ТЕМА 5.1. БЕЗОПАСНОСТЬ И КОНТРОЛЬ ДОСТУПА

В данной теме рассматриваются следующие вопросы:

- Понятие безопасности ОС.
- Основные угрозы безопасности ОС.
- Методы и защитные механизмы операционных систем.
- Авторизация и аутентификация пользователей.
- Криптографическая защита данных.
- Права доступа.

Лекции – 2 часа, лабораторные занятия – 2 часа, самостоятельная работа – 2 часа.

Экзаменационные вопросы по теме:

- Понятие безопасности ОС. Основные угрозы безопасности ОС. Методы и защитные механизмы операционных систем.

5.1.1. Понятие безопасности ОС

Для некоторых людей понятия «безопасность» и «защита» являются взаимозаменяемыми. Во избежание недоразумений для ссылки на проблемы общего плана будет использоваться термин безопасность (security), а для ссылки на специфические механизмы операционной системы, используемые для защиты информации, имеющейся на компьютере, будет использоваться термин защитные механизмы (protection mechanisms). Тем не менее четко обозначенной границы между ними не существует [1].

Во многих работах, посвященных безопасности, безопасность информационных систем разбита на три компонента: конфиденциальность, целостность и доступность. Вместе все три компонента часто называют CIA (Confidentiality, Integrity, Availability). Они показаны в табл. 5.1.1 и составляют основу свойств безопасности, которыми мы должны защититься от взломщиков и шпионов.

Таблица 5.1.1. Задачи и угрозы безопасности

Задачи	Угрозы
Конфиденциальность	Незащищенность данных
Целостность	Подделка данных
Доступность	Отказ от обслуживания

Первое свойство — конфиденциальность (confidentiality) — направлено на сохранение секретности данных. Точнее говоря, если владелец неких данных решил, что эти данные могут быть доступны строго определенному кругу лиц, система должна гарантировать невозможность допуска к данным лиц, не имеющих на это права. Как минимум владелец должен иметь возможность определить, кто и что может просматривать, а система должна обеспечить выполнение этих требований, касающихся в идеале отдельных файлов.

Второе свойство — целостность (integrity) — означает, что пользователи, не обладающие необходимыми правами, не должны иметь возможности изменять какие-либо данные без разрешения их владельцев. В этом контексте изменение данных включает в себя не только внесение в них изменений, но и их удаление или добавление в них ложных данных. Если система не может гарантировать, что заложенные в нее данные не будут подвергаться изменениям, пока владелец не решит их изменить, то она утратит свою роль хранилища данных.

Третье свойство — доступность (availability) — означает, что никто не может нарушить работу системы и вывести ее из строя. Атаки, вызывающие отказ от обслуживания (denial of service, (DOS)), приобретают все более распространенный характер. Например, если компьютер работает в роли интернет-сервера, то постоянное забрасывание его запросами может лишить его работоспособности, отвлекая все рабочее время его центрального процессора на изучение и отклонение входящих запросов. Если на обработку входящего запроса на чтение веб-страницы уходит, скажем, 100 мкс, то любой, кто сможет отправлять 10 000 запросов в секунду, способен вывести сервер из строя. Чтобы справиться с атаками, направленными на нарушение конфиденциальности и целостности данных, можно подобрать вполне подходящие модели и технологии, а вот противостоять атакам, вызывающим отказы от обслуживания, значительно труднее.

Позже было решено, что трех основных свойств для всех возможных сценариев недостаточно, и были добавлены дополнительные свойства, такие как аутентичность (authenticity), идентифицируемость (accountability), неотвергаемость (nonrepudiability), закрытость (privacy) и др. Конечно, неплохо было бы обладать всеми этими свойствами. Но даже при этом три исходных свойства по-прежнему занимают особое место в умах и сердцах большинства экспертов по вопросам безопасности.

5.1.2. Основные угрозы безопасности ОС

Организация эффективной и надежной защиты операционной системы невозможна без предварительного анализа возможных угроз ее безопасности. Угрозы безопасности операционной системы существенно зависят от условий эксплуатации системы, от того, какая информация хранится и обрабатывается в системе. Например, если операционная система используется для организации электронного документооборота, наиболее опасны угрозы, связанные с несанкционированным доступом (НСД) к файлам. Если же операционная система используется как платформа провайдера интернет-услуг, очень опасны атаки на сетевое программное обеспечение операционной системы [2].

Угрозы безопасности операционной системы можно классифицировать по различным аспектам их реализации.

Классификация угроз по цели атаки:

- несанкционированное чтение информации;
- несанкционированное изменение информации;
- несанкционированное уничтожение информации;
- полное или частичное разрушение операционной системы.

Классификация угроз по принципу воздействия на операционную систему:

- использование известных (легальных) каналов получения информации, например, угроза несанкционированного чтения файла, доступ пользователей к которому определен некорректно – разрешен доступ пользователю, которому, согласно политике безопасности, доступ должен быть запрещен;
- использование скрытых каналов получения информации, например, угроза использования злоумышленником недокументированных возможностей операционной системы;
- создание новых каналов получения информации с помощью программных закладок.

Классификация угроз по типу используемой злоумышленником уязвимости защиты:

- неадекватная политика безопасности, в том числе и ошибки администратора системы;
- ошибки и недокументированные возможности программного обеспечения операционной системы, в том числе и так называемые люки – случайно или преднамеренно встроенные в систему «служебные входы», позволяющие обходить систему защиты;
- ранее внедренная программная закладка.

Классификация угроз по характеру воздействия на ОС:

- **активное воздействие** – несанкционированные действия злоумышленника в системе;
- **пассивное воздействие** – несанкционированное наблюдение злоумышленника за процессами, происходящими в системе.

Угрозы безопасности ОС можно также классифицировать по таким признакам, как способ действий злоумышленника, используемые средства атаки, объект атаки, способ воздействия на объект атаки, состояние атакуемого объекта ОС на момент атаки.

Операционная система может подвергнуться следующим типичным атакам:

- **сканирование файловой системы**. Злоумышленник просматривает файловую систему компьютера и пытается прочесть (или скопировать) все файлы подряд. Рано или поздно обнаруживается хотя бы одна ошибка администратора. В результате злоумышленник получает доступ к информации, который должен быть ему запрещен;

- **подбор пароля**. Существует несколько методов подбора паролей пользователей:
  - тотальный перебор;
  - тотальный перебор, оптимизированный по статистике встречаемости символов или с помощью словарей;
  - подбор пароля с использованием знаний о пользователе (его имени, фамилии, даты рождения, номера телефона и т. д.);
- **кража ключевой информации**. Злоумышленник может подсмотреть пароль, набираемый пользователем, или восстановить набираемый пользователем пароль по движениям его рук на клавиатуре. Носитель с ключевой информацией (смарткарта, Touch Memory и т. д.) может быть просто украден;
- **сборка мусора**. Во многих операционных системах информация, уничтоженная пользователем, не уничтожается физически, а помечается как уничтоженная (так называемый мусор). Злоумышленник восстанавливает эту информацию, просматривает ее и копирует интересующие его фрагменты;
- **превышение полномочий**. Злоумышленник, используя ошибки в программном обеспечении ОС или политике безопасности, получает полномочия, превышающие те, которые ему предоставлены в соответствии с политикой безопасности. Обычно это достигается путем запуска программы от имени другого пользователя;
- **программные закладки**. Программные закладки, внедряемые в операционные системы, не имеют существенных отличий от других классов программных закладок;
- **жадные программы** – это программы, преднамеренно захватывающие значительную часть ресурсов компьютера, в результате чего другие программы не могут выполняться или выполняются крайне медленно. Запуск жадной программы может привести к краху операционной системы.

5.1.3. Методы и защитные механизмы операционных систем

Операционную систему называют **защищенной**, если она предусматривает средства защиты от основных классов угроз. Защищенная операционная система обязательно должна содержать средства разграничения доступа пользователей к своим ресурсам, а также средства проверки подлинности пользователя, начинающего работу с операционной системой. Кроме того, защищенная операционная система должна содержать средства противодействия случайному или преднамеренному выводу операционной системы из строя [2].

Если операционная система предусматривает защиту не от всех основных классов угроз, а только от некоторых, такую ОС называют **частично защищенной**.

Подходы к построению защищенных операционных систем

Существует два основных подхода к созданию защищенных операционных систем – фрагментарный и комплексный [2].

При **фрагментарном** подходе вначале организуется защита от одной угрозы, затем от другой. Примером фрагментарного подхода может служить ситуация, когда за основу берется незащищенная операционная система, на нее устанавливают антивирусный пакет, систему шифрования, систему регистрации действий пользователей и т. д.

При применении фрагментарного подхода подсистема защиты операционной системы представляет собой набор разрозненных программных продуктов, как правило, от разных производителей. Эти программные средства работают независимо друг от друга, при этом практически невозможно организовать их тесное взаимодействие. Кроме того, отдельные элементы такой подсистемы защиты могут некорректно работать в присутствии друг друга, что приводит к резкому снижению надежности системы.

При **комплексном** подходе защитные функции вносятся в операционную систему на этапе проектирования архитектуры операционной системы и являются ее неотъемлемой частью. Отдельные элементы подсистемы защиты, созданной на основе комплексного подхода, тесно взаимодействуют друг с другом при решении различных задач, связанных с организацией защиты информации, поэтому конфликты между ее отдельными компонентами практически невозможны. Подсистема защиты, созданная на основе комплексного подхода, может быть устроена так, что при фатальных сбоях в функционировании ее ключевых элементов она вызывает крах операционной системы, что не позволяет злоумышленнику отключать защитные функции системы. При фрагментарном подходе такая организация подсистемы защиты невозможна.

Как правило, подсистему защиты операционной системы, созданную на основе комплексного подхода, проектируют так, чтобы отдельные ее элементы были заменяемы. Соответствующие программные модули могут быть заменены другими модулями.

Административные меры защиты

Программно-аппаратные средства защиты операционной системы обязательно должны дополняться административными мерами защиты. Без постоянной квалифицированной поддержки со стороны администратора даже надежная программно-аппаратная защита может давать сбои. Перечислим основные административные меры защиты [2].

- Постоянный контроль корректности функционирования операционной системы, особенно ее подсистемы защиты. Такой контроль удобно организовать, если операционная система поддерживает автоматическую регистрацию наиболее важных событий (event logging) в специальном журнале.
- Организация и поддержание адекватной политики безопасности. Политика безопасности ОС должна постоянно корректироваться, оперативно реагируя на попытки злоумышленников преодолеть защиту операционной системы, а также на изменения в конфигурации операционной системы, установку и удаление прикладных программ.
- Осведомление пользователей операционной системы о необходимости соблюдения мер безопасности при работе с ОС и контроль за соблюдением этих мер.
- Регулярное создание и обновление резервных копий программ и данных ОС.
- Постоянный контроль изменений в конфигурационных данных и политике безопасности ОС.

Адекватная политика безопасности

Выбор и поддержание адекватной политики безопасности являются одной из наиболее важных задач администратора операционной системы. Если принятая в ОС политика безопасности неадекватна, это может привести к несанкционированному доступу злоумышленника к ресурсам системы и к снижению надежности функционирования ОС [2].

Известно утверждение: чем лучше защищена ОС, тем труднее с ней работать пользователям и администраторам. Это обусловлено следующими факторами:

- система защиты не всегда способна определить, является ли некоторое действие пользователя злонамеренным. Поэтому система защиты либо не пресекает некоторых видов несанкционированного доступа, либо запрещает некоторые вполне легальные действия пользователей. Чем выше защищенность системы, тем шире класс тех легальных действий пользователей, которые рассматриваются подсистемой защиты как несанкционированные;
- любая система, в которой предусмотрены функции защиты информации, требует от администраторов определенных усилий, направленных на поддержание

адекватной политики безопасности. Чем больше в операционной системе защитных функций, тем больше времени и средств нужно тратить на поддержание защиты;

- подсистема защиты операционной системы, как и любой другой программный пакет, потребляет аппаратные ресурсы компьютера. Чем сложнее устроены защитные функции операционной системы, тем больше ресурсов компьютера (процессорного времени, оперативной памяти и др.) затрачивается на поддержание функционирования подсистемы защиты и тем меньше ресурсов остается на долю прикладных программ;
- поддержание слишком жесткой политики безопасности может негативно сказаться на надежности функционирования операционной системы. Чрезмерно жесткая политика безопасности может привести к трудно выявляемым ошибкам и сбоям в процессе функционирования операционной системы и даже к краху ОС.

Оптимальная адекватная политика безопасности – это такая политика безопасности, которая не только не позволяет злоумышленникам выполнять несанкционированные действия, но и не приводит к описанным выше негативным эффектам.

Адекватная политика безопасности определяется не только архитектурой ОС, но и ее конфигурацией, установленными прикладными программами и т. д. Формирование и поддержание адекватной политики безопасности ОС можно разделить на ряд этапов.

1. **Анализ угроз**. Администратор операционной системы рассматривает возможные угрозы безопасности данного экземпляра ОС. Среди возможных угроз выделяются наиболее опасные, защите от которых нужно уделять максимум средств.
2. **Формирование требований к политике безопасности**. Администратор определяет, какие средства и методы будут применяться для защиты от тех или иных угроз. Например, защиту от несанкционированного доступа к некоторому объекту ОС можно решать либо средствами разграничения доступа, либо криптографическими средствами, либо используя некоторую комбинацию этих средств.
3. **Формальное определение политики безопасности**. Администратор определяет, как конкретно должны выполняться требования, сформулированные на предыдущем этапе. Формулируются необходимые требования к конфигурации ОС, а также требования к конфигурации дополнительных пакетов защиты, если установка таких пакетов необходима. Результатом данного этапа является развернутый перечень настроек конфигурации ОС и дополнительных пакетов защиты с указанием того, в каких ситуациях какие настройки должны быть установлены.
4. **Претворение в жизнь политики безопасности**. Задачей данного этапа является приведение конфигурации ОС и дополнительных пакетов защиты в соответствие с политикой безопасности, формально определенной на предыдущем этапе.
5. **Поддержание и коррекция политики безопасности**. В задачу администратора на данном этапе входят контроль соблюдения политики безопасности и внесение в нее необходимых изменений по мере появления изменений в функционировании ОС.

Специальных стандартов защищенности операционных систем не существует. Для оценки защищенности операционных систем используются стандарты, разработанные для компьютерных систем вообще. Как правило, сертификация операционной системы по некоторому классу защиты сопровождается составлением требований к адекватной политике безопасности, при безусловном выполнении которой защищенность

конкретного экземпляра операционной системы будет соответствовать требованиям соответствующего класса защиты.

Определяя адекватную политику безопасности, администратор операционной системы должен в первую очередь ориентироваться на защиту ОС от конкретных угроз ее безопасности.

Архитектура подсистемы защиты операционной системы

Подсистема защиты ОС выполняет следующие основные функции:

- **Идентификация и аутентификация**. Ни один пользователь не может начать работу с операционной системой, не идентифицировав себя и не предоставив системе аутентифицирующую информацию, подтверждающую, что пользователь действительно является тем, кем он себя заявляет.
- **Разграничение доступа**. Каждый пользователь системы имеет доступ только к тем объектам ОС, к которым ему предоставлен доступ в соответствии с текущей политикой безопасности.
- **Аудит**. Операционная система регистрирует в специальном журнале события, потенциально опасные для поддержания безопасности системы.
- **Управление политикой безопасности**. Политика безопасности должна постоянно поддерживаться в адекватном состоянии, то есть должна гибко реагировать на изменения условий функционирования ОС. Управление политикой безопасности осуществляется администраторами системы с использованием соответствующих средств, встроенных в операционную систему.
- **Криптографические функции**. Защита информации немыслима без использования криптографических средств защиты. Шифрование используется в ОС при хранении и передаче по каналам связи паролей пользователей и некоторых других данных, критичных для безопасности системы.
- **Сетевые функции**. Современные ОС, как правило, работают не изолированно, а в составе локальных и/или глобальных компьютерных сетей. ОС компьютеров, входящих в одну сеть, взаимодействуют между собой для решения различных задач, в том числе имеющих прямое отношение к защите информации.

Подсистема защиты обычно не представляет собой единый программный модуль. Как правило, каждая из перечисленных функций подсистемы защиты решается одним или несколькими программными модулями. Некоторые функции встраиваются непосредственно в ядро ОС. Между различными модулями подсистемы защиты должен существовать четко определенный интерфейс, используемый при взаимодействии модулей для решения общих задач.

5.1.4. Авторизация и аутентификация пользователей

В защищенной ОС любой пользователь (субъект доступа), перед тем как начать работу с системой, должен пройти идентификацию, аутентификацию и авторизацию [2].

Идентификация субъекта доступа заключается в том, что пользователь (субъект) сообщает операционной системе идентифицирующую информацию о себе (имя, учетный номер) и таким образом идентифицирует себя.

Для того чтобы установить, что пользователь именно тот, за кого себя выдает, в информационных системах предусмотрена процедура аутентификации, задача которой – предотвращение доступа к системе нежелательных лиц.

Аутентификация субъекта доступа заключается в том, что субъект предоставляет операционной системе, помимо идентифицирующей информации, еще и аутентифицирующую информацию, подтверждающую, что он действительно является тем субъектом доступа, к которому относится идентифицирующая информация.

Авторизация субъекта доступа происходит после успешной идентификации и аутентификации. При авторизации субъекта ОС выполняет действия, необходимые для того, чтобы субъект мог начать работу в системе.

Авторизация субъекта не относится напрямую к подсистеме защиты операционной системы. В процессе авторизации решаются технические задачи, связанные с организацией начала работы в системе уже идентифицированного и аутентифицированного субъекта доступа.

С точки зрения обеспечения безопасности ОС, процедуры идентификации и аутентификации являются весьма ответственными. Действительно, если злоумышленник сумел войти в систему от имени другого пользователя, он легко получает доступ ко всем объектам ОС, к которым имеет доступ этот пользователь. Если при этом подсистема аудита генерирует сообщения о событиях, потенциально опасных для безопасности ОС, то в журнал аудита записывается не имя злоумышленника, а имя пользователя, от имени которого злоумышленник работает в системе.

Наиболее распространенными методами идентификации и аутентификации являются следующие:

- идентификация и аутентификация с помощью имени и пароля;
- идентификация и аутентификация с помощью внешних носителей ключевой информации;
- идентификация и аутентификация с помощью биометрических характеристик пользователей.

5.1.5. Криптографическая защита данных

Криптография – это метод защиты информации путем использования закодированных алгоритмов, хэшей и подписей. Информация может находиться на этапе хранения (например, файл на жестком диске), передачи (например, электронная связь между двумя или несколькими сторонами) или использования (при применении для вычислений) [3].

Замысел криптографии заключается в том, чтобы закодировать открытый текст (plaintext) — сообщение или файл, превратив его в зашифрованный текст (ciphertext), чтобы о том, как его снова превратить в открытый текст, знали только те, кто имеет на это право. Для всех остальных зашифрованный текст будет лишь непонятным набором битов. Как бы странно это ни прозвучало для новичков, но алгоритмы (функции), используемые для шифрования и дешифрования, всегда должны быть открытыми. Попытка хранить их в секрете практически никогда не срабатывает и создает у людей, пытающихся сохранить секреты, ложное чувство безопасности. В коммерции такая тактика называется безопасностью за счет неизвестности (security by obscurity) и используется только дилетантами [1].

При реальном подходе к делу безопасность зависит от параметров алгоритмов, называемых ключами. Если P — это файл с обычным текстом, K_E — ключ шифрования, C — зашифрованный текст и E — алгоритм шифрования (то есть функция), то $C = E(P,K_E)$. Это и есть определение шифрования. Из него следует, что зашифрованный текст получается за счет использования известного алгоритма шифрования E с параметрами, в качестве которых выступает открытый текст P и секретный ключ шифрования, K_E. Идея, предполагающая использование открытого алгоритма и содержание секретности исключительно в ключах, называется принципом Керкгоффса (Kerckhoffs' Principle). Он был сформулирован голландским криптографом XIX века Огюстом Керкгоффсом. Сегодня этой идеи придерживаются все серьезные криптографы.

Аналогично прежней формуле, $P = D(C, K_D)$, где D — это алгоритм дешифрования, а K_D — ключ дешифрования. Согласно этой формуле, чтобы получить обычный текст P из

зашифрованного текста C при наличии ключа дешифрования K_D, нужно запустить алгоритм D, используя C и K_D в качестве параметров. Взаимоотношения между различными компонентами показаны на рис. 5.1.1.

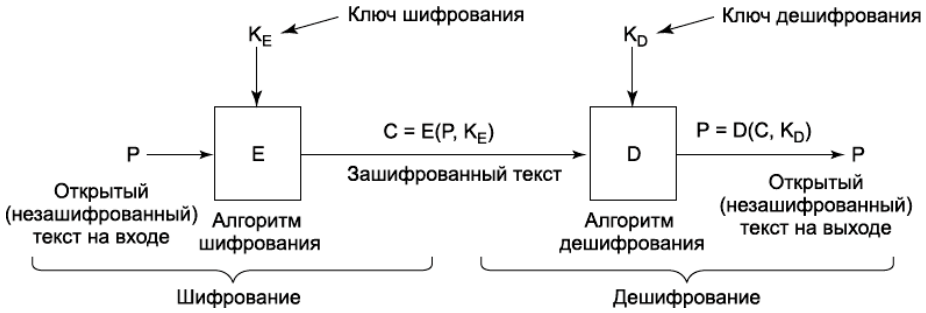

Рис. 5.1.1. Взаимоотношения между открытым и зашифрованным текстом

Криптография преследует четыре основных цели [3].

- **Конфиденциальность** – предоставляет доступ к информации только авторизованным пользователям.
- **Целостность** – гарантирует, что над информацией не производились манипуляции.
- **Подлинность** – подтверждает подлинность информации или личность пользователя.
- **Обеспечение невозможности отказа** – лишает возможности отрицать прежние обязательства или действия.

Криптография использует некоторые низкоуровневые криптографические алгоритмы для достижения одной или нескольких из этих целей информационной безопасности. Среди этих инструментов – алгоритмы шифрования, алгоритмы цифровой подписи, алгоритмы хэширования и другие функции.

Алгоритм шифрования – это процедура, которая преобразует сообщение в формате неформатированного текста в зашифрованный текст. Современные алгоритмы используют сложные математические вычисления и один или несколько ключей шифрования. Благодаря этому можно относительно легко зашифровать сообщение, но практически невозможно расшифровать его, не зная ключей.

В зависимости от того, как действуют ключи, технологии шифрования делятся на две категории: симметричные и асимметричные.

Криптография с симметричным ключом

Алгоритмы шифрования с симметричным ключом используют одни и те же криптографические ключи для шифрования простого текста и расшифровки зашифрованного. При использовании симметричного шифрования все получатели сообщения должны иметь доступ к общему ключу [3].

На рис. 5.1.2 показано, как шифрование и расшифровка работают с симметричными ключами и алгоритмами, когда все стороны используют один и тот же общий ключ.

На первой иллюстрации симметричный ключ и алгоритм используются для преобразования сообщения из обычного текста в зашифрованный. На второй иллюстрации показан тот же симметричный ключ и симметричный алгоритм, который используется для обратного преобразования зашифрованного текста в обычный.

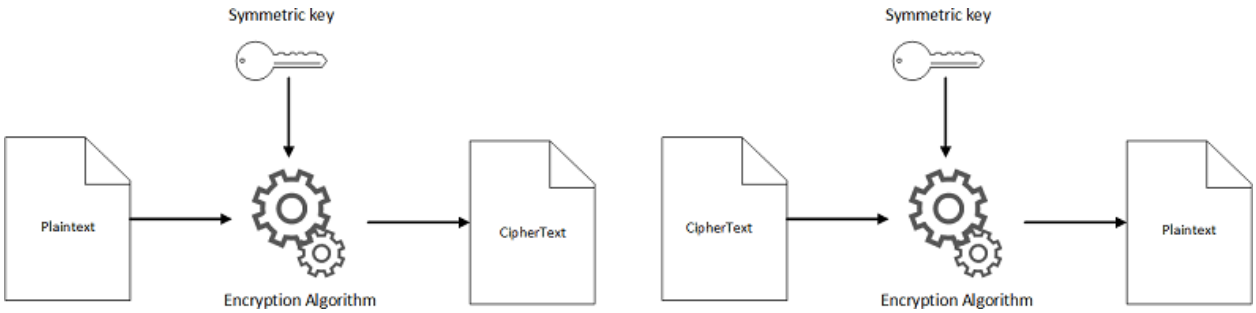

Рис. 5.1.2. Использование симметричного ключа

Одним из самых популярных блочных шифров является расширенный стандарт шифрования (Advanced Encryption Standard, AES). Этот блочный шифр поддерживает 128-, 192- и 256-разрядные ключи. AES часто используется в сочетании со счётчиком с аутентификацией Галуа (Galois/Counter Mode, GCM), который известен как AES-GCM, с целью создания алгоритма аутентифицированного шифрования.

AES – это промышленный стандарт шифрования, используемый во всем мире. Его безопасность широко известна, а эффективные программные и аппаратные реализации широко доступны. Старые алгоритмы DES и TrippleDES давно считаются взломанными.

В Республике Беларусь алгоритм шифрования с симметричным ключом определен стандартом СТБ 34.101.31-2007 «Информационные технологии и безопасность. Криптографические алгоритмы шифрования и контроля целостности».

Асимметричная криптография (с открытым ключом)

Асимметричная (с открытым ключом) криптография охватывает широкий круг алгоритмов. Они основаны на математических задачах, которые относительно легко решить в одном направлении и сложно в противоположном [3].

Одним из известнейших примеров таких задач является проблема разложения числа на множители: для четко подобранных простых чисел p и q мы можем быстро вычислить произведение N=p*q. Но если дано лишь N, очень сложно восстановить p и q.

Широко используется такой криптографический алгоритм с открытым ключом на основе задачи разложения на множители, как функция Ривеста-Шамира-Адлемана (RSA). В сочетании с соответствующей схемой заполнения можно использовать RSA во многих целях, в том числе для асимметричного шифрования.

Схема шифрования называется **асимметричной**, если в ней один ключ (открытый) используется для шифрования данных, а другой, но математически связанный (частный) – для их расшифровки.

Необходимо, чтобы было невозможно вычислить частный ключ, если известен только открытый. Поэтому общий открытый ключ можно передавать, а частный – держать в тайне и в безопасности. Эти ключи называются парой ключей.

Одной из популярных схем асимметричного шифрования является RSA-OAEP, представляющая собой сочетание функции RSA со схемой заполнения, называемой Оптимальное асимметричное шифрование с заполнением (OAEP). Обычно RSA-OAEP используется только для шифрования небольших объемов данных, поскольку работает медленно и производит шифр, который намного длиннее незашифрованного текста.

Гибридное шифрование

Поскольку алгоритмы с открытым ключом, такие как RSA-OAEP, менее эффективны, чем их симметричные аналоги, то они не имеют широкого применения для непосредственного шифрования данных. Однако они играют важную роль в криптографической экосистеме, предоставляя возможность обмена ключами [3].

Для использования симметричного шифрования стороны должны совместно использовать ключ. Несмотря на то, что этот ключ можно отправить по существующему зашифрованному каналу, нам не понадобится новый ключ, если безопасный канал уже использовался. Вместо этого мы решаем проблему обмена ключами с использованием криптографии с открытым ключом.

Ниже описаны два распространенных способа обмена симметричными ключами.

Асимметричное шифрование. Одна сторона генерирует симметричный ключ, затем зашифровывает ключ, применяя такой алгоритм, как RSA-OAEP, для открытого ключа другой стороны. Получатель может расшифровать зашифрованный текст с помощью своего частного ключа, чтобы восстановить симметричный ключ.

Обмен ключами Диффи-Хеллмана – (DH). Алгоритм Диффи-Хеллмана – это другой тип криптографического алгоритма с открытым ключом, разработанный специально для того, чтобы помочь сторонам достичь соглашения о симметричном ключе при отсутствии безопасного канала. Алгоритм Диффи-Хеллмана основан не на той же математической задаче, что и функция RSA, и менее гибок, чем RSA. Однако он отличается более эффективными конструкциями, которые предпочтительнее в некоторых примерах использования.

Это сочетание криптографии с открытым ключом для обмена ключей и симметричного шифрования для пакетного шифрования данных называется гибридным шифрованием.

В гибридном шифровании используются уникальные свойства криптографии с открытым ключом для обмена секретной информацией по недоверенному каналу с эффективностью симметричного шифрования. Оно представляет собой практически применимое сквозное решение для обеспечения конфиденциальности данных.

Гибридное шифрование широко используется в протоколах передачи данных для Интернета, таких как протокол TLS (безопасность транспортного уровня). Когда вы подключаетесь к веб-сайту, который использует HTTPS (безопасный HTTP с TLS), браузер согласовывает криптографические алгоритмы, защищающие соединение. Это алгоритмы обмена ключами, симметричного шифрования и цифровой подписи.

Цифровая подпись

Схемы цифровых подписей – это тип криптографии с открытым ключом, который гарантирует целостность, подлинность и обеспечение невозможности отказа [3].

Процесс подписания можно воспринимать как шифрование файла с использованием частного ключа. Лицо, подписывающее цифровой документ, например, файл или фрагмент кода, использует для создания «подписи» свой частный ключ.

Эта подпись является уникальной для пары документ-частный ключ и может прикрепляться к документу и проверяться с использованием открытого ключа лица, ставящего подпись. Двумя распространенными алгоритмами цифровой подписи являются RSA с вероятностной схемой подписи (RSA-PSS) и алгоритм цифровой подписи (DSA).

Код аутентификации сообщения

Код аутентификации сообщения (MAC) – это симметричная версия цифровой подписи. При использовании MAC две или больше сторон совместно используют ключ. Одна сторона создает тег MAC, который является симметричной версией цифровой подписи, и прикрепляет его к документу. Другая сторона может проверить целостность сообщения с использованием того же ключа, который использовался для создания тега.

Обратите внимание, что несколько сторон совместно используют ключ, с помощью которого создавались теги MAC, поэтому MAC невозможно использовать для аутентификации или обеспечения невозможности отказа, потому что неизвестно, какая сторона создала тег.

MAC могут быть автономными алгоритмами, например, кодами аутентификации сообщений на основе хэша (HMAC). Однако, поскольку целостность сообщения почти всегда является ценным подтверждением, она часто интегрируется в алгоритмы симметричного шифрования, такие как AES-GCM.

Эллиптическая криптография

Эллиптическая криптография (ECC) – это технология криптографии с открытым ключом, основанная на математической теории эллиптических кривых [3].

Наибольшим преимуществом ECC является то, что она может обеспечить уровень безопасности, подобный более традиционным технологиям, с меньшими ключами и более быстрой работой. Благодаря своей эффективности ECC хорошо подходит для использования в устройствах с относительно низкой вычислительной мощностью, например, для мобильных телефонов.

ECC можно использовать для эффективного обмена ключами с использованием варианта протокола Диффи-Хеллмана (ECDH) на эллиптических кривых или для цифровых подписей с использованием алгоритма цифровых подписей на эллиптических кривых (ECDSA). Благодаря своей скорости и гибкости ECC широко используется во многих интернет-приложениях.

Хэширование в криптографии

Криптографическая хэш-функция – это инструмент для преобразования произвольных данных в «отпечаток» фиксированной длины. Хэш-функции создаются таким образом, чтобы было сложно найти два различных набора входных данных, дающих один и тот же отпечаток, и чтобы было сложно найти сообщение, отпечаток которого совпадает с фиксированным значением [3].

В отличие от схем шифрования, схем подписей и MAC, хэш-функции не имеют ключа. Кто угодно может вычислить хэш для данного входного значения, и хэш-функция всегда будет генерировать одно и то же самое выходное значение для одного и того же входного.

Хэш-функции являются важным конструктивным элементом крупных криптографических алгоритмов и протоколов. Это алгоритмы цифровых подписей, алгоритмы выделенных MAC, протоколы аутентификации и хранилище паролей.

В настоящее время широко используется семейство криптографических алгоритмов SHA-2 (Secure Hash Algorithm Version 2 — безопасный алгоритм хеширования, версия 2), включающее в себя алгоритмы SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256 и SHA-512/224.

Что такое криптовалюта?

Криптовалюта – это цифровая валюта, при использовании которой транзакции подтверждаются и записи ведутся децентрализованной системой, а не централизованным органом. Криптовалюта является примером практического применения криптографии [3].

Криптовалюта использует множество низкоуровневых криптографических алгоритмов для создания доверенной и надежной платформы. Криптовалюта использует многие концепции, которые упоминаются на этой странице: эллиптическую криптографию, цифровые подписи, хэш-функции и другие концепции. В совокупности эти алгоритмы обеспечивают доверие и ответственность без наличия централизованного органа.

Что такое постквантовая криптография?

За несколько последних десятилетий много средств было инвестировано в квантовые вычисления. Квантовые компьютеры используют квантовую физику и способны решать

математические задачи, такие как разложение на множители, что с точки зрения вычислений невозможно для классических компьютеров [3].

Крупномасштабный квантовый компьютер смог бы взломать сегодняшние криптосистемы с открытым ключом, в том числе криптосистемы на основе функций Ривеста-Шамира-Адлемана (RSA). Взлом этих алгоритмом означал бы потерю конфиденциальности и аутентификации многих приложений протоколов, которыми мы пользуемся в настоящее время.

Несмотря на то, что сегодня уже существуют небольшие квантовые компьютеры, они слишком малы для взлома криптографических алгоритмов. Неизвестно, когда появится криптографически релевантный квантовый компьютер (CRQC) и появится ли он вообще. Уже сделаны значительные научные прорывы, необходимые для разработки CRQC.

Постквантовая криптография (PQC) относится к криптографическим алгоритмам, выполняемым на компьютерах, которые мы используем сегодня, и не имеющим известных уязвимостей перед крупным квантовым компьютером.

Что такое криптографические вычисления?

Представленные на данный момент инструменты дают возможность применять шифрование при хранении данных и при их передаче. Традиционно данные расшифровывались перед их использованием в вычислениях. Криптографические вычисления заполнили этот пробел, предоставив инструменты для работы непосредственно с данными, защищенными с помощью криптографии [3].

Термин «криптографические вычисления» охватывает широкий диапазон технологий, в том числе безопасные многосторонние вычисления, гомоморфное шифрование и шифрование с возможностью поиска. Несмотря на различия в подробностях реализации эти технологии обеспечивают криптографическую безопасность данных с возможностью проводить вычисления с использованием защищенных данных, сохраняя при этом их конфиденциальность.

5.1.6. Права доступа

Объектом доступа (или просто объектом) называют любой элемент операционной системы, доступ к которому пользователей и других субъектов доступа может быть произвольно ограничен. Возможность доступа к объектам ОС определяется не только архитектурой операционной системы, но и текущей политикой безопасности. Под объектами доступа понимают, как ресурсы оборудования, так и программные ресурсы. В качестве примера ресурсов оборудования можно привести процессор, принтер, жесткие диски и ленты. Каждый объект имеет уникальное имя, отличающее его от других объектов в системе, и может быть доступен через хорошо определенные и значимые операции [2].

Методом доступа к объекту называется операция, определенная для объекта. Тип операции зависит от объектов. Например, процессор может только выполнять команды, сегменты памяти могут быть записаны и прочитаны, считыватель магнитных карт может только читать, а для файлов могут быть определены методы доступа «чтение», «запись» и «добавление» (дописывание информации в конец файла).

Субъектом доступа называют любую сущность, способную инициировать выполнение операций над объектами (обращаться к объектам по некоторым методам доступа). Обычно полагают, что множество субъектов доступа и множество объектов доступа не пересекаются.

Иногда к субъектам доступа относят процессы, выполняющиеся в системе. Однако логичнее считать субъектом доступа именно пользователя, от имени которого выполняется процесс. Естественно, под субъектом доступа подразумевают не

физического пользователя, работающего с компьютером, а «логического», от имени которого выполняются процессы операционной системы.

Таким образом, объект доступа – это то, к чему осуществляется доступ, субъект доступа – это тот, кто осуществляет доступ, и метод доступа – это то, как осуществляется доступ.

Для объекта доступа может быть определен **владелец** – субъект, которому принадлежит данный объект и который несет ответственность за конфиденциальность содержащейся в объекте информации, а также за целостность и доступность объекта.

Обычно владельцем объекта автоматически назначается субъект, создавший данный объект; в дальнейшем владелец объекта может быть изменен с использованием соответствующего метода доступа к объекту. На владельца, как правило, возлагается ответственность за корректное ограничение прав доступа к данному объекту других субъектов.

Правом доступа к объекту называют право на получение доступа к объекту по некоторому методу или группе методов. Например, если пользователь имеет возможность читать файл, говорят, что он имеет право на чтение этого файла. Говорят, что субъект имеет некоторую привилегию, если он имеет право на доступ по некоторому методу или группе методов ко всем объектам ОС, поддерживающим данный метод доступа.

Разграничением доступа субъектов к объектам является совокупность правил, определяющая для каждой тройки субъект–объект–метод, разрешен ли доступ данного субъекта к данному объекту по данному методу. При избирательном разграничении доступа возможность доступа определена однозначно для каждой тройки субъект–объект–метод, при полномочном разграничении доступа ситуация несколько сложнее.

Субъекта доступа называют **суперпользователем**, если он имеет возможность игнорировать правила разграничения доступа к объектам.

Правила разграничения доступа

Правила разграничения доступа, действующие в операционной системе, устанавливаются администраторами системы при определении текущей политики безопасности. За соблюдением этих правил субъектами доступа следит монитор ссылок – часть подсистемы защиты операционной системы [2].

Правила разграничения доступа должны удовлетворять следующим требованиям:

- Правила разграничения доступа, принятые в операционной системе, должны соответствовать аналогичным правилам, принятым в организации, в которой установлена эта ОС. Иными словами, если согласно правилам организации, доступ пользователя к некоторой информации считается несанкционированным, этот доступ должен быть ему запрещен.
- Правила разграничения доступа не должны допускать разрушающие воздействия субъектов доступа на ОС, выражающиеся в несанкционированном изменении, удалении или другом воздействии на объекты, жизненно важные для нормальной работы ОС.
- Любой объект доступа должен иметь владельца. Недопустимо присутствие ничейных объектов – объектов, не имеющих владельца.
- Недопустимо присутствие недоступных объектов – объектов, к которым не может обратиться ни один субъект доступа ни по одному методу доступа.
- Недопустима утечка конфиденциальной информации.

Основные модели разграничения доступа

Существуют две основные модели разграничения доступа:

- избирательное (дискреционное) разграничение доступа;
- полномочное (мандатное) разграничение доступа.

При **избирательном разграничении доступа** (Discretionary Access Control) определенные операции над конкретным ресурсом запрещаются или разрешаются субъектам или группам субъектов.

Большинство операционных систем реализуют именно избирательное разграничение доступа.

Полномочное разграничение доступа заключается в том, что все объекты могут иметь уровни секретности, а все субъекты делятся на группы, образующие иерархию в соответствии с уровнем допуска к информации. Иногда эту модель называют моделью многоуровневой безопасности, предназначенной для хранения секретов.

Избирательное разграничение доступа

Система правил избирательного разграничения доступа формулируется следующим образом.

- Для любого объекта операционной системы существует владелец.
- Владелец объекта может произвольно ограничивать доступ других субъектов к данному объекту.
- Для каждой тройки субъект–объект–метод возможность доступа определена однозначно.
- Существует хотя бы один привилегированный пользователь (администратор), имеющий возможность обратиться к любому объекту по любому методу доступа.

Этот привилегированный пользователь не может игнорировать разграничение доступа к объектам. Например, в ОС типа Windows администратор для обращения к чужому объекту (принадлежащему другому субъекту) должен вначале объявить себя владельцем этого объекта, используя привилегию администратора объявлять себя владельцем любого объекта, затем дать себе необходимые права, и только после этого администратор может обратиться к объекту. Последнее требование введено для реализации механизма удаления потенциально недоступных объектов.

При создании объекта его владельцем назначается субъект, создавший данный объект. В дальнейшем субъект, обладающий необходимыми правами, может назначить объекту нового владельца.

При этом субъект, изменяющий владельца объекта, может назначить новым владельцем объекта только себя. Такое ограничение вводится для того, чтобы владелец объекта не мог отдать владение объектом другому субъекту и тем самым снять с себя ответственность за некорректные действия с объектом.

Для определения прав доступа субъектов к объектам при избирательном разграничении доступа используются такие понятия, как матрица доступа и домен безопасности.

С концептуальной точки зрения текущее состояние прав доступа при избирательном разграничении доступа описывается матрицей, в строках которой перечислены субъекты доступа, в столбцах – объекты доступа, а в ячейках – операции, которые субъект может выполнить над объектом.

Домен безопасности определяет набор объектов и типов операций, которые могут производиться над каждым объектом операционной системы.

В ГОСТ Р ИСО/МЭК ТО 19791-2008 дано следующее определение: **Домен безопасности** (security domain) — часть автоматизированной системы, которая реализует одни и те же политики безопасности.

Домен – это просто набор ресурсов, доступных субъекту (субъект может быть пользователем, процессом или приложением). В рамках операционной системы, процесс имеет домен, который является набором системных ресурсов, доступных процессу для выполнения им своих задач. Этими ресурсами могут быть сегменты памяти, пространство на жестком диске, службы операционной системы и другие процессы. В сетевой среде,

домен является набором доступных физических и логических ресурсов, которыми могут быть маршрутизаторы, файловые серверы, службы FTP, веб-серверы и т.д.

Термин **домен безопасности** (security domain) основывается на определении домена, добавляя к нему факт, что ресурсы в рамках этой логической структуры (домена) работают с одной и той же политикой безопасности и управляются одной группой. Таким образом, администратор может поместить компьютеры, учетные записи и сетевые ресурсы сотрудников бухгалтерии в Домен 1, а компьютеры, учетные записи и сетевые ресурсы руководства в Домен 2. Все эти элементы попадут в эти два контейнера, поскольку они (элементы) не только выполняют однотипные задачи, но также, что более важно, имеют один и тот же уровень доверия. Общий уровень доверия позволяет управлять этими элементами одной (отдельной) политикой безопасности [1].

Возможность выполнять операции над объектом есть право доступа, представляющее собой упорядоченную пару <object-name, rights-set>. Таким образом, домен есть набор прав доступа. Например, если домен D имеет право доступа <file F, {read, write}>, это означает, что процесс, выполняемый в домене D, может читать или писать в файл F, но не может выполнять других операций над этим объектом.

Пример доменов показан в таблице на рис. 5.1.3.

Домен	Объект			
	F1	F2	F3	Printer
D1	read		execute	
D2		read		
D3				print
D4	read write		read write	

Рис. 5.1.3. Специфицирование прав доступа к ресурсам. Матрица доступа. F1, F2, F3 – файлы, Printer – принтер.

Связь конкретных субъектов, функционирующих в операционных системах, может быть организована следующим образом [2]:

- каждый пользователь может быть доменом. В этом случае набор объектов, к которым может быть организован доступ, зависит от идентификации пользователя;
- каждый процесс может быть доменом. В этом случае набор доступных объектов определяется идентификацией процесса;
- каждая процедура может быть доменом. В этом случае набор доступных объектов соответствует локальным переменным, определенным внутри процедуры. Заметим, что, когда процедура выполнена, происходит смена домена.

Рассмотрим стандартную двухрежимную модель выполнения ОС. Когда процесс выполняется в режиме ядра (Kernel Mode), он может вызывать привилегированные инструкции и иметь полный контроль над компьютерной системой. С другой стороны, если процесс выполняется в пользовательском режиме (User Domain), он может вызывать только непривилегированные инструкции. Следовательно, он может выполняться лишь внутри предопределенного пространства памяти. Наличие этих двух режимов позволяет защитить ОС (Kernel Domain) от пользовательских процессов (выполняющихся в режиме User Domain). В мультипрограммных системах двух доменов недостаточно, так как появляется необходимость защиты пользователей друг от друга.

В ОС UNIX домен безопасности связан с пользователем. Каждый пользователь обычно работает со своим набором объектов.

Модель безопасности, специфицированная выше (см. рис. 5.1.3), имеет вид матрицы и называется **матрицей доступа**. Столбцы этой матрицы представляют собой объекты, строки – субъекты. В каждой ячейке матрицы хранится совокупность прав доступа, предоставленных данному субъекту на данный объект. Поскольку реальная матрица доступа очень велика (типичный объем матрицы для современной операционной системы составляет несколько десятков мегабайтов), ее никогда не хранят в системе в явном виде. В общем случае эта матрица будет разреженной, то есть большинство ее клеток будут пустыми. Матрицу доступа можно разложить по столбцам, в результате чего получаются списки прав доступа ACL (Access Control List).

> Access Control List или ACL — **список контроля доступа**, который определяет, кто или что может получать доступ к конкретному объекту, и какие именно операции разрешено или запрещено этому субъекту проводить над объектом.

Списки контроля доступа являются основой систем с избирательным управлением доступа.

В результате разложения матрицы по строкам получаются мандаты возможностей (Capability List или Capability Tickets).

Список прав доступа ACL. Каждая колонка в матрице может быть реализована как список доступа для одного объекта. Очевидно, что пустые клетки могут не учитываться. В результате для каждого объекта имеем список упорядоченных пар <domain, rights-set>, который определяет все домены с непустыми наборами прав для данного объекта.

Элементами списка прав доступа ACL могут быть процессы, пользователи или группы пользователей. При реализации широко применяется предоставление доступа по умолчанию для пользователей, права которых не указаны. Например, в ОС UNIX все субъекты-пользователи разделены на три группы (владелец, группа и остальные) и для членов каждой группы контролируются операции чтения, записи и исполнения (rwx). В итоге имеем ACL – 9-битный код, который является атрибутом разнообразных объектов UNIX.

Мандаты возможностей. Как отмечалось выше, если матрицу доступа хранить по строкам, то есть если каждый субъект хранит список объектов и для каждого объекта – список допустимых операций, то такой способ хранения называется мандатами возможностей, или перечнями возможностей. Каждый пользователь обладает несколькими мандатами и может иметь право передавать их другим. Мандаты могут быть рассеяны по системе и вследствие этого представлять большую угрозу для безопасности, чем списки контроля доступа. Их хранение должно быть тщательно продумано. Примерами систем, использующих перечни возможностей, являются Hydra, Cambridge CAP System.

Избирательное разграничение доступа является наиболее распространенным способом разграничения доступа. Это обусловлено его сравнительной простотой реализации и необременительностью правил такого разграничения доступа для пользователей. Главное достоинство избирательного разграничения доступа – гибкость; основные недостатки – рассредоточенность управления и сложность централизованного контроля.

Вместе с тем защищенность операционной системы, подсистема защиты которой реализует только избирательное разграничение доступа, в некоторых случаях может оказаться недостаточной.

В частности, в США запрещено хранить информацию, содержащую государственную тайну, в компьютерных системах, поддерживающих только избирательное разграничение доступа.

Расширением модели избирательного разграничения доступа является изолированная (или замкнутая) программная среда.

Изолированная программная среда. При использовании изолированной программной среды права субъекта на доступ к объекту определяются не только правами и привилегиями субъекта, но и процессом, с помощью которого субъект обращается к объекту. Можно, например, разрешить обращаться к файлам с расширением .doc только программам Word, Word Viewer и WPview.

Система правил разграничения доступа для модели изолированной программной среды формулируется следующим образом:

- Для любого объекта операционной системы существует владелец.
- Владелец объекта может произвольно ограничивать доступ других субъектов к данному объекту.
- Для каждой четверки субъект–объект–метод–процесс возможность доступа определена однозначно.
- Существует хотя бы один привилегированный пользователь (администратор), имеющий возможность обратиться к любому объекту по любому методу.
- Для каждого субъекта определен список программ, которые этот субъект может запускать.

Изолированная программная среда существенно повышает защищенность операционной системы от разрушающих программных воздействий, включая программные закладки и компьютерные вирусы. Кроме того, при использовании данной модели повышается защищенность целостности данных, хранящихся в системе. В то же время изолированная программная среда создает определенные сложности в администрировании операционной системы. Например, при установке нового программного продукта администратор должен изменить списки разрешенных программ для пользователей, которые должны иметь возможность работать с этим программным продуктом. Изолированная программная среда не защищает от утечки конфиденциальной информации.

Полномочное разграничение доступа с контролем информационных потоков

Полномочное, или мандатное, разграничение доступа (Mandatory Access Control) обычно применяется в совокупности с избирательным разграничением доступа. Рассмотрим именно такой случай.

Правила разграничения доступа в данной модели формулируются следующим образом:

1. Для любого объекта операционной системы существует владелец.
2. Владелец объекта может произвольно ограничивать доступ других субъектов к данному объекту.
3. Для каждой четверки субъект–объект–метод–процесс возможность доступа определена однозначно в каждый момент времени. При изменении состояния процесса со временем возможность предоставления доступа также может измениться. Вместе с тем в каждый момент времени возможность доступа определена однозначно. Поскольку права процесса на доступ к объекту меняются с течением времени, они должны проверяться не только при открытии объекта, но и перед выполнением над объектом таких операций, как чтение и запись.
4. Существует хотя бы один привилегированный пользователь (администратор), имеющий возможность удалить любой объект.
5. В множестве объектов выделяется множество объектов полномочного разграничения доступа. Каждый объект полномочного разграничения доступа имеет гриф секретности. Чем выше числовое значение грифа секретности, тем секретнее объект. Нулевое значение грифа секретности означает, что объект несекретен. Если объект не является объектом полномочного разграничения

доступа или несекретен, администратор может обратиться к нему по любому методу, как и в предыдущей модели разграничения доступа.

6. Каждый субъект доступа имеет уровень допуска. Чем выше числовое значение уровня допуска, тем больший допуск имеет субъект. Нулевое значение уровня допуска означает, что субъект не имеет допуска. Обычно ненулевое значение допуска назначается только субъектам-пользователям и не назначается субъектам, от имени которых выполняются системные процессы.
7. Доступ субъекта к объекту должен быть запрещен независимо от состояния матрицы доступа, если:
    - объект является объектом полномочного разграничения доступа;
    - гриф секретности объекта строго выше уровня допуска субъекта, обращающегося к нему;
    - субъект открывает объект в режиме, допускающем чтение информации.

    Это правило называют правилом NRU (Not Read Up – не читать выше).
8. Каждый процесс операционной системы имеет уровень конфиденциальности, равный максимуму из грифов секретности объектов, открытых процессом на протяжении своего существования. Уровень конфиденциальности фактически представляет собой гриф секретности информации, хранящейся в оперативной памяти процесса.
9. Доступ субъекта к объекту должен быть запрещен независимо от состояния матрицы доступа, если:
    - объект является объектом полномочного разграничения доступа;
    - гриф секретности объекта строго ниже уровня конфиденциальности процесса, обращающегося к нему;
    - субъект собирается записывать в объект информацию.

    Это правило разграничения доступа предотвращает утечку секретной информации. Это так называемое правило NWD (Not Write Down – не записывать ниже).
10. Понизить гриф секретности объекта полномочного разграничения доступа может только субъект, который:
    - имеет доступ к объекту согласно правилу 7;
    - обладает специальной привилегией, позволяющей ему понижать грифы секретности объектов.

При использовании данной модели разграничения доступа существенно страдает производительность операционной системы, поскольку права доступа к объекту должны проверяться не только при открытии объекта, но и при каждой операции чтения/записи.

Кроме того, данная модель разграничения доступа создает пользователям определенные неудобства, связанные с тем, что если уровень конфиденциальности процесса строго выше нуля, то вся информация в памяти процесса фактически является секретной и не может быть записана в несекретный объект.

Если процесс одновременно работает с двумя объектами, только один из которых является секретным, процесс не может записывать информацию из памяти во второй объект. Эта проблема решается посредством использования специального программного интерфейса API для работы с памятью. Области памяти, выделяемые процессам, могут быть описаны как объекты полномочного разграничения доступа, после чего им могут назначаться грифы секретности.

При чтении секретного файла процесс должен считать содержимое такого файла в секретную область памяти, используя для этого функции операционной системы, гарантирующие невозможность утечки информации. Для работы с секретной областью памяти процесс также должен использовать специальные функции. Поскольку утечка

информации из секретных областей памяти в память процесса невозможна, считывание процессом секретной информации в секретные области памяти не отражается на уровне конфиденциальности процесса. Если же процесс считывает секретную информацию в область памяти, не описанную как объект полномочного разграничения доступа, повышается уровень конфиденциальности процесса.

Из вышеизложенного следует, что пользователи операционных систем, реализующих данную модель разграничения доступа, вынуждены использовать программное обеспечение, разработанное с учетом этой модели. В противном случае пользователи будут испытывать серьезные проблемы в процессе работы с объектами операционной системы, имеющими ненулевой гриф секретности.

Определенные проблемы вызывает также вопрос о назначении грифов секретности создаваемым объектам. Если пользователь создает новый объект с помощью процесса, имеющего ненулевой уровень конфиденциальности, пользователь вынужден присвоить новому объекту гриф секретности не ниже уровня конфиденциальности процесса. Во многих ситуациях это неудобно.

Сравнительный анализ моделей разграничения доступа

Каждая из рассмотренных моделей разграничения доступа имеет свои достоинства и недостатки. Таблица на рис 5.1.4 позволяет провести их сравнительный анализ.

Свойства модели	Избирательное разграничение доступа	Изолированная программная среда	Полномочное разграничение доступа с контролем потоков
Защита от утечки информации	Отсутствует	Отсутствует	Имеется
Защищенность от разрушающих воздействий	Низкая	Высокая	Низкая
Сложность реализации	Низкая	Средняя	Высокая
Сложность администрирования	Низкая	Средняя	Высокая
Затраты ресурсов компьютера	Низкие	Низкие	Высокие
Использование ПО, разработанного для других систем	Возможно	Возможно	Проблематично

Рис.5.1.4. Модели разграничения доступа

Как видно из рис. 5.1.4, в большинстве ситуаций применение избирательного разграничения доступа наиболее эффективно.

Изолированную программную среду целесообразно использовать в случаях, когда важно обеспечить целостность программ и данных операционной системы. Полномочное разграничение доступа с контролем информационных потоков следует применять в тех случаях, когда для организации чрезвычайно важно обеспечение защищенности системы от несанкционированной утечки информации. В остальных ситуациях применение этой модели нецелесообразно из-за резкого ухудшения эксплуатационных качеств операционной системы.

Список использованных источников

1. Таненбаум, Э. Современные операционные системы. / Э. Таненбаум, Х. Бос. — 4-е изд. — СПб.: Питер, 2015. — 1120 с.

2. Принципы построения защищенной операционной системы
https://yztm.ru/lekc/l2/

3. Что такое криптография?
https://aws.amazon.com/ru/what-is/cryptography/