

Мандатной политикой безопасности будем считать любую политику, логика и присвоение атрибутов безопасности которой строго контролируются системным администратором политики безопасности.

Дискреционной политикой безопасности будем считать любую политику, в которой обычные пользователи могут принимать участие в определении функций политики и/или присвоении атрибутов безопасности.

Ролевая политика безопасности представляет собой существенно

усовершенствованную модель Харрисона-Руззо-Ульмана [4] (управление доступом в ней осуществляется как на основе матрицы прав доступа для ролей, так и с помощью правил, регламентирующих назначение ролей пользователям и их активацию во время сеансов).

Политика доменов и типов (Domain and Type Enforcement — *DTE*) модель, являющаяся расширением типизированной матрицы доступа, в которой типы приписаны не только объектам, но и субъектам.

2.2.5. Дискреционная модель Харрисона-Руззо-Ульмана

Данная модель реализует произвольное управление доступом субъектов к объектам и контроль за распространением прав доступа.

2.2.5.1. Обозначения

Обозначим:

S — множество субъектов (осуществляют доступ к информации);

O — множество объектов, содержащих защищаемую информацию;

$R = \{r_1, \ldots, r_n\}$ — конечное множество прав доступа, означающих полномочия на выполнение соответствующих действий (чтение, запись, выполнение).

Принято считать, что $S \subset O$, т.е. субъекты одновременно являются и объектами (это сделано для того, чтобы включить в область действия модели отношения между субъектами).

Поведение системы моделируется с помощью понятия *состояния*.

$O \times S \times R$ — пространство состояний системы;

M — матрица прав доступа, описывающая текущие права доступа субъектов к объектам (строки — субъекты, столбцы — объекты);

$Q = (S, O, M)$ — текущее состояние системы.

Любая ячейка матрицы $M[s, o]$ содержит набор прав доступа s к объекту o, принадлежащих множеству прав доступа R.

2.2.5.2. Поведение системы

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

command $\alpha(x_1, \ldots, x_k)$
    if r_1 in $M[x_{s_1}, x_{o_1}]$ and (условия выполнения команды)
        r_2 in $M[x_{s_2}, x_{o_2}]$ and
        ...
        r_m in $M[x_{s_m}, x_{o_m}]$ and
    then
        $op_1, op_2, \ldots, op_n$ (операции, составляющие команду)

где α — имя команды; x_i — параметры команды, являющиеся идентификаторами субъектов и объектов; s_i и o_i — индексы субъектов и объектов в диапазоне от 1 до k; op_i — элементарные операции (выполняются только

в том случае, если все условия, означающие присутствие указанных прав доступа в ячейках матрицы M, являются истинными).

2.2.5.3. Элементарные операции

В классической модели допустимы только следующие элементарные операции:

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

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

enter r into $M[s,o]$ (где $s \in S, o \in O$)

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

Данная операция вводит право r в существующую ячейку матрицы доступа. Содержимое ячейки рассматривается как множество, т.е. если это право уже имеется, то ячейка не изменяется. Операция enter называется монотонной, т.к. она только добавляет права в матрицу и ничего не удаляет. Предусловие выполнения операции — существование ячейки (существование соответствующих субъекта и объекта).

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

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

Данная операция удаляет право r из ячейки матрицы доступа, если оно там присутствует. Содержимое ячейки рассматривается как множество, т.е. если удаляемое право отсутствует в данной ячейке, то данная операция ничего не делает. Операция delete называется немонотонной, т.к. она удаляет информацию из матрицы. Предусловие выполнения операции — существование ячейки (существование соответствующих субъекта и объекта).

create subject s (где $s \notin S$)

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

Операция является монотонной. Предусловие выполнения операции — отсутствие создаваемого субъекта / объекта.
destroy subject s (где $s \in S$)
$O' = O \backslash \{s\}$
$S' = S \backslash \{s\}$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S' \times O'$
Операция является немонотонной. Предусловие выполнения операции — наличие субъекта / объекта.
create object o (где $o \notin O$)
$O' = O \cup \{o\}$
$S' = S$
$M'[x_s, x_o] = M[x_s, x_o]$, если $(x_s, x_o) \in S \times O$
$M'[x_s, o] = \varnothing$ для всех $x_s \in S'$
Операция является монотонной. Предусловие выполнения операции — отсутствие создаваемого субъекта / объекта.
destroy object o (где $o \in O \backslash S$)
$O' = O \backslash \{o\}$
$S' = S$
$M'[x_s, x_o] = M[x_s, x_o]$ для всех $(x_s, x_o) \in S' \times O'$
Операция является немонотонной. Предусловие выполнения операции — наличие субъекта / объекта.

2.2.5.4. Формальное описание системы $\sum(Q, R, C)$

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

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

Поведение системы во времени моделируется с помощью последовательности состояний Q_i, причем $Q_{i+1} = C_i(Q_i)$, где C — множество команд. Попадание системы в то или иное состояние для заданного начального состояния зависит только от условий команд из C и составляющих их операций. Каждое состояние определяет отношения доступа, которые существуют между сущностями системы в виде множества субъектов, объектов и матрицы прав. Должна существовать возможность определить множество состояний системы, в которые она сможет попасть из заданного начального состояния (т.к. для обеспечения безопасности необходимо наложить запрет на некоторые отношения доступа). Это позволит задавать такие начальные условия (интерпретацию команд C, множества объектов O_0, субъектов S_0 и матрицу доступа M_o), при которых система никогда не сможет попасть в состояния, нежелательные с точки зрения безопасности.

Следовательно для построения системы с предсказуемым поведением необходимо для заданных начальных условий получить ответ на вопрос: сможет ли некоторый субъект s когда-либо приобрести право доступа r для некоторого объекта o?

2.2.5.5. Критерий безопасности модели Харрисона-Руззо-Ульмана

Критерий формулируется следующим образом:

Критерий 2.1. *Для заданной системы начальное состояние* $Q_0 = (S_0, O_0, M_0)$ *является безопасным относительно права* r*, если не существует применимой к* Q_0 *последовательности команд, в результате которой право* r *будет занесено в ячейку памяти матрицы* M*, в которой оно отсутствовало в состоянии* Q_0.

Смысл данного критерия состоит в том, что для безопасной конфигурации системы субъект никогда не получит право r доступа к объекту, если он не имел его изначально.

Удаление субъекта или объекта приводит к уничтожению всех прав в соответствующей строке или столбце матрицы, но не влечет за собой уничтожение самого столбца или строки и сокращение размера матрицы. Следовательно, если в какой-то ячейке в начальном состоянии существовало право r, и после удаления субъекта или объекта, к которым относилось это право, ячейка будет очищена, но впоследствии появится вновь (в результате создания субъекта или объекта), и в эту ячейку с помощью соответствующей команды enter снова будет занесено право r, то это не будет означать нарушения безопасности.

Из критерия безопасности следует, что для данной модели ключевую роль играет выбор значений прав доступа и их использование в условиях команд. Хотя модель не налагает никаких ограничений на смысл прав и считает их равнозначными, те из них, которые участвуют в условиях выполнения команд, фактически представляют собой не права доступа к объектам, а права управления доступом, или права на осуществление модификаций ячеек матрицы доступа.

Таким образом, данная модель описывает не только доступ субъектов к объектам, но и распространение прав доступа от субъекта к субъекту, т.к. именно изменение содержания ячеек матрицы доступа определяет возможность выполнения команд (в том числе команд, модифицирующих саму матрицу доступа), которые потенциально могут привести к нарушению критерия безопасности.

2.2.5.6. Положительные и отрицательные стороны модели Харрисона-Руззо-Ульмана

Положительные моменты:

— данная модель является простой в реализации (т.к. не требует применения сложных алгоритмов);

— данная модель является эффективной в управлении (т.к. позволяет управлять полномочиями пользователей с точностью до операции над объектом);
— критерий безопасности данной модели является весьма сильным в практическом плане (т.к. позволяет гарантировать недоступность определенной информации для пользователей, которым изначально не выданы соответствующие полномочия).

Отрицательные моменты:

— доказано, что в общем случае не существует алгоритма, который может для произвольной системы, ее начального состояния $Q_0(S_0, O_0, M_0)$ и общего правила r решить, является ли данная информация безопасной;
— существует уязвимость к атаке с помощью «троянского коня» (т.к. в дискреционных моделях контролируются только операции доступа субъектов к объектам, а не потоки информации между ними).

2.2.6. Мандатная модель Белла-ЛаПадулы

Мандатная модель управления доступом основана на правилах секретного документооборота, принятых в государственных и правительственных учреждениях многих стран.

Основным положением политики безопасности является назначение всем участникам процесса обработки защищаемой информации и документам, в которых она содержится, специальной метки (секретно, совершенно секретно и т.д.). Такая метка называется *уровнем безопасности*. Все уровни безопасности упорядочиваются с помощью установленного отношения доминирования.

Контроль доступа осуществляется в зависимости от уровней безопасности взаимодействующих сторон на основании двух правил:

1. уполномоченное лицо (субъект) имеет право читать только те документы, уровень безопасности которых не превышает его собственный уровень безопасности;
2. уполномоченное лицо (субъект) имеет право заносить информацию только в те документы, уровень безопасности которых не ниже его собственного уровня безопасности.

Таким образом, мандатные модели управляют доступом неявным образом — с помощью назначения всем сущностям системы уровней безопасности, которые определяют все допустимые взаимодействия между ними. Следовательно, мандатное управление доступом не различает сущностей, которым присвоен одинаковый уровень безопасности, и на их взаимодействия ограничения отсутствуют. Поэтому в тех ситуациях, когда управление доступом требует более гибкого подхода, мандатная модель применяется совместно с какой-либо дискреционной, которая используется для контроля за взаимодействиями между сущностями одного уровня и для установки дополнительных ограничений, усиливающих мандатную модель.

2.2.6.1. Обозначения

Обозначим:

S — множество субъектов (осуществляют доступ к информации);

O — множество объектов, содержащих защищаемую информацию;

Принято считать, что $S \subset O$, т.е. субъекты одновременно являются и объектами (это сделано для того, чтобы включить в область действия модели отношения между субъектами).

$R = \{read, write\}$ — множество прав доступа, означающих полномочия на выполнение соответствующих действий (чтение, запись);

L — множество уровней безопасности;

Λ — решетка уровней безопасности (это формальная алгебра $(L, \leqslant, \bullet, \otimes)$, где оператор $\leqslant$ определяет частичное нестрогое соответствие порядка для базового множества уровней безопасности L);

V — множество состояний системы, которое представляется в виде набора упорядоченных пар (F, M), где

M — матрица доступа, отражающая текущую ситуацию с правами домступа субъектов к объектам (содержание матрицы аналогично матрице прав доступа в модели Харрисона-Руззо-Ульмана, но набор прав ограничен правами $read$ и $write$);

F — функция уровня безопасности;

$\sum(v_0, R, T)$ — модель системы;

v_0 — начальное состояние системы

$T : (V \times R) \to V$ — функция перехода, которая в ходе выполнения запросов переводит систему из одного состояния в другое; система, находящаяся в состоянии $v \in V$ при получении запроса $r \in R$, переходит в следующее состояние $v^* = T(v, r)$;

состояние v достижимо в системе $\sum(v_0, R, T) \Leftrightarrow$ $\exists\{(r_0, v_0), \ldots, (r_{n-1}, v_{n-1}), (r_n, v)\} : T(r_i, v_i) = v_{i+1}$ для $0 \leqslant i < n$;

v_0 тривиально достижимо.

Уровни безопасности субъектов и объектов задаются с помощью функции уровня безопасности $F : S \cup O \to L$, которая ставит в соответствие каждому объекту и субъекту уровень безопасности, принадлежащий множеству уровней безопасности L, на котором определена решетка Λ.

2.2.6.2. Решетка уровней безопасности Λ

Решетка уровней безопасности Λ — это формальная алгебра $(L, \leqslant, \bullet, \otimes)$, где оператор $\leqslant$ определяет частичное нестрогое соответствие порядка для базового множества уровней безопасности L (т.е. оператор $\leqslant$ — антисимметричен, транзитивен, рефлексивен).

Отношение $\leqslant$ на L:

1) ревлексивно, если $\forall a \in L : a \leqslant a$;
   Нет смысла запрещать потоки инвормации между сущностями одного и того же класса (сущности одного класса с точки зрения безопасности содержат одинаковую информации).
2) антисимметрично, если $\forall a_1, a_2 \in L : ((a_1 \leqslant a_2) \wedge (a_2 \leqslant a_1)) \Rightarrow a_1 = a_2$;
   Антисимметричность необходима для удаления избыточных классов (если информация может передаваться как от сущностей класса A к

сущностям класса B, так и наоборот, то классы A и B содержат одноуровневую информацию и с точки зрения безопасности эквивалентны классу (AB)).

3) транзитивно, если $\forall a_1, a_2, a_3 \in L : ((a_1 \leqslant a_2) \wedge (a_2 \leqslant a_3)) \Rightarrow a_1 \leqslant a_3$. Если информация может передаваться от сущностей класса A к сущностям класса B, а также от сущностей класса B к сущностям класса C, то очевидно, что она будет так же передаваться от сущностей класса A к сущностям класса C.

Свойство 2.1 (решетки). *Для каждой пары a_1 и a_2 элементов множества L можно указать единственный элемент наименьшей верхней границы и единственный элемент наибольшей границы.*

Эти элементы также принадлежат L и обозначаются с помощью операторов $\bullet$ и $\otimes$ соответственно:

$$a_1 \bullet a_2 = a \Leftrightarrow \Big(\big((a_1, a_2 \leqslant a) \wedge (\forall a' \in L : (a' \leqslant a))\big) \Rightarrow \big((a' \leqslant a_1) \vee (a' \leqslant a_2)\big)\Big)$$

Для пары сущностей x и y, обладающих ровнями безопасности a и b соответственно, обозначим наибольший уровень безопасности их комбинации как $(a \bullet b)$, при этом $a \leqslant (a \bullet b)$ и $b \leqslant (a \bullet b)$. Тогда, если существует некоторый уровень c такой, что $a \leqslant c$ и $b \leqslant c$, то должно иметь место отношение $(a \bullet b) \leqslant c$, поскольку $(a \bullet b)$ — это минимальный уровень субъекта, для которого доступна информация как из x, так и из y. Следовательно, $(a \bullet b)$ должен быть наименьшей верхней границей a и b.

$$a_1 \otimes a_2 = a \Leftrightarrow \Big(\big((a \leqslant a_1, a_2) \wedge (\forall a' \in L : (a' \leqslant a_1) \wedge (a' \leqslant a_2))\big) \Rightarrow (a' \leqslant a)\Big)$$

Для пары сущностей x и y, обладающих ровнями безопасности a и b соответственно, обозначим наименьший уровень безопасности их комбинации как $(a \otimes b)$, при этом $(a \otimes b) \leqslant a$ и $(a \otimes b) \leqslant b$. Тогда, если существует некоторый уровень c такой, что $c \leqslant a$ и $c \leqslant b$, то должно иметь место отношение $c \leqslant (a \otimes b)$, поскольку $(a \otimes b)$ — это максимальный уровень субъекта, для которого разрешена передача информации как в x, так и в y. Следовательно, $(a \times b)$ должен быть наибольшей нижней границей a и b.

Смысл этих определений заключается в том, что для каждой пары элементов всегда можно указать единственный элемент, ограничивающий ее сверху или снизу таким образом, что между ними и этим элементом не будет других элементов.

Функция уровня безопасности F назначает каждому субъекту и объекту некоторый уровень безопасности из L, разбивая множество сущностей системы на классы, в пределах которых их свойства с точки зрения модели безопасности являются эквивалентными. Тогда оператор $\leqslant$ определяет направление потоков информации (если $F(A) \leqslant F(B)$, то информация может передаваться от элементов класса A к элементам класса B).

Использование решетки для описания отношения между уровнями безопасности позволяет использовать в качестве атрибутов безопасности (элементов множества L) не только целые числа, для которых определено отношение «меньше или равно», но и более сложные составные элементы.

2.2.6.3. Кассическая мандатная модель Белла-ЛаПадулы

В манатных моделях функция уровня безопасности F вместе с решеткой уровней определяют все допустимые отношения доступа между сущностями системы.

Состояния системы делятся на:

— безопасные (отношения доступа не противоречат установленным в модели правилам);
— небезопасные (правила нарушаются и происходит утечка информации).

Сотояние (F, M) называется *безопасным по чтению* (или просто безопасным) тогда и только тогда, когда для каждого субъекта, осуществляющего в этом состоянии доступ чтения к объекту, уровень безопасности этого субъекта доминирует над уровнем безопасности этого объекта: $\forall s \in S, \forall o \in O, read \in M[s, o] \rightarrow F(s) \geqslant F(o)$.

Сотояние (F, M) называется *безопасным по записи* (или *-безопасным) тогда и только тогда, когда для каждого субъекта, осуществляющего в этом состоянии доступ записи к объекту, уровень безопасности этого объекта доминирует над уровнем безопасности этого субъекта: $\forall s \in S, \forall o \in O, write \in M[s, o] \rightarrow F(o) \geqslant F(s)$.

Состояние безопасно тогда и только тогда, когда оно безопасно и по чтению, и по записи.

Критерий безопасности системы

Критерий 2.2. *Система $\sum(v_0, R, T)$ безопасна тогда и только тогда, когда безопасны ее начальное состояние v_0 и все состояния, достижимые из v_0 путем применения конечной последовательности запросов из R.*

Основная теорема безопасности Белла-ЛаПадулы

Теорема 2.1. *Система $\sum(v_0, R, T)$ безопасна тогда и только тогда, когда:*

a) начальное состояние v_0 безопасно и
b) для любого состояния v, достижимого из v_0 путем применения конечной последовательности запросов из R таких, что $T(v, r) = v^*$, $v = (F, M)$ и $v^* = (F^*, M^*)$, для каждого $s \in S$ и $o \in O$ выполняются следующие условия:
   1) если $read \in M^*[s, o]$ и $read \notin M[s, o]$, то $F^*(s) \geqslant F^*(o)$;
   2) если $read \in M[s, o]$ и $F^*(s) < F^*(o)$, то $read \notin M^*[s, o]$;
   3) $write \in M^*[s, o]$ и $write \notin M[s, o]$, то $F^*(o) \geqslant F^*(s)$;
   4) $write \in M[s, o]$ и $F^*(o) < F^*(s)$, то $write \notin M^*[s, o]$.

Доказательство:

1. Необходимость.
   Если система безопасна, то состояние v_0 безопасно по определению. Пусть существует некоторое состояние v^*, достижимое из v_0 путем применения конечного числа запросов из R и полученное путем перехода из безопасного сотояния $v : T(v, r) = v^*$. Тогда, если при этом переходе нарушено хотя бы одно из первых двух ограничений, накладываемых теоремой на функцию T, то состояние v^* не будет безопасным по чтению, а если T нарушает хотя бы одно из последних двух условий теоремы, то состояние v^* не будет безопасным по записи. Таким образом, при нарушении хотя бы одного из условий теоремы система не будет безопасной.
2. Достаточность.
   Предположим, что система небезопасна. Тогда, либо v_0 небезопасно, что явно противоречит условиям теоремы, либо должно существовать небезопасное сосотояние v^*, достижимое из безопасного состояния v_0 путем применения конечного числа запросов из R. В этом случае обязательно будет иметь место переход $T(v, r) = v^*$, при котором состояние v — безопасно, а состояние v^* — нет. Но условия теоремы делают такой переход невозможным.

Таким образом по утверждению теоремы система с безопасным начальным состоянием является безопасной тогда и только тогда, когда при любом переходе ситемы из одного состояния в другое не возникает никаких новых и не сохраняется никаких старых отношений доступа, котороые будут небезопасны по отношению к функции уровня безопасности нового состояния.

Недостатки основной теоремы безопасности Белла-ЛаПадулы
Недостатоки состоят в том, что

- данная теорема является избыточной по отношению к определению безопасного сотояния, т.к. ограничения, накладываемые теоремой на функцию перехода, совпадают с критерием безопасности состояния;
- из теоремы следует только то, что все состояния, достижимые из безопасного состояния при определенных ограничениях, будут в некотором смысле безопасными, но при этом не гарантируется, что процесс осуществления перехода будет безопасным.

Таким образом, можно найти такую систему (Z-систему), в которой при попытке низкоуровнего субъекта прочитать информацию из высокоуровнего объекта будет происходить понижение уровня объекта до уровня субъекта и осуществляться чтение. Функция перехода Z-системы удовлетворяет ограничениям основной теоремы безопасности, и все состояния системы также безопасны в смысле критерия Белла-ЛаПадулы, но любой пользователь системы сможет прочитать любой файл, что несовместимо с безопасностью в обычном понимании.

Для гарантирования безопасности во время перехода между состояниями необходимо регламентировать изменения уровней безопасности во время перехода от состояния к состоянию с помощью дополнительных правил.

2.2.7. Мандатная модель Мак-Лина

Мандатная модель Мак-Лина является интерпретацией мандатной модели Белла-ЛаПадулы. Основная теорема безопасности основывается на понятии безопасного перехода, а не на понятии безопасного состояния.

При таком подходе функция уровня безопасности представляется с помощью двух функций, определнных на множестве субъектов и объектов: $F_s : S \to L$ и $F_o : O \to L$.

Функция перехода T является *безопасной по чтению*, если для любого перехода $T(r, v) = v^*$ выполняются следующие условия:

1) если $read \in M^*[s, o]$ и $read \notin M[s, o]$, то:
   $F_s(s) \geqslant F_o(o)$ и $F = F^*$
2) если $F_s \neq F_s^*$, то:
   $M = M^*$
   $F_o = F_o^*$
   для $\forall s, o$, для которых $F_s^*(s) < F_o^*(o)$, $read \notin M[s, o]$
3) если $F_o \neq F_o^*$, то:
   $M = M^*$
   $F_s = F_s^*$
   для $\forall s, o$, для которых $F_s^*(s) < F_o^*(o)$, $read \notin M[s, o]$

Функция перехода T является *безопасной по записи*, если для любого перехода $T(r, v) = v^*$ выполняются следующие условия:

1) если $write \in M^*[s, o]$ и $write \notin M[s, o]$, то:
   $F_o(o) \geqslant F_s(s)$ и $F = F^*$
2) если $F_s \neq F_s^*$, то:
   $M = M^*$
   $F_o = F_o^*$
   для $\forall s, o$, для которых $F_s^*(s) > F_o^*(o)$, $write \notin M[s, o]$
3) если $F_o \neq F_o^*$, то:
   $M = M^*$
   $F_s = F_s^*$
   для $\forall s, o$, для которых $F_s^*(s) > F_o^*(o)$, $write \notin M[s, o]$

Функция перехода является безопасной тогда и только тогда, когда она является безопасной и по чтению, и по записи.

Смысл введения перечисленных ограничений состоит в том, что нельзя изменять одновременно более одного компонента состояния системы — в процессе перехода либо возникает новое отношение доступа, либо изменяется уровень объекта, либо изменяется уровнь субъекта.

Следовательно, *функция перехода является безопасной тогда и только тогда, когда она изменяет только один из компонентов состояния и изменения не приводят к нарушению безопасности системы.*

Поскольку безопасный переход из состояния v в состояние v^* позволяет изменяться только одному элементу из v, и т.к. этот элемент может быть изменен только способами, сохраняющими безопасность состояния, была доказана следующая теорема о свойствах безопасной системы:

Теорема 2.2 (безопасности Мак-Лина). *Система безопасна в любом состоянии и в процессе перехода между ними, если ее начальное со-*

стояние является безопасным, а ее функция перехода удовлетворяет критерию Мак-Лина.

Обратное утверждение неверно. Система может быть безопасной по определению Белла-ЛаПадулы, но не иметь безопасной функции перехода.

Формулировка основной теоремы безопасности в интерпретации Мак-Лина позволяет расширить область ее применения по сравнению с классической теоремой Белла-ЛаПадулы, но используемый критерий безопасности перехода не всегда соответствует требованиям контроля доступа, возникающим на практике.

2.2.8. Модель уполномоченных субъектов

В процессе осуществления переходов могут изменятся уровни безопасности сущностей системы. Поэтому желательно контролировать этот процесс, явным образом разрешая или запрещая субъектам подобные переходы.

Для решения этой задачи Мак-Лин расширил базовую модель путем выделения подмножества уполномоченных субъектов. Таким субъектам разрешается инициировать переходы, в результате которых у сущностей системы изменяются уровни безопасности.

2.2.8.1. Обозначения

Система с уполномоченными субъектами описывается множествами S, O и L.

Состояние системы описывается набором упорядоченных пар (F, M), где

F — функция перехода;

M — матрица отношений доступа (введенные обозначения совпадают с аналогичными понятиями модели Белла-ЛаПадулы).

Функция управления уровнями $C : S \cup O \rightarrow \boldsymbol{P}(S)$ определяет подмножество субъектов, которым позволено изменять уровень безопасности для заданного объекта или субъекта.

$\boldsymbol{P}(S)$ — множество всех подмножеств S.

Модель системы $\sum(v_0, R, T^a)$ состоит из начального состояния v_0, множества запросов R и функции перехода $T^a : (S \times V \times R) \rightarrow V$, которая переводит систему из состояния в состояние по мере выполнения запросов (a — субъект, от которого исходит запрос). Система, находящаяся в состоянии $v \in V$, при получении запроса $r \in R$ от субъекта $s \in S$, переходит в состояние $v^* = T^a(s, v, r)$.

2.2.8.2. Авторизованная функция перехода

Функция перехода T^a в модели с уполномоченными субъектами называется *авторизованной функцией перехода* тогда и только тогда, когда для каждого перехода $T^a(s, v, r) = v^*$, при котором $v = (F, M)$ и

$v^* = (F^*, M^*)$, выполняется условие:

$$\forall x \in S \cup O: \text{ если } F^*(x) \neq F(x), \text{ то } s \in C(x)$$

Иными словами, в ходе авторизованного перехода уровень безопасности субъекта или объекта может изменяться только тогда, когда субъект, выполняющий переход, принадлежит множеству субъектов, уполномоченных изменять уровень этого субъекта или объекта.

2.2.8.3. Безопасность системы $\sum(v_0, R, T^a)$

Система $\sum(v_0, R, T^a)$ считается безопасной в том случае, если:

1. начальное состояние v_0 и все состояния, достижимые из него путем применения конечного числа запросов из R, являются безопасными по критерию Белла-ЛаПадулы;
2. функция перехода T^a является авторизованной функцией перехода.

Из этого определения следует только необходимое условие безопасности системы. В качестве достаточного условия может использоваться совокупность критерия авторизации функции перехода и критериев безопасного состояния Белла-ЛаПадулы, либо критериев безопасности функции перехода Мак-Лина.

2.2.9. Модель совместного доступа

Для того, чтобы мандатная модель предусматривала совместный доступ, ее необходимо модифицировать.

2.2.9.1. Обозначения

$\boldsymbol{S} = \boldsymbol{P}(S)\backslash\{\varnothing\}$ — множество непустых подмножеств S.
$\boldsymbol{M}$ — расширенная матрица прав доступа, отражающая текущее состояние доступа в системе (добавлены строки, содержащие права групповых субъектов).
$F : S \cup O \to L$ — функция уровня безопасности.
Для групповых субъектов вводятся дополнительные функции: $\boldsymbol{F}^L : \boldsymbol{S} \to L$ такая, что $\boldsymbol{F}^L(s)$ есть наибольшая нижняя граница множества $\{F(s)|s \in \boldsymbol{S}\}$ и $\boldsymbol{F}^H : \boldsymbol{S} \to L$ такая, что $\boldsymbol{F}^H$ есть наименьшая верхняя граница множества $\{F(s)|s \in \boldsymbol{S}\}$.

2.2.9.2. Критерии безопасности состояния

Критерий 2.3. *Состояние системы является безопасным по чтению тогда и только тогда, когда для каждого индивидуального или группового субъекта, имеющего в этом состоянии доступ чтения к объекту, наибольшая нижняя граница множества уровней безопасности данного субъекта доминирует над уровнем безопасности данного объекта:*

$$\forall \boldsymbol{s} \in \boldsymbol{S}, \forall o \in O, read \in \boldsymbol{M}[\boldsymbol{s}, o] \to \boldsymbol{F}^L(\boldsymbol{s}) \geqslant F(o).$$

Критерий 2.4. *Состояние системы является безопасным по записи тогда и только тогда, когда для каждого индивидуального или группового субъекта, имеющего в данном состоянии доступ записи к объекту, уровень безопасности данного объекта доминирует над наименьшей верхней границей множества уровней безопасности данного субъекта:*

$$\forall \boldsymbol{s} \in \boldsymbol{S}, \forall o \in O, write \in \boldsymbol{M}[\boldsymbol{s}, o] \rightarrow F(o) \geqslant \boldsymbol{F}^H(\boldsymbol{s}).$$

Критерий 2.5. *Состояние безопасно тогда и только тогда, когда оно одновременно безопасно и по чтению, и по записи.*

2.2.9.3. Теорема Белла-ЛаПадулы для модели совместного доступа

Благодаря тому, что множество уровней безопасности и отношение доминирования образуют решетку, удалось задать функции, определяющие границы множества уровней безопасности групповых субъектов, таким образом, что одни условия безопасности одновременно учитывают как индивидуальные, так и совместные доступы.

Переопределим функцию перехода, которая определяет следующее состояние системы после выполнения определенным субъектом некоторого запроса, как $\boldsymbol{T} : (V \times R) \rightarrow V$, где $\boldsymbol{T}(v, r) = v^*$, причем в описании состояния $v = ((F, \boldsymbol{F}^H, \boldsymbol{F}^L), \boldsymbol{M})$ и $v^* = ((F^*, \boldsymbol{F}^{*H}, \boldsymbol{F}^{*L})^*, \boldsymbol{M}^*)$ участвуют три функции уровня безопасности: F — для объектов, $\boldsymbol{F}^H$ и $\boldsymbol{F}^L$ — наименьшая верхняя и наибольшая нижняя границы для групповых субъектов.

Теорема 2.3. *Система* $\sum(v_0, R, \boldsymbol{T})$ *безопасна тогда и только тогда, когда:*

1) *начальное состояние* v_0 *безопасно и*
2) *функция перехода* $\boldsymbol{T}$ *такова, что для любого состояния* v*, достижимого из* v_0 *путем применения конечной последовательности запросов из* R*, таких, что* $\boldsymbol{T}(v.r) = v^*$*,* $v = ((F, \boldsymbol{F}^H, \boldsymbol{F}^L), \boldsymbol{M})$ *и* $v^* = ((F^*, \boldsymbol{F}^{*H}, \boldsymbol{F}^{*L})^*, \boldsymbol{M}^*)$ *для каждого* $\boldsymbol{s} \in \boldsymbol{S}$*, и* $o \in O$*, выполняются следующие условия:*
   - *(a) если* $read \in \boldsymbol{M}^*[\boldsymbol{s}, o]$ *и* $read \notin \boldsymbol{M}[\boldsymbol{s}, o]$*, то* $\boldsymbol{F}^{*L}(\boldsymbol{s}) \geqslant F^*(o)$
   - *(b) если* $read \in \boldsymbol{M}[\boldsymbol{s}, o]$ *и* $\boldsymbol{F}^{*L}(\boldsymbol{s}) < F^*(o)$*, то* $read \notin \boldsymbol{M}^*[\boldsymbol{s}, o]$
   - *(c) если* $write \in \boldsymbol{M}^*[\boldsymbol{s}, o]$ *и* $write \notin \boldsymbol{M}[\boldsymbol{s}, o]$*, то* $F^*(o) \geqslant \boldsymbol{F}^{*H}(\boldsymbol{s})$
   - *(d) если* $write \in \boldsymbol{M}[\boldsymbol{s}, o]$ *и* $F^*(o) < \boldsymbol{F}^{*H}(\boldsymbol{s})$*, то* $write \notin \boldsymbol{M}^*[\boldsymbol{s}, o]$

Аналогичные рассуждения можно провести и в отношении множественного доступа к нескольким объектам сразу (групповым объектам), когда в ходе одной операции читаются и записываются несколько объектов. В этом случае множество объектов и матрица прав доступа расширяются аналогичным образом за счет групповых объектов, для которых вводятся аналогичные функции верхней и нижней границы уровня безопасности.

2.2.9.4. Безопасная функция перехода для модели совместного доступа

Пусть F_s — функция уровней безопасности для индивидуальных субъектов (т.к. значения функций $\boldsymbol{F}^H$ и $\boldsymbol{F}^L$ границ множества уровней безопасности групповых субъектов однозначно определяются уровнями безопасности составляющих их индивидуальных субъектов, то, если в результате перехода из одного состояния в другое F_s осталась без изменений, из этого следует, что не изменились и функции $\boldsymbol{F}^H$ и $\boldsymbol{F}^L$).

Критерий 2.6. *Функция перехода $\boldsymbol{T}$ для модели совместного доступа считается безопасной по чтению, если для любого перехода $\boldsymbol{T}(v, r) = v^*$, при котором $v = ((F, \boldsymbol{F}^H, \boldsymbol{F}^L), \boldsymbol{M})$ и $v^* = ((F^*, \boldsymbol{F}^{*H}, \boldsymbol{F}^{*L})^*, \boldsymbol{M}^*)$ выполняются следующие три условия:*

1) *если $read \in \boldsymbol{M}^*[\boldsymbol{s}, o]$ и $read \notin \boldsymbol{M}[\boldsymbol{s}, o]$, то:*
   $\boldsymbol{F}^{*L}(\boldsymbol{s}) \geqslant F_o^*(o)$ *и* $F_s = F_s^*$, $F_o = F_o^*$
2) *если $F_s \neq F_s^*$, то:*
   $\boldsymbol{M} = \boldsymbol{M}^*$
   $F_o = F_o^*$
   $\forall \boldsymbol{s}, o$, *для которых* $\boldsymbol{F}^{*L}(\boldsymbol{s}) < F_o^*(o)$, $read \notin \boldsymbol{M}[\boldsymbol{s}, \boldsymbol{o}]$
3) *если $F_o \neq F_o^*$, то:*
   $\boldsymbol{M} = \boldsymbol{M}^*$
   $F_s = F_s^*$
   $\forall \boldsymbol{s}, o$, *для которых* $\boldsymbol{F}^{*L}(\boldsymbol{s}) < F_o^*(o)$, $read \notin \boldsymbol{M}[\boldsymbol{s}, \boldsymbol{o}]$

Критерий 2.7. *Функция перехода $\boldsymbol{T}$ для модели совместного доступа считается безопасной по записи, если для любого перехода $\boldsymbol{T}(v, r) = v^*$ выполняются следующие три условия:*

1) *если $write \in \boldsymbol{M}^*[\boldsymbol{s}, o]$ и $write \notin \boldsymbol{M}[\boldsymbol{s}, o]$, то:*
   $F_o^*(o) \geqslant \boldsymbol{F}^{*H}(\boldsymbol{s})$ *и* $F_s = F_s^*$, $F_o = F_o^*$
2) *если $F_s \neq F_s^*$, то:*
   $\boldsymbol{M} = \boldsymbol{M}^*$
   $F_o = F_o^*$
   $\forall \boldsymbol{s}, o$, *для которых* $\boldsymbol{F}^{*H}(\boldsymbol{s}) > F_o^*(o)$, $write \notin \boldsymbol{M}[\boldsymbol{s}, \boldsymbol{o}]$
3) *если $F_o \neq F_o^*$, то:*
   $\boldsymbol{M} = \boldsymbol{M}^*$
   $F_s = F_s^*$
   $\forall \boldsymbol{s}, o$, *для которых* $\boldsymbol{F}^{*H}(\boldsymbol{s}) > F_o^*(o)$, $write \notin \boldsymbol{M}[\boldsymbol{s}, \boldsymbol{o}]$

Критерий 2.8. *Функция перехода является безопасной тогда и только тогда, когда она одновременно безопасна и по чтению, и по записи.*

Для модели совместного доступа также будет верна и теорема Мак-Лина о свойствах безопасной системы (с учетом сформулированного критерия безопасного перехода).

2.2.10. Модель совместного доступа с уполномоченными субъектами

2.2.10.1. Обозначения

$\boldsymbol{T}^a : (\boldsymbol{S} \times V \times R) \to V$ — функция перехода (в качестве аргумента добавлен субъект a (групповой или индивидуальный), который инициирует данный переход).
Область определения функции управления уровнями $\boldsymbol{C}$ можно ограничить множествами объектов и простых субъектов, а ее область значений неюбходимо расширить за счет групповых субъектов: $\boldsymbol{C} : S \cup O \to \boldsymbol{P}(\boldsymbol{S})$ (это вызвано тем, что т.к. уровень безопасности группового субъекта $\{x, y\}$ определяется уровнями безопасности составляющих его субъектов $\{x\}$ и $\{y\}$, то достаточно контролировать только изменения уровней безопасности индивидуальных субъектов).
Система, находящаяся в состоянии $v \in V$, при получении запроса $r \in R$ от субъекта $\boldsymbol{s} \in \boldsymbol{S}$ переходит из v в состояние $v^* = \boldsymbol{T}^a(\boldsymbol{s}, v, r)$.

2.2.10.2. Авторизованная функция перехода

Функция перехода T^a является *авторизованной функцией перехода* тогда и только тогда, когда для каждого перехода $T^a(\boldsymbol{s}, v, r) = v^*$, при котором $v = ((F, \boldsymbol{F}^H, \boldsymbol{F}^{\boldsymbol{L}}), \boldsymbol{M})$ и $v^* = ((F^*, \boldsymbol{F}^{*H}, \boldsymbol{F}^{*L}), \boldsymbol{M}^*)$, выполняются условия:
1) $\forall x \in \boldsymbol{S}$ если $\boldsymbol{F}^{*H}(x) \neq \boldsymbol{F}^H(x)$ или $\boldsymbol{F}^{*L}(x) \neq \boldsymbol{F}^L(x)$, то $\boldsymbol{s} \in \boldsymbol{C}(x)$
2) $\forall o \in O$ если $F^*(o) \neq F(o)$, то $\boldsymbol{s} \in \boldsymbol{C}(o)$

2.2.10.3. Безопасность системы $\sum(v_0, R, T^a)$

Система $\sum(v_0, R, \boldsymbol{T}^a)$ с совместным доступом и уполномоченными субъектами считается безопасной в том случае, если:
1) начальное состояние v_0 и все состояния, достижимые из него путем применения конечного числа запросов из R, являются безопасными по критерию Белла-ЛаПадулы;
2) функция перехода T^a является авторизованной функцией перехода.

Из этого определения следует необходимое условие безопасности системы.

2.2.11. Применение мандатных моделей

Рассмотренные мандатные модели используют только два права доступа — чтение и запись. На практике инфрмационные системы поддерживают значительно более широкий спектр операций над информацией (например создание, удаление, передачу и т.д.). Поэтому для применения мандатной модели к реальной системе необходимо установить подходящее соответствие между чтением и записью и операциями, реализованными в конкретной системе.

Идеальным соответствием считается такое, при котором объект не может воздействовать на поведение субъекта до тех пор, пока субъект не осуществит к нему доступ чтения, и когда субъект не может воздействовать на объект, пока не осуществит к нему доступ записи.

В реальной жизни невозможно ограничиться однонаправленными потоками информации, идущими строго от субъекта к объекту, или наоборот.

Поэтому, когда в системе используется мандатная политика, все взаимодействия рассматриваются только на достаточно высоком уровне абстракции, на котором не учитываются детали реализации операций доступа. Такой подход позволяет отобразить любое множество разнообразных операций доступа в обобщенные операции чтения и записи. Для оценки возможности нарушений безопасности с использованием методов, основанных на несоответствии этих абстрактных операций и реальных механизмов доступа, приеняется анализ скрытых каналов утечки информации (т.е. выявляются действия, с помощью которых информация может передаваться в обход правил мандатной модели).

Необходимо отметить, что хотя мандатная модель управления доступом является базовой моделью безопасности, составляющей основу теории защиты информации, в реальной жизни она используется только в системах, обрабатывающих классифицированную информацию, и применяется только в отношении ограниченного подмножества субъектов и объектов.

3. Практические вопросы защиты операционных систем

В данной главе рассмотрены некоторые проекты, направленных на повышение безопасности существующих операционных систем.

В первой части рассматриваются некоторые уязвимости современных ОС и причины их существования [8, 9]. Кроме того, рассматривается технология переполнения буфера [10, 11].

Во второй части дается обзор проектов [12, 13, 14, 15, 16], которые легли в основу применяемых на дынный момент систем безопасности SELinux [17, 18] и RSBAC [19] (сами системы не рассматриваются).

3.1. Уязвимости ОС

3.1.1. Причины существования уязвимостей в ОС UNIX-системах

Рассмотрим причины возникновения нарушения безопасности UNIX и классифицируем их.

Причины, по которым происходит до 90% всех случаев вскрытия UNIX-хостов:

1. наличие демонов;
2. механизм SUID/SGID-процессов;
3. излишнее права доступа;
4. человеческий фактор с весьма разнообразными способами его проявления — от легко вскрываемых паролей у обычных пользователей до ошибок у квалифицированных системных администраторов, многие из которых как раз и открывают путь для использования механизмов доверия.

Механизмы 1 и 2, являющиеся неотъемлемой частью идеологии UNIX, были и будут наиболее предпочтительными для взломщиков, так как пользователь в этом случае взаимодействует с процессом, имеющим большие привилегии, и поэтому любая ошибка или недоработка в нем автоматически ведет к использованию этих привилегий.

Механизм 3: в UNIX много служб, использующих излишние права доступа, и они могут тем или иным способом быть обмануты.

Приведем причины, по которым демоны и SUID/SGID-процессы становятся уязвимыми.

— Возможность возникновения непредусмотренных ситуаций, связанных с ошибками или недоработками в программировании (переполнение буфера или размерности массива — конкретные атаки, несмотря на универсальность способа, они всегда будут системозависимыми и ориентированными только на конкретную платформу

и версию UNIX; неправильная обработка некоторого специального сигнала или прерывания; неправильная обработка входных данных, что является некоторым обобщением случая переполнения буфера).

— Наличие скрытых путей взаимодействия с программой, называемых «люками»
(«Люком» или «черным входом» (backdoor) часто называют оставленную разработчиком недокументированную возможность взаимодействия с системой (чаще всего — входа в нее), например, известный только разработчику универсальный пароль. Люки оставляют в конечных программах вследствие ошибки, не убрав отладочный код, или вследствие необходимости продолжения отладки уже в реальной системе из-за ее высокой сложности, или же из корыстных интересов).

— Возможность подмены субъектов и объектов различным образом
(многие особенности UNIX, такие как асинхронное выполнение процессов, развитые командный язык и файловая система, позволяют злоумышленникам использовать механизмы подмены одного субъекта или объекта другим. Например, замена имени файла, имени получателя и т. п. именем программы; подмена специальных переменных).

3.1.2. Технология переполнение буфера

Переполнение буфера (buffer overflows) — название самой распространенной уязвимости в области безопасности программного обеспечения. Первая атака с применением данной уязвимости использовалась в вирусе-черве Морриса в 1988 году. С тех пор их число увеличивается с каждым годом. В настоящее время можно говорить, что уязвимости, связанные с *переполнение буфера* являются доминирующими при удаленных атаках, где обычный пользователь сети получает частичный или полный контроль над атакуемым хостом. Очевидно, что эффективное решение данной проблемы позволит исключить большую долю самых серьезных угроз компьютерной безопасности.

Основа атак с использованием этой уязвимости — принцип функционирования операционных систем, где программа получает привилегии и права запустившего ее пользователя или процесса. Таким образом, менее привилегированный пользователь или процесс, который взаимодействует с данной программой, может использовать ее права в своих целях.

Одной из основных проблем, стоящей перед взломщиком, является необходимость исполнения написанного им кода на машине, которую он атакует. Иначе говоря, он должен указать компьютеру, с какого адреса размещается этот код, то есть занести в указатель команд (обычно он называется *instruction pointer* — IP) нужный ему адрес. Это, безусловно, может быть сделано штатными средствами операционной системы — путем запуска соответствующей программы. Но возникает две проблемы:

1. Может не быть доступа на атакуемый компьютер, то есть возможности исполнения программ.
2. Даже если доступ (login) есть, то привилегий может оказаться недо-

статочно для выполнения некоторых функций вредоносного кода. Обычная цель взломщика — получить полный контроль над машиной.

Решением может служить следующее: передача некоторому привилегированному процессу данных, которые интерпретировались бы им как код. При этом отсутствие доступа на компьютер решается передачей удаленных данных через демоны. Для выбора локальных привилегированных процессов (то есть при наличии доступа) также хорошо подходят демоны, если они запущены от имени суперпользователя или SUID root-программы.

Таким образом, необходима привилегированная программа, которая получает какие-то входные данные от непривилегированных пользователей и необходимо заставить программу исполнить эти данные как код. Такой прием получил название *buffer overflow* (переполнение буфера в стеке).

Технология переполнения локального буфера весьма универсальна и будет работать практически в любой ОС.

После передачи управления надо выполнить следующие шаги:

— Найти подходящую программу, которая выполняется с большими привилегиями. Если взломщику доступны исходные тексты, то особое внимание надо обратить на программы, содержащие функции strcat(), strcpy(), sprintf(), vsprintf(), gets(), scanf() и т. п. Если исходных текстов нет, то остается ручной поиск уязвимых программ, то есть подача на вход длинных строк и оценка результатов.
— Определить для найденной программы, какой размер буфера надо использовать, где в буфере должен располагаться адрес возврата и т.п.
— Написать код, на который осуществится переход.
— Каким-то образом внедрить свой код в систему. При этом злоумышленнику надо проверить, чтобы вызываемая функция при обработке этой строки не испортила данный код.

Также важно отметить, что технология переполнения буфера, являясь самой распространенной и эффективной для удаленного исполнения кода, то есть для реализации опасных угроз *раскрытия и целостности*, но требующая значительных усилий по формированию соответствующей строки, может применяться очень эффективно и для атак «отказ в обслуживании». Здесь нет необходимости специально подбирать буфер с правильным адресом возврата, а подойдет любой, и возврат совершится на некий случайный адрес, вызвав тем самым аварийную остановку программы или всей ОС в целом.

Таким образом, явно или неявно предполагалось, что:

— параметры функций передаются через стек;
— адрес возврата также помещается в стек;
— локальные переменные располагаются в стеке;
— стек растет вниз;
— данные в стеке могут интерпретироваться как команды;
— должны существовать процессы или программы, имеющие уязвимый код, подобный функции `process_data()`;
— некоторые процессы или функции должны иметь высокие привилегии.

Очевидно, что этим условиям удовлетворяет большинство ОС, в том числе UNIX и Windows NT.

3.2. Проекты безопасности для Linux

3.2.1. Проект Openwall

Встраивание средств защиты в ядро операционной системы обеспечивает контроль на уровне процесса, вне зависимости от того, какие программные модули и библиотеки участвовали в его создании. Участники проекта Openwall [13] работают над созданием собственного дистрибутива Linux, преследуя цель «снижения количества и последствий ошибок в программных компонентах, а также уменьшения возможного вреда от ошибок в стороннем программном обеспечении, которое пользователь может установить на свой компьютер». Это достигается путем тщательной проверки программного обеспечения и применения дополнительных средств защиты, работающих на уровне ядра.

Если переход на дистрибутив Openwall по каким-либо причинам нежелателен, то можно воспользоваться отдельными решениями, способными повысить уровень безопасности любой современной Linux-системы. Сделать это позволит пакет, содержащий набор дополнений для ядра.

При настройке ядра необходимо произвести настройку параметров меню Security options [20].

Опция `Non-executable user stack area` налагает запрет на выполнение кода, находящегося в стековой области пользовательского процесса. Ее активизация приводит к блокированию ядром любых попыток исполнения кода в стеке. Таким образом, осуществляется защита от атак, использующих механизм изменения адресов возврата функций на адреса запуска вредоносного кода, расположенного в стековой области.

Включение запрета на выполнение кода, расположенного в области данных может привести к неработоспособности некоторых программ, использующих механизм так называемых Trampoline-вызовов. Если в вашей системе имеются такие программы, то необходимо включить опцию `Autodetect and emulate GCC trampolines`. В этом случае работоспособность программ будет восстановлена, но безопасность системы несколько снизится.

Опции `Restricted links in /tmp` и `Restricted FIFOs in /tmp` включают дополнительные ограничения при работе с соответствующими элементами файловой системы. Это позволяет противостоять целому классу атак, осуществляемых посредством манипуляций с этими объектами.

Активизация режима `Restricted /proc` приводит к ограничению доступа к информации, находящейся в каталоге /proc. При этом рядовым пользователям становится недоступной некоторая системная информация и информация по чужим для них процессам.

В ОС Linux есть системный запрос setrlimit, позволяющий ограничивать число процессов, создаваемых пользователем. К сожалению,

это ограничение работает только в том случае, если порождение очередного процесса происходит посредством запроса fork. Включение опции `Enforce RLIMIT_NPROC on execve(2)` позволяет распространить действие проверок и на системный вызов `execve`.

В Linux также имеются ограничения на объем памяти, доступный выполняемому процессу. К сожалению, сегменты разделяемой памяти могут существовать независимо от процессов, их использующих, что позволяет обойти соответствующие проверки. Включение опции `Destroy shared memory segments not in use` вызывает уничтожение этих сегментов, после того как они становятся невостребованными. Активизация данного режима может привести к неработоспособности некоторых приложений.

Если в системе появляются проблемы с запуском некоторых приложений, необходимо воспользоваться утилитой chstk, позволяющей разрешить отдельным приложениям выполнять соответствующие «незаконные» операции. При этом следует учесть, что, при выключении тех или иных механизмов защиты, открываются бреши, через которые злоумышленник может проникнуть в систему. Поэтому, если требования по безопасности высоки, лучше отказаться от «неправильных» программ и как можно реже пользоваться утилитой chstk.

3.2.2. Проект PaX

Целью проекта PaX [14] является также создание механизма, позволяющего противостоять атакам, основанным на различных способах перехвата управления у нормально исполняющегося процесса. Перечень защит, предоставляемых PaX-ядром, значительно отличается от аналогичного перечня Openwall-ядра. PaX предоставляет более широкий выбор механизмов, реализующих контроль попыток вживления вредоносного кода в исполняющийся процесс, исполнения существующего программного кода в порядке, не предусмотренном изначальным алгоритмом программы и обработку процессом некорректных данных, поступающих извне. Вместе с тем, в нем нет механизмов, ограничивающих доступ к символьным ссылкам, и не осуществляется контроль ситуаций, связанных с перерасходом системных ресурсов, выделенных процессу.

Рассмотрим возможности, предоставляемые подсистемой PaX при конфигурировании ядра.

Блок опций Enforce non-executable pages отвечает за ограничения, вводимые ядром системы при выполнении программного кода. Включение опции Paging based non-executable page запрещает выполнение программного кода, расположенного в страницах памяти, содержащих данные. Активизация этой опции на системах архитектуры i386 может привести к значительному снижению быстродействия; на платформах Alpha, SPARC, SPARC64, AMD64 не произойдет снижения производительности. Для того чтобы потеря производительности была незначительной, можно выбрать опцию Segmentation based non-executable pages. В этом случае, правда, объем адресуемой памяти уменьшится вдвое, до 1,5 Гбайт.

Активизация опции Emulate trampolines снимает проблемы, возникающие при исполнении программ, использующих соответствующие вызовы.

Включение опции Restrict mprotect() запрещает изменение статуса страниц памяти на executable (выполняемый), если он не был таковым изначально. Активизация этого режима приводит к неработоспособности многих приложений (в том числе, и приложений X Window).

Опция Disallow ELF text relocations обеспечивает дополнительную защиту от несанкционированных действий, выполняемых посредством вызова mmap. Активизировать эту опцию можно только в тех системах, где все программы собраны с PIC-версией библиотеки crtl; в противном случае система может стать незагружаемой.

Режимы из блока Address Space Layout Randomization вносят элемент случайности при распределении системой адресного пространства. Дело в том, что большинство методик осуществления вторжений требуют знания определенных адресов внутри атакуемого процесса. В системе, управляемой классическим ядром, эти адреса распределяются стандартным образом и могут быть легко предсказаны. Введение же в процесс распределения адресного пространства элемента случайности значительно усложняет процедуру передачи управления вредоносному коду.

При включении опции Randomize kernel stack base ядро при каждом системном запросе будет производить перераспределение области стековой памяти ядра. Разработчики PaX предупреждают, что включение этой опции может повлечь за собой нестабильность в работе ОС, поэтому пользоваться ей нужно крайне осторожно.

Активизация режима Randomize user stack base включает механизм случайного распределения памяти под стековую область пользовательского процесса.

Включение опции Randomize mmap() base приведет к тому, что ядро будет назначать случайным образом базовый адрес запроса mmap, в результате чего все динамические библиотеки будут загружаться в память по произвольному адресу. Это значительно затруднит успешное проведение атак, в которых злоумышленник пытается выполнить существующий библиотечный код для своих целей, например, инициировать вызов shell из атакованной программы.

Управление защитой на уровне отдельных программ осуществляется при помощи утилиты chpax.

3.2.3. Medusa

Medusa DS9 security system — система, позволяющая делить процессы на группы — виртуальные среды и гарантировать их непересекаемость. Проект Medusa DS9 [21] был инициирован летом 1997 года.

Первая версия этого программного продукта представляла собой заплатку для ядра ОС Linux. В настоящее время Medusa DS9 представляет собой пакет способный значительно улучшить защищенность Linux-систем, расширяя функциональность стандартных средств обеспечения безопасности, обеспечивая при этом обратную совместимость.

Medusa DS9 поддерживает на уровне ядра авторизацию пользователей на серверах авторизации, поэтому прежде чем выполнять какую-либо операцию, ядро делает запрос на сервер, который в свою очередь либо разрешает, либо запрещает ее выполнение. Так же в некоторых случаях сервер может повлиять и на то, как эта операция должна выполняться, что в свою очередь позволяет использовать практически любую архитектуру безопасности. Если сервер авторизации сконфигурирован правильно, то при необходимости получения доступа к системным ресурсам выделяется наилучший, из соображений безопасности системы, уровень доступа и проводится аудит системы.

Пакет Medusa DS9 состоит из двух частей: патч для ядра ОС Linux и демон безопасности «констебль», который выполняет функции сервера авторизации. Заплатка ядра позволяет установить контроль над системными вызовами, действиями файловой системы и процессами. Демон безопасности связывается с ядром, используя символьное устройство `/dev/medusa` для манипуляции пакетами. Когда ядру нужно сделать запрос на подтверждение, оно записывает информацию в это устройство, при этом сам процесс переводится в спящий режим. Далее активизируется Констебль, который читает информацию из символьного устройства `/dev/medusa` и, в зависимости от настроек и конфигурации системы Medusa, посылает ответ ядру и снова засыпает. Получив ответ, ядро активизирует процесс и получает результат операции. Вообще говоря, в случае необходимости констебль может сам посылать команды ядру, даже если от ядра не поступало никаких запросов на выполнение. В своей работе демон безопасности использует специальный коммуникационный протокол, определенный заранее в ядре системы. Поэтому, зная лишь какой протокол он использует и, будучи уверенным, что ядро его поддерживает, на его основе можно организовать полноценный сервер авторизации, не боясь при этом испортить что-нибудь в ядре системы. Для передачи информации протокол использует специальные пакеты. Пакет Medusa DS9 предоставляет следующие возможности:

— управление доступом к любому файлу системы посредством VFS;
— возможность перенаправить доступ на выбранный файл;
— управление отсылкой / приемом сигналов;
— непосредственное управление выполнением важных процессов;
— контроль выполнения любых системных вызовов любого выбранного процесса;
— каждый процесс или файл является частью специализированной виртуальной подсистемы, и каждому процессу назначается уровень доступа к этой виртуальной подсистеме, что позволяет скрыть существование одних процессов и файлов от других;
— каждый процесс имеет уникальный идентификатор (uid), который назначается при первом вызове;
— низкоуровневый контроль над любым системным вызовом.

Благодаря тому, что констебль представляет собой интерпретатор со своими собственным языком описания конфигурационных файлов, напоминающим язык C, можно реализовать практически любую модель безопасности.

3.2.4. Grsecurity

Grsecurity [15] — это набор патчей для 2.4.x и 2.6.x linux ядер, объединяющий в себе множество разрозненных патчей (как собственной разработки, так и сторонних авторов: Openwall non-executable stack, PaX и т.д.), направленных на увеличение безопасности системы, обнаружение и предотвращение атак злоумышленников.

В Grsecurity реализованы ограничения на `/proc`, `fifo`, запуск процессов и манипуляции с файловыми ссылками, широкие возможности ведения логов (запускаемые процессы, смена uid, сигналы, ошибки fork), дополнительные ограничения для пользователей, изменение технологии chroot, увеличение безопасности TCP/IP стека.

Grsecurity поддерживает следующие функции Openwall: non-executable stack, PaX, the Oblivion ACL system, /proc restrictions, chroot restrictions, linking and FIFO restrictions, exec и set*id logging, secure file descriptors, trusted path execution, randomized IP IDs, randomized PIDs, randomized TCP source ports, altered ping ids, randomized TTL, better IP stack randomness, socket restrictions, fork-bomb protection, sysctl поддержка всех опций, secure keymap loading, stealth networking enhancements, signal logging, failed fork logging, time change logging и др.

3.2.5. LinSec — Linux Security Protection System

LinSec является реализацией мандатного механизма разграничения доступа для Linux. LinSec базируется на четырех составляющих:

- Capabilities — предоставление отдельных, выборочных полномочий для конкретной программы или пользователя;
- Filesystem Access Domains — списки доступа к файловой системе, например, программе или пользователю можно разрешить только доступ к mailbox и домашней директории;
- IP Labeling Lists — списки доступа на уровне сетевых сервисов, можно граничить программу или пользователя только для обслуживания входящих запросов к определенному порту.
- Socket Access Control — ограничения по созданию сетевых соединений.

3.2.6. LIDS

LIDS (Linux Intrusion Detection/Defence System) [12] — система обнаружения и защиты от вторжения. Это патч для ядра Linux, добавляющий много новых возможностей для увеличения безопасности.

LIDS позволяет запретить или ограничить доступ к файлам, к памяти, блочным устройствам, сетевым интерфейсам, запущенным программам и т.д. даже для пользователя root. Правильнее сказать, именно для пользователя root, так как ограничить доступ для простого пользователя можно и стандартными средствами Linux. Данная система предназначена для защиты от взломщика, который воспользовавшись «дырой» в какой-нибудь программе, получил права пользователя root.

В отличие от других средств защиты, входящих в поставку Linux, эту систему нельзя выключить, не зная пароль администратора LIDS, который в зашифрованном виде хранится в специальном файле и виден только программе администрирования LIDS (не администратору, а именно программе). То же самое относится и к другим конфигурационным файлам LIDS. Даже узнав каким-то образом пароль администратора, взломщик не сможет отключить LIDS, не находясь за взломанным компьютером.

LIDS позволяет распределять права доступа к файлам, устройствам и т.д. на уровне программ, а не на уровне пользователей. Например, можно запретить доступ к файлу `/etc/shadow` для всех так, что он даже при использовании команды `ls -a /etc` не будет виден, и дать к нему доступ на чтение программам `/bin/login`, `/bin/su`, на чтение-запись программе `/usr/bin/passwd` и т.д., то есть тем программам, которые в нем *действительно* нуждаются.

LIDS позволяет запретить перезапуск системы, так что человек, не находящийся в непосредственной близости к кнопке RESET, перегрузить систему не сможет.

Посредством LIDS можно запретить загрузку/выгрузку модулей ядра, что защитит систему от запуска модулей-троянов.

Информация о всех действиях, направленных на взлом защищенных при помощи LIDS объектов, записывается в логи и отправляется на e-mail, указанный в файле конфигурации LIDS, непосредственно в момент совершения взлома, что дает возможность администратору незамедлительно реагировать на все происходящее в системе.

В LIDS есть встроенный детектор сканирования портов, обнаруживающий большинство известных способов сканирования. Работает этот детектор на уровне ядра, то есть отключить его невозможно.

3.2.6.1. Правила доступа

Вся настройка LIDS делается с помощью одной программы — *lidsadm*, которая работает в двух режимах — настройки правил доступа и ввода команд администрирования. Каждому режиму соответствует несколько параметров запуска.

Установки правил доступа находятся в файле `/etc/lids/lids.conf`. Изначально он содержит наиболее часто используемые правила доступа. Этот файл просматривается и изменяется посредством *lidsadm*.

Правила доступа состоят из трех элементов: субъекта (*subject*), объекта (*object*) и цели (*target*).

Объектом является любой файл или каталог, на который и должны действовать правила доступа и защита LIDS. Если в качестве объекта указывается каталог, то все файлы в нем и вложенные каталоги с их файлами автоматически становятся объектами.

Субъектом является любая *защищенная* программа, которой дают доступ к защищаемому объекту, т.е. прежде чем использовать программу в качестве субъекта, ее саму надо защитить средствами LIDS от посягательств, применив к ней правила доступа как к объекту. Если субъект не

указан, то субъектом является любая программа.

Целью является тип доступа — доступ на чтение (*READ*), запись (*WRITE*), запрет на какой-либо доступ (*DENY*), открытие только для дозаписи (*APPEND*) и игнорирование защиты (*IGNORE*).

3.2.6.2. Привилегии

В качестве объекта правил доступа могут также служить привилегии (capabilities).

Capabilities — это привилегии программ совершать какие-либо действия.

При установке *lidsadm* в каталоге `/etc` появляется каталог `lids`, содержащий четыре файла с параметрами настройки LIDS:

- **lids.cap**
  в этом файле хранятся текущие значения установок привилегий;
- **lids.net**
  этот файл содержит настройки отправки сообщения на удаленный почтовый аккаунт;
- **lids.pw**
  здесь записан в зашифрованном (методом RipeMD-160) виде пароль администратора;
- **lids.conf**
  текущие установки правил доступа.

LIDS позволяет устанавливать и отменять большое количество привилегий, такие как перезагружать компьютер (**CAP_SYS_BOOT**), изменять владельца файла (**CAP_CHOWN**), загружать/выгружать модули ядра (**CAP_SYS_MODULE**) и многие другие.

Текущие установки привилегий хранятся в файле `/etc/lids/lids.cap` в формате: [+|−] Номер:Привилегия. «+» включает capabilitie, «−» — отключает, например **+22:CAP_SYS_BOOT** разрешает перезагрузку, **-22:CAP_SYS_BOOT** — запрещает. Изменять его можно с помощью любого текстового редактора. Выключение capabilities влияет на все программы, кроме тех, которым напрямую указана данная привилегия с помощью правил доступа *lidsadm*. Включение capabilities влияет на все программы без исключения. Нельзя включить capabilitie у всех программ, а у нескольких выключить.

Значения параметров:

CAP_CHOWN
С помощью этого параметра устанавливается привилегия программ изменять владельца и группу владельца файла.

CAP_DAC_OVERRIDE
Включает / отключает привилегию программ, запускаемых под пользователем root, не принимать во внимание режимы доступа к файлам. Например, при включенной данной привилегии root может открыть и изменить файл, который принадлежит dh и имеет режим доступа 0600, при отключенной данной опции root не в состоянии будет даже открыть данный файл, т.е. root при отключении данной привилегии приравнивается к обыкновенному пользователю при

доступе к файлам.

CAP_DAC_READ_SEARCH

Включает / отключает привилегию программ, запускаемых под пользователем root, не принимать во внимание режимы доступа к каталогам (чтение и поиск).

CAP_FOWNER

Запрещает / разрешает операции с файлами, когда владелец файла должен совпадать с пользователем, совершающим операцию. Например, изменение режима доступа к файлу (chmod). Режим доступа может изменять либо владелец файла, либо пользователь root. При отключении этой привилегии, root уже будет не в состоянии изменить режим доступа. То же относится к изменению атрибутов файлов (chattr).

CAP_FSETID

Запрещает / разрешает установку SUID'ного или SGID'ного бита на чужих файлах (не принадлежащих пользователю root).

CAP_KILL

Включает / отключает привилегию процессов, принадлежащих пользователю root, убивать чужие процессы.

CAP_SETGID

Управляет привилегией программ, принадлежащих пользователю root, сменять группу, под которой работает программа. Так работает, например, httpd, sendmail, postfix, ftpd, safe_finger и т.д.

CAP_SETUID

Управляет привилегией программ, принадлежащих пользователю root, сменять пользователя, под которым работает программа.

CAP_SETPCAP

Включает / отключает возможность программ менять привилегии.

CAP_LINUX_IMMUTABLE

Управляет привилегией снимать атрибуты S_IMMUTABLE (chattr -i) и S_APPEND (chattr -a) с файлов.

CAP_NET_BIND_SERVICE

Включает / отключает привилегию программ привязываться к порту с номером < 1024.

CAP_NET_BROADCAST

Управляет привилегией программ рассылать широковещательные пакеты.

CAP_NET_ADMIN

Этот параметр управляет большим количеством различных привилегий: конфигурирование сетевых интерфейсов, изменение правил сетевого экрана, изменение таблиц маршрутизации и многих других, связанных с сетевыми настройками Linux.

CAP_NET_RAW

Управляет привилегией программ использовать сокет-соединения.

CAP_IPC_LOCK

Управляет привилегией процессов, принадлежащих пользователю root, блокировать сегменты разделяемой памяти.

CAP_IPC_OWNER

Управляет доступом программ, принадлежащих пользователю root,

к ресурсам межпроцессорного взаимодействия чужих процессов.

CAP_SYS_MODULE
Управляет привилегией загружать/выгружать модули ядра.

CAP_SYS_RAWIO
Управляет доступом на чтение-запись к таким устройствам, как `/dev/mem`, `/dev/kmem`, `/dev/port`, `/dev/hd??`, `/dev/sd??`.

CAP_SYS_CHROOT
Управляет привилегией устанавливать корневой каталог для текущего shell'а.

CAP_SYS_PTRACE
Данный параметр включает / отключает привилегию программ использовать вызов функции **ptrace()**, которая позволяет управлять выполнением процессов-потомков процессу-родителю.

CAP_SYS_PACCT
Управляет привилегией конфигурировать учет процессов.

CAP_SYS_ADMIN
Управляет множеством привилегий: управление `/dev/random`, создание новых устройств, конфигурирование дисковых квот, настройка работы `klogd`, установка имени домена, установка имени хоста, сброс кэша, монтирование / размонтирование дисков, включение / отключение swap-партиции, установка параметров последовательных портов и др.

CAP_SYS_BOOT
Данный параметр управляет привилегией перегружать систему.

CAP_SYS_NICE
Управляет привилегией изменять приоритет чужих процессов.

CAP_SYS_RESOURCE
Привилегия изменять ограничения использования ресурсов системы: дисковые квоты, зарезервированное пространство на ext2-партициях, максимальное количество консолей и т.д.

CAP_SYS_TIME
Управляет привилегией изменять системное время.

CAP_SYS_TTY_CONFIG
Привилегия изменять настройки tty-устройств.

CAP_HIDDEN
Привилегия программ делаться невидимыми в списке процессов. Не влияет на все программы.

CAP_INIT_KILL
Привилегия убивать процессы-потомки процесса `init`. К таким процессам относятся практически все демоны.

Для первоначальной (в процессе загрузки) инициализации параметров capabilities используется команда `lidsadm -I`. Обычно ее ставят в какой-нибудь *rc*-скрипт, после запуска всех демонов. Можно поставить ее в конце `/etc/rc.d/rc.local`. Таким образом, отключение capabilities сработает только после запуска всех необходимых для работы сервера программ.

3.3. Мандатные модели в Linux

Относительно защищенный вариант Unix-системы можно построить на базе ее штатных средств и для большинства применений это будет вполне адекватный уровень безопасности. Однако в самой идеологии построения Unix имеется ряд фундаментальных изъянов, которые невозможно преодолеть только грамотным администрированием.

Фактически, в Unix предполагается всего два варианта статуса пользователя: обычный и суперпользователь (root), что сразу порождает две проблемы:

1) root никому не подконтролен: он может изменить любую настройку, имеет доступ абсолютно ко всем объектам в файловой системе, может стереть или модифицировать любые данные и записи в журналах регистрации, что неприемлемо для защищенной системы;
2) необходимость изменения текущего уровня привилегий пользователя для выполнения некоторых действий (например, для смены пароля пользователю требуется временное разрешение на запись в файл `/etc/shadow` или аналогичный), что традиционно достигалось путем установки флага SUID (или SGID) на файлы исполняемых модулей программ, в результате чего порождаемый при запуске такого файла процесс приобретает не права запустившего его пользователя, а права владельца файла.

Реализованная в Unix модель разграничения доступа является дискреционной — права доступа субъектов (пользователей, процессов) к объектам (файлам, каталогам и т.п.) задаются явно, в матричной форме. На практике же часто возникает необходимость в реализации мандатной модели, при которой права доступа субъекта к объекту определяются уровнем субъекта и классом объекта, либо комбинированной модели, включающей в себя мандатную и доступ на основе списков управления доступом. Причем, если в других операционных системах, использующих дискреционную модель (например, Novell NetWare), необходимого разграничения достичь можно, то в Unix реализация некоторых требований по разграничению может оказаться попросту невозможной. Это связано с тем, что в Unix права доступа задаются всего для трех категорий субъектов: владельца файла, ассоциированного с файлом группы, и всех прочих. В частности, невозможно задать разные права доступа к файлу для менеджера проекта, начальника отдела, менеджера направления и сотрудника, непосредственно работающего с файлом, закрыв доступ всем остальным пользователям.

Ряд других проблем (например, передача паролей и других данных в открытом виде по сети при взаимодействии по различным протоколам) решаются широким кругом стандартных средств (ssh, IPsec и т.п.) и в большинстве своем не являются Unix-специфичными.

В качестве примеров решения приведенных проблем с организацией защиты предлагается рассмотреть две системы: RSBAC (Rule Set Based Access Control) и Security-Enhanced Linux, позволяющие построить защищенную систему на базе ядра Linux.

3.3.1. RSBAC

RSBAC — это надстройка над ядром Linux и комплект утилит управления, позволяющие создать на базе любого дистрибутива Linux защищенную систему.

Одной из целей создания RSBAC является выполнение всех требований класса B1 по классификации так называемой «Оранжевой Книги» [22] (хотя набор этих требований уже несколько устарел).

Рассмотрим предоставляемые системой RSBAC механизмы обеспечения безопасности. Реализация этих механизмов выполнена на уровне ядра системы и позволяет эффективно контролировать все процессы. Системные вызовы, затрагивающие безопасность, дополняются специальным кодом, выполняющим обращение к центральному компоненту RSBAC. Этот компонент принимает решение о допустимости данного системного вызова на основе следующих параметров:

— типа запрашиваемого доступа (чтение, запись, исполнение);
— субъекта доступа;
— атрибутов субъекта доступа;
— объекта доступа;
— атрибутов объекта доступа.

3.3.1.1. Архитектура RSBAC

RSBAC, функционирующий на уровне ядра, состоит из двух компонентов: дополнения исходного кода ядра и модуля правил. RSBAC добавляет свои собственные проверки к системным вызовам. Как результат, логика работы ядра не меняется, но его интерфейсная часть дополняется новыми возможностями.

3.3.1.2. Архитектура GFAC

Архитектура GFAC (General Framework for Access Control Approach), по которой работает RSBAC, предложена одним из разработчиков мандатного доступа Ла-Падула и впервые реализована в SystemV/MLS. В этой архитектуре механизм защиты делится на три части:

1) модуль контроля системных вызовов (Access Enforcement Facility — AEF),
2) модуль принятия решения (Access Decision Facility — ADF),
3) хранилище информации о правах доступа (Access Control Information — ACI).

Системный вызов преобразуется в один из стандартных запросов к модулю принятия решения, который разрешает или запрещает дальнейшее выполнение системного вызова.

Модули защиты получают информацию для принятия решения из атрибутов участников системы, которые хранятся в ACI.

Архитектура GFAC позволяет сделать защиту независимой от типа операционной системы (достаточно заменить модуль AEF, сохранив все остальные модули) и обеспечить одновременную работу нескольких независимых друг от друга модулей правил безопасности. Архитектура GFAC

позволяет оснащать механизмом RSBAC различные операционные системы.

3.3.1.3. Модульная структура RSBAC

RSBAC имеет модульную структуру, т.е. существует несколько модулей аутентификации, каждый из которых контролирует доступ своим особым способом. Окончательное решение о предоставлении доступа или отказе в нем получается как суммарное после обсуждения этого вопроса всеми модулями. Все модули работают независимо, за исключением модуля **auth**, который используется всеми. В RSBAC имеются следующие модули:

- модуль **AUTH** осуществляет общее слежение за процессами и не позволяет им менять свой uid произвольно;
- модуль **RC** (Role Compatibility) — модель доступа, основанная на ролевой модели; с минимальными затратами решает большинство проблем разграничения доступа;
- модуль **MAC** (Mandatory Access Control) — мандатная модель доступа на основе известной модели Белла—Ла Падула, но несколько модифицированная по сравнению с ней (цель модели — не допустить перетекания информации из более секретных объектов в менее секретные);
- модуль **ACL** (Access Control Lists) — расширяет права файлов по сравнению со стандартными, позволяет контролировать такие операции как добавление файла в ядро, запуск файла, смена каталога и иные с точностью до индивидуального пользователя, группы пользователей или роли из модели RC;
- модуль **FC** (Functional Control) — реализует простую ролевую модель, в которой доступ к системной информации разрешен только администраторам системы, а доступ к информации, связанной с безопасностью, разрешен только офицерам безопасности;
- модуль **SIM** (Security Information Modification) — обеспечивает возможность модификации данных, помеченных как «security information», только администраторами безопасности;
- модуль **PM** (Privacy Model) — реализует модель безопасности, направленную на обеспечение приватности личных данных; основная идея [23] состоит в том, чтобы пользователь мог получить доступ к персональным данным только в том случае, если они ему необходимы для выполнения текущей задачи и если он авторизован на ее выполнение; кроме того, цели выполнения текущей задачи должны совпадать с целями, для которых эти данные собраны, либо должно быть получено согласие субъекта этих данных;
- модуль **MS** (Malware Scan) — обеспечивает сканирование всех файлов на наличие вредоносного кода и контролирует все запросы на чтение файлов и соединений TCP/UDP;
- модуль **FF** (File Flags) — предоставляет механизм установки и проверки флагов на файлы и каталоги, причем модифицировать флаги разрешено только офицерам безопасности системы (пока поддержи-

ваются флаги execute_only (для файлов), read_only (для файлов и каталогов), search_only (для каталогов), secure_delete (для файлов), no_execute (для файлов) и add_inherited (для файлов и каталогов)).

3.3.1.4. Модуль AUTH

Основы. Этот модуль может быть рассмотрен как модуль поддержки для всех остальных модулей. Он ограничивает смену владельца для процессов (CHANGE_OWNER) на объектах процессов (setuid): запрос разрешен, только если процесс имеет установленный флаг auth_may_setuid или в его наборе привилегий находится целевой ID пользователя. Флаг auth_may_setuid и набор способностей наследуются при выполнении от программного файла.

Возможности программных файлов могут быть установлены, если всем модулям разрешен запрос MODIFY_ATTRIBUTE для A_auth_add_f_cap или A_auth_remove_f_cap. Возможности процесса могут быть добавлены только другими процессами, которые имеют установленный флаг auth_may_set_cap, который также унаследован от их исполняемого файла.

Таким образом, возможно осуществление процесса аутентификации, базирующееся на демоне, также как ограничение системных демонов набором пользовательских ID.

Атрибуты AUTH. Все процессы имеют два auth-флага: auth_may_setuid и auth_may_set_cap с вышеупомянутыми значениями. Они унаследованы всеми процессами потомков.

Файлы и процессы также имеют наборы возможностей с теми же самыми правилами наследования. Добавление (удаление) к (из) набору(-а) возможностей ограничено в соответствии с различными схемами контроля доступа процессов и файлов: заголовки процесса могут быть установлены любым процессом, который имеет набор auth_may_set_flag, кто бы ни был владельцем процесса. Флаги файла установлены от имени пользователей для любой программы.

Специальные значения. До версии 1.0.9a, наборы привилегий файлов могли иметь специальное значение 65533 (UID -3), которое замещалось владельцем процесса во время копирования набора (время выполнения). Эти процессы могут вернуться назад к оригинальному ID пользователя после его изменения. Это значение было изменено на 4294967292 (снова -3) в 1.0.9b, которое поддерживает 32-битный ID пользователя.

Администрирование. AUTH только ограничивает собственное администрирование, если это позволено в конфигурации ядра. Администрирование разрешено пользователю с ролью system_role security_officer, и все задействованные атрибуты защищены.

В дальнейшем каждый модуль может ограничить администрацию AUTH, если это зависит от надлежащей аутентификации.

Администрирование в основном базируется на атрибутах файла и процесса и наборах возможностей обозначенных выше.

3.3.1.5. Модуль MAC

Модуль MAC реализует модель мандатного доступа [24], предложенную в свое время Беллом и Ла Падулом [22].

3.3.1.6. Модуль ACL

Помимо мандатного доступа необходимо контролировать права доступа к файлу с точностью до пользователя или группы пользователей. Для этого используется модуль ACL, который добавляет к обычным правам списки контроля доступа, которые определяют, какой субъект к какому объекту может иметь доступ и с каким запросом.

Объекты группируются по видам, но каждый имеет собственный список ACL. Если права доступа к объекту не заданы явно, они наследуются от родительского объекта с учетом маски наследования прав. Эффективные права доступа субъекта к объекту складываются из прав, полученных непосредственно, и прав, полученных через назначение на роль или членство в группе ACL.

Права доступа к файлу:

- ADD_TO_KERNEL — добавить модуль в ядро (для драйвера);
- ALTER — изменить контрольную информацию для IPC;
- APPEND_OPEN — открыть для добавления;
- CHANGE_GROUP — сменить группу;
- CHANGE_OWNER — сменить владельца;
- CHDIR — сменить каталог;
- CLONE — клонировать процесс;
- CLOSE — запрос на закрытие;
- CREATE — запрос на создание;
- DELETE — запрос на удаление;
- EXECUTE — запрос на запуск;
- GET_PERMISSIONS_DATA — прочитать права UNIX;
- GET_STATUS_DATA — получить информацию о файле;
- LINK_HARD — сделать жесткую ссылку;
- MODIFY_ACCESS_DATA — модифицировать информацию о времени доступа к файлу;
- MODIFY_ATTRIBUTE — изменить атрибут rsbac;
- MODIFY_PERMISSIONS_DATA — изменить права Unix;
- MODIFY_SYSTEM_DATA — изменить системные данные;
- MOUNT — осуществить операцию монтирования;
- READ — произвести чтение;
- READ_ATTRIBUTE — произвести чтение атрибута RSBAC;
- READ_OPEN — открыть файл для чтения;
- READ_WRITE_OPEN — открыть файл для чтения-записи;
- REMOVE_FROM_KERNEL — удалить модуль из ядра (для драйверов);

— RENAME — переименовать;
— SEARCH — произвести поиск;
— SEND_SIGNAL — послать сигнал другим процессам;
— SHUTDOWN — выключить / перезагрузить систему;
— SWITCH_LOG — изменить параметры протоколирования RSBAC;
— SWITCH_MODULE — включить / выключить модули;
— TERMINATE — остановить процесс;
— TRACE — трассировать процесс;
— TRUNCATE — усечь файл;
— UMOUNT — выполнить демонтирование;
— WRITE — произвести запись;
— WRITE_OPEN — открыть файл для записи.

Если не существует ACL вхождения для субъекта к объекту, то права к родительскому объекту унаследованы. Наследственность может быть ограничена масками наследственности.

Наверху иерархии наследственности находится значение ACL по умолчанию для каждого типа объекта (ФАЙЛ, КАТАЛОГ, ...).

Для изменения значения ACL объекта необходимы специальные права контроля доступа для этого объекта. Специальные права Forward позволяют дать права, которыми обладает пользователь, кому-нибудь еще, но в последствии отменить их нельзя.

Специальные права Supervisor включают все остальные права, которые никогда не могут быть замаскированы (если не разрешено в конфигурации ядра) и могут быть установлены только пользователями, имеющими их. Эти права установлены для пользователя 400 в его конфигурации ACL по умолчанию и не могут быть прочитаны с диска во время работы, например во время новой установки.

Поддерживаются все типы объектов. Только объекты IPC, USER и PROCESS имеют общий ACL по умолчанию, который всегда наследуется всеми объектами этого типа — эти объекты слишком недолговечны для администрирования полезных индивидуальных ACL.

Групповой менеджмент позволяет каждому пользователю определять глобальные и частные группы без ограничений. Глобальные группы могут быть использованы любым пользователем, приватные — только владельцем группы. Также только владельцу группы позволено добавлять или удалять членов группы или изменять установки группы — название, владельца и тип. Так как владелец может быть изменен, то группа является передаваемой (таким образом ее можно сделать недоступной для настоящего владельца).

Права группы могут быть добавлены к правам пользователя и роли. Так как нет глобального группового администратора, каждый пользователь может выполнять работу по обслуживанию благоразумной структуры группы.

3.3.1.7. Модуль RC

Данный модуль определяет 64 роли и 64 типа для каждого вида объекта. Виды объектов могут быть следующими: file, dir, dev, ipc (interprocess

communication), scd (system control data), process. Для каждой роли отношение к различным типам и другим ролям настраивается индивидуально, в зависимости от вида запроса. Используя данный модуль, можно настроить разделение обязанностей между администраторами, избежав при этом назначения избыточных прав.

Введем такие термины как цели и запросы.

Субъекты (суть процессы) делают запросы на доступ к цели. Запросы могут быть самые разнообразные и совпадать с правами ACL.

Целью может быть:

- **FILE** — файл,
- **DIR** — каталог,
- **DEV** — устройство,
- **IPC** — объект IPC: семафоры, разделяемая память и т.д.,
- **SCD** — системные данные, такие как имя машины, системный журнал.

Все они как правило только для чтения.

- **USER** — пользователи,
- **PROCESS** — процессы,
- **NONE** — пустой объект,
- **FD** — файловый дескриптор.

Причем каждый тип цели может иметь свой подтип. Например, файл может быть *обычный, секретный* или *системный*.

Роль определяет некий класс субъектов, задавая, какие права имеют члены этого класса по отношению к определенным подтипам цели и другим классам. Рассмотрим эти параметры более подробно:

- **Name** — название роли;
- **Role Comp** — совместимые роли (то есть роли, на которые данная роль может переключиться без смены владельца);
- **Admin Roles** — роли, которые данная роль может администрировать;
- **Assign Roles** — роли, которая данная роль может назначать пользователям или процессам;
- **Type comp FD** — здесь указываются, какие ACL-права имеет данная роль при обращении к тому или иному подтипу типа FD;
- **Type comp DEV** — аналогично для типа DEV;
- **Type comp Process** — аналогично для типа Process;
- **Type comp IPC** — аналогично для типа IPC;
- **Type comp SCD** — аналогично для типа SCD;
- **Admin Type** — устаревший параметр, указывающий тип этой роли: Системный Администратор, Ролевой Администратор (Офицер безопасности) или Простой Пользователь (можно вместо него пользоваться первыми четырьмя параметрами);
- **Default FD Create Type** — при создании объекта типа FD будет использован соответствующий подтип (т.е., например, может быть роль, создающая только секретные файлы и каталоги, пользоваться которыми смогут только роли с соответствующими ACL-правами);
- **Default Process CreateType** — аналогично для создаваемых (клонируемых) процессов;

— **Default Process Chown Type** — подтип процесса после смены владельца (setuid);
— **Default Process Execute Type** — подтип процесса после запуска;
— **Default IPC Create Type** — подтип новых IPC каналов, семафоров и т.д.

Такое количество настроек для роли позволяет не только усложнить себе жизнь, но и гибко настроить систему безопасности, достигая при этом совершенно фантастических результатов.

3.3.1.8. Модуль FF

При помощи данного модуля можно назначать атрибуты целым деревьям каталогов. Например, можно назначить флаг *"read_only"(только для чтения)* для каталогов `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin`.

3.3.1.9. Functional Control (FC)

Простейшая ролевая модель, которая назначает каждому пользователю роль, например главный пользователь, офицер безопасности или системный администратор. Каждый объект получает категорию, например, главный, секретный или системный объект.

FC может быть полностью выражен RC-моделью.

3.3.1.10. Simone Fischer-Hubner's Privacy Model (PM)

Эта модель была представлена на 17-ой Национальной Конференции Компьютерной Безопасности в Балтиморе, США, в 1994 году ее разработчиком Simone Fischer-Hubner. Это следствие из правил Федерального Немецкого Закона о Секретности и директивы Европейского Союза о безопасности.

Модель сфокусирована на безопасности. Конфиденциальность, целостность и доступность представлены для личных данных и процедур по определению необходимых доступов.

Контрольные системные данные, такие как глобальные настройки или данные аутентификации, могут быть защищены только путем объявления их личными данными. Если для каких-то данных это сделать невозможно, то они не могут быть защищены.

Эта модель может быть использована для хранения и обработки персональных данных. Для защиты системных данных без администрирования наверху, рассматривая их как персональные данные, должна использоваться другая модель, например FC, RC или ACL.

3.3.1.11. Malware Scan (MS)

Это не настоящая модель контроля доступа, но она предпочтительнее системной модели защиты от нежелательного программного кода. Выполнение, чтение и передача файлов, инфицированных нежелательным программным кодом, может быть предотвращена.

Если данные, особенно программы, перемещены из других систем, то во избежание широкого распространения инфекции должна быть использована эта модель. Однако определен может быть только известный алгоритму сканера инфицированный код. На Linux — это вирус bliss, варианты A и B, и некоторые вирусы DOS.

3.3.2. Security-Enhanced Linux

Security-Enhanced Linux (SELinux) [17] имеет такое же назначение, как RSBAC, и также представляет собой дополнения к ядру и набор утилит. Обе системы доступны под лицензией GPL, однако, разработка Security Enhanced Linux продвигается Агентством национальной безопасности США.

Security-Enhanced Linux обеспечивает гибкую архитектуру принудительного контроля доступа (flexible mandatory access control architecture — FLASK) [18], использующую развитый язык описания конфигураций политики безопасности. С использованием этого языка описаний разработана конфигурация системы, реализующая идеологию Type Enforcement.

3.3.2.1. DTE

Type Enforcement — это матрица доступа, организованная для эффективности в классы эквивалентности. Действительно, обычная матрица доступа представляет собой двухмерную таблицу, по одной оси которой располагаются субъекты доступа (активные именованные сущности — процессы, пользователи), по другой — объекты доступа (пассивные сущности — файлы, каталоги, устройства и другие ресурсы), а в ячейках на пересечении указываются операции, которые субъект может выполнять над объектом. В системе с большим количеством сущностей построить такую матрицу может оказаться попросту невозможно. Но в этом и нет необходимости: многие объекты схожи по своим свойствам, а функции многих субъектов совпадают, что позволяет объединить их в классы эквивалентности и строить матрицу для классов, проведя, фактически, типизацию сущностей.

В терминологии данной системы политика безопасности определяет набор доменов и типов. Каждый субъект (процесс) в каждый момент времени ассоциирован с определенным доменом, а каждый объект — с определенным типом. Как и в классической матрице, в политике определяются возможные виды доступа доменов к типам и допустимые способы взаимодействия между доменами. В частности, в зависимости от используемых типов, может происходить автоматическое переключение домена.

3.3.2.2. Роли

В политике безопасности также определен набор ролей. Для пользователей первичная установка роли происходит в процедуре регистрации (login), а переключение роли — при помощи команды newrole (в некотором роде аналог su). Системные процессы работают с ролью system_r,

обычные пользователи — с ролью usr_r, а для системных администраторов зарезервирована роль sysadm_r.

3.3.2.3. Домены

Для каждой роли в политике безопасности задается набор доменов, в которых допускается работа с этой ролью. Всем пользовательским ролям назначается стартовый домен: user_t для роли user_r и sysadm_t для роли sysadm_r. По мере выполнения программ, запускаемых из стартовой оболочки shell, может происходить автоматическое перемещение в другие домены для обеспечения изменения привилегий. Выбор домена, в который должно быть произведено перемещение, осуществляется не только исходя из типа запускаемой программы, но и с учетом текущего домена. В частности, при запуске браузера Netscape обычным пользователем (текущий домен user_t), произойдет перемещение в домен user_netscape_t, а при запуске этой же программы администратором (текущий домен sysadm_t) перемещение произойдет в другой домен — sysadm_netscape_t. Такой подход не позволит программе выполнить потенциально опасные действия — соответствующий домен серьезно ограничит права администратора. В обычных дистрибутивах Linux в случае с браузером Netscape проблема решалась проще — в настройке по умолчанию программа просто не запускалась с правами root.

Для администраторов в Security-Enhanced Linux заданы достаточно жесткие ограничения. В частности, нельзя зайти в систему удаленно — при необходимости такой процедуры необходимо сначала произвести вход обычным пользователем, а потом переключиться на административную роль при помощи команды newrole, производящей дополнительную аутентификацию. Впрочем, и в большинстве современных Unix-подобных системах администратору также запрещен удаленный вход и для этой цели используется команда su.

Пример с браузером не случаен — ему уделено особое внимание в перечне целей политики безопасности. Во избежание исполнения браузером вредоносного динамического кода (Java-апплеты, сценарии JavaScript), для него определен специальный домен (точнее, два домена — для пользователей и администраторов), ограничивающий полномочия. Причем, определяются два подтипа: один ограничивает доступ браузера к локальным файлам только чтением, другой допускает запись в них.

3.3.2.4. Политика безопасности

Политика безопасности должна контролировать различные формы прямого доступа к данным, поэтому в ней определяются различные типы для устройств памяти ядра (kernel memory device), дисковых устройств, специальных файлов в каталоге `/proc`, а также различные домены для процессов, которым необходим доступ к этим ресурсам. Обеспечение целостности ядра достигается определением различных типов для загрузочных файлов, объектных файлов подключаемых модулей, утилит для работы с модулями и файлов конфигурации модулей. Соответствующим процессам,

которым требуется право записи в эти файлы, назначаются специальные домены.

Системное ПО, файлы конфигурации и журналы также нуждаются в защите. Для этой цели также определяются отдельные типы для системных библиотек, исполняемых файлов, файлов конфигурации и журналов, а работающим с ними программам и ролям назначаются специальные домены. В результате запись журналов может вести только система регистрации (syslog), а модификация системного ПО производится только администраторами. Есть решение и для уже упомянутой проблемы, присущей программам с повышенными привилегиями (с установленными флагами suid или sgid). Для таких программ также назначаются отдельные домены, не позволяющие им превысить необходимые полномочия.

Политика безопасности уделяет особое внимание тщательному разделению процессов по привилегиям и защите привилегированных процессов от выполнения ошибочного или опасного кода. Путем установки атрибута executable только на необходимые для исполнения привилегированным процессом программы, политика безопасности может достичь того, что при входе в привилегированный домен процессу будет позволено выполнение только стартовой программы для данного домена, динамического компоновщика и системных разделяемых библиотек. Администратор ограничен выполнением системных программ и программ, созданных им самим — выполнение программ, созданных другими пользователями запрещено. Более того, система минимизирует взаимодействие между обычными пользовательскими процессами и системными. Только системным процессам и администраторам разрешен доступ к записям в procfs, относящимся к другим доменам; ограничено использование вызова ptrace по отношению к другим процессам и доставка сигналов между разными доменами. При использовании файловой системы также приняты дополнительные меры предосторожности: домашние каталоги администраторов и обычных пользователей отнесены к разным типам; создаваемым в каталогах общего пользования (`/tmp` и др.) файлам также присваиваются разные типы, в зависимости от создавшего их домена.

3.3.2.5. Сравнение RSBAC и SELinux

Вся дополнительная информация, используемая RSBAC, хранится в дополнительном каталоге, который доступен только ядру системы. В зависимости от описанного уровня абстракции исполняемых задач и необходимой степени защиты выбираются различные сочетания модулей. Можно обойтись списками ACL — для большинства применений этого хватит, можно защитить системную информацию и существенную с точки зрения безопасности информацию, используя ролевую модель.

RSBAC дает возможность избавить систему от общих недостатков Unix. Появление должности «офицер безопасности» (security officer, security administrator) решает проблему не подконтрольности основного администратора системы, а категории «офицер по защите данных» (data protection officer) децентрализует администрирование, позволяя практически реализовать принцип «четырех глаз», в соответствии с которым

все критичные операции не должны производиться в одиночку. Действительно, возможно построение системы, в которой нового пользователя попрежнему заводит root, однако, его уровень безопасности и права доступа к ресурсам задаются офицером безопасности и офицером по защите данных.

Стоит, однако, помнить об одной простой вещи: любая защита накладывает ограничения, а степень защищенности всегда обратно пропорциональна удобству работы с системой. Если говорить конкретно о модулях RSBAC, то в частности, для модуля MAC при попытке перехода в защищенный каталог может не работать команда `cd`. Модуль интерпретирует данную операцию, как попытку открытия объекта с более высокой меткой безопасности при открытом объекте с менее высокой меткой, а это запрещено идеологией модели — создается предпосылка перетока «секретной» информации в «несекретный» объект. Погоня за секретностью и безопасностью может привести к временной блокировке доступа к данным, а в отдельных случаях — и к их потере. Впрочем, эта проблема не является уже специфичной для RSBAC.

RSBAC не занимается аутентификацией пользователей, проверкой целостности файлов, контролем доступа к фрагментам файлов и другим операциям, которыми не может управлять ядро. Так, если программа имеет права на доступ к файлу `/etc/passwd` для записи нового пользователя, то это означает, что ей позволено изменять весь файл целиком, а не отдельную запись; однако такая «неизбирательность» не всегда допустима. Поэтому в защищенной системе механизм RSBAC логично сочетать с другими средствами защиты, которые работают на более высоком уровне приложений. С помощью RSBAC можно добиться реализации принципа минимизации полномочий процессов и пользователей, причем не только на уровне отдельных идентификаторов пользователя UID, но и в зависимости от роли, которую исполняет процесс с конкретным UID.

RSBAC можно интегрировать с любым дистрибутивом Linux, необходимо лишь внести соответствующие исправления в исходные тексты ядра системы. Эта процедура осуществляется при помощи стандартной утилиты patch и прилагаемого файла-«заплатки», который предлагается для разных версий ядра. После этого происходит его обычная настройка (make config, make menuconfig и т.п.), при этом в настройке появляется несколько новых пунктов. После загрузки ядра настройка системы производится при помощи прилагаемых утилит администрирования. Настройка возможна как при помощи иерархической системы меню, так и при помощи командной строки с параметрами.

По сравнению с RSBAC система Security-Enhanced Linux имеет хорошую предопределенную политику безопасности. Ее сложно применить для «небольшого усиления» стандартного дистрибутива Linux — настройка достаточно сложна и невозможна без изучения специального языка конфигурации. Но это не недостатки, а скорее следствия целей создания системы. Но не стоит пытаться создавать на базе идеологии Type Enforcement операционную систему общего назначения, которая автоматически обеспечит надежность, безопасность, выполнение принципа минимальных привилегий [25] и другие свойства любым запускаемым в ней приложениям. Type Enforcement — это механизм, позволяющий произвести инте-

грацию приложений и менеджера ресурсов, сведя при этом к минимуму присущие им недостатки. Это средство построения специализированных защищенных систем, в которых все лишние функции убраны, а поведение оставленных жестко определено матрицей Type Enforcement.

3.3.3. Выводы

Наиболее эффективную защиту обеспечивают средства, функционирующие на уровне ядра системы. Однако они, как правило, плохо интегрируются между собой. Это затрудняет использование в рамках одной системы защитных механизмов, взятых из разных проектов. Кроме того, даже применение таких средств не сможет обеспечить 100-процентной защиты от взлома. Тем не менее, их применение оправдано по целому ряду причин. Прежде всего, серьезно затрудняется нелегальное проникновение в систему и ее инфицирование. Кроме того, появление в системном журнале записей от соответствующих средств защиты облегчает администратору обнаружение как «неблагонадежного» программного обеспечения, так и следов попыток вторжений злоумышленников.

По поводу рассмотренных проектов можно сказать следующее. Openwall является достаточно хорошим набором патчей к ядру ОС, но развивается достаточно медленно.

PaX предназначен исключительно для противостояния атакам, связанным с переполнением буфера. Следовательно PaX должен использоваться совместно с другими механизмами обеспечения безопасности ОС.

Medusa также как Openwall очень медленно развивается, но может применяться для систем с ядром 2.4 и ниже.

Grsecurity сочетает в себе набор патчей и мандатную модель на основе идентификатора (IBAC). Но в связи с прекращением финансирования этого проекта неизвестно, будет ли он развиваться в дальнейшем.

LIDS является довольно интересным проектом, в котором широко применяются capabilities. Но в России данная система не распространена.

Наиболее предпочтительнымы представляются программные комплексы RSBAC и SELinux, которые являются наиболее мощными и гибкими из применяемых на данный момент.

4. Основы криптографии

4.1. Основные определения

Криптография — искусство и наука защиты сообщений.

Криптоанализ — искусство и наука взлома шифротекстов.

Открытый текст — исходное сообщение.

Зашифрование — процесс маскировки сообщения способом, позволяющим скрыть его суть.

Расшифрование — процесс обратного преобразования шифрованного текста в открытый текст.

Шифрование — зашифрование и расшифрование.

Криптографический алгоритм (шифр) — математическая функция, используемая для зашифрования и расшифрования.

Симметричный алгоритм — алгоритм, в котором ключ зашифрования может быть вычислен из ключа расшифрования, и наоборот.

Потоковый симметричный алгоритм — симметричный алгоритм, который обрабатывает текст побитово.

Блочный симметричный алгоритм — симметричный алгоритм, который обрабатывает группы битов открытого текста, называемые блоками.

Несимметричный алгоритм — алгоритм, в котором ключи зашифрования и расшифрования всегда разные и не могут быть вычислены один из другого. При этом ключ зашифрования является *несекретным (открытым) ключом*, а ключ расшифрования является *секретным ключом*. Текст может быть зашифрован с помощью открытого ключа любым источником, а расшифрован лишь приемником, знающим секретный ключ расшифрования.

Операция XOR — операция «исключающего ИЛИ» (сложение по модулю 2), обозначаемая в математике знаком $\oplus$:

$$
\begin{aligned}
0 \oplus 0 &= 0 \\
0 \oplus 1 &= 1 \\
1 \oplus 0 &= 1 \\
1 \oplus 1 &= 0 \\
a \oplus a &= 0 \\
a \oplus b \oplus b &= a
\end{aligned}
$$

Однонаправленная функция — сравнительно легко вычисляемая функция, но вычисление обратной ей функции затруднено или невозможно.

S-блок (Substitution Box) — узел замены.

Отбеливание — преобразование, при котором выполняется операция XOR над входом блочного алгоритма и частью ключа и XOR над выходом блочного алгоритма и другой частью ключа.

4.2. Криптографические примитивы и механизмы

Криптография является одним из основных инструментов, обеспечивающих конфиденциальность, доверие, авторизацию, электронные платежы, корпоративную безопасность и т.п.

Криптографические методы могут применятся для решения следующих проблем безопасности:

— конфидециадьности передаваемых или хранимых данных
— аутентификации
— целостности передаваемых и хранимых данных
— обеспечения подлинности документов

Базовые методы преобразования информации:

— шифрование(симметричное и несимметричное)
— вычисление хэш-функций
— генерация электронно-цифровой подписи
— генерация последовательности псевдо-случайных чисел

Шифрование — обратимое преобразование данных с целью их сокрытия от посторонних.

Почти все методы шифрования используют ключ шифрования. Ключ шифрования — секретная кодовая последовательность, используемая в процессе преобразования информации.

4.2.1. Классификация ключей по типам алгоритмов

4.2.1.1. Асимметричные ключи

Криптование использует пару — закрытый и открытый ключи, обеспечивая тем самым то, что данные могут быть зашифрованы одним ключом, а расшифрованы могут быть только другим ключом. В действительности ключи похожи и могут использоваться альтернативно: один ключ пары шифрует, другой — расшифровывает.

Ключи основаны на больших простых числах, длина которых в битах ювляется достаточной для того, чтобы было невозможно расшифровать сообщение, не зная второго ключа пары.

Открытый ключ может быть доступен каждому, в то время как второй (секретный) ключ должен храниться в надежном месте. Любой может отправить Вам зашифрованное Вашим открытым ключом сообщение, но только Вы можете его открыть (расшифровать и прочитать). И наоборот, Вы можете засертифицировать сообщение так, чтобы было ясно, что именно Вы зашифровали его Вашим секретным ключом и только ассоциированный открытый ключ может корректно расшифровать сообщение.

Криптография с открытыми ключами обеспечивает все требования, предъявляемые к криптографическим системам. Но реализация алгоритмов требует больших затрат процессорного времени. Поэтому в чистом виде криптография с открытыми ключами в мировой практике обычно не применяется. Для шифрования данных используются симметричные (се-

ансовые) ключи, которые в свою очередь шифруются с использованием открытых ключей для передачи сеансовых ключей по сети.

4.2.1.2. Симметричный ключ

Симметричным ключом называется ключ, используемый как для шифрования, так и для расшифровки данных. Симметричный ключ является потенциально небезопасным. Если злоумышленник сможет перехватить ключ, то информация не будет более являться секретной. Нельзя передавать симметричный ключ в открытом виде. Симметричный ключ вкладывается внутрь сообщения, зашифрованного асимметричным ключом.

Преимущества криптографии с симметричными ключами:

— Производительность
Производительность алгоритмов с симметричными ключами очень велика.
— Стойкость
Криптография с симметричными ключами очень стойкая, что делает практически невозможным процесс дешифрования. При прочих равных условиях (общий алгоритм) стойкость определяется длиной ключа. При длине ключа 256 бит необходимо произвести 10^{77} переборов для определения ключа.

Недостатки криптографии с симметричными ключами:

— Распределение ключей
Так как для шифрования и расшифрования используется один и тот же ключ, при использовании криптографии с симметричными ключами требуются очень надежные механизмы для распределения ключей.
— Масштабируемость
Так как используется единый ключ между отправителем и каждым из получателей, количество необходимых ключей возрастает в геометрической прогрессии. Для 10 пользователей нужно 45 ключей, а для 1000 уже 499500.
— Ограниченное использование
Так как криптография с симметричными ключами используется только для шифрования данных и ограничивает доступ к ним, при ее использовании невозможно обеспечить аутентификацию и неотрекаемость.

4.2.2. Алгоритмы шифрования

4.2.2.1. Симметричные блочные шифры

Симметричные блочные шифры для шифрования и расшифрования информации используют один и тот же ключ и шифруют информацию блоками. Ниже в табл. 4.1 перечислены основные используемые в настоящее время алгоритмы, их длины блока и длины ключа.

Таблица 4.1

Симметричные блочные шифры

Алгоритм	Длина ключа (в битах)	Длина блока (в битах)
DES	64	64
Blowfish	Переменная, до 448	64
IDEA	128	64
ГОСТ 28147-89	256	64
RC5	Переменная	Переменная

DES. DES предназначен для шифрования данных 64-битовыми блоками. Алгоритм представляет собой комбинацию двух методов шифрования: рассеивания и перемешивания. Фундаментальным строительным блоком DES является применение к тексту единичной комбинации этих методов (подстановка, а за ней — перестановка), зависящей от ключа. Такой блок называется «раундом» или «основным циклом». DES включает 16 раундов, одна и та же комбинация методов применяется к открытому тексту 16 раз.

Схема алгоритма. DES оперирует 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины по 32 бита. Затем выполняется 16 раундов одинаковых действий (рис. 4.1), называемых функцией f, в которых данные объединяются с ключом. После 16-го раунда правая и левая половины объединяются, и алгоритм завершается заключительной перестановкой (обратной к первоначальной).

На каждом раунде биты ключа сдвигаются, а затем из 56 битов ключа выбираются 48 битов. Правая половина данных увеличивается до 48 бит путем перестановки с расширением, складывается операцией XOR с 48 битами смещенного и переставленного ключа, проходит через 8 S-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией f. Затем результат исполнения функции f складывается с левой половиной с помощью еще одной операции XOR. В итоге появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 раундов алгоритма DES.

Если обозначить B_i результат i-й итерации, L_i и R_i — левую и правую половины B_i, K_i — 48-битовый ключ для раунда i, а f — функцию, выполняющую все подстановки, перестановки и операцию XOR с ключом, то раунд можно представить так:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Расшифрование DES. Алгоритм DES позволяет использовать для шифрования и расшифрования блока одну и ту же функцию. Единственное отличие состоит в том, что ключи должны использоваться в обрат-

ном порядке, т.е. если в раундах зашифрования использовались ключи $K_1, K_2, \ldots, K_{16}$, то ключами расшифрования будут $K_{16}, K_{15}, \ldots, K_1$.

Алгоритм, который создает ключ для каждого этапа, тоже циклличен. Ключ сдвигается направо, а число позиций сдвига равно 0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1.

Режимы работы DES. Определены четыре режима работы: ECB, CBC, OFB и CFB. Банковские стандарты ANSI определяют для шифрования режимы ECB и CBC, а для аутентификации — режимы OFB и n-битовый CFB.

Из-за своей простоты в большинстве существующих коммерческих программ используется режим ECB, хотя этот режим наиболее уязвим к вскрытию. Режим CBC используется редко, хотя и более безопасен.

Варианты DES.

— Многократный DES.
  В ряде реализаций DES используется тройной алгоритм DES.
— DES с независимыми подключами.
  В каждом раунде вместо создания ключей из одного 56-битового ключа используются различные подключи в каждом раунде.
— DESX.
  Вариант DES, разработанный RSA Data Security, Inc. Для маскировки входов и выходов DES в алгоритме DESX использован метод, называемый «отбеливанием» (отбеливанием называют такое преобразование, при котором выполняется операция XOR над входом блочного алгоритма и частью ключа и XOR над выходом блочного алгоритма и другой частью ключа).
— CRYPT.
  CRYPT представляет собой вариант DES, используемый в UNIX. В основном он используется в качестве однонаправленной функции для хранения паролей, но иногда и для шифрования. В отличие от DES, в CRYPT включена зависящая от ключа расширяющая перестановка, обеспечивающая 2^{12} варианта перестановок.
— Обобщенный DES.
  Обобщенный DES (GDES) спроектирован для ускорения исполнения DES и повышения устойчивости алгоритма. GDES работает с блоками открытого текста переменной длины. Блоки шифрования делятся на q 32-битовых подблоков, точное число которых зависит от полного размера блока. В общем случае q равно размеру блока, деленному на 32.
  Функция f рассчитывается один раз для каждого раунда для крайнего правого блока. Результат с помощью операции XOR объединяется со всеми остальными частями, которые затем циклически смещаются направо. В последний раунд внесено незначительное изменение, с тем, чтобы процессы зашифрования и расшифрования отличались только порядком применения подключей.

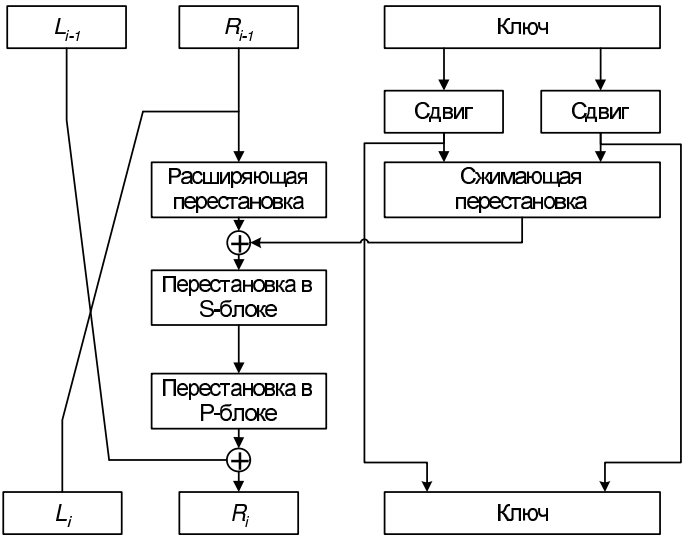


Рис. 4.1. Один раунд алгоритма DES

Алгоритм RC2. RC2 предназначен для замены DES и представляет собой шифр с 64-битовым блоком и имеет переменную длину ключа. Скорость шифрования не зависит от размера ключа. *S*-блоки в алгоритме RC2 не используются.

Алгоритм IDEA. International Data Encription Algorithm (IDEA) — симметричный блочный шифр, работающий с 64-битовыми блоками открытого текста и ключами длиной 128 бит. IDEA использует рассеивание и перемешивание. В основе конструкции алгоритма лежит «смешение операций различных алгебраических групп». Смешиваются три алгебраические группы: операция XOR, сложение по модулю 2^{16}, умножение по модулю $2^{16} + 1$. Все эти операции работают с 16-битовыми блоками.

Описание алгоритма IDEA. 64-битовый блок делится на четыре 16-битовых подблока X_1, X_2, X_3 и X_4. Эти подблоки используются как входы первого раунда алгоритма. Всего в алгоритме 8 раундов. На каждом раунде четыре подблока подвергаются операциям XOR, сложению и умножению друг с другом и шестью 16-битовыми подключами. Между раундами второй и третий блоки меняются местами. В выходном преобразовании четыре подблока объединяются с четырьмя подключами.

На каждом раунде события происходят в следующем порядке (рис. 4.2):

1. Перемножаются X_1 и первый подключ.
2. Складываются X_2 и второй подключ.
3. Складываются X_3 и третий подключ.
4. Перемножаются X_4 и четвертый подключ.
5. Выполняется операция XOR над результатами раундов 1 и 3.
6. Выполняется операция XOR над результатами раундов 2 и 4.
7. Перемножаются результаты раунда 5 и пятый подключ.
8. Складываются результаты раундов 6 и 7.
9. Перемножаются результаты раунда 8 и шестой подключ.
10. Складываются результаты раундов 7 и 9.
11. Выполняется операция XOR над результатами раундов 1 и 9.
12. Выполняется операция XOR над результатами раундов 3 и 9.
13. Выполняется операция XOR над результатами раундов 1 и 10.
14. Выполняется операция XOR над результатами раундов 4 и 10.

На выходе раунда создаются четыре подблока — результаты действий 11–14. Если поменять местами два внутренних подблока в любом (но не в последнем) раунде, то получатся исходные данные для следующего раунда.

После восьмого раунда выполняется заключительное преобразование:

1. Перемножаются X_1 и первый подключ.
2. Складываются X_2 и второй подключ.
3. Складываются X_3 и третий подключ.
4. Перемножаются X_4 и четвертый подключ.

Наконец четыре подблока снова соединяются, образуя шифрованный текст.

В алгоритме используются 52 подключа — шесть в каждом из восьми раундов и еще четыре в заключительном преобразовании. Сначала 128-битовый ключ разделяется на восемь 16-битовых подключей. Затем ключ циклически сдвигается налево на 25 битов и снова делится на восемь подключей. Первые четыре подключа используются в раунде 2, а оставшиеся четыре — в раунде 3. Ключ циклически сдвигается налево на 25 битов для получения следующих восьми подключей, и т.д. до завершения алгоритма.

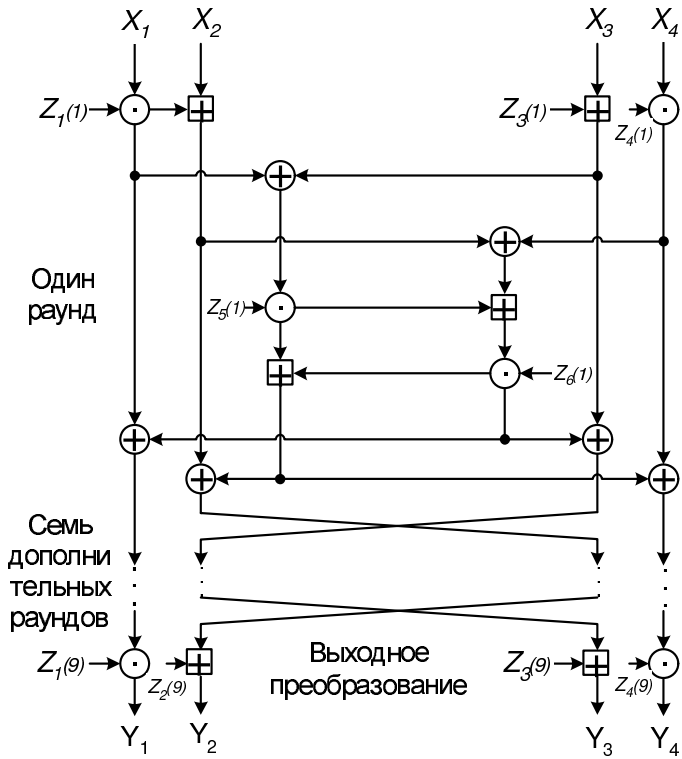
Расшифрование выполняется аналогично шифрованию, за исключением того, что подключи инвертируются и немного изменяются.

Алгоритм ГОСТ 28147-89. ГОСТ представляет собой 64-битовый алгоритм с 356-битовым ключом. Кроме того, применяется добавочный ключ. В процессе работы алгоритма последовательно, на 32 раундах, выполняется простой алгоритм шифрования (рис. 4.3).

Вначале шифруемый текст разбивается на левую L и правую R половины. На раунде i используется подключ K_i. На каждом i-ом раунде алгоритма выполняются следующие операции:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Функция f проста. Сначала над правой половиной шифруемого текста и i-ым подключом выполняется операция сложения по модулю 2^{32}.



X_i : 16-биовый подблок открытого текста
Y_i : 16-биовый подблок шифротекста
$Z_{j(r)}$: 16-биовый подблок ключа
⊕ : побитовая операция XOR над 16-битовыми подблоками
⊞ : сложение по модулю 2^{16} 16-разрядных целых чисел
⊙ : сложение по модулю $2^{16}+1$ 16-разрядных целых чисел, причем нулевой подблок соответствует 2^{16}

Рис. 4.2. Алгоритм IDEA

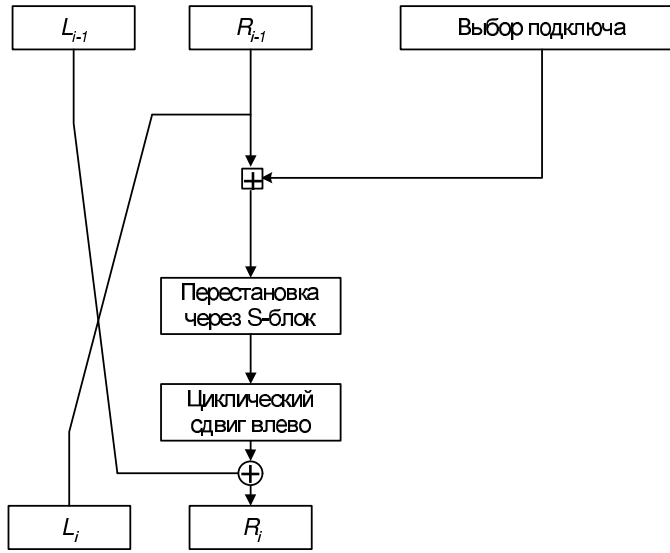


Рис. 4.3. Один раунд алгоритма ГОСТ

Результат операции разбивается на восемь 4-битовых фрагмента; каждый из этих фрагментов поступает на вход своего S-блока («узла замены»). В алгоритме используются восемь различных S-блоков. Первый 4-битовый фрагмент поступает в первый S-блок, следующий 4-битовый фрагмент — во второй S-блок и т.д. Каждый из S-блоков является перестановкой чисел от 0 до 15. Все восемь S-блоков различны и должны храниться в секрете.

Выходы всех восьми S-блоков объединяются в одно 32-х битовое слово; затем все это слово сдвигается циклически влево на 11 битов. На завершающем этапе результат объединяется операцией XOR с левой половиной шифруемого текста, создавая новую правую половину шифруемого текста, а правая половина становится новой левой половиной. После выполнения этих операций 32 раза все будет завершено.

Подключи генерируются следующим образом: ключ размером 256 бит разбивается на восемь 32-битовых блоков $k_1, k_2, \ldots, k_8$. На каждом раунде используется свой подключ. Расшифрование выполняется точно так же, как шифрование, просто инвертируется порядок подключей k_i.

Алгоритм Blowfish. Blowfish представляет собой 64-битовый блочный алгоритм шифрования с ключом переменной длины. Алгоритм состоит из двух частей: расширения ключа и шифрования данных. Расширение ключа преобразует ключ длиной 448 бит в несколько массивов подключей общим размером 4168 байт.

Шифрование данных заключается в последовательном исполнении простой функции 16 раз. На каждом раунде выполняются зависимая от ключа перестановка и зависимая от ключа и данных подстановка. Используются только операции сложения и XOR над 32-битовыми словами. Единственные дополнительные операции каждого раунда — четыре взятия данных из индексированного массива.

В алгоритме используется множество подключей, которые должны быть вычислены до начала шифрования и расшифрования данных.

Шифрование 64-битного блока входного текста включает в себя (рис. 4.4):

1) разбиение блока на два 32-полублока x' и x'';
2) выполнение на каждом раунде $i = 1, \ldots, 16$ следующих операций:
   - левый полублок x' суммируется по модулю 2 с соответствующим подключом K_i;
   - полученная сумма x разбивается на четыре 8-битных подблока a, b, c, d, каждый из которых является адресом элемента соответствующей подстановки S_j, $1 \leqslant j \leqslant 4$. 32-битный результат обработки полублока оператором F формируется по формуле: $F(x) = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d}$, где „$\oplus$“ — сложение по модулю 2, „$+$“ — сложение по модулю 2^{32};
   - компоненты левого зашифрованного полублока $F(x')$ суммируются по модулю 2 с компонентами правого полублока x'';
   - компоненты левого и правого полублоков меняются местами (на последнем раунде этот шаг не выполняется).
3) после 16 раундов левая и правая половины полученного текста суммируются по модулю 2 с подключами K_{18} и K_{17} соответственно.

Выходным зашифрованным текстом является конкатенация двух полублоков, полученных на шаге 3.

Расшифрование осуществляется аналогично, с обратным порядком использования подключей: $K_{18}, K_{17}, \ldots, K_1$.

Подключи $K_1, K_2, \ldots, K_{18}$ и S-блоки S_1, S_2, S_3, S_4 вырабатываются из секретного ключа KS с использованием алгоритма шифрования, а именно:

1. Начальный массив K_i подключей и S-блоков инициируются фиксированной 128-битной строкой (дробной частью числа $pi = 3,14159\ldots$ в шестнадцатиричном представлении).
2. K_1 суммируется по модулю 2 с первыми 32 битами ключа KS; K_2 суммируется по модулю 2 со следующими 32 битами ключа KS и т.д. для всех разрядов ключа. В случае короткого ключа KS для построения массива K_i используется конкатенация ключа вида: $KSKS$, $KSKSKS$ и т.д.
3. 64-битный блок $0 = (0, 0, \ldots, 0)$ зашифровывается алгоритмом Blowfish на подключах, полученных на шагах 1 и 2.
4. K_1 и K_2 заменяются полученным на шаге 3 результатом шифрования $C0 = Blowfish(0)$.
5. Шифротекст $C0$ зашифровывается алгоритмом Blowfish на модифицированных подключах.
6. K_3 и K_4 заменяются полученным на шаге 5 реультатом шифрования $C1 = Blowfish(C0)$.

7. Процесс продолжается циклически до тех пор, пока не будут получены сначала все пары подключей (9 пар), а затем все пары элементов S-блоков (512 пар). Таким образом, всего выполняется 521 раунд шифрования, на каждом из которых происходит модификация подключей.

Полученные массивы подключей и S-блоков используются для шифрования данных.

Алгоритм RC5. RC5 представляет собой блочный шифр с множеством параметров: размером блока, размером ключа, и числом раундов. В RC5 предусмотрены три операции: XOR, сложение и циклические сдвиги. Переменные циклические сдвиги представляют собой нелинейную функцию.

Секретный b-байтный ($0 \leqslant b \leqslant 255$) ключ KS преобразуется в расширенный ключ $K = K_0 K_1 \dots K_{2R+1}$, где R — число циклов, а размер W слова K_i может быть 16, 32 или 64 бита.

Построение расширенного ключа включает в себя следующие операции:

— представление b-байтного ключа в виде последовательности W-битных слов $L = L_0 L_1 \dots L_{r-1}$, где $r = \frac{8(b-1)}{W+1}$;
— построение фиксированной (не зависящей от ключа KS) псевдослучайной последовательности $S = S_0 S_1 \dots S_{2R+1}$ на основе чисел $e = 2.71828$ (основание натурального логарифма) и $\varphi = 1.67803$ (золотое сечение);
— получение расширенного ключа K из последовательности L и S по алгоритму, аналогичному алгоритму шифрования, в котором на каждом шаге: $a \leftarrow (S_i + a + b) \lll 3$; $S_i \leftarrow a$; $b \leftarrow (L_j + a + b) \lll (a + b)$; $L_j \leftarrow b$, где „+“ — сложение по модулю 2^W, „$\lll x$“ — циклический сдвиг влево на x бит.

Исходными значениями для a, b, i, j являются нули, а число итераций определяется размером ключа KS.

Алгоритм шифрования:

1. блок входного открытого текста длины $2W$ бит разбивается на два полублока A и B по W бит;
2. первые два слова расширенного ключа K_0 и K_1 суммируются по модулю 2^W с полублоками A и B соответственно;
3. выполняется циклическая итерация: $A \leftarrow ((A \oplus B) \lll W_B) + K_{2i}$, $B \leftarrow ((B \oplus A) \lll W_A) + K_{2i+1}$, $i = 1, \dots, R$, где „+“ — сложение по модулю 2^W, „$\lll W_X$“ — циклический сдвиг влево на X, т.е. сдвиг на $X (\mathrm{mod} W)$.

Выходным зашифрованным текстом является конкатенация итоговых значений полублоков A и B после R циклов.

При расшифровании операции выполняются в обратном порядке с тем же расширенным ключом.

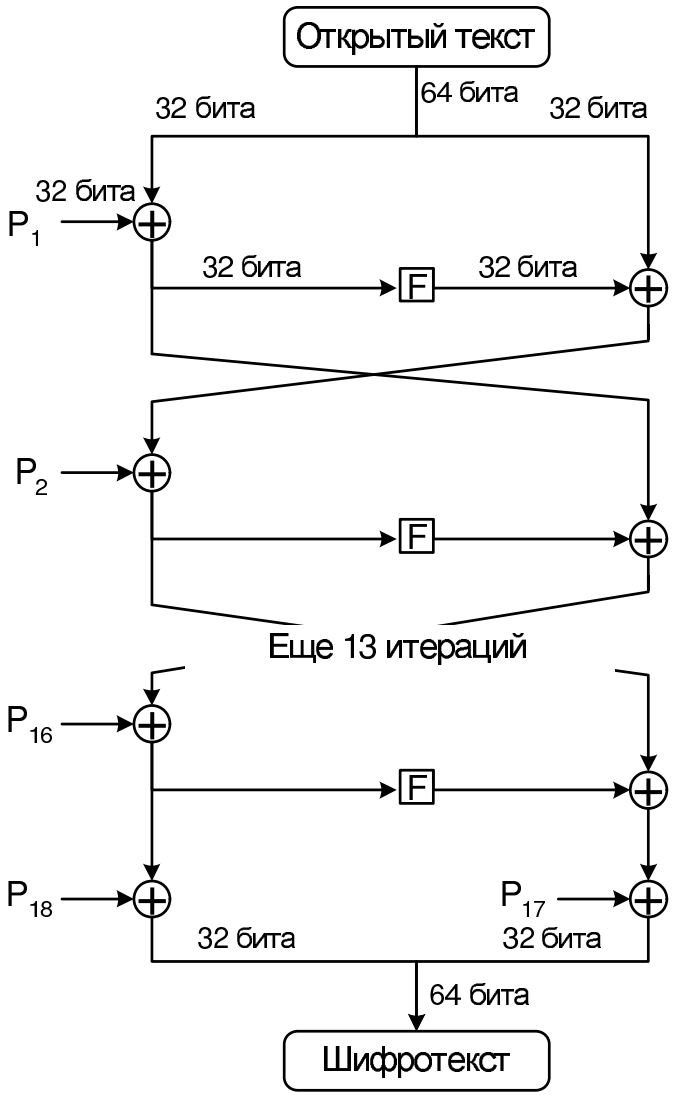


Рис. 4.4. Алгоритм Blowfish

4.2.2.2. Симметричные поточные шифры

Симметричные поточные шифры используют симметричный ключ, но выполняют шифрование входного потока побайтно или иногда побитно.

Идея поточного шифра состоит в том, что на основе симметричного ключа вырабатывается «ключевая последовательность» или «гамма» — последовательность, которая складывается по модулю два (операция XOR) с входным потоком. Поточные шифры, как правило, более производительны, чем блочные и используются для шифрования речи, сетевого трафика и иных данных с заранее неизвестной длинной.

Алгоритм RC4. RC4 — потоковый шифр с переменным размером ключа. Алгоритм работает в режиме OFB, т.е. гамма не зависит от открытого текста. Используется блок $8 * 8$: $S_0, S_1, \ldots, S_{255}$. Элементы представляют собой перестановку чисел от 0 до 255, а перестановка зависит от ключа переменной длины. В алгоритме применяются два счетчика i и j с нулевыми начальными значениями.

Чтобы сгенерировать случайный байт, необходимо выполнить следующие операции:

$$i = (i + 1) \bmod 256$$
$$j = (j + S_i) \bmod 256$$

Поменять местами S_i и S_j

$$t = (S_i + S_j) \bmod 256$$
$$K = S_t$$

Байт K используется в операции XOR с открытым текстом для получения шифрованного текста или в операции XOR для получения открытого текста.

Для инициализации S-блока необходимо заполнить его линейно: $S_0 = 0, S_1 = 1, \ldots, S_{255} = 255$. Затем ключом заполняется другой 256-байтовый массив. Если необходимо, ключ повторяется, чтобы заполнить весь массив: $K_0, K_1, \ldots, K_{255}$. Затем:

$$j = 0$$
для i от 0 до 256
$$j = (j + S_i + K_i) \bmod 256$$
Поменять местамиS_i и S_j

4.2.2.3. Асимметричные алгоритмы

Алгоритм RSA. Безопасность алгоритма RSA основана на трудоемкости разложения на множители (факторизации) больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел,

разрядностью 100–200 десятичных цифр. Предполагается, что восстановление открытого текста по шифрованному тексту и открытому ключу равносильно разложению числа на два больших простых множителя.

Для генерации двух ключей применяются два больших случайных простых числа p и q. Для обеспечения максимальной безопасности p и q должны иметь одинаковую длину. Рассчитывается произведение $n = pq$.

Затем случайным образом выбирается ключ шифрования e так, чтобы e и $(p-1)(q-1)$ являлись взаимно простыми числами. С помощью расширенного алгоритма Евклида вычисляется ключ расшифрования d, такой что:

$$ed \equiv 1(\mathrm{mod}(p-1)(q-1)).$$

Другими словами:

$$d = e^{-1}(\mathrm{mod}(p-1)(q-1)).$$

Заметим, что d и n также взаимно простые числа.

Числа e и n — это открытый ключ, а число d — закрытый. Числа p и q могут быть отброшены, но они не должны быть раскрыты.

При шифровании сообщение m разбивается на цифровые блоки m_i, размерами меньше n (для двоичных данных выбирается самая большая степень числа 2, меньшая n). Зашифрованное сообщение c будет состоять из блоков c_i, причем длина блока c_i равна длине блока m_i. Таким образом, формула шифрования имеет вид:

$$c_i = {m_i}^e \bmod n$$

При расшифровке сообщения для каждого зашифрованного блока c_i вычисляется:

$$m_i = {c_i}^d \bmod n$$

Так как

$${c_i}^d = ({m_i}^e)^d = {m_i}^{k(p-1)(q-1)+1} = m_i,$$

все по $\mathrm{mod}n$, то формула восстанавливает сообщение.

Таким образом

$$d = e^{-1}(\mathrm{mod}(p-1)(q-1)) \text{ — закрытый ключ}$$
$$c = m^e \bmod n \text{ — шифрование}$$
$$m = c^d \bmod n \text{ — расшифрование .}$$

Алгоритм Эль-Гамаля. Открытый ключ:

1) простое число p;
1) образующая g группы ${F_p}^*$, порядок которой имеет большой простой общий делитель;
1) экспонента $y = g^x(\mathrm{mod}p)$.

Секретным ключом расшифрования служит целое число x, $0 < x < p-1$. Для шифрования открытого текста $P \in F_p$ источник выполняет следующие действия:

1) выбирает случайное число k, $0 < k < p - 1$;
2) вычисляет $r \equiv y^k (\mathrm{mod} p)$.

Шифротекстом является пара (C_1, C_2), где $C_1 = g^k (\mathrm{mod} p)$, $C_2 = rP (\mathrm{mod} p)$.

Алгоритм расшифрования:

1) r восстанавливается при помощи своего секретного ключа x, т.е. $r = {C_1}^x (\mathrm{mod} p)$;
2) восстанавливается открытый текст P: $P = r^{-1} C_2 (\mathrm{mod} p)$.

4.2.3. Хэш-функция

Криптографическими методами можно обеспечить не только конфидециальность, но и проконтролировать целостность передаваемых или хранимых данных. Контроль целостности в основном производится путем расчета некоторой «контрольной суммы» данных.

Для многих приложений простой контрольной сумма оказывается достаточно, особенно если важна скорость обработки данных и заранне не известен объем данных.

Проблема простых алторитмов вычисления контрольной суммы в том, что достаточно легко подобрать несколько массивов данных, имеющих одинаковую контрольную сумму. Криптографически стойкие контрольный суммы вычисляются как результат применения к исходному тексту хэш-функции.

Все используемые в настоящее время хэш-функции являются так называемыми кандидатами в односторонние функции.

Под односторонней функцией понимается функция, определенная, например, на множестве натуральных чисел и не требующая для вычисления своего значения больших вычислительных ресурсов. Но вычисление обратной функции (т.е. по известному значению функции восстановить значение аргумента) оказывается невозможно теоретически или невозможно вычислительно. Строгое существование односторонних функций пока не доказано.

Основными свойствами криптографически-«хорошей» хэш-функции являются свойство рассеивания, свойство стойкости к коллизиям и свойство необратимости.

О необратимости уже было сказано. Коллизией хэш-функции H называется ситуация, при которой существуют два рзличных текста T_1 и T_2, но $H(T_1) = H(T_2)$. Значение хэш-функции всегда имеет фиксированную длину, а на длину исходного текста не накладывается никаких ограничений. Из этого следует, что коллизии существуют. Требование стойкости к коллизиям означает, что для криптографически-«хорошей» хэш-функции для заданного текста T_1 вычислительно невозможно найти текст T_2, вызывающий коллизию.

Свойство рассеивания требует, чтобы минимальные изменения текста, подлежащего хэшированию, вызывали максимальные изменения в значении хэш-функции.

Основными применяеммыми на сегодняшний день алгоритмами, реализующими хэш-функции, являются MD2, MD4, MD5, SHA и его вариант

SHA1, российский алгоритм, описанный стандартон ГОСТ Р 34.11–94. Наиболее часто используются MD5, SHA1 и в Росии ГОСТ Р 34.11–94.

4.2.3.1. MD2, MD4, MD5

Хэш-функции в алгоритмах MD2, MD4, MD5 преобразуют входное сообщение произвольной длины в сжатый 128-битный образ.

Основные операции: сложение по модулю 2^{32}, циклический сдвиг, логические операции $\oplus, \wedge, \vee$.

Алгоритм MD2 отличается от MD4, MD5 способом дополнения сообщения (с применением 16-байтной контрольной суммы), значением стартового вектора хэширования и использованием 256-байтной перестановки, а обработка одного сообщения в нем выполняется за 18 раундов.

В алгоритме MD4 обработка одного блока выполняется за три раунда, каждый из которых включает в себя 16 операций. На каждом раунде используется своя цикловая функция f_j, $1 \leqslant j \leqslant 3$, где $f_1 = (X \wedge Y) \vee (\bar{X} \wedge Z)$, $f_2 = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$, $f_3 = X \oplus Y \oplus Z$.

Наиболее используемым является алгоритм MD5, который и будет рассмотрен.

Пусть исходное сообщение m задано t-битной строкой $m_1, m_2, \ldots, m_t$.

Строка сообщения m дополняется до длины, сравнимой с 448 по модулю 512 (т.е. длине сообщения не достает ровно 64 бит, чтобы быть кратной 512), а именно: к сообщению присоединяется одна „1“, а затем необходимое количество нулей. Причем дополнение строки выполняется даже в том случае, если ее длина уже сравнима с 448 по модулю 512 (в этом случае к сообщению добавляется 512 бит).К полученной строке присоединяется 64-битное представление числа t. Если $t \geqslant 2^{64}$, то используются только младшие 64 бита числа t.

Пусть $M = M_1 M_2 \ldots M_n$ (где M_i — 16-словный блок с размером слова 32 бита) — дополненное сообщение, а n кратно 16.

Стартовый вектор хэширования MD_0 имеет длину 128 бит и представляет собой конкатенацию четырех 32-битных слов $md_0 || md_1 || md_2 || md_3$, где $md_0 = 01234567$, $md_1 = 89abcdef$, $md_2 = fedcba98$, $md_3 = 76543210$.

В алгоритме MD5 используются четыре цикловые функции f_j, $1 \leqslant j \leqslant 4$, где $f_1(X, Y, Z) = (X \wedge Y) \vee (\bar{X} \wedge Z)$, $f_2(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \bar{Z})$, $f_3(X, Y, Z) = X \oplus Y \oplus Z$, $f_4(X, Y, Z) = Y \oplus (X \vee \bar{Z})$.

Все операции в этих функциях выполняются побитно, т.е. каждый выходной бит зависит только от соответствующих ему входных битов.

Вычисление хэш-функции:

Вход: $M = M_1 M_2 \ldots M_n$ (n блоков по 512 бит).

Алгоритм: Для всех $i = 1, \ldots, n$ $MD_i \leftarrow g(MD_{i-1}, M_i)$.

Выход: $h(M) = MD_n$.

При вычислении используется накопитель E, содержаий четыре 32-битных слова A, B, C, D. Исходным заполнителем накопителя E являются слова стартового вектора хэширования MD_0.

Обработка 16-словного блока M_i, $1 \leqslant j \leqslant n$, сообщения M осуществляется за четыре раунда, каждый из которых включает в себя 16 шагов.

На каждом шаге j-го цикла $A \leftarrow B + ((A + f_j(B,C,D) + M_i[s] + r) \lll k)$, $1 \leqslant j \leqslant 4$, где $M_i[s]$ — слово, выбранное из M_i; s, r, k — параметры шага; $X \lll k$ — циклический сдвиг слова X влево на k бит; „+“ — операция сложения по модулю 2^{32}. Таким образом, на каждом шаге выполняется четыре операции сложения, одна операция сдвига и вычисляется значение одной цикловой функции.

Для каждой итерации MD_i представляет собой конкатенацию текущих значений четырех слов накопителя E. После обработки блока M_n итоговым сжатым образом сообения будет 128-битная строка из четырех слов $MD_n = A||B||C||D$.

4.2.3.2. SHA-1

SHA-1 является модифицированной версией алгоритма SHA (Secure Hash Algorithm), который используется в стандарте на хэш-функцию SHD (Secure Hash Standart).

Алгоритм SHA-1 сопоставляет сообщению длиной до 2^{64} бит сжатый 160-битный образ, используемый, в частности, для генерации и проверки цифровой подписи в стандарте DSS. Кроме того, алгоритм SHA-1 создан по образцу MD4.

Основные операции алгоритма: сложение по модулю 2^{32} и по модулю 2, циклический сдвиг.

t-битная строка битов исходного сообщения $m = m_1 m_2 \dots m_t$ дополняется до длины, кратной 512. Если $t \geqslant 2^{64}$, то используются только младшие 64 бита числа t.

Дополнение сообщения выполняется так же, как и для MD-функций, т.е. сначала к m присоединяется одна „1“, затем необходимое количество нулей и 64-битное представление числа t — длины исходного сообщения.

Таким образом, дополнение сообщения состоит из n 512-битных блоков, т.е. имеет вид $M = M_1 M_2 \dots M_n$ (где M_i — 16-словный блок с размером слова 32 бита)

Стартовый вектор хэширования SHA_0 имеет длину 160 бит и представляет собой конкатенацию пяти 32-битных слов $sha_0||sha_1||sha_2||sha_3||sha_4$, где $sha_0 = 67452301$, $sha_1 = efcdab89$, $sha_2 = 98badcfe$, $sha_3 = 10325476$, $sha_4 = c3d2e1f0$.

В алгоритме SHA-1 используются последовательность функций $f_0, f_1, \dots, f_79$, каждая из которых сопоставляет трем 32-битным словам X, Y, Z одно 32-битное слово f_j, $0 \leqslant j \leqslant 79$. функции f_j имеют вид:

$$f_j(X,Y,Z) = (X \wedge Y) \vee (\bar{X} \wedge Z), \quad 0 \leqslant j \leqslant 19$$
$$f_j(X,Y,Z) = X \oplus Y \oplus Z, \quad 20 \leqslant j \leqslant 39, \quad 60 \leqslant j \leqslant 79$$
$$f_j(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), \quad 40 \leqslant j \leqslant 59.$$

Вычисление хэш-функции h:

Вход: $M = M_1 M_2 \dots M_n$ (n блоков по 512 бит).

Алгоритм: Для всех $i = 1, \dots, n$ $SHA_i \leftarrow g(SHA_{i-1}, M_i)$.

Выход: $h(M) = SHA_n$.

Стандарт SHS предусматривает два альтернативных способа вычисления шаговой функции хеширования g, дающих в результате один и тот же сжатый образ сообщения.

При вычислении используется последовательность 32-битных слов $\{W_i\}$, $0 \leqslant j \leqslant 79$, и два накопителя F и H, содержащие по пять 32-битных слов. Слова в первом накопителе обозначены как A, B, C, D, E, во втором — $H_0, \ldots, H_4$. Исходным заполнением накопителя H являются слова стартового вектора хэширования SHA_0.

Обработка каждого из 512-битных блоков $M_1, M_2, \ldots, M_n$ осуществляется по следующему алгоритму:

1. Блок M_i разбивается на 16 слов $W_0, W_1, \ldots, W_{15}$ по 32 бита.
2. Для $j = 16, \ldots, 79$ $W_j \leftarrow S^1(W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16})$, где операция циклического сдвига $S^n(X)$, $0 \leqslant n < 32$, определяется как $S^n(X) = (X \lll n) \vee (X \ggg 32 - n)$.
3. Содержимое второго накопителя H переписывается в первый F.
4. Для $j = 0, \ldots, 79$

$$\{TEMP \leftarrow S^5(A) + f_j(B, C, D) + E + W_j + K_j;$$
$$E \leftarrow D;\ D \leftarrow C;\ C \leftarrow S^{30}(B);\ B \leftarrow A;\ A \leftarrow TEMP\},$$

   где K_j, $0 \leqslant j \leqslant 79$ — фиксированные константы, значения которых определяются параметром j; $TEMP$ — временная переменная; „+“ — операция сложения по модулю 2^{32}.
5. Содержимое первого накопителя F суммируется по модулю 2^{32} с содержимым второго накопителя H и записывается в H.

Для каждой итерации SHA_i представляет собой конкатенацию текущих значений пяти слов накопителя H. После обработки блока M_n итоговым сжатым образом сообщения будет 160-битная строка из пяти слов $SHA_n = H_0||H_1||H_2||H_3||H_4$.

4.2.3.3. ГОСТ Р34.11-94

Алгоритм вычисления хэш-функции ГОСТ Р34.11-94 сопоставляет сообщению произвольной длины 256-битный сжатый образ, который используется, в частности, для генерации и проверки цифровой подписи в стандарте ГОСТ Р34.10-94.

Пусть исходное сообщение m задано t-битной строкой $m_1 m_2 \ldots m_t$.

Строка сообщения m дополняется необходимым количеством нулей до длины, кратной 256.

Пусть $M = M_1 M_2 \ldots M_n$ — дополненное сообщение, которое состоит из n блоков, каждый из которых имеет длину 256 бит.

Параметрами хэш-функции ГОСТ Р34.11-94 являются 256-битный стартовый вектор хэширования $GOST_0 = gost_0||gost_1||gost_2||gost_3||$, на выбор которого ограничения не накладываются, и блок подстановки, используемый в алгоритме шифрования ГОСТ 28147-89.

Вычисление хэш-функции h:

Вход: $M = M_1 M_2 \dots M_n$ (n блоков по 256 бит)
Алгоритм: Для всех $i = 1, \dots, n$ $GOST_i \leftarrow g(GOST_{i-1}, M_i)$
Выход: $h(M) = GOST_n$

При вычислении шаговой функции хэширования используется накопитель H, содержащий четыре 64-битных слова $H_0, \dots, H_3$. Исходным заполнителем накопителя H являются слова стартового вектора хэширования $GOST_0$.

Алгоритм вычисления шаговой функции хэширования g для 256-битного блока M_i включает в себя следующие этапы:

1) генерацию четырех 256-битных ключей $K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}$;
2) шифрование заполнения накопителя H на этих ключах с помощью алгоритма ГОСТ 28147-89;
3) перемешивание результата шифрования.

На первом этапе блок M_i рассматривается как вектор m над полем F_2. С помощью четырех различных невырожденных аффинных над F_2 преобразований из этого вектора вырабатываются четыре 256-битных ключа $K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}$:

$$K_j^{(i)} = A_j m + c_j, \quad j = 1, \dots, 4$$

где A_j — блочная матрица; c_j — вектор сдвига.

На втором этапе каждое из четырех 64-битных слов $H_0, \dots, H_3$ накопителя H зашифровывается в режиме простой замены на соответствующем ключе $K_j^{(i)}$. Результатом шифрования является вектор $s = (s_1, s_2, s_3, s_4)$, где

$$s_j = E_{K_j^{(i)}}(H_j), \quad 1 \leqslant j \leqslant 4.$$

На третьем этапе выполняется перемешивание, представляющее собой композицию невырожденных линейных над F_2 преобразований, применяемую к векторам h, m, s, где h — представленное в виде вектора содержимое накопителя H. Результат перемешивания заносится в накопитель H и является текущим значением $GOST_i$ шаговой функции хэширования g для 256-битного блока M_i.

После обработки блок M_n итоговым сжатым образом сообщения будет 256-битная строка из четырех слов $GOST_n = H_1 || H_2 || H_3 || H_4$.

4.3. Электронные цифровые подписи

4.3.1. Обеспечение подлинности электронных документов

Цифровая подпись зависит от содержания подписываемого документа и некоего секретного элемента (ключа), которым обладает только лицо, участвующее в защищенном обмене.

Цифровая подпись, во-первых, должна подтверждать, что подписывающее лицо не случайно подписало электронный документ.

Во-вторых, цифровая подпись должна подтверждать, что только подписывающее лицо, и только оно, подписало электронный документ.

В-третьих, цифровая подпись должна зависеть от содержания подписываемого документа и времени его подписания.

В-четвертых, подписывающее лицо не должно иметь возможности в последствии отказаться от факта подписи документа.

4.3.1.1. Подпись документов при помощи симметричных криптосистем

Первые варианты цифровой подписи были реализованы при помощи симметричных криптосистем, в которых абоненты, участвующие в обмене сообщениями, используют один и тоже секретный ключ для простановки и проверки подписи под документом. В качестве алгоритма криптографического преобразования может использоваться любая из симметричных криптосистем, обладающая специальными режимами функционирования (например, DES, ГОСТ 28147-89 и т.п.).

Однако, данная схема применима только в тех сетях, в которых можно дать стопроцентную гарантию надежности каждого из абонентов, т.к. в противном случае существует потенциальная возможность мошенничества со стороны одного из абонентов, владеющих секретным ключом.

Для устранения указанного недостатка была предложена схема с доверенным арбитром.

В случае возникновения спорных ситуаций, например, связанных с отказом отправителя от факта подписания документа, используются услуги третьей, независимой стороны, так называемого арбитра, который расследует каждую такую ситуацию и принимает решение в пользу одной из сторон, участвующих в обмене данными.

Можно заметить, что самым критичным звеном этой схемы является именно арбитр. Во-первых, он должен быть действительно независимой ни от кого стороной. А во-вторых, арбитр должен быть абсолютно безошибочным. Ошибка в рассмотрении даже одной из нескольких тысяч спорных ситуаций подорвет доверие не только к арбитру, но и ко всем предыдущим подписанным документам, достоверность которых удостоверялась арбитром.

Именно поэтому, несмотря на теоретическую возможность применения симметричных криптосистем, на практике для выработки подписи они не используются, поскольку требуют дорогостоящих и громоздких мероприятий по поддержанию достаточного уровня достоверности передаваемых данных.

4.3.1.2. Подпись документов при помощи криптосистем с открытыми ключами

Поиски методов, устраняющих указанные выше недостатки, велись по нескольким направлениям. Наиболее впечатляющих результатов добились криптографы-математики Уитфилд Диффи (W. Diffie) и Мартин Хеллман (M. Hellman), а также Ральф Меркль (Ralph Merkle), которые в конце

70-х годов опубликовали результаты своих исследований. В своих работах авторы показали, что существует возможность построения криптосистем, не требующих передачи секретного ключа между абонентами, участвующими в обмене защищаемой информацией. В таких криптосистемах нет необходимости и в арбитрах.

Суть разработанного подхода заключается в том, что в обмене защищаемыми документами каждый абонент использует пару взаимосвязанных ключей — открытый и секретный. Отправитель подписываемого документа передает получателю открытый ключ. Он может это сделать любым несекретным способом или поместить ключ в общедоступный справочник. При помощи открытого ключа получатель проверяет подлинность получаемой информации. Секретный ключ, при помощи которого подписывалась информация, хранится в тайне от всех.

Можно заметить, что в данной схеме абоненты используют различные ключи, что не позволяет мошенничать ни одной из сторон.

Собственно ЭЦП документа — это его хэш-сумма, зашифрованная секретным ключом. Проверка ЭЦП документа сводится к вычислению хэш-суммы документа, расшифрованию хэш-суммы, содержащейся в подписи, и сравнению двух величин. Если значения вычисленной и сохраненной в подписи хэш-сумм совпали, то считается, что подпись под документом верна.

В настоящий момент широкого известны цифровые подписи, построенные по алгоритмам RSA, Эль-Гамаля, Шнорра, Рабина и математического аппарата эллиптических кривых.

Схема подписи RSA. Для создания подписи сообщения M претендент:
1) вычисляет сжатый образ $r = h(m)$ сообщения M с помощью хэш-функции (например, MD4 или MD5);
2) зашифровывает полученный сжатый образ $h(M)$ на своем секретном ключе d, т.е. вычисляет экспоненту $s \equiv r^d \mod n$, которая и является подписью.

Для проверки подписи верификатор:
1) расшифровывает подпись s на окрытом ключе e претендента, т.е. вычисляет $r' \equiv s^e \mod n$ и таким образом, восстанавливает предполагаемый сжатый образ r' сообщения M;
2) вычисляет сжатый образ $r = h(M)$ сообщения M с помощью той же самой хэш-функции, которую использовал претендент;
3) сравнивает полученные значения r и r'. Если они совпадают, то подпись правильная, претендент действительно является тем, за кого себя выдает, и сообщение не было изменено при передаче.

Схема подписи Эль-Гамаля. Секретным ключом создания подписи служит случайное целое число x, $0 < x < q$, открытым ключом проверки подписи являются простое число p, образующая g подгруппы, экспонента $y = g^x \pmod p$.

Для создания подписи сообщения M, $0 < M < q$, претендент:
1) генерирует случайное число k, $0 < k < q$;

2) вычисляет $r \equiv g^k (\mod p)$;
3) находит число s решением уравнения $M \equiv xr + ks (\mod q)$:

$$s \equiv (M - xr)k^{-1} (\mod q).$$

Так как числа k и q взаимно простые, то k обратимо по модулю q, т.е. число s всегда существует и оно единственное.

Подписью для сообщения M является пара (r, s).

Для проверки подписи верификатор проверяет выполнение сравнения $s \equiv (M - xr)k^{-1} (\mod q)$ для экспонент $g^M \equiv y^r r^s (\mod p)$.

4.3.1.3. ГОСТ Р34.10-94

ГОСТ Р34.10-94 основан на задаче дискретного логарифмирования в подгруппе простого порядка q простого поля характеристики p. Длина числа q — 256 бит и используется хэш-функция $H(x)$ ГОСТ Р34.11-94.

ГОСТ Р34.10-94 использует следующие параметры:

- p — большое простое число длиной от 509 до 512 бит или от 1020 до 1024 бит;
- q — простой сомножитель числа $(p - 1)$ длиной $254 \ldots 256$ бит;
- a — любое число, меньшее $(p - 1)$, причем такое, что $a^q \mod p = 1$;
- x — некоторое число, меньшее q;
- $y = a^x \mod p$.

При этом p, q, a — открытые параметры; x — секретный ключ; y — открытый ключ.

Для подписания некоторого сообщения m выполняются следующие шаги:

1) отправитель генерирует случайное число k, причем $k < q$;
2) отправитель вычисляет значения:

$$r = (a^k \mod p) \mod q$$
$$s = (xr + k(H(m))) \mod q.$$

Если $H(m) \mod q = 0$, то значение $H(m) \mod q$ принимают равным единице. Если $r = 0$, то выбирают другое значение k и начинают снова.

Цифровая подпись представляет собой два числа: $r \mod 2^{256}$ и $s \mod 2^{256}$, которые должны быть отправлены другому получателю.

3) получатель проверяет полученную подпись, вычисляя:

$$v = (H(m))^{q-2} \mod q$$
$$z_1 = (sv) \mod q$$
$$z_2 = ((q - r)v) \mod q$$
$$u = ((a^{z_1} y^{z_2}) \mod p) \mod q$$

Если $u = r$, то подпись считается верной.

4.3.1.4. Схема подписи с восстановлением сообщения

Схема подписи с восстановлением сообщения может использоваться для подписи коротких сообщений (например, ключей). Подписывание выполняется несимметричным алгоритмом без хэш-функций. Ключ шифрования является секретным ключом создания подписи. Ключ расшифрования является открытым ключом проверки подписи (например, в системе RSA). Исходное сообщение дополняется избыточными битами, включающими в себя признак начала сообщения и число добавленных битов, и зашифровывается на секретном ключе претендента. Восстановление исходного текста по подписи происходит в процессе проверки подписи. Верификатор расшифровывает полученное сообщение и проверяет правильность избыточных битов, выполняя действия претендента в обратном порядке.

4.3.2. Стандарты

В России в 1994 году были приняты стандарты ГОСТ Р 34.10-94 «Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма» и ГОСТ Р 34.11-94 «Криптографическая защита информации. Функция хэширования». Данные стандарты, во-первых, должны гарантировать криптостойкость, т.е. надежность реализованных по ним алгоритмов. Во-вторых, применение стандартов должно обеспечить совместимость продуктов различных производителей. Однако есть и проблемы при использовании принятых стандартов.

Стандарты ГОСТ Р 34.10-94 и ГОСТ Р 34.11-94 описывают лишь процедуры выработки и проверки ЭЦП и хэш-функции. За пределами их рассмотрения остается такой важный вопрос, как распространение и генерация ключей, защита от несанкционированного доступа к ключевой информации и т.д. Поэтому зачастую продукты, реализующие один и тот же стандарт, несовместимы между собой. На российском рынке средств защиты информации существует несколько известных систем, в которых реализованы указанные стандарты (Верба-О, Криптон и т.д.), но между собой эти системы не совместимы.

Как уже отмечалось, использование стандарта должно гарантировать, что документы, подписанные при помощи ГОСТ Р 34.10-94, теоретически не могут быть подделаны за приемлемое для злоумышленника время. Однако на практике дело не всегда обстоит таким образом. Связано это с тем, что стандарт описывает алгоритм математическим языком, в то время как пользователи сталкиваются уже с его реализацией. А при реализации могут быть допущены различные ошибки, которые сводят на нет все достоинства алгоритма. Кроме того, эффективное применение систем ЭЦП зависит от их правильной эксплуатации. Например, хранение секретных ключей для генерации цифровой подписи на доступном всем жестком диске позволяет злоумышленнику получить к ним доступ и в дальнейшем подделывать документы, подписанные на этих ключах.

Поэтому далеко не всегда указанные системы обеспечивают необходимый уровень защищенности.

4.3.3. Требования пользователей

Если обратиться к документации на различные системы, реализующие ЭЦП, то можно заметить, что производители, особенно в России, уделяют максимум внимания математическим аспектам реализованных алгоритмов. Десятки страниц посвящены тому, какова криптостойкость алгоритма и сколько лет потребуется злоумышленнику на подделку подписанного документа. Но как ни парадоксально это прозвучит, на практике пользователей очень мало волнуют эти вопросы. Если система реализует некоторый стандарт, то для конечного пользователя этого факта достаточно. Тем более что проверить правильность приводимых в документации выкладок сможет только квалифицированный математик-криптограф, которых в России очень мало. В первую очередь пользователи интересуются потребительскими свойствами предлагаемых систем, возможностям их встраивания в уже существующую технологию обработки информации и т.п. Рассмотрим более подробно эти и другие вопросы, задаваемые пользователями при приобретении систем, реализующих электронную цифровую подпись.

Скорость — это один из основных параметров, на которые следует обращать внимание при выборе системы цифровой подписи. Особенно в системах связи, в которых осуществляется очень интенсивный обмен данными и передаваемая информация должна защищаться от подделки. Данный параметр складывается из двух составляющих: скорости генерации подписи и скорости ее проверки, и существенно зависит от скорости выработки хэш-функции, а также типа ЭВМ, на которой осуществляется генерация или проверка ЭЦП.

Немаловажным параметром является длина подписи. Например, в системах диспетчерского управления, в которых постоянно передается большое число переменных малой длины, использование российского стандарта для подписи всех данных неэффективно.

Поскольку при приобретении системы цифровой подписи, как правило, у заказчика уже сложилась информационная инфраструктура, то очень часто на первое место выходит вопрос об интеграции приобретаемой системы в принятую технологию обработки информации. Например, если в качестве средства отправки электронной почты используется Microsoft Outlook, то необходимо, чтобы система ЭЦП могла быть встроена в эту почтовую программу. Такую возможность предлагают многие зарубежные и некоторые российские системы ЭЦП, например, PGP (Pretty Good Privacy), которая также может быть встроена в почтовую программу The Bat!, широко распространенную в России. Если система ЭЦП не поддерживает используемое у заказчика программное обеспечение (например, потому что оно разработано самим заказчиком), то поставщик должен поставлять интерфейс (API) для встраивания возможностей работы с цифровой подписью в систему заказчика. Такую возможность предлагают многие российские производители (например, МО ПНИЭИ, ЛАН КРИПТО, НИП «Информзащита»). Причем желательно, чтобы данный интерфейс существовал для различных операционных систем и платформ (Windows NT, Windows 9x, MS DOS, HP UX, AIX и т.д.).

Необходимо обратить внимание на предлагаемые механизмы или меры

защиты от несанкционированного доступа к системе электронной цифровой подписи. Должны быть предусмотрены действия, выполняемые в случае компрометации ключей одного из пользователей (например, занесение их в «черные списки» и рассылка всем пользователям системы). Кроме того, должна контролироваться целостность как системы ЭЦП в целом, так и ее компонентов (например, журналов регистрации действий). В документации на некоторые российские системы ЭЦП есть рекомендации по применению систем защиты информации от несанкционированного доступа. Данные системы, в частности, позволяют ограничить круг лиц, имеющих право запуска системы цифровой подписи. Одной из таких систем является сертифицированная в Гостехкомиссии России система Secret Net, разработанная Научно-инженерным предприятием «Информзащита».

Немаловажным аспектом является юридическая поддержка предлагаемого решения. Если предложение системы ЭЦП исходит от компании-разработчика программного обеспечения, то она должна предоставить проект договора об обмене электронными документами с использованием ЭЦП. Если же компания предлагает обслуживание с применением систем ЭЦП, то следует внимательно ознакомиться с текстом заключаемого договора. В таком договоре или его проекте должно быть предусмотрено решение следующих вопросов:

— наличие процедуры урегулирования конфликтных ситуаций;
— описание состава комиссии, расследующей возникающие конфликты;
— ответственность сторон (в т.ч. и фирмы-разработчика).

Для реализации процедуры расследования конфликтных ситуаций система, в которой реализована цифровая подпись, должна поддерживать возможность хранения всех используемых ключей.

Окончательный выбор системы ЭЦП может быть определен наличием или отсутствием следующих дополнительных возможностей:

— постановка нескольких подписей под одним документом и их выборочная проверка;
— хранение цифровой подписи не только в подписываемом документе, но и в отдельном файле;
— использование командной строки для работы с системой ЭЦП;
— подпись и проверка группы файлов;
— постановка и проверка подписи под заданными фрагментами (полями) документа;
— выработка и проверка групповой подписи;
— совместное использование функций шифрования и цифровой подписи;
— постановка подписи и ее проверка для участка оперативной памяти;
— архивация использованных ключей и т.д.

4.3.4. Атаки на цифровую подпись

Пользователю системы электронной цифровой подписи необходимо знать, каким образом злоумышленник может осуществить атаку на ЭЦП. В документации на многие системы цифровой подписи очень часто упоминается число операций, которые надо осуществить для перебора всех

возможных ключей. Однако это только один из возможных вариантов реализации атак.

Квалифицированный злоумышленник далеко не всегда использует такой «грубый» перебор (brute force search) всех возможных ключей. Рассмотрим подробнее некоторые из типов атак.

4.3.4.1. Атаки на алгоритмы

Многие разработчики, несмотря на наличие в России государственных стандартов цифровой подписи и хэш-функции, пытаются разработать свои собственные алгоритмы. Однако, из-за низкой квалификации авторов данные алгоритмы не обладают свойствами, присущими качественным алгоритмам, разработанными математиками-криптографами. К ошибкам, которые существуют в алгоритмах криптографов-самоучек, можно отнести:

— периодическое повторение одних и тех же значений алгоритмами генерации случайных чисел, которые получили широкое распространение в криптографии;

— возможность генерации одинаковой хэш-функции для двух различных документов. Такое событие называется коллизией и надежный алгоритм хэш-функции должен противостоять такого рода неприятностям;

— многие разработчики пытаются сохранить разработанный алгоритм в секрете, тем самым надеясь гарантировать его надежность. Однако согласно приведенному выше правилу Киркхоффа, надежность цифровой подписи должна определяться только секретностью ключа, используемого для подписи сообщения.

Кроме того, иногда в российских журналах или телеконференциях в сети Internet и FIDOnet можно прочитать сообщения об оптимизации существующих стандартов (например, ГОСТ 28147-89 или DES), которые якобы не снижают надежности самих стандартов, а скорость работы при этом возрастает на один-два порядка. К таким сообщениям надо относиться с определенной степенью скептицизма. Выгоды от применения «оптимизированных» алгоритмов сомнительны, а вот вероятность фальсификации документов, подписанных с их помощью, увеличивается.

4.3.4.2. Атаки на криптосистему

Под криптосистемой понимается не только используемый алгоритм выработки и проверки цифровой подписи, но также механизм генерации и распределения ключей и ряд других важных элементов, влияющих на надежность криптосистемы. Ее надежность складывается из надежности отдельных элементов, составляющих эту криптосистему. Поэтому в некоторых случаях нет необходимости атаковать алгоритм. Достаточно попытаться атаковать один из компонентов криптосистемы. Например, механизм генерации ключей. Если датчик случайных чисел, реализованный в криптосистеме для генерации ключей, недостаточно надежен, то говорить об эффективности такой системы не приходится. Даже при наличии хорошего алгоритма ЭЦП.

Для алгоритмов, основанных на открытых ключах, например, RSA или Эль-Гамаля, существует ряд математических проблем, которые не всегда учитываются при построении криптосистемы. К таким моментам можно отнести выбор начальных значений, на основе которых создаются ключи. Есть определенные числа, которые позволяют очень быстро вычислить секретный ключ ЭЦП. В то же время правильный выбор начальных значений позволяет гарантировать невозможность лобовой атаки в течение нескольких сотен лет при современном развитии вычислительной техники.

4.3.4.3. Атаки на реализацию

Атаки именно этого типа наиболее часто используются злоумышленниками. Связано это с тем, что для их реализации нет необходимости обладать обширными познаниями в области математики. Достаточно быть квалифицированным программистом. Примеры неправильной реализации, приводящей к атаке на нее:

— секретный ключ ЭЦП хранится на жестком диске;

— после завершения работы системы ЭЦП, ключ, хранящийся в оперативной памяти, не затирается;

— обеспечивается безопасность сеансовых ключей и недостаточное внимание уделяется защите главных ключей;

— открыт доступ к черным спискам скомпрометированных ключей.

— отсутствует контроль целостности программы генерации или проверки ЭЦП, что позволяет злоумышленнику подделать подпись или результаты ее проверки.

Вопросы, касающиеся атак на аппаратную реализацию, в данной публикации рассматриваться не будут в связи с тем, что в России практически нет средств, реализующих цифровую подпись на аппаратном уровне. Можно добавить, что существуют варианты атак на аппаратные элементы хранения ключей ЭЦП (таблетки Touch Memory, смарт-карты и т.п.). Такого рода атаки получили очень широкое распространение в последнее время.

4.3.4.4. Атаки на пользователей

Не стоит забывать, что конечный пользователь также является элементом криптосистемы и также подвержен атакам наравне со всеми остальными элементами. Пользователь может передать дискету с секретным ключом ЭЦП своему коллеге для подписи документов в свое отсутствие. Пользователь может потерять такую дискету или другой носитель секретных ключей и не сообщать об утере до того момента, пока эта дискета не понадобится вновь.

Во многих системах пользователь может сам создавать себе ключи для выработки подписи. Генерация ключа основывается на паролях, выбираемых самим пользователем. Как известно фантазия в выборе таких паролей у пользователя очень слаба. Поэтому выбираются легко запоминаемые слова или фразы, которые легко угадываются злоумышленниками.

4.3.5. PGP

Pretty Good Privacy (PGP) используется для защиты сообщений и файлов с помощью их шифрования и цифровой подписи. Удобно использовать PGP вместе с одним из популярных пакетов электронной почты, обеспечивающим поддержку встраиваемых модулей. PGP позволяет шифровать, подписывать, расшифровывать и проверять сообщения во время отправки и чтения электронной почты. В PGP применяются стойкие криптографические алгоритмы CAST, тройной DES и IDEA. Для выработки сеансового ключа используются алгоритмы RSA и Диффи-Хеллмана, для подписи — RSA и DSS.

PGP обеспечивает интегрированную поддержку распространения и поиска открытых ключей на серверах ключей.

Перед использованием PGP необходимо сгенерировать открытый и секретный ключи. Открытый ключ может быть передан абоненту как сообщение электронной почты, как файл, или можно поместить его на сервер открытых ключей. Получив копию чьего-либо открытого ключа, пользователь может добавить его на свою связку открытых ключей. Затем пользователь должен убедится, что ключ не был подделан. Сделать это можно, сравнивая уникальный отпечаток своей копии ключа с отпечатком оригинальной копии. Отпечаток — это строка из цифр и букв, уникальным образом идентифицирующая владельца ключа. После того, как пользователь убедился в действительности ключа, он подписывает его, чтобы дать PGP знать, что он считает безопасным его использование. Кроме того, пользователь может указать степень доверия, которую он испытывает к владельцу ключа, в смысле его способности ручаться за подлинность ключей третьих лиц. Таким образом, формируется сеть поручительства абонентов сети за достоверность распространяемых ключей, так называемая «сеть доверия».

4.4. Инфраструктура открытых ключей

4.4.1. Концепция инфраструктуры открытых ключей

Инфраструктура открытых ключей — Public Key Infrastructure (PKI) — это набор служб, которые призваны обеспечить работу криптографических методов с открытыми ключами.

Технология PKI заключается в использовании двух математически-связанных цифровых ключей, имеющих следующие свойства:

— один ключ может использоваться для шифрования сообщения, которое может быть расшифровано только с помощью второго ключа;

— даже если известен один ключ, с помощью вычислений абсолютно невозможно определить второй. Один из ключей открыт для всех, второй же имеет частный характер и хранится в защищенном месте. Эти ключи могут использоваться для аутентификации, шифрования или цифровой подписи электронных данных.

PKI служит не только для создания цифровых сертификатов, но и для хранения огромного количества сертификатов и ключей, обеспечения

резервирования и восстановления ключей, взаимной сертификации, ведения списков аннулированных сертификатов и автоматического обновления ключей и сертификатов после истечения срока их действия.

4.4.2. Хранение ключей

Согласно правилу Кирхгоффа стойкость шифра и, как следствие, стойкость ЭЦП, должна определяться только секретностью ключа, используемого для шифрования или подписи сообщения. Как следствие, секретные ключи никогда не должны храниться в явном виде на носителях, которые могут быть скопированы. Во многих существующих разработках этим условием пренебрегают, оставляя защиту секретных ключей на совести пользователя системы ЭЦП. В некоторых случаях разработчики предлагают варианты хранения на носителях, которые, по их словам, трудно копируются. Например, «таблетки» Touch Memory, смарт-карты или бесконтактные карты Proximity. Однако в последнее время за рубежом участились случаи «удачных» атак на такие носители. Эти случаи преимущественно зафиксированы за рубежом, но российские «умельцы» ни в чем не уступают своим западным коллегам, и можно предсказать, что скоро случаи попыток «взлома» аппаратных носителей будут зафиксированы и в России.

Для повышения надежности таких схем хранения рекомендуется секретные ключи шифровать на других ключах, которые, в свою очередь, могут быть тоже зашифрованы. Самый последний ключ в этой иерархии называется главным или мастер-ключом и не должен шифроваться. Однако к нему предъявляются очень жесткие требования к хранению в защищенной части компьютера или аппаратуры, реализующей функции цифровой подписи.

4.4.3. Распределение ключей

Очень важный вопрос при выборе системы ЭЦП — это распределение ключей между абонентами, участвующими в обмене защищаемыми документами. Такое распределение может осуществляться двумя способами.

— Путем создания центра генерации и распределения ключей.
Недостаток такого подхода очевиден. Центр обладает полной информацией о том, кто и какой ключ использует. Компрометация центра распределения приводит к компрометации всей передаваемой между абонентами этого центра информации. Кроме того, знание секретных ключей абонентов позволяет нечистым на руку сотрудникам центра фальсифицировать определенные документы, передаваемые в системе обмена информацией.

— Путем прямого обмена ключами между абонентами, которые хотят обмениваться подписанными сообщениями.
В этом случае основная задача — подтверждение подлинности каждого абонента, участвующего в обмене.

Подтверждение подлинности абонентов в последнем случае может осуществляться следующим образом.

— Непосредственно между абонентами.
Данный метод применяется в том случае, если абонентов всего двое. Для обмена ключами в данном случае может быть использован алгоритм распределения ключей, например, разработанный в 1976 году криптографами Диффи и Хеллманом. Существуют и другие варианты обмена ключами. Например, при помощи симметричных криптосистем или фельдъегерской службы. Однако, в распределенных сетях, насчитывающих не один десяток абонентов, такие варианты не применимы из-за возникающих сложностей.

— С использованием посредника (арбитра).
Данный метод может применяться в корпоративных сетях, в которых существует так называемый центр верификации или сертификации ключей. Данный центр удостоверяет ключи, используемые для проверки подписи. Подтверждение подлинности ключей может реализовываться или путем формирования справочника открытых ключей, или путем выдачи сертификатов, которые передаются вместе с сообщением, требующим проверки. Данный сертификат представляет собой ключ для проверки подписи и некоторую аутентифицирующую информацию, скрепленные подписью Центра сертификации. В данном случае достаточно проверить подпись Центра, указанную в сертификате, чтобы удостовериться в подлинности ключа абонента.

— С использованием двух и более посредников.
Этот метод, являющийся комбинацией двух предыдущих, может применяться в том случае, когда необходимо обеспечить обмен подписанными сообщениями между несколькими корпоративными сетями, в каждой из которых существует свой центр сертификации.

4.4.4. Подходы к реализации PKI

В настоящее время можно выделить несколько подходов к реализации инфраструктуры открытых ключей [26]:
- защищённая DNS;
- PKI стандарта OpenPGP;
- PKIX — PKI, основанная на сертификатах формата X.509 (PKI for X.509 certificates).

4.4.4.1. Защищенная DNS

Предложено и реализовано расширение DNS, выполняющее аутентификацию данных путём применения цифровых подписей. Расширение предусматривает хранение открытых ключей в DNS.

Некоторые DNS позволяют включать помимо записей, содержащих информацию об IP-адресе и о сервере имен, записи *KEY* и *SIG*, которые обеспечивают распространение ключа и аутентификацию исходных данных.

Запись *KEY* позволяет связывать открытые ключи с именами DNS. Из-за относительно простого формата имён DNS запись *KEY* содержит

флаговые биты, которые указывают вид имени DNS и ограничения использования ключа. Предполагается, что некоторые ключи защищённой DNS будут использоваться совместно с другими протоколами интернет. Байт протокола указывает, что данный ключ может использоваться в других протоколах. Байт алгоритма определяет алгоритм шифрования, для которого предназначен данный ключ.

Запись *SIG* используется для аутентификации других записей, связывая их с доменным именем. Запись *SIG* должна содержать следующие поля: тип записей, защищаемых данной записью, алгоритм, счётчик меток, время жизни, срок прекращения действия подписи, время подписания, имя подписавшего, подпись.

К сожалению, организационная часть, связанная с выделением сертификатов доменам, не выполнена и вряд ли в ближайшей перспективе сертификаты будут выданы. Поэтому защищённая DNS имеет исключительно теоретический интерес.

4.4.4.2. PGP

PGP используется для защиты сообщений и файлов с помощью их шифрования и цифровой подписи. PGP позволяет шифровать, подписывать, расшифровывать и проверять сообщения. В PGP используются следующие алгоритмы: CAST, 3DES, IDEA; для выработки сеансового ключа используются алгоритмы RSA и Диффи–Хэллмана; для подписи используются алгоритмы RSA и DSS [27].

PGP обеспечивает интегрированную поддержку распространения и поиска открытых ключей на серверах ключей.

Перед использованием PGP генерятся открытый и закрытый ключи. Открытый ключ передаётся непосредственно абоненту либо помещается на сервер открытых ключей. Получив копию чьего-либо открытого ключа, пользователь может добавить его на свою связку открытых ключей. Пользователь должен сам проверить валидность полученного ключа, после чего он подписывает его. Кроме того, пользователь должен указать степень доверия к владельцу данного ключа в смысле его способности ручаться за подлинность ключей третьих лиц. Таким образом, формируется сеть поручительства абонентов сети за достоверность распространяемых ключей, так называемая *сеть доверия*.

4.4.4.3. PKIX

Данный вариант PKI [28, 29] стал фактически индустриальным стандартом.

4.4.5. Компоненты инфраструктуры открытых ключей и их функции

Эффективная PKI должна включать следующие элементы:
— доверенный центр;
— архив сертификатов;

— систему аннулирования сертификатов;
— систему создания резервных копий и восстановления ключей;
— систему поддержки невозможности отказа от цифровых подписей;
— систему автоматической корректировки пар ключей и сертификатов;
— систему управления «историей» ключей;
— систему поддержки взаимной сертификации;
— клиентское программное обеспечение, взаимодействующее со всеми этими подсистемами безопасным, согласованным и надежным способом.

Сертификат (Certificate) и соответствующий ему секретный ключ позволяют идентифицировать их владельца. Универсальное применение сертификатов обеспечивает стандарт Международного Телекоммуникационного Союза X.509, который является базовым и поддерживается целым рядом протоколов безопасности.

Стандарт X.509 задает формат цифрового сертификата. Основными атрибутами сертификата являются имя и идентификатор субъекта, информация об открытом ключе субъекта, имя, идентификатор и цифровая подпись уполномоченного по выдаче сертификатов, серийный номер, версия и срок действия сертификата, информация об алгоритме подписи и др. Важно, что цифровой сертификат включает в себя цифровую подпись на основе секретного ключа доверенного центра.

Центр сертификации (ЦС, Certificate Authority—CA) *или Доверенный центр* [1] — объект, который авторизован создавать, подписывать и публиковать сертификаты. CA имеет полномочия идентифицировать пользователей. Основными операциями, которые выполняет CA, являются:
— издание сертификата;
— обновление сертификата;
— аннулирование сертификата.

Действия CA ограничены политикой сертификации, которая диктует ему, какую информацию он должен помещать в сертификат. CA публикует свою политику сертификации таким способом, чтобы пользователи могли проверить соответствие сертификатов этой политике.

Список аннулированных сертификатов (Certificate Revocation List—CRL) — список сертификатов, признанных недействительными в период их действия в случае компрометации секретного ключа или изменения атрибутов сертификата с момента его выпуска.

[1] В настоящее время не существует общепризнанного русского аналога термина, который берет начало в области шифрования с открытыми ключами, — Certificate Authority. Это понятие получило множество совершенно разных названий: служба сертификации, уполномоченный по выпуску сертификатов, распорядитель сертификатов, орган выдачи сертификатов, доверенный центр, центр сертификации, сертификатор и т. д. В случае отсутствия общепринятого русского аналога здесь будут использоваться англоязычные аббревиатуры.

Хранилище сертификатов — специальный объект PKI, где хранятся выпущенные сертификаты и списки отозванных сертификатов. Оно не является обязательным компонентом PKI, но оно значительно упрощает доступ к ресурсам и управление системой. К хранилищу предъявляют следующие требования:

— простота доступа;
— доступ должен быть стандартным;
— обновление информации;
— встроенная защищенность;
— простое управление;
— совместимость с другими хранилищами (необязательное требование).

Хранилище упрощает систему распространения CRL.

Фактически действующим стандартом доступа к хранилищам является LDAP (Lightweight Directory Access Protocol, упрощенный протокол доступа к каталогу). Он наиболее адекватен в качестве стандарта для сохранения и извлечения сертификатов после их генерации, поддерживается большинством серверных операционных систем и баз данных и достаточно открыт для того, чтобы его могли поддерживать практически любые инфраструктуры с открытыми ключами.

Центр регистрации (Registration Authority—RA) является дополнительным компонентом системы PKI, который авторизован CA аутентифицировать пользователей и проверять информацию, которая заносится в сертификат. В его функции может входить генерация и архивирование ключей, сообщение об аннулировании сертификатов и др. В некоторых системах CA выполняет функции RA. CA выдаёт сертификат RA (если он присутствует в системе), и RA выступает как объект, подчинённый CA. Но RA не может выпускать сертификаты и CRL.

Конечный пользователь (End Entity—EE) — пользователь сертификата PKI и/или владелец сертификата. То есть конечный пользователь — это объект, который использует некоторые сервисы и функции системы PKI. Конечный пользователь может быть владельцем сертификата (человек, организация или иная сущность) или объектом, запрашивающим сертификат или CRL.

4.4.6. Сервисы PKI

4.4.6.1. Сервисы управления сертификатами

Сервисы управления сертификатами — это сервисы, образующие ядро инфраструктуры с открытыми ключами. К ним относятся:

— Выпуск сертификата.

Сертификаты выпускаются для пользователей (физических и юридических лиц), для доверенных центров, находящихся на более низких уровнях иерархии доверия, а также для других доверенных центров в случае их взаимной сертификации.

— Аннулирование сертификата.

Если пользователь теряет свой секретный ключ, если ключ похищается или компрометируется, или есть вероятность наступления таких событий, действие сертификата должно быть прекращено. После получения подтверждения запроса пользователя об аннулировании сертификата CA уведомляет об аннулировании все заинтересованные стороны, используя CRL.

Аналогично аннулированию осуществляется приостановление действия сертификата. Оно заключается в однократной отмене сертификата на определенный период времени в течение срока его действия. После этого действие сертификата возобновляется автоматически или же сертификат аннулируется. Приостановление действия сертификата осуществляется в тех ситуациях, когда невозможно установить подлинность лица, обращающегося с запросом об аннулировании.

— Публикация сертификата.

Выпущенный однократно сертификат включается в каталог (в соответствии со спецификациями стандарта X.500 или иными требованиями), чтобы третьи стороны могли иметь к нему доступ. В одних случаях каталог контролируется доверенным центром, в других — третьей стороной. Доступ к каталогу может быть ограничен. Если необходимо соблюдение прав приватности абонентов, применяются профилактические меры для защиты от лиц, не имеющих полномочий доступа.

— Хранение сертификата в архиве.

Выпускаемые сертификаты и списки аннулированных сертификатов хранятся в архиве длительное время. Это делается потому, что заверенный цифровой подписью документ продолжает свое существование и после истечения срока действия сертификата. Следовательно, сертификаты с истекшим сроком действия должны быть по-прежнему доступны.

— Выработка политики CA.

Для реализации операций сертификации формируется политика операционной работы CA, работы с персоналом и оборудованием и политика выпуска сертификатов на основе критериев контроля за созданием сертификатов для пользователей и других доверенных центров.

4.4.6.2. Вспомогательные сервисы

В инфраструктуре с открытыми ключами могут поддерживаться также различные дополнительные сервисы.

— Регистрация.

Регистрационные сервисы обеспечивают регистрацию и контроль индивидуальной информации, а также аутентификацию, необходимую для выпуска или аннулирования сертификатов (от имени доверенного центра). Фактический выпуск сертификатов осуществляет CA.

— Хранение информации в архиве.

Сервисы хранения информации в архиве предназначены для долговременного хранения и управления цифровыми документами и другой информацией. Сервисы обеспечивают создание резервных копий и восстановление информации в случае уничтожения или старения среды хранения.

— Нотариальная аутентификация.
Нотариальная аутентификация включает аутентификацию отправителя сообщения, подтверждение целостности и юридической силы цифровых документов.

— Создание резервных копий и восстановление ключей.
СА должен иметь возможность восстановить зашифрованную информацию в случае потери пользователями их ключей шифрования. Это означает, что доверенному центру, к которому относится пользователь, необходима система создания резервных копий и восстановления этих ключей. Этот процесс известен как коммерческое создание резервных копий и восстановление ключей, и он отличается от принудительного депонирования ключей третьей стороной (обычно правоохранительными органами), которая получает доступ к ключам для расшифровки необходимой информации. Коммерческие сервисы восстановления ключей обеспечивают заблаговременное засекречивание копии ключа на случай утери ключа пользователем, его ухода с работы, забывания пароля, необходимого для доступа к ключу, и восстановление ключа в ответ на запрос пользователя или его работодателя. В одних случаях ключ является секретным ключом из алгоритма с открытыми ключами, в других — это распределяемый ключ.

— Каталог.
Сервисы каталога осуществляют всестороннее управление и обеспечение информацией, имеющей отношение к пользователю. К атрибутам пользователя относится не только сертификат, но и номер телефона, адрес электронной почты абонента и т. д.

— Поддержка невозможности отказа от цифровых подписей.
При бумажной технологии подписи лиц законно связывают их с документами, что не позволяет в дальнейшем отказаться от подписания документа. При электронных технологиях обычная подпись заменяется цифровой. Самое главное требование для невозможности отказа от цифровой подписи состоит в том, что ключ, используемый для выработки цифровых подписей — ключ подписи, должен создаваться и безопасно храниться все время исключительно под контролем пользователя. Когда пользователи забывают свои пароли или теряют свои ключи подписи, на резервирование или восстановление предыдущей пары ключей подписи не накладывается никаких технических ограничений (в отличие от аналогичной ситуации с парами ключей шифрования сообщений). В таких случаях допускается генерация и дальнейшее использование пользователями новых пар ключей подписи.
Параллельное функционирование систем резервного копирования и восстановления ключей и системы поддержки невозможности отказа от цифровых подписей вызывает определенные проблемы. При резервном копировании и восстановлении ключей должны создаваться копии секретных ключей пользователя. Для обеспечения невозможности отказа от цифровой подписи не должны создаваться резервные копии секретных ключей пользователя, используемых для выработки цифровой подписи. Для соблюдения этих требований в инфраструктуре с открытыми ключами должны поддерживаться две пары ключей для каждого пользователя. В любой момент времени пользователь должен иметь одну пару ключей для шиф-

рования и дешифрования, а другую пару — для выработки или проверки цифровой подписи.

— Корректировка ключей и управление историями ключей.

В ближайшем будущем в распоряжении пользователей будет находиться огромное количество пар ключей, которые должны будут поддерживаться как криптографические ключи, даже если никогда не будут использоваться. Ключи шифрования должны со временем обновляться и должна поддерживаться история всех ключей, использованных ранее.

Процесс корректировки пар ключей должен быть «прозрачен» для пользователя. Это означает, что пользователи не должны понимать, что осуществляется обновление ключей, и никогда не должны получать отказ сервиса из-за недействительности своих ключей. Для удовлетворения этого требования пары ключей пользователя должны автоматически обновляться до истечения срока их действия. При обновлении пары ключей подписи предыдущий ключ подписи безопасно уничтожается. Тем самым предотвращается получение несанкционированного доступа к ключу подписи и устраняется необходимость хранения предыдущих ключей подписи.

— Другие сервисы.

В ряде случаев необходимы и другие сервисы, например, сервисы генерации пар ключей и записи их на смарт-карты.

4.4.7. Форматы сертификатов

Наиболее распространен формат сертификата, установленный Международным Телекоммуникационным Союзом (ITU Rec. X.509 | ISO/IEC 9594-8). Сертификат содержит элементы данных, определенные в приведенной ниже табл. 4.2, сопровождаемые цифровой подписью.

Цифровая подпись для всех элементов вырабатывается при помощи ключа «authority key identifier». Элементы, включенные в версию v1, являются обязательными; элементы, включенные в последующие версии — необязательные. Имя субъекта сначала было обязательным элементом и должно было быть уникальным. Начиная с версии v3, оно стало необязательным. В формат уникального имени включается такая информация, как название страны, региона и данного имени, которые объединяются так, чтобы образовать уникальный идентификатор. Использование одинаковых идентификаторов запрещено.

Таблица 4.2

Формат сертификата X.509

Версия	Элемент	Описание
v1	version	Версия (0 означает v1, 2 означает v3)
	serialNumber	Серийный номер сертификата

Продолжение таблицы 4.2

Версия	Элемент	Описание
	signature.algorithmIdentifier algorithm parameters	Тип алгоритма подписи
	issuer	Уникальное название центра, выпускающего сертификат
	Validity	Период действия
	NotBefore	Дата и время начала действия
	notAfter	Дата и время конца действия
	subject	Уникальное имя субъекта
	SubjectPublicKeyInfo	Информация об открытом ключе субъекта
	Algorithm	Криптографический алгоритм
	subjectPubkicKey	Ключ (строка битов)
v2	issuerUniqueID	Уникальный идентификатор центра, выпускающего сертификат
	subjectUniqueID	Уникальный идентификатор субъекта
v3	AuthorityKeyIdentifier	Идентификатор ключа, используемого для подтверждения подписи CA
	keyIdentifier	Идентификатор ключа
	authorityCertIssuer	Общее название CA
	authorityCertSerialNumber	Серийный номер сертификата CA
	subjectKeyIdentifier	Идентификатор, используемый тогда, когда субъект имеет более одного ключа (например, во время возобновления сертификата)
	keyUsage	Применение ключа (строки битов) Цифровая подпись Невозможность отказа получателя/отправителя сообщения от факта его передачи/приема и содержания Шифрование ключа Шифрование информации Соглашение о ключе Подписание сертификата Подписание CRL

Продолжение таблицы 4.2

Версия	Элемент	Описание
	privateKeyUsagePeriod	Период действия секретного ключа CA для подписи. Стандартно он короче периода действия соответствующего открытого ключа
	CertificatePolicies	Политика CA
	policyIdentifier	Идентификатор политики (как для ISO/IEC 9834-1)
	PolicyQualifiers	Критерии сертификации
	PolicyMappings	Используется только для сертификата CA
	IssuerDomainPolicy SubjectDomainPolicy	Оговаривает, что политика эмитента и политика сертификации субъекта одинаковы
	SupportedAlgorithms AlgorithmIdentifier IntendedUsage intendedCertificatePolicies	Определяют атрибуты каталога. Используются, чтобы сделать атрибуты известными заранее в случаях, когда партнер по связи использует данные каталога
	SubjectAltName	Альтернативное имя субъекта. Свободный выбор имени.
	OtherName	Произвольное имя
	rfc822Name	Адрес электронной почты
	dNSName	Имя домена
	x400Address	Адрес X.400 отправителя/получателя
	directoryName	Имя каталога EDI-имя
	ediPartyName	Унифицированный указатель ресурсов WWW
	uniformResourceIdentifier	URL
	iPAddress registeredID	IP-адрес объекта
	issuerAltName	Альтернативное имя CA
	subjectDirectoryAttributes	Необязательные атрибуты субъекта, например, почтовый адрес, номер телефона и т. п.
	BasicConstraints	Отличает ключ CA от ключей конечных пользователей (используется только для сертификата CA)

Продолжение таблицы 4.2		
Версия	Элемент	Описание
	cA	Для ключа CA cA истинно
	pathLenConstraint	Ограничение длины пути
	NameConstraints	Используется только при сертификации CA. Определяет сертификацию домена по имени по отношению к CA более низкого уровня в пределах пути, устанавливаемого параметром BasicConstraints
	PermittedSubtrees	CA более низкого уровня и домен его поддерева
	Base	Имя CA более низкого уровня
	minimum	Верхний предел домена
	maximum	Нижний предел домена
	excludedSubtree	CA более низкого уровня, исключенный из домена
	PolicyConstraints PolicySet InhibitPolicyMapping	Ограничения политики (используется только для requireExplicitPolicy CA)
	cRLDistributionPoints	Пункты распределения CRL
	distributionPoint	Имя центра распределения Abbreviates cRLIssuer
	reasons	Вид списка, распределяемого данным пунктом
	keyCompromise	Скомпрометированный ключ конечного пользователя
	cACompromise	Скомпрометированный ключ CA
	affiliationChanged	Измененная информация в сертификате (не повреждение)
	superseded	Приостановленный ключ
	cessationOfOperation	Завершение использования
	certificateHold	Приостановление использования
	cRLIssuer	Имя центра-эмитента CRL

4.4.7.1. Описание полей

— Версия.

Данное поле описывает версию сертификата. При использовании дополнений версия должна быть установлена как X.509 version 3 (значение равно 2). Если дополнения не используются, версия должна быть 1 (значение должно быть не установлено).

— Идентификатор алгоритма ЭЦП.

Поле содержит идентификатор криптографического алгоритма, используемого CA для выработки ЭЦП сертификата.

— Серийный номер сертификата.

Серийный номер является целым числом, устанавливаемым CA для каждого сертификата. Значение должно быть уникальным для каждого сертификата, выпущенного данным CA. Имя Издателя и серийный номер сертификата совместно являются уникальным идентификатором сертификата.

— Имя Издателя сертификата.

Поле Издатель идентифицирует объект (субъект), который сформировал ЭЦП и издал сертификат. Значение в поле Издатель должно содержать ненулевое значение DN (*distinguished name*). Поле Издатель определено в рекомендациях *X*.501 как тип *Name*. Значение поля состоит из набора иерархических атрибутов (*AttributeType*), таких как код страны и соответствующего ему значения (*AttributeValue*, например, RU). Тип компонентов *AttributeValue*, входящих в имя, определяется типом атрибута *AttributeType* и в основном используется *DirectoryString*.

— Срок действия сертификата.

Данное поле определяет срок действия (в виде временного интервала), в течение которого CA управляет сертификатом (отслеживает состояние). Поле представляет последовательность двух дат: дата начала действия сертификата (*notBefore*) и дата окончания срока действия сертификата (*notAfter*). Оба этих значения могут быть закодированы либо как *UTCTime*, либо как *GeneralizedTime*.

— Имя Владельца сертификата.

Поле Владелец идентифицирует объект (субъект), являющийся обладателем секретного ключа, соответствующего открытому ключу в сертификате.

— Открытый ключ Владельца.

Данное поле используется для хранения открытого ключа и идентификации алгоритма, соответствующего открытому ключу. Поле *parameters* идентификатора может содержать дополнительные данные, присущие определенному алгоритму.

— Уникальный идентификатор Издателя и Владельца.

Данное поле может использоваться только в сертификатах версии 2 или 3. Поле было предусмотрено в версии 2 сертификатов X.509 для целей обеспечения использования одинакового имени Владельца или Издателя в разных сертификатах. С введением дополнений в версии 3 такая необходимость отпала.

— Подпись Центра Сертификации.

Битовая последовательность, содержащая значение ЭЦП, сформированной с использованием секретного ключа Центра Сертификации и алгоритмов хеширования и ЭЦП, указанных в поле Идентификатор алгоритма ЭЦП.

4.4.7.2. Сертификат X.509 версии 3

Данный раздел описывает версию 3 сертификата X.509. Версия 3 описала механизм дополнений, дополнительной информации, которая может быть помещена в сертификат. X.509 и рекомендации RFC 2459 описывают набор *стандартных* дополнений, но вместе с тем не ограничивают возможности использования любых других дополнений путем регистрации идентификатора (ISO или IANA).

Каждое дополнение состоит из трех полей:
- tipe;
- critical;
- value.

Дополнение представляет собой структуру, содержащую:
- идентификатор дополнения;
- признак «критичное / не критичное» дополнение;
- собственно значение дополнения, представленное в бинарном виде (OCTET STRING).

В свою очередь само дополнение может являться какой угодно сложной структурой (от простого текстового значения до сложной структуры), формат и интерпретация которого определяются идентификатором дополнения.

Основная цель критичных дополнений – предохранить сертификат, изданный CA, от возможности использования его в приложениях, которые не могут обработать такие дополнения. Таком образом, правила обработки дополнений, изложенные в рекомендациях, требуют от прикладного ПО отвергнуть сертификат, если дополнение отмечено критичным и прикладное ПО не может его интерпретировать. В свою очередь, требование отвергнуть дополнение прикладным ПО, отмеченное как критичное, при невозможности его интерпретации, требует от прикладного ПО детального разбора дополнений сертификатов и затрудняет процесс модификации как прикладного ПО, так и ПО, обеспечивающего реализацию PKI.

Дополнения X.509 версии 3 Дополнения, используемые в сертификатах версии 3, определены рекомендациями X.509 версии 3 ITU-T и рекомендациями IETF RFC 2459.

Базовый идентификатор дополнений, определенный рекомендациями X.509: id-ce OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 29}, где id-ce обозначает: Object identifier assignments for ISO certificate extensions.

Все дополнения, определенные указанными рекомендациями, можно разделить на две категории: ограничивающие и информационные дополнения. Первые ограничивают область применения ключа, определенного сертификатом, или самого сертификата. Вторые содержат дополнительную информацию, которая может быть использована в прикладном ПО пользователем сертификата.

К ограничивающим дополнениям относятся:
- базовые ограничения (*basicConstraints*);
- область применения ключа (*keyUsage*);
- расширенная область применения ключа (*extendedKeyUsage*);

— регламенты сертификата (модифицируемые ограничениями регламентов и соответствием регламентов) (*certificatesPolicies*);
— ограничения имен (*nameConstraints*).

К информационным дополнениям относятся:
— идентификаторы ключей (*subjectKeyIdentifier, authorityKeyIdentifier*);
— альтернативные имена (*subjectAltName, issuerAltName*);
— точка распространения СОС (*cRLDistributionPoint, issuingDistributionPoint*);
— способ доступа к информации СА (*authorityAccessInfo*).

Вместе с этим стандарт X.509 позволяет определить любые другие дополнения, необходимость которых определяется их использованием в конкретной системе (например: протокол SET).

Идентификатор ключа Издателя. Дополнение Идентификатор ключа Издателя (*authorityKeyIdentifier*) используется для идентификации открытого ключа, соответствующего секретному ключу ЭЦП, использованному СА при подписи издаваемого сертификата (или СОС). Данное дополнение может быть использовано в случае, когда Издатель сертификата (СА) имеет несколько различных секретных ключей ЭЦП (например при плановой их смене), а также для однозначного построения цепочек сертификатов.

Идентификатор ключа Владельца. Данное дополнение используется для идентификации различных сертификатов, содержащих открытый ключ. Для упрощения процедуры построения цепочки данное дополнение должно устанавливаться во всех сертификатах СА, которые включают дополнение *basicConstraints* с установленным значением *cA TRUE*. Во всех издаваемых СА сертификатах значение *keyIdentifier* в дополнении *authorityKeyIdentifier* должно быть идентично значению *subjectKeyIdentifier* сертификата СА.

Для сертификатов значение *subjectKeyIdentifier* должно вырабатываться из открытого ключа или с использованием метода генерации уникальных значений. Рекомендациями RFC 2459 предлагается два метода генерации идентификатора на основе значения открытого ключа:

1. Значение *keyIdentifier* определяется как 160 бит хэш-функции, вычисляемой по алгоритму SHA-1 из значения BIT STRING subjectPublicKey (исключая тэг, длину и неиспользованные биты).
2. Значение *keyIdentifier* определяется как 4-х битовое поле со значением 0100 и последующим за ним 60 битами наименьшей значимой части хэш-функции, вычисляемой по алгоритму SHA-1 из значения BIT STRING subjectPublicKey.

Для идентификации без использования открытого ключа можно также использовать монотонно возрастающую последовательность целых чисел.

Для сертификатов конечного пользователя данное дополнение используется для идентификации приложением различных сертификатов, содержащих определенный открытый ключ. Если конечный пользователь обладает несколькими сертификатами, особенно от разных СА, данное до-

полнение позволяет быстро определить требуемый сертификат. Для этих целей данное дополнение должно добавлять во все сертификаты конечных пользователей.

Значение данного дополнения для сертификатов конечных пользователей должно формироваться из значения открытого ключа способом, описанным выше.

Область применения ключа. Данное дополнение определяет область применения секретного ключа, соответствующего открытому, содержащемуся в сертификате.

Таблица 4.3

Область применения ключа

Флаг	Применение ключа
digitalSignature	Ключ может быть использован для целей обеспечения целостности и авторства информации. Формирование и проверка ЭЦП электронных документов и информации, установление идентичности в процессе аутентификации и т.д.
nonRepudiation	Ключ может быть использован в приложениях, обеспечивающих неотрекаемость.
keyEncipherment	Ключ может быть использован для шифрования других ключей.
dataEncipherment	Ключ может быть использован для целей обеспечения конфидециальности и целостности информации.
keyAgreement	Ключ может быть использован в процессе формирования других ключей (например, по алгоритму Деффи-Хелмана).
keyCertSign	Ключ может быть использован для целей формирования ЭЦП сертификатов.
CRLSign	Ключ может быть использован для целей формирования ЭЦП CRL.
EncipherOnly	Ключ может быть использован только для шифрования.
DecipherOnly	Ключ может быть использован только для расшифрования.

Расширенная область применения ключа. Данное дополнение *(Extended keyUsage)* предназначено для задания дополнительных областей применения ключа по требованиям прикладного ПО.

Значение *Область применения ключа* (*KeyPurposeId*) данного дополнения может быть определена любой организацией в зависимости от конкретных целей.

Срок действия секретного ключа. Данное дополнение позволяет Издателю сертификата задать различные сроки действия секретного ключа

и сертификата. Дополнение содержит два опциональных компонента: *notBefore и notAfter*. Секретный ключ, соответствующий открытому в сертификате, не должен быть использован до или после времен, указанных соответствующими компонентами. CA не должен формировать сертификат, в котором не указан ни один из компонентов.

Регламенты использования сертификата. Данное дополнение представляет собой последовательность, состоящую из одного или нескольких *информаторов регламента (PolicyInformation*), каждый из которых содержит *идентификатор объекта* (OID) и опциональный *квалификатор*.

Данный *информатор регламента* отображает регламент, с учетом которого сертификат был издан, и цели, для которых сертификат может быть использован. Опциональный *квалификатор*, который может присутствовать, не предусмотрен для целей изменения регламента, определенного информатором.

Приложения с определенными требованиями функционирования, должны содержать внутренний список регламентов, удовлетворяющих данным требованиям, для целей сравнения *идентификаторов объектов* в сертификате с имеющимся внутренним списком приложения. Если данное дополнение критичное, ПО, производящее обработку, должно обладать возможностью интерпретации данного дополнения (включая опциональный *квалификатор*). В противном случае сертификат должен быть отвергнут.

Соответствие регламентов. Данное дополнение предназначено для использования в сертификатах CA. Оно содержит список парных *Идентификаторов Объектов* (OID). Каждая пара в свою очередь включает *Регламент Зоны Издателя* (*issuerDomainPolicy*) и *Регламент Зоны Владельца* (*subjectDomainPolicy*). Такая парность означает, что CA, выступающий в роли Издателя (*issuing CA*), принимает *Регламент Зоны Издателя* эквивалентным *Регламенту Зоны Владельца* для подчиненного CA (*subject CA*).

Пользователи, относящиеся к Издающему CA (*issuing CA*), могут принимать *Регламент Зоны Издателя*(*issuerDomainPolicy*) для соответствующих приложений. Дополнение *Соответствие Регламентов* ставит в известность пользователей Издающего CA о том наборе *Регламентов*, (*subject CA*), которые сравнимы с регламентами, соответствующими их требованиям.

Альтернативное имя Владельца. Дополнение *Альтернативное Имя Владельца* может использоваться для двух целей. Во-первых, оно позволяет расширить границы идентификации Владельца сертификата. Для этого используются заранее определенные идентификаторы, которые включают адрес электронной почты, имя в DNS, IP адрес и URI. Во-вторых, оно предоставляет набор дополнительной справочной информации о Владельце сертификата. Для этого используется представление имени в различных видах и множественное представление имен. При необходимости введения

Таблица 4.4

Поля дополнения «Альтернативное Имя»

Наименование	Тип	Назначение	Идентификатор
otherName	ANY DEFINED BY type ID	Любое, определяемое Издателем	CHOICE[0]
rfc822Name	IA5String	Адрес электронной почты rfc822 [30]	CHOICE[1]
DNSName	A5String	Имя DNS	CHOICE[2]
directoryName	IA5String	X.500 DN имя	CHOICE[4]
uniformResource identifier	IA5String	Адрес URI	CHOICE[6]
ipAddress	OCTET STRING	Адрес IP	CHOICE[7]
registeredID	OBJECT IDENTIFIER	Идентификатор ASN.1 объекта	CHOICE[8]

такой дополнительной идентификации в сертификат должно использоваться дополнение *Альтернативное Имя Владельца* или *Альтернативное Имя Издателя*.

В связи с тем, что альтернативное имя может быть использовано для целей определения соответствия Владельца и открытого ключа, все части, идентифицирующие его и входящие в альтернативное имя, должны быть проверены CA. Уровень проверки дополнительной информации определяется Регламентом CA.

Альтернативное Имя Владельца может быть ограничено тем же способом, что и поле *Владелец* в сертификате, используя дополнение *nameConstraintsExtension*.

Кроме зарегистрированных типов полей, Издающий Центр может определить и использовать свои собственные типы имен, определив их в поле *otherName*.

Альтернативное имя Издателя. Так же как и дополнение *Альтернативное Имя Владельца*, дополнение *Альтернативное имя Издателя* (*issuerAltName*) служит целям дополнительной ассоциации *Издателя* сертификата. Правила использования данного дополнения аналогичны использованию дополнения *Альтернативное Имя Владельца*.

Атрибуты Справочника Владельца сертификата. Дополнение предусмотрено для хранения дополнительной информации, связанной с атрибутами каталога X.500.

Основные ограничения. Дополнение *Базовые ограничения* идентифицирует, является ли Владелец сертификата Центром Сертификации, и ка-

кова длина цепочки сертификатов для этого CA.

Поле *pathLenConstraint* имеет смысл только при условии, если значение *cA* установлено в *TRUE*. В этом случае оно обозначает максимальное количество сертификатов CA, которые следуют за данным сертификатом в цепочке. Значение ноль означает, что только сертификаты конечного пользователя могут следовать в цепочке за данным сертификатом. При использовании значение *pathLenConstraint* больше или равно нулю. Если значение не установлено, это означает, что лимит на длину цепочки не определен.

Ограничения имени. Дополнение *Ограничение имени* должно использоваться только в сертификатах CA. Оно указывает пространство имен, внутри которого должны быть расположены все имена Владельцев издаваемых сертификатов. Ограничения могут быть применимы как к имени Владельца (*Subject DN*), так и к альтернативному имени.

Ограничения определены в терминах допускаемого (*permittedSubtrees*) или исключаемого (*excludedSubtrees*) поддерева имен. Любое имя, совпадающее с ограничением в исключающем поддереве, является некорректным в независимости от возможного его присутствия в допускаемом поддереве.

При реализации данного дополнения RFC 2459 рекомендуется:

— не использовать поля *minimum* и *maximum* ни в какой из форм имен, так что *minimum* всегда ноль, а *maximum* всегда отсутствует;
— использовать только поля *permittedSubtrees* для задания разрешенного диапазона имен и не использовать *excludedSubtrees*, что согласуется с организационной или территориальной схемой иерархии.

Ограничение регламента. Данное дополнение может быть использовано в сертификатах, издаваемых для CA. Дополнение *Ограничение регламента* накладывает ограничения на проверяемую цепочку в двух направлениях. Оно может использоваться для запрещения проверки соответствия регламентов (*policy mapping*) или требовать, чтобы каждый сертификат в цепочке содержал приемлемый идентификатор регламента (*policy identifier*).

Точка распространения CRL. Точка распространения CRL является дополнением, которое определяет механизм и расположение CRL определенного типа в сети, и определяет зону действия CRL как:

— только для конечных пользователей;
— только для CA;
— или ограничивает коды мотивации.

Коды мотивировки, ассоциированные с точкой распространения, должны специфицироваться в поле *onlySomeReasons*. Если поле *onlySomeReasons* отсутствует, точка распространения должна содержать отзываемые сертификаты для всех кодов. CA может использовать Точку распространения CRL как основу для управления потоками данных при компрометации.

В этом случае отзывы сертификатов с кодами *keyCompromise* и *cACompromise* располагаются в одной точке распространения, а все остальные — в другой.

Способ доступа к информации CA. Данное дополнение определено в рекомендациях IETF RFC 2459 (в отличие от остальных стандартных дополнений, которые определены как в рекомендациях X.509, так и в RFC 2459).

Данное дополнение указывает на способы доступа к информации и сервисам CA, издавшим сертификат, в котором это дополнение установлено. Информация и сервис могут включать процедуры интерактивной проверки и получения Регламента (Регламентов) CA. Способ получения CRL не специфицируется данным дополнением, для этого используется дополнение *cRLDistributionPoints*.

4.4.8. Верификация цепочки сертификатов

Доверие любому сертификату пользователя определяется на основе цепочки сертификатов. Причем начальным элементом цепочки является сертификат CA, хранящийся в защищенном персональном справочнике пользователя.

Процедура верификации цепочки сертификатов описана в рекомендациях X.509 и RFC 2459 [31] и проверяет связанность между именем Владельца сертификата и его открытым ключом. Процедура верификации цепочки подразумевает, что все «правильные» цепочки начинаются с сертификатов, изданных одним CA. Под доверенным центром понимается главный CA, открытый ключ которого содержится в самоподписанном сертификате. Такое ограничение упрощает процедуру верификации, хотя наличие самоподписанного сертификата и его криптографическая проверка не обеспечивают безопасности. Для обеспечения доверия к открытому ключу такого сертификата должны быть применены специальные способы его распространения и хранения, так как на данном открытом ключе проверяются все остальные сертификаты.

Алгоритм верификации цепочек использует следующие данные:

- X.500 имя Издателя сертификата;
- X.500 имя Владельца сертификата;
- открытый ключ Издателя;
- срок действия открытого (секретного) ключа Издателя и Владельца;
- ограничивающие дополнения, используемые при верификации цепочек;
- CRL для каждого Издателя (даже если он не содержит отзываемых сертификатов).

Цепочка сертификатов представляет собой последовательность из n сертификатов, в которой:

- для всех x в $\{1, (n-1)\}$, Владелец сертификата x есть Издатель сертификата $x+1$;
- сертификат $x = 1$ есть самоподписанный сертификат;
- сертификат $x = n$ есть сертификат конечного пользователя.

Одновременно с цепочкой сертификатов используется цепочка CRL, представляющая собой последовательность из n CRL, в которой:

— для всех CRL x в $\{1, n\}$, Издатель сертификата x есть Издатель CRL x;
— CRL $x = 1$ есть CRL, изданный Владельцем самоподписанного сертификата;
— CRL $x = n$ есть CRL, изданный Издателем сертификата конечного пользователя.

После построения двух цепочек (сертификатов и CRL) выполняется:

— криптографическая проверка сертификатов и CRL в цепочках;
— проверка сроков действия сертификатов и CRL;
— проверка соответствия имен Издателя и Владельца с использованием дополнения *nameConstraints*;
— проверка длины цепочки с использованием дополнения *basicConstraints*;
— проверка на отзыв сертификатов, причем, если сертификат промежуточного центра был отозван CRL вышестоящего центра, все сертификаты, изданные промежуточным центром, считаются недействительными;
— проверка приемлемых регламентов использования сертификата и приемлемых областей использования ключа с использованием дополнений *certificatesPolicies* и *extendedKeyUsage*.

4.4.9. Стандарты в области PKI

Стандарты в области PKI делятся на две группы: часть из них описывает собственно реализацию PKI, а вторая часть, которая относится к пользовательскому уровню, использует PKI, не определяя ее.

Стандартизация в области PKI позволяет различным приложениям взаимодействовать между собой с использованием единой PKI.

В особенности стандартизация важна в области:

— процедуры регистрации и выработки ключа;
— описания формата сертификата;
— описания формата CRL;
— описания формата криптографически защищенных данных;
— описания протоколов обмена информацией.

Основным центром по выпуску согласованных стандартов в области PKI является рабочая группа PKI (PKI working group) организации IETF (Internet Engineering Task Force), известная как группа PKIX (от сокращения PKI for X.509 certificates).

4.4.9.1. Стандарты PKIX

Спецификации PKIX основаны на двух группах стандартов: X.509 ITU-T (Международный комитет по телекоммуникациям) и PKCS (Public Key Cryptography Standards) firmy RSA Data Security. X.509 изначально был предназначен для спецификации аутентификации при использовании

в составе сервиса X.500 директории. Фактически же, синтаксис электронного сертификата, предложенный в X.509, признан стандартом де-факто и получил всеобщее распространение независимо от X.500. Однако X.509 ITU-T не был предназначен для полного определения PKI. В целях применения стандартов X.509 в повседневной практике пользователи, поставщики и комитеты по стандартизации обращаются к стандартам PKCS.

PKIX группа издала следующие стандарты Internet (RFC):

— Internet X.509 Public Key Infrastructure Certificate and CRL Profile (RFC 2459) [31];
— Internet X.509 Public Key Infrastructure Certificate Management Protocols (RFC 2510) [32];
— Internet X.509 Certificate Request Message Format (RFC 2511) [33];
— Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework (RFC 2527) [34];
— Internet X.509 Public Key Infrastructure Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates (RFC 2528) [35];
— Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2 (RFC 2559) [36];
— Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP (RFC 2585) [37];
— Internet X.509 Public Key Infrastructure LDAPv2 Schema (RFC 2587) [38];
— X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — OCSP (RFC 2560) [39].

Стандарт X.509 ITU-T является фундаментальным стандартом, лежащим в основе всех остальных, используемых в PKI. Основное его назначение — определение формата электронного сертификата и CRL.

4.4.9.2. PKCS

Из серии стандартов, изданных фирмой RSA Data Security, наиболее важными и используемыми в PKI являются:

— PKCS #7 Cryptographic Message Syntax Standard;
— PKCS #10 Certificate Request Syntax Standard;
— PKCS #12 Personal Information Exchange Syntax Standard.

Вместо устаревшего стандарта RSA PKCS #7, описывающего форматы криптографических сообщений, в июне 1999 года был принят RFC 2630 «Cryptographic Message Syntax».

4.4.9.3. Стандарты, основанные на PKI

Большинство стандартов, использующих криптографию, разработано с учетом использования PKI.

— S/MIME
  Стандарт S/MIME определен IETF для обеспечения защищенного обмена сообщениями. S/MIME использует PKI для формирования

ЭЦП и шифрования информации. В группе стандартов S/MIME наиболее важными являются следующие: Cryptographic Message Syntax, Message Specification, Certificate Handling и Certificate Request Syntax.

— SSL и TLS
Протокол SSL (разработанный фирмой Netscape) и соответствующий ему стандарт IETF TLS (RFC 2246) являются наиболее используемыми стандартами для обеспечения защищенного доступа к Web. Вместе с этим, SSL и TLS широко используются для создания клиент-серверных приложений, не использующих Web. Оба эти протокола в своей основе используют PKI.

— Secure Electronic Transactions (SET)
Протокол SET был разработан фирмами Visa и MasterCard и предназначен для обеспечения системы электронных банковских расчетов с использованием пластиковых карт. В данном протоколе PKI является фундаментом, на котором базируется вся система аутентификации участников расчетов.

— IPSec
Протокол IPSEC (Internet Protocol Security Protocol) разработан IETF как протокол для шифрования IP и является одним из основных протоколов, используемых для построения VPN. PKI в протоколе IPSec используется для аутентификации и шифрования. В настоящее время протокол еще широко не распространен, но повсеместное развитие PKI приводит к возрастанию количества реализаций IPSec.

Рекомендуемая литература

1. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments / P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer et al. — 1998. — http://www.nsa.gov/selinux/inevit-abs.html. 6, 34

2. The Flask Security Architecture: System Support for Diverse Security Policies / R. Spencer, S. Smalley, P. Loscocco et al. — 1999. — http://www.nsa.gov/selinux/flask-abs.html. 6, 34

3. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации. — 1992. — http://www.gostexkom.ru. 10

4. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000. — 452 с. 35, 36, 37

5. DOD 5200.28-STD // Department of Defense Trusted Computer System Evaluation Criteria. — 1985. 36

6. *Ferraiolo D., Kuhn R.* Role-Based Access Control // Proceedings of the 15th National Computer Security Conference. — 1992. 36

7. DTOS General System Security and Assurability Assessment Report: Technical report md a904-93-c-4209 cdrl a011 / Secure Computing Corporation. — 1997. — http://www.securecomputing.com/randt/HTML/dtos.html. 36

8. *Медведовский И. Д., Семьянов П. В., Леонов Д. Г.* Атака на Internet. — Издательство ДМК, 1999. 53

9. *Тайтуров К.* Доспехи для «Пингвина» // Открытые системы. — № 11. — 2003. 53

10. Smashing The Stack For Fun And Profit. — www.opennet.ru/base/sec/p49-14.txt.html. 53

11. *Колищак А.* Атаки на переполнение буфера. — 1999. — http://www.security.nnov.ru/articles/bo.asp. 53

12. Libsafe. — www.research.avayalabs.com/project/libsafe/. 53, 60

13. Openwall. — www.openwall.com. 53, 56

14. PaX. — pageexec.virtualave.net/docs/pax.txt. 53, 57

15. Grsecurity. — http://www.grsecurity.net/. 53, 60

16. *Ерижоков А. А.* LIDS — Система обнаружения атак и защиты от вторжения. — 2000. — http://www.opennet.ru/docs/RUS/lids/lids.html. 53

17. Security-Enhanced Linux. — http://www.nsa.gov/selinux/index.html. 53, 73

18. *Loscocco P., Smalley S.* Integrating Flexible Support for Security Policies into the Linux Operating System. — http://www.nsa.gov/selinux/slinux-abs.html. 53, 73

19. Rule Set Based Access Control (RSBAC) for Linux. — http://rsbac.org/. 53

20. *Тайтуров К.* Linux Kernel HOWTO. — www.opennet.ru/docs/HOWTO-RU/Kernel-HOWTO.html. 56

21. Medusa DS9. — http://medusa.fornax.sk. 58

22. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000. — 452 с. 66, 69

23. *Fischer-Hubner S., Ott A.* From a Formal Privacy Model to its Implementation. — http://www.rsbac.org/niss98.htm. 67

24. *Barkley J.* Mandatory Access Control. — 1994. — http://csrc.nist.gov/publications/nistpubs/800-7/node35.html. 69

25. *Wheeler D. A.* Secure programmer: Minimizing privileges. — 2004. — http://www-106.ibm.com/developerworks/linux/library/l-sppriv.html. 76

26. *Давыдов А. Н.* Обзор инфраструктур открытых ключей // Труды научно-технической конференции «Безопасность информационных технологий». Секция № 2: Информационная безопасность сложных систем. — Т. 1. — Пенза: 2001. — С. 34–36. — http://beda.stup.ac.ru/rv-conf/v01/015/. 107

27. *Шнайер Б.* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. — М.: Триумф, 2003. — С. 816. — http://www.ssl.stu.neva.ru/psw/crypto/appl_rus/appl_cryp.htm. 108

28. *Янглав Р.* PKI: как это работает. — http://www.iso.ru/journal/articles/96.html. 108

29. Инфраструктура открытых ключей. — http://www.atnn.ru/default.html?/pki.html. 108

30. *Crocker D. H.* Standard for the Format of ARPA Internet Text Messages. — 1982. — http://ietf.org/rfc/rfc0822.txt?number=822. 122

31. Internet X.509 Public Key Infrastructure. Certificate and CRL Profile. / R. Housley, W. Ford, W. Polk, D. Solo. — 1999. — http://ietf.org/rfc/rfc2459.txt. 124, 126

32. *Adams C.* Internet X.509 Public Key Infrastructure. Certificate Management Protocols. — 1999. — http://ietf.org/rfc/rfc2510.txt?number=2510. 126

33. Internet X.509 Certificate Request Message Format / M. Myers, C. Adams, D. Solo, D. Kemp. — 1999. — http://ietf.org/rfc/rfc2511.txt?number=2511. 126

34. *Chokhani S., Ford W.* Internet X.509 Public Key Infrastructure. Certificate Policy and Certification Practices Framework. — 1999. — http://ietf.org/rfc/rfc2527.txt?number=2527. 126

35. *Housley R., Polk W.* Internet X.509 Public Key Infrastructure. Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates. — 1999. — http://ietf.org/rfc/rfc2528.txt?number=2528. 126

36. *Boeyen S., Howes T., Richard P.* Internet X.509 Public Key Infrastructure. Operational Protocols - LDAPv2. — 1999. — http://ietf.org/rfc/rfc2559.txt?number=2559. 126

37. *Housley R., Hoffman P.* Internet X.509 Public Key Infrastructure. Operational Protocols: FTP and HTTP. — 1999. — http://ietf.org/rfc/rfc2585.txt?number=2585. 126

38. *Boeyen S., Howes T., Richard P.* Internet X.509 Public Key Infrastructure. LDAPv2 Schema. — 1999. — http://ietf.org/rfc/rfc2587.txt?number=2587. 126

39. X.509 Internet Public Key Infrastructure. Online Certificate Status Protocol - OCSP / M. Myers, R. Ankney, A. Malpani et al. — 1999. — http://ietf.org/rfc/rfc2560.txt?number=2560. 126