Проскурин В. Г.

ЗАЩИТА В ОПЕРАЦИОННЫХ СИСТЕМАХ

Рекомендовано федеральным казенным государственным образовательным учреждением высшего профессионального образования «Академия Федеральной службы безопасности Российский Федерации» в качестве учебного пособия для студентов (слушателей) высших учебных заведений, обучающихся по специальностям 10.05.01 – «Компьютерная безопасность», 10.05.03 – «Информационная безопасность автоматизированных систем» и 10.05.04 – «Информационно-аналитические системы безопасности», по направлению подготовки 10.03.01 – «Информационная безопасность», уровень бакалавр

Москва
Горячая линия – Телеком
2014

УДК 004.732.056(075.8)
ББК 32.973.2-018.2я73
П82

Проскурин В. Г.

П82 Защита в операционных системах. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2014. – 192 с.: ил.

ISBN 978-5-9912-0379-1.

Подробно рассмотрены основные средства и методы обеспечения информационной безопасности в современных операционных системах: управление доступом, аутентификация, аудит и обнаружение вторжений. Кроме того, отдельно рассматриваются некоторые специфические вопросы, косвенно связанные с обеспечением безопасности операционных систем: централизованное управление политиками безопасности в доменах Windows, особенности обеспечения безопасности операционных систем мобильных устройств, концепция виртуализации операционных систем и ее влияние на информационную безопасность. Изложение теоретического материала иллюстрируется практическими примерами. В конце каждой главы приведен перечень вопросов для самопроверки, в конце пособия – методические рекомендации по его изучению.

Для студентов (слушателей) вузов, обучающихся по специальностям 10.05.01 – «Компьютерная безопасность», 10.05.03 – «Информационная безопасность автоматизированных систем» и 10.05.04 – «Информационно-аналитические системы безопасности», по направлению подготовки 10.03.01 – «Информационная безопасность», уровень бакалавр.

ББК 32.973.2-018.2я73

Адрес издательства в Интернет WWW.TECHBOOK.RU

Учебное издание

Проскурин Вадим Геннадьевич

ЗАЩИТА В ОПЕРАЦИОННЫХ СИСТЕМАХ

Учебное пособие

Редактор Ю. Н. Чернышов
Компьютерная верстка Ю. Н. Чернышова
Обложка художника О. В. Карповой

Подписано в печать 25.12.2013. Формат 60×88/16. Уч. изд. л. 12. Тираж 500 экз.
ООО «Научно-техническое издательство «Горячая линия – Телеком»

ISBN 978-5-9912-0379-1

Предисловие

В настоящее время дисциплина «Защита в операционных системах», предусмотренная федеральным стандартом высшего профессионального образования по специальности **090301 «Компьютерная безопасность»** (квалификация специалист), обеспечена учебно-методической литературой в явно недостаточной мере. Единственное учебное пособие по данной дисциплине [10], покрывающее примерную учебную программу дисциплины более чем на 20 %, было издано в 2000 году и к настоящему времени сильно устарело.

Предлагаемое учебное пособие призвано заполнить данный пробел в методическом обеспечении специальности 10.05.01 — «Компьютерная безопасность», а также смежных с ней специальностей. Пособие построено на основе 17-летнего опыта преподавания дисциплины «Защита в операционных системах» в ИКСИ. Согласно учебному плану, на изучение дисциплины «Защита в операционных системах» отводится восьмой семестр, в конце семестра студенты сдают экзамен. В рамках основного лекционного курса изучаются пять основных тем, каждой из которых соответствует одна глава пособия. Кроме того, в пособие включены две главы, описывающие два сравнительно новых перспективных направления в деле обеспечения безопасности операционных систем. Соответствующие темы предполагается включить в программу дисциплины «Защита в операционных системах» при следующей плановой переработке учебно-методического комплекса.

Помимо вышеупомянутой дисциплины, предлагаемое пособие может использоваться для преподавания следующих дисциплин:

- «Безопасность операционных систем» специальностей 10.05.03 — «Информационная безопасность автоматизированных систем» и 10.05.04 — «Информационно-аналитические системы безопасности»;
- «Программно-аппаратные средства защиты обеспечения информационной безопасности» для специальности 10.05.02 — «Информационная безопасность телекоммуникационных систем»;
- «Программно-аппаратные средства защиты информации» для бакалавров направления подготовки 10.03.01 — «Информационная безопасность».

1 Понятие защищенной операционной системы

1.1. Основные определения

Мы будем называть операционную систему защищенной, если она предусматривает средства защиты от основных угроз конфиденциальности, целостности и доступности информации, актуализированных с учетом особенностей эксплуатации данного конкретного экземпляра операционной системы. В практически значимых ситуациях защищенная операционная система обычно содержит средства управления доступом пользователей к различным ресурсам, средства проверки подлинности пользователя, начинающего работу с операционной системой, а также средства регистрации действий пользователей, потенциально опасных с точки зрения безопасности. Кроме того, защищенная операционная система должна содержать средства противодействия случайному или преднамеренному выводу операционной системы из строя.

Мы будем называть политикой безопасности набор норм, правил и практических приемов, регламентирующих порядок хранения и обработки ценной информации. В применении к операционной системе политика безопасности определяет то, какие пользователи могут работать с операционной системой, какие пользователи имеют доступ к каким объектам операционной системы, какие события должны регистрироваться в системных журналах и т. д.

Адекватной политикой безопасности мы будем называть такую политику безопасности, которая обеспечивает достаточный уровень защищенности операционной системы. Следует особо отметить, что адекватная политика безопасности не обязательно является той политикой безопасности, при которой достигается максимально возможная защищенность системы.

1.2. Основные подходы к построению защищенных операционных систем

Существуют два основных подхода к созданию защищенных операционных систем — фрагментарный и комплексный. При фрагментарном подходе вначале организуется защита от одной угрозы, затем от другой и т. д. Примером фрагментарного подхода может

служить ситуация, когда за основу берется незащищенная операционная система, на нее устанавливаются антивирусный пакет, затем система шифрования, система регистрации действий пользователей и т. д.

Основной недостаток фрагментарного подхода очевиден — при применении этого подхода подсистема защиты операционной системы представляет собой набор разрозненных программных продуктов, как правило, произведенных разными производителями. Эти программные средства работают независимо друг от друга, организовать их тесное взаимодействие практически невозможно. Кроме того, отдельные элементы такой подсистемы защиты могут некорректно работать в присутствии друг друга, что приводит к резкому снижению общей надежности системы. Поскольку подсистема защиты, созданная на основе фрагментарного подхода, не является неотъемлемой компонентой операционной системы, при отключении отдельных защитных функций в результате несанкционированных действий пользователя-нарушителя остальные элементы операционной системы продолжают нормально работать, что еще сильнее снижает надежность защиты.

При комплексном подходе к организации защиты защитные функции вносятся в операционную систему еще 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. Поскольку вся подсистема защиты разрабатывается и тестируется в совокупности, 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 подсистемы защиты она вызывает аварийное завершение работы операционной системы, что не позволяет нарушителю отключать защитные функции системы. При использовании фрагментарного подхода такая организация подсистемы защиты невозможна.

Как правило, подсистему защиты операционной системы, созданную на основе комплексного подхода, проектируют так, что отдельные ее элементы являются заменяемыми и соответствующие программные модули могут быть заменены другими модулями, реализующими предусмотренный и должным образом документированный интерфейс взаимодействия соответствующего программного модуля с другими элементами подсистемы защиты.

1.3. Административные меры защиты

Организация эффективной и надежной защиты операционной системы невозможна с помощью одних только программно- аппаратных средств. Эти средства обязательно должны дополняться административными мерами защиты. Без постоянной квалифицированной поддержки со стороны администратора даже самая надежная программно-аппаратная защита оборачивается фикцией.

К основным административным мерам защиты относятся следующие:

- постоянный контроль корректности функционирования операционной системы и, в особенности, ее подсистемы защиты. При этом могут и должны использоваться средства аудита, встроенные в операционную систему, и, при необходимости, дополнительные средства аудита;
- организация и поддержание адекватной политики безопасности. Политика безопасности должна постоянно корректироваться, оперативно реагируя на изменения в конфигурации операционной системы, установку и удаление и изменение конфигурации прикладных программных продуктов и расширений операционной системы, попытки злоумышленников преодолеть защиту операционной системы и т. д.;
- инструктирование пользователей операционной системы о необходимости соблюдения мер безопасности при работе с операционной системой, контроль над соблюдением пользователями этих мер;
- регулярное создание и обновление резервных копий программ и данных операционной системы;
- постоянный контроль изменений в конфигурационных данных и политике безопасности операционной системы. Информацию об этих изменениях часто дублируют на неэлектронные носители информации, чтобы нарушителю, преодолевшему защиту операционной системы, было труднее замаскировать свои несанкционированные действия.

В конкретных конфигурациях операционных систем могут потребоваться и другие административные меры защиты информации.

1.4. Адекватная политика безопасности

Задача выбора и поддержания адекватной политики безопасности является важнейшей и одной из сложнейших задач, стоящих перед администратором операционной системы. Если принятая в операционной системе политика безопасности неадекватна, это может

приводить к фактам несанкционированного доступа пользователя-нарушителя к ресурсам защищаемой системы, а также к снижению надежности ее функционирования.

Не всякая адекватная политика безопасности применима на практике. В общем случае верно следующее утверждение: чем лучше операционная система защищена, тем труднее с ней работать пользователям и администраторам. Это обусловлено следующими факторами.

1. Система защиты, не обладающая интеллектом, не всегда способна определить, является ли некоторое действие пользователя злонамеренным. Поэтому система защиты либо не пресекает некоторые виды 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. Если, например, некоторому пользователю запрещено создавать файлы на жестком диске компьютера, то этот пользователь не сможет запустить ни одну программу, для нормального функционирования которой требуется создавать временные файлы. С точки зрения рассматриваемой политики безопасности создание временного файла является несанкционированным действием и в том, что оно пресекается, нет ошибки. Просто в данной политике безопасности класс несанкционированных действий настолько широк, что это препятствует нормальной работе пользователей с операционной системой.

2. 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.

3. Подсистема защиты операционной системы, как и любой другой программный пакет, потребляет аппаратные ресурсы компьютера. Чем сложнее устроены защитные функции операционной системы, тем больше процессорного времени, оперативной памяти и других аппаратных ресурсов затрачивается на поддержание функционирования подсистемы защиты и тем меньше ресурсов остается на долю прикладных программ. В отдельных случаях подсистема защиты операционной системы может потреблять более половины аппаратных ресурсов компьютера.

4. Поддержание слишком жесткой политики безопасности может негативно сказаться на надежности функционирования опера-

ционной системы. Если, например, в Windows запретить псевдопользователю SYSTEM, от имени которого выполняются системные процессы, доступ к исполняемым файлам системных процессов, операционная система не сможет загрузиться. В данном случае чрезмерно жесткая политика безопасности приводит к моментальному краху операционной системы, в других случаях подобная политика безопасности может приводить к трудновыявляемым ошибкам и сбоям в процессе функционирования операционной системы, что еще более опасно.

Таким образом, при определении адекватной политики безопасности не следует пытаться достигнуть максимально возможного уровня защищенности операционной системы. Оптимальная адекватная политика безопасности — такая политика безопасности, которая не только не позволяет нарушителям выполнять несанкционированные действия, но и не приводит к вышеописанным негативным последствиям.

Не существует единой адекватной политики безопасности на все случаи жизни. То, какая политика безопасности будет адекватной, определяется не только архитектурой операционной системы, но и ее конфигурацией, ассортиментом установленного программного обеспечения и т. д. Политика безопасности, адекватная для некоторой операционной системы, скорее всего, будет неадекватна для другого экземпляра той же операционной системы. Большинство современных операционных систем достаточно универсальны и могут применяться для решения самых разных задач. Одна и та же операционная система может использоваться для обеспечения функционирования автоматизированной банковской системы, веб-сервера, системы электронного документооборота. Очевидно, что угрозы безопасности для всех трех указанных применений будут различны и адекватная политика безопасности в каждом из трех случаев будет своя.

Формирование и поддержание адекватной политики безопасности операционной системы в общем случае можно разделить на следующие этапы.

1. **Анализ угроз.** Администратор операционной системы рассматривает возможные угрозы безопасности данного экземпляра операционной системы. Среди возможных угроз выделяются наиболее опасные, защите от которых нужно уделять максимум сил и средств.

2. **Формирование требований к политике безопасности.** На этом этапе администратор определяет, какие средства и методы будут применяться для защиты от тех или иных угроз. Например,

защиту от несанкционированного доступа к некоторому объекту операционной системы можно решать либо средствами разграничения доступа, либо криптографическими средствами, либо используя некоторую комбинацию этих средств, либо используя какие-то иные средства. На данном этапе администратор должен сделать подобный выбор для каждой угрозы безопасности операционной системы, выбирая оптимальные средства защиты от каждой угрозы. Одновременно администратор анализирует возможные побочные эффекты различных вариантов политики безопасности, оценивая, в какой мере в каждом варианте политики безопасности будут проявляться побочные негативные факторы. Как правило, администратору приходится идти на компромисс, смиряясь либо с недостаточной защищенностью операционной системы от отдельных угроз, либо с определенными трудностями пользователей при работе с системой. Результатом данного этапа является набор требований наподобие: «В операционной системе должно быть предусмотрено дискреционное разграничение доступа с минимизацией полномочий пользователей и частичной реализацией правил изолированной программной среды».

3. **Формальное определение политики безопасности.** Данный этап заключается в том, что администратор четко определяет, как конкретно должны выполняться требования, сформулированные на предыдущем этапе. Администратор решает, можно ли добиться выполнения этих требований только встроенными средствами операционной системы или необходима установка дополнительных пакетов защиты. В последнем случае производится выбор необходимого программного обеспечения. На данном этапе формулируются необходимые требования к конфигурации операционной системы, а также требования к конфигурации дополнительных пакетов защиты, если установка таких пакетов необходима. Кроме того, на данном этапе администратор должен предусмотреть порядок внесения необходимых изменений в политику безопасности в чрезвычайных ситуациях, например, при обнаружении факта несанкционированного входа в систему пользователя-нарушителя. Результатом данного этапа является развернутый перечень настроек конфигурации операционной системы и дополнительных пакетов защиты с указанием того, в каких ситуациях какие настройки должны быть выставлены.

4. **Претворение в жизнь политики безопасности.** К началу этого этапа у администратора операционной системы имеется четкое представление о том, какой должна быть адекватная политика безопасности. Задачей этапа является приведение конфигу-

рации операционной системы и дополнительных пакетов защиты в соответствие с политикой безопасности, формально определенной на предыдущем этапе.

5. **Поддержание и коррекция политики безопасности.** На данном этапе операционная система функционирует в соответствии с политикой безопасности, определенной на третьем этапе. Задачей администратора является контроль над соблюдением политики безопасности и внесение в нее необходимых изменений по мере появления изменений в функционировании операционной системы. Например, если в операционной системе устанавливается новый программный продукт, может потребоваться коррекция политика безопасности таким образом, чтобы этот программный продукт мог нормально функционировать.

1.5. Стандарты безопасности операционных систем

Анализ угроз, с которого начинается формирование политики безопасности, является весьма трудоемкой и трудноформализуемой процедурой. Как правило, угрозы, от которых предполагается защищать компьютерную систему или сеть, очень разнородны, сравнивать их между собой и выделять среди них наиболее опасные обычно крайне затруднительно. Иногда эту проблему пытаются решать путем количественного выражения элементарных рисков в некоторых условных единицах с использованием формул наподобие:

Риск = (стоимость ресурса * вероятность угрозы) / величина уязвимости

Практическая реализация данного подхода в конкретных операционных системах сталкивается с рядом трудностей. Наиболее серьезная проблема количественного анализа рисков заключается в том, что для исходных числовых данных, используемых в количественном анализе рисков, часто затруднительно обосновать погрешность присвоения тем или иным качественным характеристикам конкретных числовых значений. Например, погрешность оценки вероятности угрозы может быть корректно вычислена только для тех угроз, которые реализуются регулярно. Если же попытки реализации некоторой угрозы в исследуемых операционных системах ни разу не регистрировались, вероятность данной угрозы, как правило, можно оценить лишь с точностью до порядка. Соответственно, риск данной угрозы тоже может быть вычислен лишь с точностью до порядка. Это может поставить под сомнение обоснованность окончательных выводов, сделанных в результате проведенного анализа.

К сожалению, часто наблюдаются ситуации, когда эксперты, выполняющие анализ рисков той или иной операционной системы, совсем не уделяют внимания оценкам погрешностей используемых количественных показателей, в результате чего анализ рисков вырождается в «жонглирование цифрами», позволяющее получить любой желаемый результат, подобрав исходные данные соответствующим образом.

Альтернативный подход к управлению безопасности основан на том, чтобы привести политику безопасности защищаемой системы в соответствие с тем или иным набором стандартов безопасности или иных нормативных документов. Основным преимуществом такого подхода является то, что он позволяет существенно сократить затраты на разработку политики безопасности, сведя к минимуму анализ рисков для защищаемой системы. Фактически, анализ рисков в данном случае сводится к обоснованию выбора стандартов безопасности, используемых в качестве основы для планирования политики безопасности операционной системы.

Если защищаемая операционная система имеет типовую конфигурацию и хорошо описывается существующими стандартами безопасности, применение данного подхода к управлению безопасностью является вполне оправданным. С другой стороны, известны случаи бездумного применения к управлению безопасностью конкретных систем заведомо неподходящих стандартов безопасности.

В последнее время заметной становится тенденция к сближению двух рассмотренных подходов. Стандарты информационной безопасности становятся все более подробными и детализированными, с каждым годом все больше практически значимых ситуаций оказываются описанными в тех или иных стандартах. Большое влияние на роль и место стандартов безопасности в процессе управления безопасностью конкретных компьютерных систем оказал состоявшийся в последнее десятилетие переход от линейных шкал классов защищенности к более гибкой системе профилей защиты. Если раньше стандарты информационной безопасности позволяли удовлетворительно описать требования к безопасности лишь для наиболее типовых конфигураций операционных систем, то теперь профиль защиты может быть построен для практически любой конфигурации операционной системы, исключая лишь самые экзотические. При этом для конкретной операционной системы выбор и обоснование выбора профиля защиты выполняется путем анализа рисков, соответствующих тем или иным профилям защиты.

В 2002–2003 годах Гостехкомиссией России на основе международного стандарта ISO 15408 был разработан комплект профилей защиты [2], специально предназначенных для описания требований к безопасности операционных систем. В качестве примера кратко рассмотрим один из этих профилей защиты, а именно «Операционные системы. Базовый профиль защиты» [1].

В данном профиле защиты перечислены основные функциональные требования, предъявляемые к безопасности операционных систем, сгруппированные следующим образом.

Аудит безопасности:

- автоматическая генерация средствами аудита предупреждений для уполномоченного администратора при обнаружении возможного нарушения безопасности;
- возможность генерации записей аудита для определенных классов событий (в профиле защиты приведены 44 примера таких классов событий);
- возможность однозначного ассоциирования каждого события, потенциально подвергаемого аудиту, с идентификатором пользователя, послужившего инициатором данного события;
- возможность просмотра уполномоченными администраторами всех накопленных данных аудита с использованием специального средства доступа к журналу аудита;
- запрет доступа неуполномоченных пользователей к накопленным данным аудита;
- возможность выборочного просмотра накопленных данных аудита с использованием определенных критериев поиска и сортировки (приведены четыре примера таких критериев);
- возможность включения событий, потенциально подвергаемых аудиту, в совокупность событий, подвергающихся аудиту, или их исключения из этой совокупности по определенным критериям (приведены четыре примера таких критериев);
- защита накопленных данных аудита от несанкционированной модификации или удаления;
- предотвращение потери накопленных данных аудита при переполнении журнала аудита.

Защита данных пользователя:

- дискреционное управление доступом для заданного множества субъектов, объектов и операций над ними;
- использование уникальных идентификаторов пользователей и групп пользователей при описании политики управления доступом;

- наличие правил управления доступа, заданных по умолчанию и применяемых, когда атрибуты безопасности субъекта не соответствуют атрибутам безопасности объекта;
- особый порядок управления доступом уполномоченных администраторов к объектам операционной системы;
- особые дополнительные правила для явного отказа в доступе при выполнении определенных условий;
- недоступность предыдущего информационного содержания ресурса при перераспределении или освобождении ресурса.

Идентификация и аутентификация:

- ограничение максимально возможного количества неуспешных попыток аутентификации в одном подключении к операционной системе;
- поддержка заданного набора атрибутов безопасности пользователя (приведены четыре примера таких атрибутов);
- требования к стойкости механизмов верификации секретов к подбору аутентификационной информации;
- явное ограничение списка действий, возможных для неидентифицированного или неаутентифицированного пользователя;
- недопустимость открытой обратной связи в ходе аутентификации (например, выдачи вводимого пароля на экран компьютера в незамаскированном виде);
- возможность однозначного ассоциирования каждого процесса операционной системы с атрибутами безопасности пользователя (приведены пять примеров таких атрибутов).

Управление безопасностью:

- разрешение управления подсистемой безопасности операционной системы только уполномоченным администраторам;
- разрешение управления дискреционной политикой безопасности только уполномоченным администраторам, а также владельцам соответствующих объектов;
- наличие явно заданных значений по умолчанию для атрибутов безопасности вновь создаваемых объектов;
- наличие явно заданных правил управления доступом уполномоченных администраторов к атрибутам учетных данных пользователей, в том числе и к аутентификационным данным пользователей;
- наличие механизмов, позволяющих уполномоченным администраторам назначать аутентификационным данным пользователей сроки действия, а также блокировать учетные записи пользователей.

Защита функций безопасности:

- возможность самотестирования подсистемы безопасности операционной системы при запуске, периодически во время работы или по запросу уполномоченного администратора;
- обеспечение конфиденциальности и целостности внутренних данных подсистем безопасности операционных систем в процессе их сетевого взаимодействия;
- наличие средств восстановления подсистемы безопасности операционной системы после сбоев и аварий;
- невозможность взаимодействия пользователей и приложений с операционной системой в обход функций подсистемы безопасности;
- средства изоляции подсистемы безопасности операционной системы, не допускающие искажения ее кода и данных недоверенными субъектами доступа;
- реализация надежных меток времени, которые не могут быть искажены недоверенными субъектами доступа;
- согласованное и надежное взаимодействие подсистем защиты операционных систем при взаимодействии операционных систем по сети.

Использование ресурсов:

- наличие механизма квотирования оперативной памяти, процессорного времени и, возможно, других аппаратных ресурсов.

Доступ к операционной системе:

- возможность ручного или автоматического (по истечении заданного интервала времени, в течение которого пользователь бездействует) блокирования сеанса взаимодействия пользователя с операционной системой, требование повторной аутентификации пользователя в ходе разблокирования сеанса;
- возможность автоматической выдачи предупреждающего сообщения пользователю, начинающему сеанс работы с операционной системой;
- возможность получения пользователем информации о своих последних попытках работы с операционной системой, в том числе и о неудачных попытках.

Доверенный маршрут/канал:

- канал связи удаленного пользователя с операционной системой должен быть защищен от раскрытия и модификации передаваемых данных, как минимум, до завершения аутентификации пользователя.

Рассматриваемый профиль защиты (как и любые другие профили защиты) не следует рассматривать как перечень требований, которые должны неукоснительно соблюдаться в любой защищенной операционной системе. Для каждой конкретной конфигурации операционной системы формулируется свой индивидуальный профиль защиты, учитывающий индивидуальные особенности той или иной системы. При этом базовый профиль может использоваться в качестве шаблона, чтобы администратору безопасности было легче сформулировать необходимые требования к конфигурации защищаемой системы.

Вопросы для самопроверки

1. Что называется защищенной операционной системой?
2. Какие подходы к построению защищенных операционных систем вы знаете?
3. Какие административные меры защиты вы знаете?
4. Какую политика безопасности называют адекватной?
5. Почему неограниченный рост защищенности операционной системы неизбежно приводит к снижению ее эксплуатационных качеств?
6. К каким негативным последствиям может привести поддержание чрезмерно высокого уровня защищенности системы?
7. Каковы основные этапы процесса формирования и поддержания адекватной политики безопасности операционной системы?
8. Когда заканчивается поддержание и коррекция адекватной политики безопасности?
9. С какими проблемами сталкиваются попытки количественного анализа рисков для тех или иных операционных систем?
10. Каковы роль и место стандартов безопасности в деле управления безопасностью операционной системы?
11. Какие преимущества дает применение профилей защиты в стандартах безопасности компьютерных систем?
12. Какие функциональные требования предъявляются к безопасности операционных систем согласно базовому профилю защиты стандарта ISO/МЭК 15408 в части, касающейся защиты данных пользователя?
13. Какие функциональные требования предъявляются к безопасности операционных систем согласно базовому профилю защиты стандарта ISO/МЭК 15408 в части, касающейся идентификации и аутентификации?
14. Какие функциональные требования предъявляются к безопасности операционных систем согласно базовому профилю защиты стандарта ISO/МЭК 15408 в части, касающейся защиты функций безопасности?
15. Какие функциональные требования предъявляются к безопасности операционных систем согласно базовому профилю защиты стандарта ISO/МЭК 15408 в части, касающейся доступа к операционной системе?

2 Управление доступом

2.1. Основные определения

Объектом доступа (или просто *объектом*) мы будем называть любой элемент операционной системы, доступ к которому пользователей и других субъектов доступа может быть произвольно ограничен. Ключевым словом в данном определении является слово «произвольно». Если правила, ограничивающие доступ субъектов к некоторому элементу операционной системы, определены жестко и не допускают изменения с течением времени, этот элемент операционной системы мы не будем считать объектом. Другими словами, возможность доступа к объектам операционной системы определяется не только архитектурой операционной системы, но и текущей политикой безопасности.

Методом доступа к объекту называется операция, определенная для некоторого объекта. Например, для файлов могут быть определены методы доступа «чтение», «запись» и «добавление» (дописывание информации в конец файла).

Субъектом доступа (или просто *субъектом*) мы будем называть любую сущность, способную инициировать выполнение операций над объектами (обращаться к объектам по некоторым методам доступа). Например, пользователи являются субъектами доступа.

Обычно к субъектам доступа относят не только пользователей, работающих в системе, но и порожденные ими процессы. Данный подход является оправданным и, более того, единственно верным, во всех случаях, когда в область рассмотрения включаются программные закладки, функционирующие автономно и преследующие свои собственные задачи, не совпадающие с целями пользователя, работающего в системе. Однако в данном пособии мы будем (за редкими исключениями) рассматривать только «чистые» операционные системы, не зараженные вредоносным программным обеспечением. Программные закладки и их взаимодействие с атакованными операционными системами подробно рассматриваются в дисциплине «Защита программ и данных» [11].

Таким образом, в данном учебном пособии разделе везде, где явно не оговорено противное, субъектом доступа мы будем считать не процесс (или поток процесса-сервера), выполняющий некоторую

операцию, а пользователя, от имени которого этот процесс (или поток) выполняется.

Итак, 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, несущий ответственность за конфиденциальность содержащейся в объекте информации (если эта информация конфиденциальна), 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. Владелец объекта не может быть лишен некоторых прав на доступ к этому объекту, на владельца, как правило, возлагается ответственность за корректное ограничение прав доступа к данному объекту других субъектов.

Правом доступа к объекту мы будем называть право на выполнение доступа к объекту по некоторому методу или группе методов. В последнем случае право доступа дает субъекту возможность осуществлять доступ к объекту по любому методу из данной группы. Говорят, что субъект имеет некоторое право на доступ к объекту (или «субъект имеет право доступа к объекту» или «субъект имеет право на объект»), если он имеет возможность осуществлять доступ к объекту по соответствующему методу или группе методов. Например, если пользователь имеет возможность читать файл, говорят, что он имеет право на чтение этого файла.

Понятие метода доступа и понятие права доступа не идентичны. Например, в операционных системах семейства UNIX право на запись в файл дает возможность субъекту обращаться к файлу как по методу «запись», так и по методу «добавление», при этом, поскольку право доступа «добавление» в UNIX отсутствует, невозможно разрешить субъекту операцию добавления, одновременно запретив операцию записи.

Говорят, что субъект имеет некоторую *привилегию*, если он имеет возможность выполнять в операционной системе некоторые действия, не выражаемые или трудновыражаемые в терминах доступа субъекта к объектам. Например, в операционной системе Windows поддерживаются привилегии перезагружать компьютер и перенастраивать часы компьютера. Как частный случай, привилегией является возможность применения некоторого права доступа или группы прав доступа ко без исключения объектам операцион-

ной системы, поддерживающим соответствующие методы доступа. Например, если субъект операционной системы Windows имеет привилегию отладки, он имеет право доступа ко всем объектам типа «процесс» и «поток» по группе методов, используемых отладчиками при отладке программ (фактически, по всем поддерживаемым операционной системой методам доступа).

Полномочиями субъекта доступа называется совокупность всех предоставленных ему прав и привилегий.

Управлением доступом субъектов к объектам называется совокупность правил, определяющая для каждой тройки субъект–объект–право, разрешена ли реализация данного права данным субъектом в отношении данного объекта. При дискреционном управлении доступом возможность доступа определяется для каждой тройки субъект–объект–право априорно, при мандатном управлении доступом ситуация несколько сложнее.

Мы будем называть субъекта доступа *суперпользователем*, если он имеет возможность игнорировать правила управления доступом к объектам.

Правила управления доступом, действующие в защищаемой компьютерной системе, устанавливаются администраторами системы при определении текущей политики безопасности. За соблюдением этих правил субъектами доступа следит *монитор ссылок* или *монитор безопасности объектов* — часть подсистемы защиты компьютерной системы.

Правила управления доступом должны удовлетворять следующим очевидным требованиям.

- Правила управления доступом, принятые в компьютерной системе, должны соответствовать аналогичным правилам, принятым в организации, в которой эксплуатируется данная система. Другими словами, если, согласно правилам организации, доступ пользователя к некоторой информации считается несанкционированным, то в операционной системе этот доступ тоже должен быть ему запрещен. Под несанкционированным доступом здесь подразумевается не только несанкционированное чтение информации, но и несанкционированное изменение, копирование или уничтожение информации.
- Правила управления доступа должны не допускать (или, по крайней мере, затруднять) разрушающие воздействия субъектов доступа, не обладающих соответствующими полномочиями, на операционную систему, выражающиеся в несанкционированном изменении, удалении или другом воздействии на объекты,

критически важные для обеспечения нормального функционирования системы.

- Любой объект системы должен иметь владельца. Присутствие в системе *ничейных объектов* — объектов, не имеющих владельца, должно быть недопустимо.
- Присутствие в системе *недоступных объектов* — объектов, к которым не может обратиться ни один субъект доступа ни по одному методу доступа, должно быть недопустимо. Недоступные объекты фактически бесполезно растрачивают аппаратные ресурсы компьютера.
- Утечка конфиденциальной информации из защищаемой системы должна быть недопустима. Поскольку реализовать выполнение данного требования программно-аппаратными средствами весьма сложно, оно предъявляется лишь в редких случаях. Как правило, предотвращение утечки конфиденциальной информации из защищаемой системы обеспечивается одними только организационными мерами.

2.2. Типовые модели управления доступом

2.2.1. Дискреционное управление доступом

Система правил *дискреционного*, или *избирательного* управления доступом (*discretionary access control*) формулируется следующим образом:

1. Для любого объекта системы существует владелец.

2. Владелец объекта может произвольно ограничивать доступ других субъектов к данному объекту.

3. Для каждой тройки субъект–объект–право возможность доступа определена однозначно.

4. Существует хотя бы один привилегированный пользователь (администратор), имеющий возможность обратиться к любому объекту по любому методу доступа. Это не означает, что этот пользователь может игнорировать разграничение доступа к объектам и поэтому является суперпользователем. Не всегда для реализации возможности доступа к объекту операционной системы администратору достаточно просто обратиться к объекту. Например, в Windows администратор для обращения к чужому (принадлежащему другому субъекту) объекту должен вначале объявить себя владельцем этого объекта, использовав привилегию администратора объявлять себя владельцем любого объекта, затем дать себе необходимые права, и только после этого администратор может обратиться к объекту.

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

Для определения прав доступа субъектов к объектам при избирательном разграничении доступа используется *матрица доступа*. Строки этой матрицы представляют собой объекты, столбцы — субъекты (или наоборот). В каждой ячейке матрицы доступа хранится совокупность прав доступа, предоставленных данному субъекту на данный объект.

Поскольку матрица доступа обычно очень велика, она никогда не хранится в системе в явном виде. Для сокращения объема матрицы доступа используется объединение субъектов доступа в группы. Права, предоставленные группе субъектов для доступа к некоторому объекту, тем самым автоматически предоставляются каждому субъекту группы.

Вместе с каждым объектом доступа хранятся его *атрибуты защиты*, описывающие, кто является владельцем объекта и каковы права доступа к данному объекту различных субъектов. Атрибуты защиты фактически представляют собой совокупность идентификатора владельца объекта и строки матрицы доступа в кодированном виде.

На практике используются два способа кодирования строки матрицы доступа в атрибутах защиты объекта:

- *вектор доступа* (UNIX) — вектор фиксированной длины, разбитый на несколько подвекторов. Каждый подвектор описывает права доступа к данному объекту некоторого субъекта. С помощью вектора доступа можно описать права доступа к объекту только фиксированного числа субъектов, что накладывает существенные ограничения на систему разграничения доступа;
- *список доступа* (Windows, VAX/VMS) — список переменной длины, элементами которого являются структуры, содержащие:
- идентификатор субъекта;
- права, предоставленные данному субъекту на данный объект;
- различные флаги и атрибуты.

Фактически вектор доступа представляет собой список доступа фиксированной длины и является частным случаем списка доступа.

Кодирование матрицы доступа в виде совокупности списков доступа позволяет реализовать более мощный и гибкий механизм управления доступом. С другой стороны, данный механизм требует гораздо больше оперативной и дисковой памяти для хранения атрибу-

тов защиты объектов, усложняет техническую реализацию правил управления доступом и создает проблему, связанную с тем, что значения элементов списка доступа могут противоречить друг другу. Предположим, один элемент списка доступа разрешает некоторому пользователю доступ к объекту, а другой элемент того же списка запрещает доступ к объекту группе, в которую входит этот пользователь. При использовании списков доступа правила управления доступом должны включать в себя правила разрешения подобных противоречий.

При создании нового объекта владелец объекта должен определить права доступа различных субъектов к этому объекту. Если владелец объекта этого не сделал, новому объекту либо назначаются атрибуты защиты по умолчанию, либо новый объект наследует атрибуты защиты от объекта-контейнера, в котором создается объект.

Дискреционное управление доступом является наиболее распространенным механизмом управления доступом. Это обусловлено сравнительной простотой реализации данной модели и сравнительной необременительностью правил дискреционного управления доступом для пользователей. Вместе с тем, общая защищенность компьютерной системы, подсистема защиты которой реализует только лишь дискреционное управление доступом, во многих случаях недостаточна.

2.2.2. Изолированная программная среда

Изолированная, или *замкнутая программная среда* представляет собой расширение модели дискреционного управления доступом. Здесь правила управления доступом формулируются следующим образом:

1. Для любого объекта системы существует владелец.

2. Владелец объекта может произвольно ограничивать доступ других субъектов к данному объекту.

3. Для каждой четверки субъект-объект-право-процесс возможность доступа определена однозначно.

4. Существует хотя бы один привилегированный пользователь (администратор), имеющий возможность обратиться к любому объекту по любому методу.

5. Для каждого субъекта определен список процессов, которые данный субъект может порождать.

При использовании изолированной программной среды права субъекта на доступ к объекту определяются не только правами и

привилегиями субъекта, но и процессом, посредством которого субъект обращается к объекту. Можно, например, разрешить обращаться к файлам с расширением .DOC только программе Microsoft Word.

Изолированная программная среда существенно повышает защищенность системы от программных закладок. В то же время изолированная программная среда создает определенные сложности в администрировании защищаемой системы. Например, при установке нового программного продукта администратор должен модифицировать списки разрешенных программ для пользователей, которые должны иметь возможность работать с этим программным продуктом. Поэтому использование в изолированной программной среде прикладного программного обеспечения, разработанного без учета особенностей данной модели управления доступом, затруднительно.

2.2.3. Мандатное управление доступом

Данная модель управления доступом в основном предназначена для предотвращения утечки конфиденциальной информации из защищаемой системы. Основная идея мандатного управления доступом заключается в том, что объектам системы присваиваются *мандатные метки*, формально описывающие секретность информации, содержащейся в данном объекте в данный момент. При каждой передаче информации от объекта к объекту вместе с информацией передается мандатная метка. Если, например, пользователь попытается скопировать информацию из файла с мандатной меткой «секретно» в файл с мандатной меткой «несекретно», мандатная метка файла-приемника будет изменена на «секретно» либо операция копирования будет запрещена. На практике мандатные метки чаще всего образуют единую линейную шкалу возрастающих грифов секретности, например: «Несекретно», «Для служебного пользования», «Секретно», «Совершенно секретно». Иногда данную шкалу дополняют так называемыми *неиерархическими категориями* информации, например: «Проект атомной бомбы», «План порабощения мира». При передаче информации, которая отнесена к определенной неиерархической категории (т. е. соответствующему объекту присвоена соответствующая мандатная метка) объект-приемник также должен быть отнесен к данной неиерархической категории либо операция копирования должна быть запрещена.

Рассмотрим правила управления доступом для наиболее типичного случая реализации мандатного управления доступом, когда

оно применяется совместно с дискреционным, а среди используемых мандатных меток нет неиерархических категорий.

1. Для любого объекта системы существует владелец.

2. Владелец объекта может произвольно ограничивать доступ других субъектов к данному объекту.

3. Для каждой четверки субъект–объект–право–процесс возможность доступа определена однозначно в каждый момент времени. При изменении состояния процесса со временем возможность предоставления доступа также может измениться. Т.е. если в некоторый момент времени к некоторому объекту разрешен доступ некоторого субъекта посредством некоторого процесса, это не означает, что в другой момент времени доступ тоже будет разрешен. Вместе с тем, в каждый момент времени возможность доступа определена однозначно — никаких случайных величин здесь нет. 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. Каждый объект доступа имеет *гриф секретности*. Чем выше числовое значение грифа секретности, тем секретнее объект. Нулевое значение грифа секретности означает, что объект несекретен. Если объект несекретен, администратор может обратиться к нему по любому поддерживаемому 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. Если:

- гриф секретности объекта строго выше уровня допуска субъекта, обращающегося к нему,
- субъект открывает объект в режиме, допускающем чтение информации,

то доступ субъекта к объекту должен быть запрещен независимо от состояния матрицы доступа. Это так называемое *правило NRU* (not read up — не читать выше).

8. Каждый процесс операционной системы имеет *уровень конфиденциальности*, равный максимуму из грифов секретности объ-

ектов, открытых процессом на протяжении своего существования. Уровень конфиденциальности фактически представляет собой гриф секретности информации, хранящейся в оперативной памяти, доступной процессу.

9. Если:

- гриф секретности объекта строго ниже уровня конфиденциальности процесса, обращающегося к нему,
- субъект собирается записывать в объект информацию,

то доступ субъекту к объекту должен быть запрещен независимо от состояния матрицы доступа. Это правило разграничения доступа предотвращает утечку секретной информации. Это так называемое *правило NWD* (not write down — не записывать ниже).

10. Понизить гриф секретности объекта может только субъект, который:

- имеет доступ к объекту согласно правилу 7;
- обладает специальной привилегией, позволяющей ему понижать грифы секретности объектов.

Если линейная шкала грифов секретности дополнена неиерархическими категориями, правила NRU и NWD видоизменяются очевидным образом. Иногда в этом случае употребляются вместо NRU и NWD формулировки NRI (not read in) и NWO (not write out).

При реализации в операционной системе мандатного управления доступом существенно страдает производительность, поскольку права доступа к объекту должны проверяться не только при открытии объекта, но и перед каждой операцией записи.

Кроме того, данная модель управления доступом создает серьезные проблемы, 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 своей оперативной памяти во второй объект. Если, например, запустить внутри операционной системы, поддерживающей мандатное разграничение доступа, текстовый редактор Word и отредактировать секретный документ, то при завершении работы Word не сможет обновить орфографический словарь. Действительно, уровень конфиденциальности процесса имеет значение «секретно», гриф секретности орфографического словаря — несекретно и запись данных должна быть запрещена согласно правилу NWD.

Данная проблема может быть решена путем применения специального программного интерфейса (API) для работы с оперативной памятью. Области памяти, выделяемые процессам, могут быть описаны как объекты доступа, после чего им могут назначаться грифы секретности. При чтении секретного файла процесс должен считать содержимое такого файла в секретную область памяти, используя специальные 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 «обычную» 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 доступа, уровень конфиденциальности процесса повышается.

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

Также вызывает проблемы вопрос о назначении грифов секретности вновь 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.

Каждая из приведенных моделей разграничения доступа имеет свои достоинства и недостатки. Приведенная ниже таблица позволяет провести их сравнительный анализ.

Дискреционное управление доступом является наиболее простой и наименее обременительной для пользователей и администраторов системой правил управления доступом. Данная модель управления доступом подходит тем операционным системам, к безопасности которых не предъявляется особых требований.

Если для организации чрезвычайно важно обеспечение защищенности системы от несанкционированной утечки конфиденциальной информации (например, если данное требование отражено в нормативных документах, регламентирующих политику безопасности в компьютерных системах данной организации), без мандатного управления доступом не обойтись. В остальных случаях применение

Таблица

Модель разграничения доступа	Управление доступом		
	избирательное	изолированная среда	полномочное
Защита от утечки информации	отсутствует	отсутствует	имеется
Защищенность от программных закладок	низкая	высокая	низкая
Сложность реализации	низкая	средняя	высокая
Сложность администрирования	низкая	средняя	высокая
Затраты ресурсов компьютера	низкие	низкие	высокие
Использование программного обеспечения, разработанного для других систем	возможно	проблематично	проблематично

этой модели представляется нецелесообразным из-за резкого ухудшения эксплуатационных качеств операционной системы.

Что касается изолированной программной среды, то ее целесообразно использовать в случаях, когда предъявляются повышенные требования к защищенности операционной системы от вредоносного программного обеспечения. В последние годы наблюдается тенденция дополнять подсистемы управления доступом популярных операционных систем отдельными элементами изолированной программной среды. Возможно, в будущем, по мере нарастания угроз со стороны вредоносного программного обеспечения, бесконтрольно циркулирующего в Internet, изолированная программная среда станет де-факто стандартом для распространенных операционных систем.

2.3. Управление доступом в Windows

2.3.1. Объекты доступа

В операционных системах семейства Microsoft Windows все объекты операционной системы являются объектами доступа. Другими словами, доступ субъектов к любому объекту операционной системы может быть произвольно ограничен. Атрибуты защиты объекта Windows входят в число обязательных атрибутов, объект, не имеющий атрибутов защиты, физически не может существовать. Даже те объекты, которые не могут иметь атрибуты защиты из-за внутренних особенностей реализации (например, файлы, физически размещенные на файловой системе FAT, не могут иметь атрибуты защиты, поскольку те не поддерживаются файловой системой), при открытии получают временный набор атрибутов защиты «объект общедоступен».

Интересной особенностью Windows по сравнению с другими операционными системами является то, что поддерживаемый набор типов объектов доступа не задан жестко в коде операционной системы, но может расширяться системным программным обеспечением, в том числе и разработанным третьими фирмами.

При загрузке операционной системы все поддерживаемые типы объектов регистрируются путем создания в директории дерева объектов \ObjectTypes специальных объектов типа «тип объекта», каждый из которых соответствует одному из поддерживаемых типов объектов. Любая программа, имеющая право создавать объекты в директории \ObjectTypes, может регистрировать в операционной системе нестандартные типы объектов, которые начиная с момента регистрации обрабатываются Windows наравне со стандартными типами.

В Windows 7 SP1 определено 42 стандартных типа объектов, большинство из которых используются внутри операционной системы и недоступны прикладным программам. Как правило, прикладные программы Windows работают только с объектами следующих типов:

- файловые объекты: файлы, дисковые директории, устройства, именованные каналы и т. п.;
- ключи реестра;
- секции разделяемой памяти;
- процессы;
- потоки;
- события;
- мьютексы;
- семафоры;
- порты;
- маркеры доступа;
- рабочие столы;
- оконные станции;
- пакетные задания;
- директории дерева объектов;
- символические связи (линки) дерева объектов.

2.3.2. Субъекты доступа

Операционная система Windows поддерживает следующие типы субъектов доступа:

1. Пользователи, включая псевдопользователей. К псевдопользователям в Windows относятся следующие субъекты доступа:

- SYSTEM — операционная система локального компьютера. Данный псевдопользователь всегда входит в группу Administrators и всегда имеет все привилегии;
- LOCAL SERVICE — псевдопользователь, от имени которого выполняются локальные (несетевые) сервисы;
- NETWORK SERVICE — псевдопользователь, от имени которого выполняются сетевые сервисы;
- ANONYMOUS — «бесправный» псевдопользователь, от имени которого выполняются сетевые запросы, сделанные в рамках нуль-сессии (null session)*;
- ⟨имя_компьютера⟩$ — псевдопользователи, соответствующие компьютерам, входящим в домен. Эти псевдопользователи используются при взаимной аутентификации компьютеров в лесу доменов, кроме того эти псевдопользователи используются для делегирования полномочий псевдопользователя SYSTEM одного компьютера на другие компьютеры леса доменов.

2. Группы пользователей. В Windows группы пользователей могут пересекаться, т. е. каждый пользователь Windows может входить в потенциально неограниченное количество групп. Среди всех групп, в которые входит пользователь, выделяется одна *первичная группа*, которая используется исключительно для совместимости со стандартом POSIX, как та самая единственная группа, в которую должен входить любой пользователь POSIX-совместимых систем. К политике безопасности операционной системы первичная группа не имеет никакого отношения.

3. *Специальные (временные) группы.* В отличие от обычных групп членство пользователя в таких группах определяется не администратором, а самой операционной системой в зависимости от способа взаимодействия пользователя с системелй. К специальным группам относятся:

- INTERACTIVE — пользователи, работающие с системой локально (обычно не более одного);
- NETWORK — пользователи, работающих с системой через сеть;
- DIAL_UP — пользователи, работающих с системой по модему;

---

* В некоторых источниках ANONYMOUS считается не пользователем, а специальной группой. Судя по MSDN, где в одних статьях ANONYMOUS называют псевдопользователем, а в других — специальной группой, единого мнения по данному вопросу нет даже в Microsoft. Однако, поскольку ANONYMOUS не может выступать в роли пользователя, от имени которого выполняется процесс, данный вопрос имеет чисто терминологическое значение.

- BATCH — пользователи и псевдопользователи, от имени которых запущены пакетные задания (batch jobs);
- SERVICE — пользователи и псевдопользователи, от имени которых выполняются сервисы (службы);
- TERMINAL SERVER USER — пользователи, работающие с системой через терминальную сессию.

4. *Относительные субъекты.* Эти субъекты определяются относительно объекта, для которого определяются права доступа. Существуют следующие относительные субъекты:

- CREATOR_OWNER — владелец объекта;
- CREATOR_GROUP — первичная группа владельца объекта.

Относительные субъекты используются, если нужно описать права доступа пользователей к объектам по принципу «что кому принадлежит, то ему и доступно».

Для идентификации субъектов доступа в Windows используется особый тип идентификатора, называемый SID (security id). Субъекты доступа SYSTEM, LOCAL SERVICE, NETWORK SERVICE, ANONYMOUS, Everyone (группа, в которую входят все пользователи, возможно, за исключением псевдопользователя ANONYMOUS), INTERACTIVE, NETWORK, DIAL_UP, BATCH, SERVICE, TERMINAL SERVER USER, CREATOR_OWNER и CREATOR_GROUP имеют стандартные идентификаторы, общие для всех экземпляров операционной системы. Идентификаторы остальных субъекты доступа уникальны в пределах всей вселенной*.

2.3.3. Методы и права доступа

Операционная система Windows поддерживает до 22 *методов доступа* субъектов к объектам каждого типа (за исключением объектов активного каталога, особенности реализации прав доступа к ним будут рассмотрены в п. 2.3.6). Шесть методов доступа представляют собой *стандартные методы*, поддерживаемые для объектов всех типов:

- удаление объекта;
- получение атрибутов защиты объекта;
- изменение списка доступа объекта;

---

* За исключением ситуации, когда несколько экземпляров операционной системы на нескольких однотипных компьютерах некорректно восстановлены из одной резервной копии. В этом случае имеющаяся неуникальность идентификаторов субъектов приводит к множеству разнообразных ошибок в функционировании «клонированных» операционных систем.

- изменение владельца объекта;
- получение и изменение параметров аудита в отношении объекта;
- ожидание объекта.

Для каждого типа объекта поддерживается до 16 *специфичных методов доступа.* Следующая таблица описывает специфичные методы доступа, определенные для некоторых типов объектов.

Таблица

Объект	Методы
Файл	чтение запись добавление информации в конец выполнение получение атрибутов изменение атрибутов получение расширенных атрибутов изменение расширенных атрибутов
Дисковая директория	просмотр создание нового файла создание поддиректории проход (traverse) удаление файла или поддиректории получение атрибутов изменение атрибутов получение расширенных атрибутов изменение расширенных атрибутов
Ключ реестра	чтение значений изменение значений создание подключа перечисление подключей требование оповещения при доступе к ключу другого потока создание символической связи
Процесс	завершение создание нового потока изменение атрибутов страниц адресного пространства чтение адресного пространства запись в адресное пространство дублирование хэндлов получение приоритета изменение приоритета получение информации о процессе изменение квоты
Поток	завершение приостановка/возобновление получение контекста[1] изменение контекста получение приоритета изменение приоритета назначение маркера доступа

Продолжение таблицы

Объект	Методы
Диспетчер сервисов	подключение получение статуса списка сервисов перечисление сервисов создание нового сервиса блокирование списка сервисов
Сервис	запуск останов приостановка/возобновление получение текущего состояния обновление текущего состояния перечисление зависимых сервисов получение конфигурации изменение конфигурации метод доступа, специфичный для данного сервиса[2]
Рабочий стол	чтение элементов рабочего стола изменение элементов рабочего стола создание окна создание меню установка фильтра (hook setting) запись макрокоманды (journal recording) воспроизведение макрокоманды (journal playback) перечисление (используется функцией EnumDesktops) отображение рабочего стола на экране
Оконная станция	чтение содержимого экрана закрытие получение атрибутов[4] изменение атрибутов обращение к карману (clipboard) обращение к таблице атомов создание нового рабочего стола перечисление рабочих столов перечисление самой оконной станции (используется функцией EnumWindowStations)
Секция	получение информации о текущем состоянии отображение для чтения отображение для записи отображение для выполнения изменение размера
Маркер доступа	чтение получение информации о подсистеме, создавшей маркер доступа включение/выключение групп включение/выключение привилегий изменение атрибутов защиты по умолчанию изменение идентификатора сессии назначение процессу назначение потоку копирование

Окончание таблицы

Объект	Методы
Событие	получение состояния изменение состояния
Семафор	получение состояния изменение состояния
Мьютекс	получение состояния изменение состояния

[1] Контекст потока в Windows — аппаратно-зависимая структура данных, в которой сохраняются значения регистров приостановленного или прерванного потока.

[2] Все 128 нестандартных операций управления, специфичных для конкретного сервиса, рассматриваются подсистемой управления доступом как единый метод доступа.

[3] Элементами рабочего стола Windows являются окна, контексты устройств (DC), шрифты и т. д.

[4] Атрибутами оконной станции являются цветовые настройки, используемые обои и хранитель экрана и т. д.

Следующие методы доступа требуют наличия у субъекта доступа специальных привилегий:

- создание нового сервиса;
- блокирование списка сервисов;
- запуск сервиса;
- останов сервиса;
- приостановка/возобновление сервиса;
- назначение процессу маркера доступа;
- получение или изменение параметров аудита в отношении объекта.

Каждому специфичному методу доступа, поддерживаемому в Windows, соответствует *право* на его осуществление. Эти права доступа называются *специфичными*, поскольку они специфичны для каждого типа объектов. Для каждого типа объектов может поддерживаться до шестнадцати специфичных прав доступа.

Каждому стандартному методу доступа, за исключением метода «получение и изменение параметров аудита в отношении объекта», также соответствует право доступа, дающее возможность реализации соответствующего метода доступа. Такие права доступа называются *стандартными*.

Заметим, что для некоторых объектов стандартные и специфичные права доступа реализованы не вполне корректно. Например, при попытке получения атрибутов защиты объекта типа «процесс» проверяется не стандартное право «получение атрибутов за-

щиты», а специфичное право «получение информации о процессе». В Windows 2000 при попытке изменения списка доступа файла проверяется не только стандартное право «изменение списка доступа», но и специфичное право «изменение расширенных атрибутов файла». Видимо, эти странности обусловлены ошибками программистов. В пользу этого предположения говорит то, что странность реализации изменения атрибутов защиты файлов имеет место лишь в Windows 2000, но не в более поздних версиях Windows.

Также Windows поддерживает так называемые *общие* (*generic*), или *отображаемые* (*mapped*) права доступа. Поддерживаются четыре отображаемых права доступа:

- чтение (GENERIC_READ);
- запись (GENERIC_WRITE);
- выполнение (GENERIC_EXECUTE);
- все действия (GENERIC_ALL).

Каждое из отображаемых прав доступа представляет собой некоторую комбинацию стандартных и специфичных прав доступа. Другими словами, отображаемое право доступа дает возможность на осуществление некоторого набора методов доступа к объекту. Отображаемые права могут быть предоставлены для доступа к объекту любого типа, однако конкретное содержание отображаемого права доступа зависит от типа объекта.

Следует иметь в виду, что порядок отображения отображаемого права доступа в набор стандартных и специфичных прав не обязательно совпадает с интуитивным смыслом общего права доступа. Например, следующие специфичные права:

- подключение к сервисам — для объекта «диспетчер сервисов»;
- реализация специфичных для конкретного сервиса методов доступа — для объектов типа «сервис»;

почему-то не включены в отображаемое право GENERIC_READ.

Но чаще всего порядок отображения отображаемых прав все же совпадает с интуитивно ожидаемым.

Отображаемые права доступа позволяют пользователю устанавливать права доступа к объекту, ничего не зная о специфике объектов данного типа. Например, если пользователь желает, чтобы все пользователи могли читать некоторый файл, он просто предоставляет группе пользователей Everyone отображаемое право на чтение файла. При этом пользователь не обязан отдельно предоставлять группе Everyone права на получение различных атрибутов файла, поскольку все эти права автоматически предоставляются группе Everyone при отображение отображаемого права доступа «чтение

объекта». Пользователь может даже не знать, что чтение информации, содержащейся в файле и чтение атрибутов файла реализуются разными методами доступа.

Последним классом прав доступа, поддерживаемых Windows, являются *виртуальные права доступа*. Виртуальные права доступа не могут быть предоставлены субъекту, но могут быть им запрошены. Поддерживаются два виртуальных права доступа:

- MAXIMUM_ALLOWED;
- ACCESS_SYSTEM_SECURITY.

Запрашивая виртуальное право MAXIMUM_ALLOWED на доступ к объекту, субъект тем самым требует открытия объекта с максимально доступными ему правами. Это виртуальное право позволяет субъекту открыть объект с максимально доступными правами, не производя детального анализа того, какие именно права доступны данному субъекту по отношению к данному объекту. Операционная система сама проводит такой анализ в процессе проверки прав доступа субъекта к объекту.

Виртуальное право ACCESS_SYSTEM_SECURITY — это право на получение и изменение параметров аудита по данному объекту. Возможность доступа к объектам по этому методу полностью регулируется соответствующей привилегией субъекта доступа. Субъект, обладающий этой привилегией, может обращаться по данному методу доступа к любому объекту операционной системы, а субъект, не обладающий этой привилегией, не может применять данный метод доступа ни к одному объекту. Таким образом, субъект, имеющий доступ к параметрам аудита некоторого объекта, имеет доступ к параметрам аудита любого объекта операционной системы. Разрешить или запретить доступ конкретного субъекта к конкретному объекту по методу «доступ к параметрам аудита по объекту» в Windows невозможно, и поэтому данное право доступа является виртуальным.

2.3.4. Привилегии субъектов доступа

Каждый пользователь и псевдопользователь Windows обладает некоторым (возможно, пустым) набором привилегий. Привилегии представляют собой права на выполнение субъектом действий, касающихся всей системы в целом, а не отдельных ее объектов. Перечислим основные привилегии, поддерживаемые в современных версиях Windows:

- завершать работу операционной системы и перезагружать компьютер с локальной консоли;
- завершать работу операционной системы и перезагружать компьютер с удаленной консоли;

- устанавливать системное время;
- анализировать производительность одного процесса;
- анализировать производительность всей операционной системы в целом;
- создавать постоянные объекты в оперативной памяти;
- создавать резервные копии информации, хранящейся на жестких дисках;
- восстанавливать информацию на жестких дисках с резервных копий;
- назначать процессам и потокам высокие приоритеты;
- повышать квоты процессов;
- изменять системные переменные среды;
- отлаживать программы — позволяет обращаться ко всем объектам типа «процесс» и «поток» по всем методам доступа, поддерживаемым для данных объектов;
- загружать и выгружать драйверы и сервисы;
- работать с подсистемой аудита — просматривать и очищать журнал аудита, получать информацию о политике аудита, изменять политику аудита, осуществлять доступ к параметрам аудита по любому объекту операционной системы;
- добавлять записи в журнал аудита;
- объявлять себя владельцем любого объекта;
- создавать маркеры доступа;
- назначать процессам маркеры доступа;
- выступать как часть операционной системы;
- получать оповещения от файловых систем;
- извлекать компьютер из стойки (docking station);
- добавлять компьютеры в домен;
- синхронизировать домен;
- создавать объекты, глобальные для всех терминальных сессий данного сервера;
- делегировать полномочия клиентов на другие компьютеры;
- олицетворять клиентов;
- выполнять задачи по обслуживанию логических дисков (дефрагментация и т. п.).

При входе в систему пользователь получает привилегии, предоставленные ему индивидуально, а также привилегии, предоставленные всем группам, в которые входит пользователь. Назначать привилегии субъектам доступа может только администратор. Если

привилегии пользователя изменились за время его работы с системой, изменения начинают действовать только после того, как пользователь выйдет из системы и снова войдет в нее.

Некоторые из перечисленных привилегий позволяют обладающим ими субъектам, преодолевать те или иные элементы защиты операционной системы, например:

- привилегия создавать резервные копии информации позволяет пользователю игнорировать правила управления доступом при чтении файлов, дисковых директорий, ключей и значений реестра;
- привилегия восстанавливать информацию с резервных копий позволяет пользователю игнорировать правила управления доступом при создании файлов, дисковых директорий, ключей и значений реестра, а также при записи в файлы и значения реестра;
- привилегия отлаживать программы позволяет пользователю обращаться к любому процессу по любому методу доступа. В частности, программа, запущенная таким пользователем, может изменить произвольным образом содержимое адресного пространства любого процесса операционной системы, что предоставляет такому пользователю практически неограниченные полномочия;
- привилегия загружать и выгружать драйверы и сервисы позволяет пользователю выполнять произвольный код от имени и с правами операционной системы (псевдопользователя SYSTEM). Пользователь может внедрять в операционную систему программные закладки под видом драйверов и сервисов. Учитывая, что драйверы устройств Windows могут игнорировать большинство защитных функций операционной системы, эта привилегия дает субъекту, ей обладающему, практически неограниченные полномочия;
- привилегия переназначать владельца любого объекта позволяет пользователю получать доступ к любому объекту по любому методу (за исключением доступа к параметрам аудита по данному объекту);
- привилегия добавлять записи в журнал аудита позволяет пользователю записывать в журнал аудита произвольную информацию, в том числе и информацию, компрометирующую других пользователей.

Администраторы операционной системы должны подходить к назначению пользователям привилегий с максимальной ответствен-

ностью. Особое внимание следует уделять вышеперечисленным опасным привилегиям.

Помимо явно определенных привилегий, в Windows существуют также привилегии, неявно определенные через предопределенные группы. Например, группа Administrators обладает целым рядом неявно определенных привилегий, например, привилегией выполнять команду at, привилегией регистрировать любых пользователей и т. д.

2.3.5. Маркер доступа пользователя

В Windows каждый пользователь (в том числе и каждый псевдопользователь), работающий в системе, имеет свой *маркер доступа* (*access token*). Каждый процесс, порожденный пользователем, получает свою копию маркера доступа пользователя, эта копия является обязательным атрибутом процесса. Процесс, не имеющий маркера доступа, не может существовать. Также маркеры доступа могут назначаться отдельным потокам.

Маркер доступа представляет собой объект специального вида, содержащий следующую информацию (пока мы просто перечислим элементы маркера доступа, не вдаваясь в подробные пояснения, которые последуют ниже):

- TokenUser — личный идентификатор пользователя;
- TokenGroups — список групп и специальных групп, в которые входит пользователь;
- TokenPrivileges — список привилегий, предоставленных пользователю;
- TokenOrigin — идентификатор сеанса работы пользователя с операционной системой;
- TokenSessionId — идентификатор терминальной сессии, в рамках которой пользователь работает с операционной системой. Если пользователь не работает в рамках терминальной сессии, данный идентификатор равен нулю;
- TokenOwner — идентификатор субъекта доступа, который по умолчанию назначается владельцем всех объектов, создаваемых пользователем в ходе работы с системой. Обычно совпадает с личным идентификатором пользователя, в некоторых конфигурациях операционной системы пользователям-администраторам в качестве TokenOwner назначается идентификатор группы Administrators;
- TokenPrimaryGroup — идентификатор группы, которая по умолчанию назначается первичной группой владельца всех объектов

операционной системы, создаваемых пользователем в ходе работы с системой. Всегда совпадает с первичной группой пользователя, которому выдан данный маркер доступа;
- TokenDefaultDacl — список доступа, назначаемый по умолчанию новым объектам операционной системы, созданным данным пользователем в текущем сеансе работы;
- TokenRestrictedSids — список ограничивающих идентификаторов, поддерживается только для ограниченных маркеров доступа;
- TokenImpersonationLevel — уровень олицетворения, поддерживается для маркеров доступа, назначенных потокам, но не процессам.
- TokenType — тип маркера доступа: первичный маркер доступа (назначается процессу) или маркер олицетворения (назначается потоку);
- TokenSource — имя и идентификатор элемента подсистемы аутентификации, выдавшего данный маркер доступа;
- TokenStatistics — дополнительная служебная информация.

Маркер доступа содержит всю информацию о пользователе, необходимую подсистеме управления доступом для принятия решений о предоставлении пользователю доступа к тем или иным объектам операционной системы.

Маркер доступа пользователя создается в ходе авторизации пользователя, на одном из последних этапов процедуры входа пользователя в систему. Создавать маркеры доступа могут только процессы, выполняющиеся от имени субъектов, обладающих соответствующей привилегией. Обычно этой привилегией обладает только псевдопользователь SYSTEM. Единственным способом создать маркер доступа является использование системного вызова LsaLogonUser, получающего в качестве входных параметров идентификационную и аутентификационную информацию пользователя и возвращающего (в случае успешного выполнения) в выходном параметре хэндл созданного маркера доступа. Создание маркера доступа «вручную» невозможно.

Доступ прикладных и системных программ к маркерам доступа возможен только лишь с использованием соответствующих системных вызовов, предоставляющих весьма ограниченные возможности по работе с маркерами доступа. Прямой доступ из пользовательского режима к отдельным элементам маркеров доступа невозможен. Это сделано специально, во избежание несанкционированного изменения полномочий пользователей путем модификации маркеров

доступа вредоносными программами. Лишь программный код, выполняющийся в режиме ядра, может получить прямой доступ к элементам маркера доступа. Впрочем, программный код, выполняющийся в режиме ядра, обладает ничем не ограниченными полномочиями и может получать прямой и ничем не ограниченный доступ к любым объектам операционной системы.

Каждому процессу Windows назначается так называемый *первичный маркер доступа* (*primary access token*). Если процесс порожден с помощью системной функции CreateProcess или одной из более высокоуровневых функций, ее использующих (подавляющее большинство процессов порождаются именно так), первичный маркер доступа представляет собой точную копию первичного маркера доступа процесса-родителя, т. е. маркер доступа пользователя, работающего в системе. Субъект, обладающий привилегией «назначать маркеры доступа процессам», может порождать процессы от имени других пользователей и псевдопользователей.

Отдельным потокам процесса могут назначаться свои маркеры доступа (так называемые *маркеры олицетворения — impersonation access tokens*). Механизм олицетворения обычно используется процессами-серверами. Когда процесс-сервер обслуживает запрос процесса-клиента, для выполнения запроса внутри процесса-сервера создается поток, которому назначается маркер доступа пользователя, инициировавшего запрос, и в дальнейшем данный поток работает с правами того пользователя, от имени которого выполняется процесс-клиент. Если бы олицетворение клиентов в Windows не поддерживалось, при каждом обращении процесса-клиента к объекту операционной системы сервера процессу-серверу приходилось бы явно проверять достаточность прав доступа ползователя-клиента к данному объекту сервера. При этом достаточно пропустить всего лишь одну проверку прав доступа клиента и в системе появляется критическая уязвимость. Но если в операционной системе поддерживается механизм олицетворения, процесс-сервер может не заботиться о полномочиях клиента, все необходимые проверки делаются автоматически.

В ранних версий Windows (до версии XP SP1 включительно) возможность олицетворять клиентов предоставлялась всем пользователям и псевдопользователям операционной системы без всяких ограничений. Это было серьезной потенциальной уязвимостью, легко приводящей к реальным уязвимостям, чаще всего критическим с точки зрения безопасности. Если низкопривилегированному пользователю удавалось зарегистрировать свой процесс как

сервер некоторого многопользовательского интерфейса и заставить высокопривилегированного пользователя подключиться к этому серверу, низкопривилегированный пользователь мог легко повысить свои полномочия, проведя олицетворение высокопривилегированного пользователя-клиента. Известен целый ряд подобных уязвимостей в Windows NT, Windows 2000 и Windows XP, одна из них (AdminTrap) была обнаружена автором настоящего пособия в 1997 году.

Начиная с Windows XP SP2, олицетворение клиента требует наличия у пользователя или псевдопользователя, от имени которого выполняется процесс-сервер, специальной привилегии «олицетворять клиентов». При отсутствии у сервера данной привилегии олицетворение не происходит, хотя соответствующие системные функции, как правило, не сообщают об ошибке*. Исключением являются ситуации, когда олицетворение клиента заведомо безопасно (например, когда маркер доступа клиента создан на основе имени и пароля, явно указанных сервером), в этих случаях олицетворение происходит даже при отсутствии данной привилегии в маркере доступа процесса-сервера.

Начиная с Windows 2000, в маркер доступа потока включается особый атрибут «уровень олицетворения». Он может принимать следующие значения:

- SecurityAnonymous, анонимный — «пустой» маркер доступа, операционная система воспринимает его как отсутствие маркера доступа у потока. Обычно такие маркеры доступа возникают в результате сетевых ошибок в ходе аутентификации удаленных клиентов;
- идентификационный — для маркера доступа поддерживаются все операции, кроме олицетворения. Если необходимо провести проверку прав доступа с использованием данного маркера доступа, эта проверка должна быть осуществлена путем явного вызова системной функции NtAccessCheck или одной из более

---

* Так сделано специально для обеспечения обратной совместимости с программным обеспечением, разработанным для предшествующих версий Windows. Дело в том, что ошибка в ходе олицетворения клиента низкопривилегированным сервером чаще всего никак не сказывается на дальнейшем функционировании ни сервера, ни клиента. Если бы функции олицетворения при отсутствии у сервера достаточных полномочий выдавали сообщения об ошибках, в большинстве случаев это приводило бы к неоправданным отказам в обслуживании клиентов.

высокоуровневых функций, вызывающих NtAccessCheck внутри себя;

- олицетворение — маркер доступа может быть использован для олицетворения на локальном компьютере;
- делегирование — маркер доступа может быть использован для олицетворения в пределах всего леса доменов, к которому принадлежит данный компьютер*. Маркер доступа данного типа может создаваться только для тех пользователей, учетные записи которых имеют атрибут «обратимое шифрование аутентификационных данных». При этом политики безопасности компьютера, домена и всего леса должны удовлетворять некоторым условиям. Большинство экспертов считают, что маркеры доступа уровня делегирования создают потенциальную угрозу безопасности сети и должны применяться только в тех случаях, когда без них безусловно нельзя обойтись (например, когда терминальный сервер должен использоваться в качестве клиента для подключения к другому терминальному серверу).

Назначать маркер доступа процессу, а также создавать маркеры доступа могут только субъекты, обладающие соответствующими привилегиями.

Каждая привилегия, содержащаяся в маркере доступа, в каждый момент времени может находиться в одном из двух состояний — включенном либо выключенном. Если привилегия выключена, субъект, которому принадлежит маркер доступа, не может пользоваться данной привилегией до тех пор, пока не включит ее. По умолчанию большинство привилегий выключены. После того, как привилегия включена и применена, Microsoft рекомендует выключить привилегию, как только это станет возможным, однако большинство разработчиков программного обеспечения не придерживаются этой рекомендации.

Для включения и выключения привилегий Microsoft рекомендует применять системную функцию AdjustTokenPrivileges из библиотеки advapi32.dll. Однако эта функция весьма неудобна в применении и большинство программистов предпочитают «хакерский» способ, основанный на использовании недокументированной функции RtlAdjustPrivilege из библиотеки ntdll.dll. Эта функция имеет следующий прототип:

```
NTSTATUS RtlAdjustPrivilege (ULONG Privilege, BOOL Enable,
```

---

* Политики безопасности организационных единиц могут ограничивать область делегирования полномочий.

```
BOOL ThreadToken,
PBOOL pOldState);
```

Параметры функции имеют следующий смысл:

- Privilege — идентификатор привилегии. В Windows каждая привилегия идентифицируется 64-битным целым числом, в котором старшие 32 бита всегда равны нулю (но это нигде не документировано). В 32-разрядных версиях Windows эта функция принимает лишь младшие 32 бита идентификатора привилегии;
- Enable — включить (TRUE) или выключить (FALSE) привилегию;
- ThreadToken — операция проводится над маркером доступа потока (TRUE) или процесса (FALSE);
- pOldState — указатель на переменную, в которую будет занесено предыдущее состояния данной привилегии.

Функция возвращает статус выполнения в формате NT API.

Маркер доступа создается подсистемой аутентификации операционной системы в процессе авторизации пользователя, при этом псевдопользователь SYSTEM использует свою привилегию создавать маркеры доступа. После того, как маркер доступа создан, информация о группах, в которые входит пользователь, и о привилегиях пользователя не может быть ни добавлена в маркер доступа, ни удалена из него (за одним исключениям, которое будет описано ниже).

В Windows 2003 маркеры доступа поддерживают новую операцию, отсутствовавшую в предшествующих версиях Windows — необратимое отключение привилегий. После того, как эта операция выполнена над некоторой привилегией, включение данной привилегии в данном экземпляре маркера доступа становится невозможно. Эксперты Microsoft рекомендуют программистам, разрабатывающим программное обеспечение для высокопривилегированных процессов, применять данную операцию в ходе инициализации процесса ко всем привилегиям, которые заведомо не будут включаться в текущем сеансе работы процесса. Это позволяет заметно снизить опасность эксплуатации нарушителей уязвимостей в программном коде процесса (переполнения буферов и т. п.). Когда в контексте процесса активизируется чужеродный вредоносный код, он может пользоваться только теми привилегиями, которые содержатся в маркере доступа процесса. Теми привилегиями, которые необратимо удалены из маркера доступа, эксплойт воспользоваться не сможет. Например, вредоносный код, активизировавшийся в контексте веб-сервера, отказавшегося от неиспользуемых привилегий, не сможет пользоваться привилегиями, необходимыми для работы отладчика или консо-

ли администрирования. К сожалению, большинство программистов пока не принимают данную рекомендацию во внимание.

Начиная с Windows 2000, операционной системой поддерживаются так называемые *ограниченные* (restricted) маркеры доступа, создаваемые из обычных маркеров доступа с помощью системной функции CreateRestrictedToken. Ограниченные маркеры доступа имеют следующие отличия от обычных маркеров доступа:

- Некоторые группы в ограниченном маркере доступа могут быть помечены флагом «только для запрета» (SE_GROUP_USE_FOR_DENY_ONLY). При проверке доступа субъекта к объекту такие группы учитываются только при анализе тех элементов списка доступа объекта, которые **запрещают** те или иные права доступа к объекту. Если некоторый элемент списка доступа **разрешает** группе с флагом «только для запрета» доступ к объекту, данный элемент при проверке прав доступа игнорируется.
- Некоторые привилегии, входившие в оригинальный маркер доступа, могут быть не скопированы в ограниченный маркер доступа. В Windows XP и более ранних версиях это единственный способ необратимо удалить привилегию из маркера доступа. В Windows 2003 Server, как уже отмечалось выше, для необратимого удаления привилегий не требуется предварительного создания ограниченного маркера доступа.
- В ограниченный маркер доступа могут быть добавлены *ограничивающие* идентификаторы пользователей и групп (restricting SIDs)*. При проверке прав доступа пользователя или псевдопользователя, чей маркер доступа содержит ограничивающие идентификаторы, выполняются две проверки прав доступа: одна для основного списка (личный идентификатор пользователя, а также группы и специальные группы, в которые входит пользователь) и другая для списка ограничивающих идентификаторов. Доступ субъекта к объекту разрешается только в том случае, если обе проверки закончились с результатом «доступ разрешен»**. Фактически список ограничивающих идентифи-

---

* Иногда их ошибочно называют *ограниченными*. К сожалению, эта ошибка (restricted вместо restricting) встречается даже в официальной документации Microsoft.

** Если пользователь запрашивает виртуальное право MAXIMUM_ALLOWED, ему предоставляются те и только те права, которые предоставлены в результате **обеих** проверок, т. е. пересечение двух списков прав доступа.

каторов задает верхнюю границу, которую не могут превосходить полномочия субъекта, обладающего ограниченным маркером доступа. Например, если ограничивающий список субъекта включает в себя единственную группу Everyone, это означает, что данный субъект может получать доступ только к общедоступным объектам, при этом к некоторым общедоступным объектам доступ данного субъекта может быть запрещен (если проверка основного списка идентификаторов субъекта не увенчалась успехом).

Ограниченные маркеры доступа представляют собой чрезвычайно мощный инструмент, позволяющий реализовывать принцип минимизации полномочий пользователей практически в полном объеме. К сожалению, на практике ограниченные маркеры доступа используются редко.

2.3.6. Дескриптор защиты объекта

Атрибуты защиты объекта Windows описываются специальной структурой данных, называемой *дескриптором защиты* (security descriptor). Любой объект Windows может иметь дескриптор защиты. Дескриптор защиты содержит следующую информацию:

- идентификатор владельца объекта;
- идентификатор первичной группы владельца объекта;
- список дискреционного управления доступом (discretionary access control list, DACL), полностью описывающий права различных субъектов на объект;
- системный список управления доступом (system access control list* SACL) — используется для генерации сообщений аудита, связанных с доступом к объекту;
- флаги.

Если объект не имеет дескриптора защиты, при обращениях к нему субъектов никакие права доступа не проверяются. В этом случае любой субъект имеет абсолютные права на данный объект.

Если объект хранится на диске или ином внешнем устройстве, дескриптор защиты хранится вместе с объектом, при этом формат хранения объекта должен предоставлять такую возможность. Поскольку файловые системы, отличные от NTFS, не поддерживают хранение на диске дескрипторов защиты для файлов, только файлы и директории, расположенные на логических дисках с файловой

* Иногда аббревиатуру SACL расшифровывают как Security Access Control List.

системой NTFS, могут иметь дескрипторы защиты. Ключи реестра могут иметь дескрипторы защиты независимо от файловой системы диска, на котором размещается реестр.

Флаги дескриптора защиты представляют собой 32-битную маску, отдельные биты которой имеют следующий смысл:

0 — установлен, если идентификатор владельца объекта был установлен по умолчанию при создании объекта и с тех пор не менялся;

1 — установлен, если идентификатор первичной группы владельца объекта ьыл установлен по умолчанию при создании объекта и с тех пор не менялся;

2 — установлен, если в дескрипторе защиты присутствует DACL. Если данный флаг сброшен, дескриптор защиты допускает полный доступ к данному объекту всех субъектов доступа;

3 — установлен, если DACL объекта назначен по умолчанию при создании объекта на основе маркера доступа субъекта-создателя;

4 — установлен, если в дескрипторе защиты присутствует SACL. Если данный флаг сброшен, дескриптор защиты не предусматривает генерации сообщений аудита при доступе субъектов к данному объекту;

5 — установлен, если SACL объекта назначен по умолчанию при создании объекта на основе маркера доступа субъекта-создателя;

8 — установлен, если любые изменения в DACL объекта должны быть автоматически унаследованы существующими дочерними объектами;

9 — установлен, если любые изменения в SACL объекта должны быть автоматически унаследованы существующими дочерними объектами;

10 — установлен, если DACL объекта подвергался процедуре автоматического наследования изменений в DACL объекта-родителя;

11 — установлен, если SACL объекта подвергался процедуре автоматического наследования изменений в SACL объекта-родителя;

12 — установлен, если DACL объекта **не** должен автоматически наследовать изменений в DACL объекта-родителя;

13 — установлен, если SACL объекта **не** должен автоматически наследовать изменений в SACL объекта-родителя;

15 — установлен, если дескриптор защиты представлен в упакованном формате (для хранения на диске или в другом внешнем хранилище, но не для чтения или модификации). Большинство операций над упакованным дескриптором защиты требуют его предварительной распаковки. Распаковка дескриптора защиты (как и упа-

ковка) является чисто технической процедурой и никак не влияет на порядок управления доступом к объекту, к которому относится дескриптор защиты.

Биты 8 и 9 имеют смысл только для объектов-контейнеров (дисковые и объектовые директории, ключи реестра и т. п.).

Когда субъект доступа открывает объект, субъект должен сообщить операционной системе права, необходимые ему для работы с данным объектом. Эти права кодируются с помощью битовой маски, каждый бит которой соответствует некоторому праву доступа. Например, если пользователь открывает объект типа «файл», второй параметр системной функции CreateFile является битовой маской, описывающей запрашиваемые права доступа к объекту. Если субъект открывает файл для чтения и записи, эта маска доступа должна быть равна FILE_READ_DATA | FILE_WRITE_DATA | FILE_APPEND_DATA (или GENERIC_READ | GENERIC_WRITE).

При каждом открытии объекта субъектом операционная система получает маркер доступа субъекта и дескриптор защиты объекта и вызывает функцию ядра SeAccessCheck (обычно не непосредственно, а через несколько промежуточных функций). SeAccessCheck реализует проверку прав доступа субъекта к объекту по алгоритму, который будет описан ниже.

Элементы списка дискреционного управления доступом называются *элементами управления доступом* (access control entries, ACE). Каждый элемент управления доступом разрешает или запрещает некоторому субъекту определенные права на доступ к объекту. В состав элемента управления доступом могут входить следующие основные поля:

- тип элемента: разрешающий, запрещающий или регистрирующий;
- идентификатор субъекта;
- битовая маска прав доступа;
- флаги;
- GUID1 (необязательное поле);
- GUID2 (необязательное поле).

В DACL объекта могут присутствовать только разрешающие и запрещающие ACE, разрешающие ACE разрешают указанным субъектам доступ к объекту по указанным правам, запрещающие — запрещают. В SACL объекта могут присутствовать только регистрирующие ACE.

Начиная с Windows NT 4.0, дескрипторы защиты объектов, помимо обычных элементов контроля доступа, могут содержать так

называемые compound ACE. В отличие от других ACE, compound ACE содержит два идентификатора субъектов, один из которых должен присутствовать в маркере доступа потока, обращающегося к объекту, а другой — в маркере доступа процесса, в контексте которого выполняется данный поток. Другими словами, compound ACE позволяет описать особые права доступа для любой заданной пары пользователь-клиент + пользователь-сервер. Этот механизм крайне неудобен для практического использования и, насколько известно автору, никогда никем не используется.

Поля GUID1 и GUID2 могут присутствовать только в ACE, относящихся к объектам активного каталога доменов Windows. Объекты активного каталога обычно представляют собой совокупности *подобъектов*[*] — небольших записей, содержащих текстовые строки, числовые значения или короткие бинарные данные. Поскольку подобъекты, как правило, занимают всего несколько байт памяти, отдельные дескрипторы защиты подобъектам не назначаются, вместо этого объекту, содержащему подобъекты, назначается один общий дескриптор защиты, регламентирующий доступ ко всем подобъектам объекта.

Подобъекты объекта активного каталога могут содержать в себе другие подобъекты, подобно тому, как дисковые и объектовые директории могут содержать в себе файлы и поддиректории. Для объекта поддерживается до 5 уровней вложенности подобъектов, идентифицируемых числовыми значениями от 0 до 4:

0 — сам объект;

1 — наборы свойств (property sets);

2 — свойства (properties);

3–4 — в современных версиях Windows не используются.

Для объектов активного каталога определены два специфичных права доступа:

- ADS_RIGHT_DS_READ_PROP — читать подобъект;
- ADS_RIGHT_DS_WRITE_PROP — изменять подобъект.

Также для отдельных типов объектов активного каталога могут вводиться расширенные (extended) права доступа, каждое такое

---

[*] Специальный термин «подобъект» введен в настоящем тексте для того чтобы устранить терминологическую путаницу с двумя различными значениями англоязычного термина «child object» — так называется, с одной стороны, «полноценный» объект операционной системы, имеющий собственный дескриптор защиты, по отношению к контейнеру, в котором данный объект создан, а с другой стороны — не имеющий собственного дескриптора защиты подобъект объекта активного каталога.

право идентифицируется с помощью GUID. Например, для объекта DMD (directory management domain) определены следующие расширенные права:

- получить обновления базы данных с некоторого сервера;
- синхронизировать базу данных с некоторым сервером;
- поработать с топологией репликаций;
- обновить кэш схемы.

Обычные элементы управления доступом, примененные к объекту активного каталога, управляют доступом ко всему объекту в целом и, в частности, ко всем его подобъектам.

Для объектов активного каталога определены три особых типа элементов управления доступом, не поддерживаемых для других объектов операционной системы:

- объектно-специфический разрешающий;
- объектно-специфический запрещающий;
- объектно-специфический регистрирующий.

Элементы управления доступом, относящиеся к перечисленным типам, отличаются от других элементов управления доступом наличием полей GUID1 и GUID2. Поле GUID1 может содержать:

- тип подобъекта — в этом случае ACE регулирует доступ ко всем подобъектам данного объекта, принадлежащим к указанному типу;
- подобъект — ACE регулирует доступ к данному подобъекту и подобъектам, вложенным в него, если таковые имеются;
- расширенное право доступа — ACE регулирует доступ к объекту по данному праву, битовая маска с правами доступа игнорируется;

Поле GUID2 может присутствовать только в объектах-контейнерах активного каталога. Это поле содержит тип дочерних объектов, которые должны наследовать данный ACE. Если GUID2 не указан, ACE наследуется дочерними объектами любых типов.

2.3.7. Порядок проверки прав доступа субъекта к объекту

Перед тем, как перейти к описанию алгоритма проверки прав доступа субъекта к объекту Windows, введем следующие обозначения:

$\sim$ — операция побитового отрицания;

& — операция побитовой конъюнкции (побитовое логическое И);

| — операция побитовой дизъюнкции (побитовое логическое ИЛИ);

$\mathbf{m}$ — маска доступа, описывающая права, запрошенные субъектом и пока не предоставленные ему. В начале алгоритма m содержит все права доступа, запрошенные субъектом;

$\mathbf{a}^{(i)}$ — маска доступа i-го ACE;

$\mathbf{n}$ — количество ACE в дескрипторе защиты объекта;

$\mathbf{x}_i$ — i-й бит маски доступа $\mathbf{x}$.

Алгоритм проверки прав доступа субъекта к объекту в Windows выглядит следующим образом*.

1. В масках доступа $\mathbf{m}$, $\mathbf{a}^{(1)}$, ..., $\mathbf{a}^{(n)}$ транслируются все транслируемые права доступа. Таким образом, в дальнейшем все маски доступа, используемые алгоритмом, включают только специфичные и стандартные права доступа (за исключением маски доступа m, которая может содержать виртуальные права доступа).

2. Создаются маски доступа $\mathbf{g} = 0$ и $\mathbf{d} = 0$. В дальнейшем маска доступа $\mathbf{g}$ содержит права доступа к объекту, разрешенные субъекту, а маска доступа $\mathbf{d}$ — права доступа, запрещенные субъекту.

3. Если субъект является владельцем объекта и запрашивает право чтения и/или право изменения DACL, то соответствующий бит выставляется в маске доступа $\mathbf{g}$ и сбрасывается в маске доступа $\mathbf{m}$. Другими словами, владелец объекта всегда имеет право читать и модифицировать атрибуты защиты объекта**.

4. Если субъект обладает привилегией овладевать любыми объектами, эта привилегия включена и субъект запрашивает право овладения данным объектом, то соответствующий бит выставляется в маске доступа $\mathbf{g}$ и сбрасывается в маске доступа $\mathbf{m}$. Другими словами, администратор может объявить себя владельцем любого объекта.

5. Если субъект обладает привилегией аудитора, эта привилегия включена и субъект запрашивает право получения доступа к SACL данного объекта, то соответствующий бит выставляется в маске доступа $\mathbf{g}$ и сбрасывается в маске доступа $\mathbf{m}$.. Другими словами, аудитор всегда может получать доступ к SACL любого объекта. Более того, поскольку данное право доступа является виртуальным и не может содержаться в масках доступа $\mathbf{a}^{(1)}$, ..., $\mathbf{a}^{(n)}$, субъект, не

---

* Чтобы чрезмерно не усложнять восприятие алгоритма, мы предполагаем, что DACL объекта не содержит ACE с непустыми полями GUID1 и что субъект не запрашивает расширенных прав на доступ к объекту.

** До Windows NT 3.51 включительно данное правило распространялось только на право модификации DACL, но не на право получения атрибутов защиты объекта.

обладающий привилегией аудитора, не может обращаться к SACL любого объекта операционной системы.

6. Если в маске доступа $\mathbf{m}$ присутствует виртуальное право MAXIMUM_ALLOWED, соответствующий бит в маске доступа $\mathbf{m}$ сбрасывается, а затем выполняется цикл по i от 1 до $\mathbf{n}$. На каждой итерации цикла выполняются следующие действия:

- если i-й ACE является разрешающим и i-й ACE относится к субъекту, обращающемуся к объекту*, то выполняется присваивание $\mathbf{g} = \mathbf{g} \mid (\mathbf{a}^{(i)} \& \sim \mathbf{d})$. Другими словами, права доступа из $\mathbf{a}^{(i)}$, которые не содержатся также и в $\mathbf{d}$, добавляются в $\mathbf{g}$;
- если i-й ACE является запрещающим и i-й ACE относится к субъекту, обращающемуся к объекту, то выполняется присваивание $\mathbf{d} = \mathbf{d} \mid (\mathbf{a}^{(i)} \& \sim \mathbf{g})$. Другими словами, права доступа из $\mathbf{a}^{(i)}$, которые не содержатся также и в $\mathbf{g}$, добавляются в $\mathbf{d}$.

По окончании цикла маска доступа $\mathbf{g}$ содержит все права, предоставленные субъекту на данный объект, а маска доступа $\mathbf{d}$ — все права доступа к данному объекту, явно запрещенные субъекту.

7. Если $\mathbf{m} \neq \mathbf{0}$ и $\mathbf{m} \neq \mathbf{d}$ (т. е. существует право доступа, относительно которого решение еще не принято), выполняется цикл по i от 1 до $\mathbf{n}$. На каждой итерации цикла выполняются следующие действия:

- если i-й ACE является разрешающим и i-й ACE относится к субъекту, обращающемуся к объекту, то выполняются присваивания $\mathbf{g} = \mathbf{g} \mid (\mathbf{a}^{(i)} \& \mathbf{m} \& \sim \mathbf{d})$, $\mathbf{m} = \mathbf{m} \& \sim \mathbf{g}$. Другими словами, права доступа из $\mathbf{a}^{(i)}$, которые также содержатся в $\mathbf{m}$ и не содержатся в $\mathbf{d}$, добавляются в $\mathbf{g}$. Права доступа, предоставленные субъекту на основании данного ACE, удаляются из $\mathbf{m}$. Если по окончании итерации цикла $\mathbf{m} = \mathbf{0}$, цикл завершается;
- если i-й ACE является запрещающим и i-й ACE относится к субъекту, обращающемуся к объекту, то выполняется присваивание $\mathbf{d} = \mathbf{d} \mid (\mathbf{a}^{(i)} \& \mathbf{m} \& \sim \mathbf{g})$. Другими словами, права доступа

---

* Здесь и далее мы будем говорить, что ACE относится к субъекту S, если:
идентификатор субъекта ACE равен идентификатору субъекта S или
идентификатор субъекта ACE равен идентификатору группы, в которую входит субъект S, или
идентификатор субъекта ACE равен идентификатору относительного субъекта, от имени которого субъект S выступает по отношению к данному объекту.

Compound ACE относится к субъекту S1, олицетворяющему субъекта S2, если идентификатор субъекта-сервера в compound ACE равен S1, а идентификатор субъекта-клиента — соответственно S2.

из $\mathbf{a}^{(i)}$, которые также содержатся в $\mathbf{m}$ и не содержатся в $\mathbf{g}$, добавляются в $\mathbf{d}$. Если по окончании итерации цикла $\mathbf{m} = \mathbf{d}$, цикл завершается.

По окончании цикла, как нетрудно убедиться, маски доступа содержат следующую информацию:

$\mathbf{m}$ — права доступа, запрошенные субъектом и не предоставленные ему;

$\mathbf{g}$ — права доступа, предоставленные субъекту;

$\mathbf{d}$ — права доступа, запрошенные субъектом и явно запрещенные ему.

8. Если $\mathbf{m} = 0$, субъект получает доступ к объекту. При этом маска $\mathbf{g}$ описывает права доступа, предоставленные субъекту. Если $\mathbf{m} \neq 0$, субъект получает отказ в доступе к объекту. При этом маска $\mathbf{g}$ описывает максимальное подмножество запрошенных прав доступа к объекту, которые могут быть предоставлены субъекту.

2.3.8. Назначение дескрипторов защиты создаваемым объектам

При создании в Windows нового объекта ему назначаются атрибуты защиты согласно следующим правилам:

- Если процесс, создающий объект, явно указывает корректный дескриптор защиты для создаваемого объекта, создаваемому объекту назначаются указанный дескриптор защиты.
- Если процесс, создающий объект, указывает, что атрибуты защиты должны быть установлены по умолчанию, или если указанный дескриптор защиты некорректен, дескриптор защиты объекта создается с помощью описанного ниже механизма наследования.
- Если по каким-то причинам наследование дескриптора защиты невозможно, объекту присваивается дескриптор защиты на основе данных, хранящихся в маркере доступа субъекта, создающего объект.

Владельцем созданного объекта, как правило, назначается пользователь, создавший данный объект, т. е. содержимое соответствующего поля создаваемого дескриптора защиты совпадает с содержимым поля TOKEN_USER текущего маркера доступа. Также владельцем объекта может быть назначена группа, для которой в текущем маркере доступа установлен флаг SE_GROUP_OWNER. Другие субъекты доступа владельцами создаваемого объекта назначаться не могут.

Если процесс, создающий объект, указал, что дескриптор защиты объекта должен быть построен по умолчанию, идентификатор

владельца объекта извлекается из поля TOKEN_OWNER текущего маркера доступа. Как правило, данное поле содержит личный идентификатор пользователя, т. е. копию поля TOKEN_USER. В некоторых версиях и конфигурациях Windows пользователи, входящие в группу Administrators, получают в качестве TOKEN_OWNER идентификатор группы Administrators, для которой в списке TOKEN_GROUPS установлен флаг SE_GROUP_OWNER. Объектам, создаваемым с использованием такого маркера доступа, в качестве идентификатора владельца назначается идентификатор группы Administrators.

Первичной группой владельца объекта может назначаться любая группа, идентификатор которой присутствует в текущем маркере доступа. По умолчанию идентификатор первичной группы владельца извлекается из поля TOKEN_PRIMARY_GROUP текущего маркера доступа.

DACL создаваемого дескриптора защиты формируется при наследовании из ACE, входящих в DACL объекта-родителя. То, какие ACE объекта-родителя будут включены в DACL создаваемого объекта, определяется следующими флагами ACE:

- CONTAINER_INHERIT_ACE (**c**) — если этот флаг установлен и создаваемый объект является контейнером, данный ACE должен включаться в DACL создаваемого объекта;
- OBJECT_INHERIT_ACE (**o**) — если этот флаг установлен и создаваемый объект не является контейнером, данный ACE должен включаться в DACL создаваемого объекта;
- NO_PROPAGATE_INHERIT_ACE (**n**) — если этот флаг установлен, при наследовании ACE флаги **c** и **o** сбрасываются. Другими словами, при наличии этого флага ACE наследуется только один раз;
- INHERIT_ONLY_ACE (**i**) — если этот флаг установлен, данный ACE игнорируется при проверке прав доступа к объекту и используется только при наследовании.

Если создаваемый объект не является контейнером, в DACL создаваемого объекта включаются те и только те ACE объекта-родителя, в которых установлен флаг **o**. Все флаги унаследованных ACE сбрасываются. Если маска доступа ACE объекта-родителя содержит отображаемые права доступа, перед наследованием ACE производится их отображение.

Если же создаваемый объект является контейнером, наследуются следующие ACE:

- ACE, в которых установлен флаг **c**, при этом, если в ACE установлен флаг **n**, после наследования флаги **c** и **o** сбрасываются;
- ACE, в которых не установлены ни флаг **c**, ни флаг **n**, но установлен флаг **o**. При этом в унаследованном ACE устанавливается флаг **i**.

Как правило, DACL контейнера содержит две группы ACE:

- ACE для разграничения доступа к объекту — установлен флаг **c**;
- ACE для назначения объектам, создаваемым внутри контейнера — установлены флаги **o** и **i**.

В маски доступа ACE из второй группы не должны включаться специфичные права доступа, поскольку при создании в контейнере объекта, не являющегося контейнером, специфичные права доступа имеют для этого объекта совершенно иной смысл, чем для контейнера. Если внутри одного контейнера создаются объекты разных типов и в DACL контейнера содержится ACE, маска доступа которого содержит специфичные права доступа, то при наследовании данного ACE специфичные права доступа будут для объектов разных типов интерпретироваться по-разному. В большинстве случаев это приводит к ошибкам.

SACL создаваемого объекта наследуется по тем же правилам, что и DACL.

В активном каталоге при наследовании DACL дополнительно учитывается поле GUID2 каждого ACE.

Начиная с версии 2000, в Windows поддерживается функция автоматического наследования изменений в DACL и SACL, управляемая соответствующими флагами дескриптора защиты (по умолчанию они включены для всех объектов). В этом случае при каждом изменении в дескрипторе защиты контейнера наследование списков доступа автоматически повторяется для всех объектов, лежащих внутри контейнера, за исключением тех, в дескрипторах защиты которых в поле флагов установлен бит 12 «запретить автоматическое наследование DACL» или, соответственно, бит 13 «запретить автоматическое наследование SACL». При этом подконтейнеры контейнера обходятся рекурсивно*. Данная функция заметно упрощает администрирование, позволяя администратору одной операцией

---

* Допускается ситуация, когда DACL или SACL дочернего объекта вместо ACE содержит ссылку на соответствующий список доступа в дескрипторе защиты объекта-родителя. В этом случае при изменении дескриптора защиты рекурсивный обход дочерних объектов не нужен, но при каждой проверке прав доступа к дочернему объекту приходится обращаться к дескриптору защиты объекта-родителя.

вносить сходные изменения в дескрипторы защиты сразу всех объектов некоторого поддерева дерева объектов операционной системы, при этом отдельные ветви дерева могут быть заблокированы от таких изменений.

Унаследованные ACE всегда располагаются после явно назначенных и, следовательно, имеют более низкий приоритет.

2.3.9. Мандатный контроль целостности

Начиная с Windows Vista, в операционных системах семейства Windows реализован механизм мандатного контроля целостности (mandatory integrity control, MIC), основанный на формальной модели Биба [4]. Основная идея MIC заключается в том, что объекты операционной системы разделяются на несколько уровней в зависимости от степени доверия к коду обращающегося к объекту процесса, при этом модификация доверенных объектов недоверенными процессами не допускается. Мандатный контроль целостности во многом похож на мандатное управление доступом, разница состоит в том, что мандатный контроль целостности решает задачу обеспечения не конфиденциальности, а целостности информации. Так же, как и при мандатном управлении доступом, при реализации мандатного контроля целостности каждому объекту операционной системы присваивается мандатная метка. В Windows мандатная метка целостности физически представляет собой целое число.

В Windows 7 поддерживаются следующие мандатные метки целостности:

- SECURITY_MANDATORY_UNTRUSTED_RID (0) — минимальный уровень целостности, не используется;
- SECURITY_MANDATORY_LOW_RID (4096) — низкий уровень целостности, не используется;
- SECURITY_MANDATORY_MEDIUM_RID (8192) — средний уровень целостности, назначается процессам прикладных программ по умолчанию;
- SECURITY_MANDATORY_HIGH_RID (12288) — высокий уровень целостности, может быть назначен процессу прикладной программы администратором;
- SECURITY_MANDATORY_SYSTEM_RID (16384) — системный уровень целостности, автоматически назначается всем системным процессам, процессам прикладных программ назначаться не может.

Поскольку числовые значения мандатных меток целостности разделены большими интервалами, в будущих версиях Windows возможно введение промежуточных уровней мандатной целостности,

позволяющих администраторам операционной системы более тонко манипулировать степенью доверия тем или иным процессам.

В имеющихся версиях Windows вышеперечисленные мандатные метки интерпретируются как уровни целостности. Чем больше числовое значение метки, тем более высоким считается уровень целостности. Основная идея мандатного контроля целостности заключается в том, что обращения процессов к объектам, потенциально нарушающие целостность объектов (операции записи и удаления, понимаемые в широком смысле этих слов) разрешаются только в том случае, когда уровень целостности субъекта не уступает уровню целостности объекта, к которому обращается данный субъект. Другими словами, модификация «низкоцелостным» субъектом «высокоцелостного» объекта запрещается независимо от дискреционных полномочий субъекта доступа, от имени которого функционирует процесс, и атрибутов защиты объекта. Таким образом, модификация объектов операционной системы, важных с точки зрения ее безопасности, разрешается только доверенным процессам, выполняющимся от имени специально уполномоченных субъектов.

Кроме того, процесс, выполняющийся на низком уровне целостности, не может получать доступ к процессам, выполняющимся на более высоких уровнях целостности, и, в том числе, не может направлять оконные сообщения их окнам.

Уровень целостности процесса физически хранится в его маркере доступа в списке групп. Каждому определенному в системе уровню целостности соответствует свой идентификатор SID, имеющий вид S-1-16-RID, где RID — числовое значение хранящейся в нем мандатной метки целостности. Помимо характерного вида поля SID, элемент списка групп, содержащий уровень целостности процесса, отличается от других элементов данного списка установленным флагом SE_GROUP_INTEGRITY.

Уровень целостности объекта доступа хранится в его дескрипторе защиты в списке SACL в виде особого элемента ACE типа SYSTEM_MANDATORY_LABEL_ACE. Данный ACE содержит SID уровня целостности вместо SID субъекта доступа. Большинство утилит администрирования, позволяющих просматривать SACL, не позволяют работать с данным ACE.

Каждому процессу, выполняющемуся от имени администратора операционной системы на среднем уровне мандатной целостности, назначается особый ограниченный маркер доступа, отличающийся от обычного маркера доступа следующими деталями:

- все привилегии, кроме пяти наименее опасных, необратимо удалены;
- группа Administrators помечена флагом «только для запрета»;
- в список ограничивающих SID добавлен единственный SID группы Users.

Таким образом, полномочия процесса, выполняющегося на среднем уровне мандатной целостности от имени администратора операционной системы, практически не отличаются от полномочий процесса, выполняющегося от имени непривилегированного пользователя. Это позволяет реализовать принцип минимизации полномочий без создания отдельных учетных записей для повседневной работы и для администрирования операционной системы. По умолчанию все программы, запускаемые администратором, запускаются на среднем уровне мандатной целостности. Если администратору нужно запустить какой-то процесс на высоком уровне мандатной целостности, это может быть сделано с помощью пункта Run as administrator контекстного меню Windows Explorer. Следует отметить, что поясняющий текст для данной строки меню выбран разработчиками операционной системы явно неудачно. Автору неоднократно приходилось слышать удивленные реплики пользователей: «Что значит «от имени администратора»? Я и есть администратор!»

Некоторые утилиты администрирования запускать бессмысленно на среднем уровне мандатной целостности. Для упрощения работы пользователя с такими программами в манифест EXE-файла включен специальный атрибут requestedExecutionLevel, который может принимать следующие значения:

- asInvoker (по умолчанию) — выполнять программу на том же уровне мандатной целостности, на котором выполняется процесс-родитель;
- highestAvailable — выполнять программу на максимально доступном в настоящий момент уровне мандатной целостности;
- requireAdministrator — выполнять программу только на высоком или системном уровнях мандатной целостности.

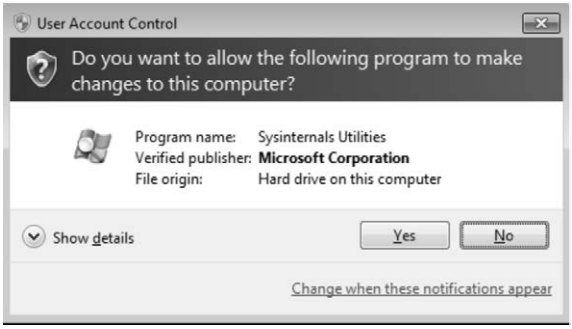
Уровень мандатной целостности, на котором работает процесс, определяется при старте процесса и не может быть изменен в дальнейшем. Некоторые программы могут создавать у пользователя иллюзию, что при нажатии определенной кнопки программа перемещается на более высокий уровень мандатной целостности. Так, например, программа Task Manager при нажатии кнопки Show processes from all users создает свою копию, выполняющуюся на высоком

уровне мандатной целостности, после чего завершает работу. У пользователя создается иллюзия, что процесс Task Manager переместился со среднего уровня мандатной целостности на высокий.

Для того чтобы мандатный контроль целостности реально повышал защищенность операционной системы, в ней должен быть реализован дополнительный механизм, затрудняющий несанкционированный запуск вредоносных приложений на высоком уровне мандатной целостности. В Windows такой механизм введен начиная с Windows Vista, он называется User Account Control (контроль учетных записей, UAC), его назначение и реализация слабо соотносятся с названием механизма.

При включенном UAC любая попытка запуска программы на высоком уровне мандатной целостности сопровождается выдачей пользователю запроса на подтверждение данного действия. Если EXE-файл запускаемой программы входит в состав дистрибутива операционной системы, данный запрос имеет следующий вид:

Если программный файл подписан цифровой подписью компании Microsoft или другой доверенной компании-разработчика, запрос UAC имеет вид



Если же компания-разработчик запускаемого файла неизвестна разработчикам Windows или не указана, запрос UAC имеет вид

Очевидно, в последнем случае окно предупреждения выглядит наиболее угрожающе.

Заметим, что текстовое содержание вопроса, написанного в верхней части приведенных окон, не вполне соответствует сути запроса UAC*. Выдавая запрос UAC, операционная система фактически спрашивает пользователя-администратора, доверяет ли он запускаемой программе в достаточной мере, чтобы позволить ей выполняться на высоком уровне мандатной целостности. При этом совершенно не обязательно, чтобы эта программа вносила изменения в конфигурацию компьютера. Если программа, например, инжектирует мобильный код в адресное пространство системного процесса, никаких изменений в конфигурации компьютера не происходит.

При выдаче запроса UAC видеосистема компьютера, как правило, переключается на особый защищенный рабочий стол, недоступный прикладным программам, выполняющимся на пользовательском рабочем столе**. Непосредственно перед переключением делается снимок экрана пользовательского рабочего стола, этот снимок подвергается затемнению цветов (функция MaskBlt или аналогичная) и назначается защищенному рабочему столу в качестве обоев. Это создает иллюзию, что запрос UAC задается на том же рабочем столе, на котором выполняются прикладные программы этого пользователя.

Администратор операционной системы может управлять «назойливостью» UAC, выбирая из следующих четырех основных конфигураций:

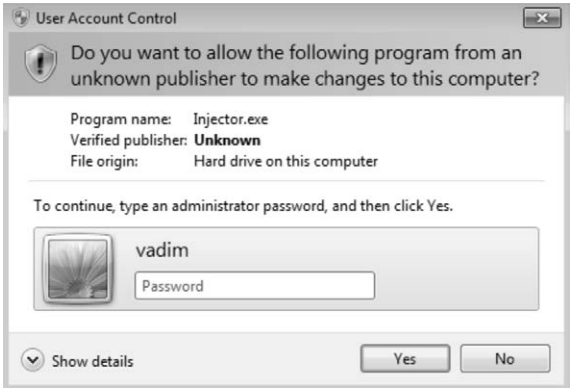
---

* В русской версии Windows этот запрос, кроме того, содержит откровенно безграмотную фразу «внести изменения на этом компьютере».

** Такой же механизм применяется в Windows для защиты пароля, вводимого пользователем при входе в систему, от перехвата программными закладками.

- UAC полностью отключен, все прикладные программы запускаются администратором на высоком уровне мандатной целостности;
- UAC включен, но запросы UAC выводятся на пользовательский рабочий стол;
- UAC включен, запросы UAC выводятся на защищенный рабочий стол, запуск большинства штатных утилит администрирования, входящих в дистрибутив Windows, не сопровождается запросами UAC (конфигурация по умолчанию для Windows 7);
- UAC включен, запросы UAC выводятся на защищенный рабочий стол, запуск любой программы на высоком уровне мандатной целостности сопровождается запросом UAC без каких бы то ни было исключений (конфигурация по умолчанию для Windows Vista).

Многие администраторы, не понимая смысла UAC, воспринимают его как источник назойливых глупых вопросов и отключают сразу после установки Windows. Такая политика безопасности нарушает принцип минимизации полномочий пользователей и тем самым делает конфигурацию операционной системы уязвимой для воздействий вредоносного программного обеспечения, в частности, компьютерных вирусов. Единственное разумное основание отключать UAC имеет место, когда пользователь-администратор в силу индивидуальных психологических особенностей автоматически всегда нажимает кнопку «Да», не задумываясь над заданным вопросом. В этом случае такой администратор должен завести себе для повседневной работы особую непривилегированную учетную запись. Как альтернативное решение, можно установить конфигурацию UAC, в которой администратор, отвечая на запрос UAC, должен не просто нажать кнопку «Да», но и повторно ввести свой пароль:



В большинстве конфигураций Windows запросы UAC не выдаются встроенному пользователю-администратору, автоматически создаваемому в ходе установки операционной системы. Поскольку эта учетная запись предназначена для использования только в экстренных случаях, но не для повседневной работы, данная политика обычно не приводит к существенному снижению безопасности системы.

В целом MIC и UAC являются очень мощным защитным механизмом. Подобно механизмам su и sudo в операционных системах семейства UNIX, MIC и UAC в Windows позволяют осуществлять разумное ограничение полномочий пользователей-администраторов, не прибегая к созданию дополнительных учетных записей. Это делает повседневную работу пользователей-администраторов заметно более удобной.

2.3.10. Элементы изолированной программной среды

Начиная с Windows XP, в подсистему разграничения доступа Windows интегрирован интерфейс SAFER, вносящий в операционную систему элементы изолированной программной среды. Этот интерфейс позволяет распределить установленные в системе программные модули по уровням надежности и определить для каждого уровня круг пользователей, которые могут запускать программные модули данного уровня. В терминах системы правил, приведенной в п. 2.2.2, интерфейс SAFER поддерживает правило 5 изолированной программной среды, но не поддерживает правило 4.

Правила SAFER встроены в групповую политику активного каталога, доступ к правилам SAFER реализуется посредством контейнера Security Settings\Software Restriction Policies групповой политики организационного подразделения, при этом в качестве организационного подразделения может выступать группа пользователей и даже конкретный пользователь. Порядок наследования элементов групповой политики, входящих в данный контейнер, ничем не отличается от порядка наследования, принятого для других контейнеров.

Контейнер Security Settings\Software Restriction Policies содержит два вложенных контейнера.

В контейнере Security Levels перечисляются уровни SAFER, определенные в данной системе. В современных версиях Windows поддерживается три уровня:

- Disallowed — запуск программы запрещен независимо от состояния матрицы доступа;

- Basic User (поддерживается начиная с Windows Vista) — запуск программы может быть разрешен только на уровне мандатной целостности не выше среднего;
- Unrestricted — возможность запуска программы полностью определяется правами доступа субъекта к исполняемому файлу программы, функции SAFER на данном уровне не задействуются.

Для каждого поддерживаемого уровня хранится текстовое описание, а также сведения о том, является ли данный уровень уровнем, заданным по умолчанию. Если по умолчанию задан уровень Disallowed, в системе разрешена загрузка только тех программных модулей, которые явно перечислены в контейнере Additional Rules. Отметим, что установка уровня Disallowed в качестве уровня по умолчанию требует от администратора операционной системы полного и точного перечисления исполняемых модулей установленных приложений в контейнере Additional Rules, в противном случае система станет неработоспособной.

Контейнер Additional Rules содержит правила SAFER, позволяющие изменять уровни конкретных программных модулей. Могут устанавливаться отдельные правила для программных модулей:

- имеющих указанный сертификат;
- имеющих указанное значение хеш-функции;
- загруженных из указанной зоны Internet;
- имеющих заданное полное имя.

Правила, основанные на хеш-функциях, позволяют запрещать загрузку конкретных программных модулей независимо от имени, под которым данный программный модуль загружается в данный момент. Правила данной группы удобно использовать для запрещения запуска пользователями распространенных компьютерных игр (Lines, Zuma и т. п.). Но если пользователь изменяет содержимое запрещенного к исполнению программного файла, например, приписав к нему один дополнительный байт, этот программный файл перестает быть запрещенным.

Следует отметить, что правила SAFER, как и остальные правила групповой политики, могут применяться не сразу после установки, а по прошествии некоторого времени. Если необходимо обеспечить немедленное применение установленных правил SAFER, операционную систему следует перезагрузить.

Интерфейс SAFER документирован в MSDN, однако документация носит крайне фрагментарный характер и не позволяет составить

целостное представление о программной реализации данного интерфейса и возможностях его использования программными средствами сторонних производителей.

2.4. Управление доступом в UNIX

В операционных системах семейства UNIX к объектам доступа относятся файлы, директории, ссылки (links), устройства и именованные каналы (named pipes). В большинстве современных версий UNIX выполняющиеся процессы также рассматриваются как объекты доступа, размещенные в специальной директории /proc. Атрибуты защиты этих объектов могут учитываться при доступе отладчика к адресному пространству процесса, при отправке процессу сигналов и т. п.

К субъектам доступа в UNIX относятся пользователи (включая псевдопользователей) и группы пользователей. В ранних версиях UNIX действовало ограничение, заключающееся в том, что каждый пользователь операционной системы мог входить в одну и только одну группу, т. е. весь список пользователей операционной системы разбивался на некоторое количество непересекающихся групп. В современных UNIX-системах данное ограничение обходится теми или иными способами, однако неявное предположение о том, что пользователь может входить только в одну группу, оказало заметное влияние на внутреннюю структуру подсистемы управления доступом UNIX.

Среди пользователей UNIX выделяется предопределенный пользователь root, обладающий в операционной системе абсолютными полномочиями и имеющий возможность игнорировать любые принятые в системе правила управления доступом.

Пользователи и группы пользователей идентифицируются в UNIX числовыми идентификаторами, уникальными в пределах одного экземпляра операционной системы. Пользователь root всегда имеет идентификатор 0. Идентификаторы пользователей обозначаются стандартным обозначением UID (User ID), идентификаторы групп — GID (Group ID).

Все методы доступа ко всем объектам регулируются в UNIX тремя правами доступа:

- Read (R) — чтение;
- Write (W) — запись;
- Execute (X) — выполнение.

Для файлов смысл всех трех прав доступа вполне очевиден. Для каталогов право Read дает возможность получения списка вло-

женных файлов и подкаталогов, право Write — возможность создания, переименования и/или удаления вложенных файлов и подкаталогов*, право Execute — переход в каталог (команда cd). Для устройств каждое право доступа дает возможность обращения к соответствующей функции драйвера, обслуживающего данное устройство. Для именованного канала право Read требуется для получения информации, право Write — для отправки, право Execute не определено.

Понятие привилегии субъекта доступа в ранних версиях UNIX-систем отсутствовало. В дальнейшем эта концепция вводилась в разных ветвях эволюции UNIX независимо друг от друга. В наиболее распространенной на сегодняшний день ее реализации, применяемой, в частности, в Linux, привилегии называются *возможностями* (*capabilities*). Так, в ядре Linux 2.6.34 определены, в том числе, следующие возможности субъектов доступа:

- CAP_CHOWN — позволяет модифицировать идентификаторы владельца или группы владельца любого объекта;
- CAP_DAC_OVERRIDE — позволяет игнорировать любые ограничения на доступ к объектам, задаваемые списками управления доступом;
- CAP_FOWNER — позволяет игнорировать задаваемые векторами доступа ограничения на доступ к объектам, владельцем которых является текущий субъект доступа;
- CAP_FSETID — позволяет устанавливать в векторе доступа объекта биты SUID и SGID для объектов, владельцем которых не является текущий субъект доступа;
- CAP_KILL — позволяет направлять сигналы процессам других субъектов доступа;
- CAP_SETGID — позволяет устанавливать бит SGID в векторе доступа объекта;
- CAP_SETUID — позволяет устанавливать бит SUID в векторе доступа объекта;
- CAP_SETPCAP — позволяет включать или выключать любую привилегию у любого процесса;
- CAP_NET_BROADCAST — позволяет отправлять широковещательные сетевые пакеты;
- CAP_NET_ADMIN — предоставляет практически неограниченные возможности управления сетевой подсистемой операционной системы;

---

* Вопреки распространенному мнению, для удаления объекта в UNIX вовсе не обязательно иметь право Write на сам этот объект.

- CAP_IPC_OWNER — позволяет процессу осуществлять межпроцессное взаимодействие с процессами, выполняющимися от имени других субъектов доступа;
- CAP_SYS_MODULE — позволяет загружать и выгружать модули расширения ядра операционной системы;
- CAP_SYS_RAWIO — позволяет выполнять низкоуровневые операции ввода-вывода (ioperm, iopl), взаимодействовать с USB-устройствами посредством виртуального файла /proc/bus/usb;
- CAP_SYS_CHROOT — позволяет использовать системный вызов chroot;
- CAP_SYS_PTRACE — позволяет использовать системный вызов ptrace в отношении любых процессов операционной системы;
- CAP_SYS_ADMIN — позволяет выполнять большинство операций по администрированию операционной системы, не регламентируемых другими привилегиями;
- CAP_SYS_BOOT — позволяет перезагружать операционную систему;
- CAP_SYS_NICE — позволяет управлять диспетчеризацией процессов, выполняющихся от имени других субъектов доступа;
- CAP_SYS_RESOURCE — позволяет управлять квотами аппаратных ресурсов, предоставляемых различным компонентам операционной системы, а также прикладным процессам;
- CAP_SYS_TIME — позволяет управлять системными таймерами;
- CAP_SYS_TTY_CONFIG — позволяет управлять консольными терминалами.

Каждому процессу UNIX присваиваются четыре числовых идентификатора:

- UID — идентификатор пользователя, породившего данный процесс;
- GID — идентификатор группы пользователя*, породившего данный процесс;
- EUID — эффективный идентификатор пользователя;
- EGID — эффективный идентификатор группы пользователя.

Обычно EUID совпадает с UID, а EGID — с GID.

---

* Здесь и далее под словосочетанием «группа пользователя» мы понимаем ту единственную группу, в которую мог входить пользователь в старых версиях UNIX. В современных UNIX-системах, где пользователь может одновременно входить в несколько групп, соответствующие аспекты разграничения доступа реализуются более сложно, причем особенности реализации заметно различаются в разных версиях.

Каждый объект UNIX может иметь атрибуты защиты, включающие в себя следующие три элемента:

- идентификатор (UID) владельца объекта;
- идентификатор группы (GID) владельца объекта;
- вектор доступа.

Вектор доступа включат в себя следующие элементы:

- является ли объект каталогом — 1 бит (D);
- бит SUID;
- бит SGID;
- бит sticky;
- права доступа владельца — 3 бита (RWX);
- права доступа пользователей, входящих в группу владельца — 3 бита (RWX);
- права доступа всех остальных пользователей — 3 бита (RWX).

Проверка прав доступа субъекта к объекту осуществляется по следующему алгоритму.

1. Если EUID процесса равен нулю (т. е. процесс выполняется от имени суперпользователя), доступ к объекту предоставляется без каких бы то ни было дополнительных проверок.

2. Если EUID процесса совпадает с UID владельца объекта, права доступа, запрошенные процессом, сравниваются с правами, разрешенными владельцу данного объекта. Если все права, запрошенные процессом, разрешены владельцу объекта, доступ разрешен, в противном случае — запрещен.

3. Если EUID процесса отличается от UID владельца объекта, но EGID процесса совпадает с GID владельца объекта, права доступа, запрошенные процессом, сравниваются с правами, разрешенными пользователям, входящим в группу владельца объекта. Если все права, запрошенные процессом, разрешены группе владельца объекта, доступ разрешен, в противном случае — запрещен

4. Если EUID процесса отличается от UID владельца объекта и EGID процесса отличается от GID владельца объекта, права доступа, запрошенные процессом, сравниваются с правами, разрешенными пользователям, не входящим в группу владельца объекта (всем остальным пользователям). Если все права, запрошенные процессом, разрешены пользователям, не входящим в группу владельца объекта, доступ разрешен, в противном случае — запрещен.

Бит D всегда выставлен у объектов-каталогов и сброшен у всех остальных объектов.

Бит sticky в современных UNIX-системах применяется только в отношении директорий. Если в векторе доступа директории уста-

новлен данный бит, удаление файлов и поддиректорий разрешается только их владельцам. Обычно бит sticky устанавливают на общедоступные для записи директории, предназначенные для обмена файлами между пользователями. Создавать объекты в такой директории могут все пользователи, но удалять уже созданные объекты может только их владелец или суперпользователь.

В процессе функционирования любой операционной системы время от времени возникают ситуации, когда пользователь для выполнения некоторых действий должен получить полномочия, недоступные ему в другое время. Например, пользователь должен иметь возможность изменять свой пароль, но для этого пользователь должен иметь доступ на запись к базе паролей, что недопустимо — тогда пользователь сможет менять и чужие пароли.

В операционных системах семейства UNIX для решения данной проблемы используется механизм SUID/SGID, позволяющий пользователю запустить программу от имени другого пользователя.

Если в векторе доступа исполняемого файла установлен бит SUID, то процесс, порожденный посредством данного файла, в качестве EUID получает не EUID родительского процесса, а UID владельца запускаемого файла. Аналогично, если в векторе доступа файла установлен бит SGID, процесс получает в качестве EGID GID владельца запускаемого файла.

Когда пользователь UNIX-системы меняет свой пароль, он запускает утилиту passwd, владельцем которой является суперпользователь root и в векторе доступа которой установлены биты SUID и SGID. В результате программа passwd запускается с полномочиями суперпользователя, что позволяет ей получить доступ к файлу, в котором хранятся аутентификационные данные пользователей операционной системы.

Схожим образом решаются и другие задачи, требующие временного повышения полномочий пользователя.

Наличие в UNIX суперпользователя root с ничем не ограниченными полномочиями создает серьезные проблемы не только для безопасности, но и для надежности операционной системы. Любая ошибка, допущенная суперпользователем при работе с командной строкой, может стать фатальной. Если, например, в команде rm -rf /*.bak ошибочно поставить пробел между звездочкой и точкой, сама операционная система и все хранящиеся в ней данные будут немедленно и необратимо уничтожены. Большинство администраторов UNIX в настоящее время избегают авторизоваться в операционной системе в качестве суперпользователя на протяжении всего сеанса

работы с системой. Таким образом, принцип минимизации полномочий пользователей соблюдается в UNIX-системах фактически сам собой — после нескольких ошибок администрирования, фатальных для операционной системы, администраторы неизбежно приучаются руководствоваться им в повседневной работе.

Обычно администратор UNIX-системы большую часть времени работает в системе с полномочиями обычного пользователя. Если администратору необходимо выполнить какие-то действия, требующие повышенных полномочий, он может воспользоваться утилитой командной строки su. Эта утилита получает имя и пароль пользователя (на практике чаще всего используется имя root, его можно не вводить, считается, что оно задано по умолчанию), и, если введены корректные данные, инициирует сеанс работы указанного пользователя на текущем терминале, запуская новую копию командного интерпретатора. Прекращение сеанса работы пользователя, инициированного командой su, происходит естественным образом при завершении командного интерпретатора, например, командой exit.

Во многих UNIX-системах пользоваться утилитой su по умолчанию (если администратор не менял настройки аутентификации для программы su) разрешено только пользователям, входящим в группу wheel. Иногда эту группу неформально сравнивают с группой администраторов Windows, а использование утилиты su — с запуском программы Windows на повышенном уровне мандатной целостности.

На практике вместо su чаще применяется утилита sudo, реализующая выполнение от имени суперпользователя (или другого пользователя, если указан флаг -u) команды, являющейся параметром команды sudo. Например, команда sudo mc запускает от имени суперпользователя файловый менеджер Midnight Commander. Среди UNIX-администраторов распространена шутка, что команда sudo подобна слову «пожалуйста» в устной речи (ее применение повышает вероятность успешного выполнения запроса).

При первом вводе sudo пользователь должен повторно ввести свой пароль (именно свой, а не пароль того пользователя, от имени которого будет выполнена команда). После этого в течение некоторого времени (по умолчанию пять минут) пользователь может использовать команду sudo без пароля. Порядок использования команды sudo описывается конфигурационным файлом /etc/sudoers, полное описание формата и содержания которого выходит за рамки настоящего пособия. Мы ограничимся лишь несколькими простыми примерами настроек /etc/sudoers.

```
# Члены группы wheel могут выполнять через sudo любые команды
 %wheel ALL = (ALL) ALL
```

```
# Суперпользователь root может выполнять через sudo любые
# команды, не вводя никаких паролей (обычно суперпользователь
# не нуждается в sudo, но команда sudo может встретиться в
# скрипте, исполняемом от имени суперпользователя)
root ALL = NOPASSWD: ALL
# Пользователь jack может выполнять через sudo любые команды,
# если данный компьютер относится к одной из указанных
# подсетей (файл /etc/sudoers часто копируют на все компьютеры
# защищаемой сети без изменений)
Host_Alias CSNETS = 128.138.243.0 128.138.204.0/24 128.138.242.0
jack CSNETS ALL
```

В настоящее время стандартная для UNIX система управления доступом на основе векторов фиксированной длины оценивается большинством экспертов как устаревшая. Во многих современных версиях UNIX в дополнение к векторам доступа поддерживаются списки доступа переменной длины. Как правило, в реализации списков доступа в UNIX заметно влияние особенностей реализации аналогичного механизма в Windows. Так, например, в операционной системе Mac OS типы, стандартный порядок расположения и флаги наследования элементов управления доступом точно соответствуют аналогичным элементам подсистемы управления доступом Windows. Кроме того, в Mac OS стандартные права read, write и execute дополнены следующими правами доступа:

- delete — удалить объект;
- append — дописать информацию в файл или создать поддиректорию в директории;
- read_attributes — получить атрибуты объекта;
- write_attributes — изменить атрибуты объекта;
- read_extended — получить расширенные атрибуты объекта;
- write_extended — изменить расширенные атрибуты объекта;
- read_permissions — получить список доступа объекта;
- write_permissions — изменить список доступа объекта;
- take_ownership — овладеть объектом (назначить себя новым владельцем объекта).

Данный набор прав доступа почти полностью (кроме права доступа «ожидать объект») совпадает с набором стандартных и специфичных прав доступа, определенных для файловых объектов Windows. Это не случайное совпадение, оно введено преднамеренно, для обеспечения совместимости с активным каталогом Windows. Более того, Mac OS, подобно Windows, поддерживает разрешающие и запрещающие элементы управления доступом. Принятые в Mac OS правила разрешения противоречий при анализе списков доступа в

точности совпадают с аналогичными правилами, принятыми в Windows. Фактически между алгоритмами проверки прав доступа, реализованными в Mac OS и в Windows, есть только одно существенное различие — в Mac OS существует суперпользователь root, имеющий полный доступ ко всем объектам операционной системы, в Windows подобного пользователя нет.

В отличие от большинства операционных систем семейства UNIX, в Mac OS поддерживается назначение атрибутов защиты создаваемым объектам путем наследования. Алгоритм наследования явно скопирован с аналогичного алгоритма Windows и совпадает с ним с точностью до нескольких несущественных деталей.

Анализ списков доступа объекта при проверке прав доступа субъекта к объекту реализован в Mac OS в особом модуле расширения ядра, носящем имя Kauth. Допускается дополнение этого модуля другими модулями, в том числе и других производителей, осуществляющими дополнительные проверки прав доступа субъекта к объекту.

По умолчанию списки доступа в Mac OS не включены ни на каких разделах жесткого диска. Включить или выключить их поддержку для конкретного раздела можно с помощью специальной утилиты командной строки, носящей имя fsaclctl.

Вопросы для самопроверки

1. Что называется объектом доступа, субъектом доступа, методом доступа?
2. Что называется правом доступа субъекта к объекту?
3. Что называется управлением доступа субъектов к объектам?
4. Каким требованиям должны удовлетворять правила управления доступом субъектов к объектам?
5. Как формулируется система правил дискреционного управления доступом?
6. Как в реальных операционных системах кодируется и хранится матрица доступа?
7. Как формулируется система правил изолированной программной среды?
8. Каково основное предназначение изолированной программной среды?
9. Каково основное предназначение мандатного управления доступом?
10. Как формулируется система правил мандатного управления доступом?
11. Какие проблемы возникают при практической реализации мандатного управления доступом в операционной системе?
12. Как решаются проблемы, возникающие при практической реализации мандатного управления доступом в операционной системе?
13. Каковы достоинства и недостатки рассмотренных систем правил управления доступом?
14. В каких ситуациях целесообразно применение мандатного управления доступом, изолированной программной среды?

15. Какие типы объектов доступа операционной системы Windows вы знаете?

16. Каких стандартных псевдопользователей Windows вы знаете?

17. Какие специальные группы пользователей Windows вы знаете?

18. Какие стандартные методы доступа Windows вы знаете?

19. Какие специфичные методы доступа определены в Windows для объектов типа «файл»?

20. Какие специфичные методы доступа определены в Windows для объектов типа «процесс»?

21. Какие специфичные методы доступа определены в Windows для объектов типа «сервис»?

22. Какие специфичные методы доступа определены в Windows для объектов типа «событие»?

23. Как связаны между собой специфичные и стандартные методы и права доступа Windows?

24. Какие отображаемые права доступа определены в Windows?

25. Какие виртуальные права доступа определены в Windows?

26. Какому стандартному методу доступа Windows соответствует не стандартное, а виртуальное право?

27. Какие привилегии субъектов доступа Windows вы знаете?

28. Какие потенциально опасные действия позволяет выполнять пользователю Windows привилегия «Отлаживать программы»?

29. Что такое маркер доступа?

30. Какие основные элементы включает в себя маркер доступа?

31. В какие моменты и каким образом создаются маркеры доступа?

32. Чем отличаются первичные маркеры доступа от маркеров олицетворения?

33. Какая уязвимость ранних версий Windows устранена введением привилегии выполнять олицетворение?

34. Какие уровни олицетворения поддерживаются в современных версиях Windows?

35. Каким образом осуществляется включение и выключение привилегий в маркере доступа?

36. С какой целью в Windows 2003 введено необратимое удаление привилегий из маркера доступа?

37. Что такое ограниченные маркеры доступа?

38. Что такое дескриптор защиты объекта Windows?

39. Какие флаги дескриптора защиты вы знаете?

40. В какой компоненте операционной системы Windows выполняются все проверки прав доступа субъектов к объектам?

41. Что такое DACL дескриптора защиты?

42. Чем различаются объекты и подобъекты активного каталога Windows?

43. Каков смысл полей GUID1 и GUID2 в ACE объектов активного каталога?

44. Какой алгоритм проверки прав доступа субъектов к объектам реализуется в Windows?

45. Какие особенности имеет проверка прав доступа субъекта к объекту Windows, если запрашивается виртуальное право MAXIMUM_ALLOWED?

46. Как при проверке прав доступа субъекта к объекту Windows осуществляется сравнение запрошенных прав доступа с предоставленными правами доступа?

47. Как реализуется в Windows назначение дескрипторов защиты создаваемым объектам?

48. Какие флаги наследования могут устанавливаться в ACE объектов доступа Windows?

49. Что такое автоматическое наследование атрибутов защиты объектов Windows?

50. Что такое мандатный контроль целостности?

51. Какие уровни мандатной целостности поддерживаются в Windows 7?

52. Какие особенности имеет выполнение процессов, выполняющихся в Windows от имени администратора операционной системы на среднем уровне мандатной целостности?

53. Когда и как определяется уровень мандатной целостности, на котором выполняется процесс Windows?

54. Для чего в Windows Vista введен UAC?

55. В чем смысл назойливых вопросов, задаваемых пользователю подсистемой UAC?

56. Какие основные конфигурации UAC поддерживаются в Windows 7?

57. Что надо делать, если администратор Windows в силу индивидуальных особенностей психики все время дает положительные ответы на вопросы UAC?

58. Какие элементы изолированной программной среды поддерживаются в Windows?

59. Как в Windows формулируются правила, устанавливающие правила изолированной программной среды для конкретных программных модулей?

60. Какие субъекты и объекты доступа поддерживаются в операционных системах семейства UNIX?

61. Какие методы и права доступа поддерживаются в операционных системах семейства UNIX?

62. Какие возможности (capabilities) субъектов доступа поддерживаются в современных версиях UNIX?

63. Какие субъекты доступа могут в современных версиях UNIX модифицировать идентификаторы владельца или группы владельца любого объекта?

64. Какие четыре числовых идентификатора, связанные с безопасностью, назначаются каждому процессу UNIX?

65. Как в UNIX осуществляется проверка прав доступа субъектов к объектам?

66. Как работает механизм динамического изменения полномочий SUID/SGID?

67. Какие средства минимизации полномочий пользователей реализуются в операционных системах семейства UNIX?

68. Для чего предназначена и как работает команда sudo в UNIX?

69. Какие дополнительные права доступа субъектов к объектам поддерживаются в Mac OS?

70. Почему дополнительные права доступа, поддерживаемые в Mac OS, так похожи на стандартные и специфичные права доступа, поддерживаемые в Windows?

3 Аутентификация

3.1. Общие сведения

В защищенной операционной системе любой субъект доступа, перед тем как начать работу с системой, должен пройти идентификацию, аутентификацию и авторизацию.

Идентификация субъекта доступа заключается в том, что субъект сообщает системе *идентификационную информацию* о себе (имя, учетный номер и т. д.), и таким образом идентифицирует себя.

Аутентификация субъекта доступа заключается в том, что субъект предоставляет системе помимо идентификационной информации еще и *аутентификационную информацию*, подтверждающую, что он действительно является тем субъектом доступа, к которому относится идентификационная информация.

Пусть, например, пользователь, входя в систему, ввел имя и пароль. В этом случае имя пользователя является идентификационной информацией, а известный только ему пароль — аутентификационной информацией. Вводя пароль, пользователь подтверждает, что введенное имя принадлежит именно ему.

Авторизация субъекта доступа происходит после успешной идентификации и аутентификации. При авторизации субъекта операционная система выполняет действия, необходимые для того, чтобы субъект мог начать работу в системе — загружает индивидуальные настройки пользователя, запускает программу-оболочку и т. п.

Авторизация субъекта не относится напрямую к защите операционной системы. В процессе авторизации решаются чисто технические задачи, связанные с организацией начала работы в системе уже идентифицированного и аутентифицированного субъекта доступа. Заметим, что в ряде источников, в том числе и в MSDN, термин «авторизация» употребляется как синоним термина «управление доступом». Это вносит определенную путаницу в терминологию в данной области.

С точки зрения обеспечения безопасности компьютерной системы процедура аутентификации являются весьма ответственной. Если нарушитель сумел войти в систему от имени другого пользователя, тем самым нарушитель легко получает доступ ко всем объектам системы, к которым имеет доступ данный пользователь. Если при

этом в процессе работы нарушителя с операционной системой подсистема аудита генерирует сообщения о потенциально опасных событиях, то в журнал аудита будет занесено не имя нарушителя, а имя пользователя, от имени которого нарушитель работает в системе.

Хотя аутентификация может осуществляться как для физических пользователей, так и для псевдопользователей, наибольший интерес с точки зрения обеспечения информационной безопасности представляет аутентификация физических пользователей. Если в системе реализуется адекватная политика безопасности, физический пользователь просто не может войти в систему от имени псевдопользователя. Если псевдопользователь обладает большими полномочиями, вход нарушителя в систему от имени псевдопользователя дает нарушителю большие возможности для осуществления несанкционированного доступа, однако на практике осуществить такую атаку обычно крайне трудно. Поэтому в дальнейшем мы будем рассматривать аутентификацию только обычных пользователей.

Обычно подсистема аутентификации операционной системы строится по одной из трех следующих схем:

- парольная аутентификация;
- аутентификация с использованием внешних носителей информации;
- биометрическая аутентификация.

Также возможно использование комбинаций двух или даже всех трех схем аутентификации в одной системе.

3.1.1. Парольная аутентификация

Данная процедура применяется для идентификации и аутентификации пользователей в большинстве современных компьютерных систем. В этом случае для идентификации пользователь должен ввести свое имя, а для аутентификации ввести *пароль* — текстовую строку, известную только ему. Имя пользователя, как правило, назначается ему администратором системы.

Процедура идентификации и аутентификации с использованием пароля предельно проста. Пользователь вводит с клавиатуры имя и пароль, операционная система ищет в списке пользователей запись, относящуюся к данному пользователю, и сравнивает пароль, хранящийся в списке пользователей, с паролем, введенным пользователем. Если запись, относящаяся к входящему в систему пользователю, присутствует в списке пользователей и соответствующий ей пароль совпадает с введенным, считается, что идентификация и аутентификация прошли успешно и начинается авторизация пользователя. В противном случае пользователь получает отказ в доступе

и не может работать с операционной системой до тех пор, пока он не будет успешно идентифицирован и аутентифицирован.

Если идентификация и аутентификация пользователя происходят в процессе входа пользователя на удаленный сервер, имя и пароль пользователя пересылаются по сети (как правило, в зашифрованном виде).

Для обеспечения надежной защиты операционной системы пароль каждого пользователя должен быть известен только этому пользователю и никому другому, в том числе и администраторам системы. На первый взгляд то, что администратор знает пароль некоторого пользователя, не отражается негативно на безопасности системы, поскольку администратор, войдя в систему от имени обычного пользователя, получает права, меньшие, чем те, которые он получит, зайдя в систему от своего собственного имени. Однако, входя в систему от имени другого пользователя, администратор получает возможность обходить систему аудита, а также совершать действия, компрометирующие данного пользователя, что недопустимо в защищенной системе.

Из вышеизложенного следует, что пароли пользователей не должны храниться в операционной системе в открытом виде. Поскольку администратор системы для выполнения своих обязанностей должен иметь доступ к списку пользователей (это необходимо, например, для регистрации новых пользователей), то, если пароли хранятся там открыто, администратор получает к ним доступ. Тем самым администратор получает возможность входить в систему от имени любого зарегистрированного пользователя.

Обычно для шифрования паролей в списке пользователей применяют одну из известных *криптографически стойких хеш-функций* — функций, обладающих следующими тремя свойствами:

- сложность вычисления функции $h(X)$ линейно зависит от размерности входа;
- сложность вычисления обратной функции $h^{-1}(Y)$ экспоненциально зависит от размерности входа. Другими словами, функция h однонаправленна (необратима) — самым быстрым способом вычисления обратной функции является последовательный перебор всех элементов области определения исходной функции;
- для всех возможных значений Y мощности прообразов $h^{-1}(Y)$ близки друг к другу. Другими словами, все значения функции встречаются с примерно одинаковой вероятностью, нет аномальных значений, встречающихся много реже или много чаще, чем другие.

При применении для шифрования паролей криптографически стойкой хеш-функции в списке пользователей хранится не сам пароль X, а *образ пароля* $h(X)$, являющийся результатом применения к паролю хеш-функции. Однонаправленность хеш-функции не позволяет восстановить пароль по образу пароля, но позволяет, вычислив хеш-функцию, получить образ введенного пользователем пароля и, сравнив его с эталонным образом пароля, проверить правильность введенного пароля.

В современных операционных системы чаще всего применяются функции, специально разработанные для использования в качестве криптографически стойких хеш-фукнций. Реже встречается ситуация, года криптографически стойкая хеш-функция создается на основе практически стойкого симметричного криптографического преобразования следующим образом.

Пусть

$$Y = F(X, K),$$

где Y — шифртекст; F — криптографическое преобразование; X — открытый текст; K — ключ шифрования. Тогда функция

$$h(X) = F(C, X),$$

где C — произвольная константа, будет криптографически стойкой хеш-функцией.

Криптографическая стойкость хеш-функции $h(X)$ следует из того факта, что для любого практически стойкого симметричного криптографического преобразования задача восстановления неизвестного ключа по известным открытому и зашифрованному текстам имеет экспоненциальную сложность, а примерно равная мощность прообразов значений хеш-функции вытекает из требования отсутствия плохих ключей, предъявляемого к практически стойким криптографическим преобразованиям.

Хеш-функция, используемая при генерации образов паролей, обязательно должна быть криптографически стойкой. Дело в том, что обеспечить хранение образов паролей в тайне от всех пользователей системы практически невозможно. Администратор операционной системы, используя свои привилегии, легко сможет прочитать образы паролей из файла или базы данных, в которой они хранятся. При сетевой аутентификации пользователя образ пароля передается по открытым каналам связи и может быть перехвачен любым монитором сетевого трафика. Если нарушитель, зная значение хеш-функции (образ пароля пользователя) сможет за приемлемое время подобрать аргумент функции, соответствующий этому

значению (пароль пользователя или эквивалентный ему пароль), то тем самым нарушитель сможет полностью преодолеть защиту, реализуемую подсистемой аутентификации защищаемой системы.

Вышесказанное отнюдь не означает, что образы паролей должны быть общедоступны. Хранение образов паролей в файле или базе данных, к которой имеют доступ только системные процессы, создает дополнительный эшелон защиты, которым не следует пренебрегать.

В процедуре генерации образа пароля обязательно должен участвовать *маркант* — данные, генерируемые случайным образом, не являющиеся секретом и хранящиеся в открытом виде вместе с образом пароля. Другими словами, вместо $h(X)$ в системе должна храниться пара $(M, h(X, M))$. Это необходимо для того, чтобы одинаковым паролям разных пользователей соответствовали разные образы. В противном случае нарушитель может осуществить целый ряд атак, наиболее простая из которых заключается в следующем.

Нарушитель берет какой-либо электронный словарь и для каждого слова из словаря генерирует в точности такую хеш-функцию, какая применяется в атакуемой системе для генерации образа пароля. Слова и соответствующие им хеш-функции сохраняются в базе данных. Перехватив образ пароля пользователя, нарушитель ищет в этой базе слово, соответствующее перехваченному образу пароля. Это и есть искомый пароль или пароль, эквивалентный искомому. Вероятность успешного получения пароля по образу может быть сделана достаточно высокой — для этого нужно всего лишь иметь достаточно большой словарь. При этом для пополнения словаря нарушителю совсем не обязательно иметь доступ к атакуемой системе. Более того, нарушитель может хранить словарь вне атакуемой системы, например, на своем домашнем компьютере.

Данная атака может быть реализована только в том случае, когда одинаковым паролям соответствуют одинаковые образы паролей. Если же при генерации образа пароля используется маркант, описываемая атака нереализуема.

В последние годы получили распространение искусственно усложненные хеш-функции, для которых вычисление каждого значения занимает на современных процессорах несколько миллисекунд или даже секунд. При проверке пароля это замедление несущественно поскольку хеш-функция вычисляется однократно, но при подборе пароля нарушителем, когда хеш-функция отдельно вычисляется при каждом опробовании очередного пароля, искусственное усложнение хеш-функции существенно затрудняет подбор.

Если пользователь, входящий в систему, неправильно ввел свое имя или пароль, система должна выдать ему сообщение об ошибке, не указывая, какая именно информация некорректна. В противном случае подбор пароля существенно упрощается.

Для системы парольной аутентификации существуют две основные угрозы — компрометация пароля и подбор пароля.

Для обеспечения надежной защиты от компрометации паролей подсистема защиты должна удовлетворять следующим требованиям:

- пароль, вводимый пользователем, не отображается на экране компьютера;
- ввод пароля из командной строки недопустим.

Кроме того, пользователи должны быть проинструктированы о:

- необходимости хранения пароля в тайне от других пользователей, включая администраторов системы;
- необходимости немедленной смены пароля после его компрометации;
- необходимости регулярной смены пароля;
- недопустимости записи пароля на бумагу или в файл (кроме случаев, когда угроза компрометации пароля неактуальна, например, если речь идет о доступе пользователя в Интернет с домашнего компьютера).

Что касается подбора паролей, прежде чем перейти к описанию средств защиты от этой угрозы, сначала рассмотрим более подробно методы подбора паролей.

В научно-популярной литературе под подбором паролей обычно понимается ситуация, когда нарушитель опробует варианты пароля непосредственно в той системе, к которой он пытается осуществить несанкционированный доступ. Однако на практике такие попытки подбора паролей применяются крайне редко в силу недостаточной эффективности — простейшие защитные меры наподобие оповещения администратора о нескольких последовательных неудачных попытках аутентификации одного и того же пользователя даже в случае успешного подбора пароля нарушителем затрудняют использование подобранного пароля в течение сколько-нибудь длительного времени. В тех случаях, когда пароль подбирается для однократного доступа к атакуемой системе (например, с целью внедрения программной закладки путем эксплуатации уязвимости, позволяющей несанкционированно повысить полномочия пользователя) и когда есть основания предполагать, что пароль пользователя легко-

подбираем, данный метод может приводить к успеху. Но подобные ситуации являются скорее исключением, чем правилом.

Гораздо чаще применяется так называемый оффлайн-подбор, осуществляемый вне атакуемой системы. В этом случае нарушитель должен предварительно получить из атакуемой системы хеш-образы подбираемых паролей. Это может сделать следующими способами:

- непосредственно извлечь хеш-образы паролей из защищенного хранилища, получив кратковременный административный доступ к атакуемой системе либо загрузив на атакуемой компьютере собственную операционную систему с внешнего носителя информации;
- извлечь хеш-образы паролей из перехваченных сетевых пакетов, в которых они передаются в ходе удаленной аутентификации или смены пароля. Заметим, что в случае применения современных протоколов передачи паролей по сети данный метод, как правило, малоэффективен.

Далее в настоящем разделе мы будем рассматривать только оффлайн-подбор паролей.

При оффлайн-подборе паролей применяются следующие методы.

Тотальный перебор. В этом случае злоумышленник последовательно опробует все возможные варианты пароля. Если пароль длиннее шести-восьми символов, метод малоэффективен.

Как правило, в практических реализациях данного метода учитывается тот факт, что разные символы встречаются в паролях пользователей с разной вероятностью. Например, вероятность того, что в пароле пользователя встретится буква «а», гораздо выше вероятности того, что в пароле встретится символ «^». Согласно различным исследованиям, статистика встречаемости символов в алфавите паролей близка к статистике встречаемости символов в естественном языке.

В ходе тотального перебора паролей нарушитель обычно вначале опробует пароли, состоящие из наиболее часто встречающихся символов, за счет чего время перебора существенно сокращается. Иногда при подборе паролей используется не только статистика встречаемости символов, но также статистика встречаемости биграмм и триграмм — комбинаций двух и трех последовательных символов соответственно.

Для подбора паролей с использованием данного метода в разное время было написано множество программ. Среднее время подбора

пароля из 6–8 символов, не включающего ни цифр, ни знаков препинания, варьируется от нескольких десятков секунд до нескольких суток в зависимости от вычислительной мощности компьютера, на котором осуществляется подбор, и эффективности реализации алгоритма генерации хеш-функции в программе, подбирающей пароли.

Подбор пароля по словарю. Пароли пользователей часто представляют собой слова английского или русского языка. Поскольку пользователю гораздо легче запомнить осмысленное слово, чем бессмысленную последовательность символов, пользователи предпочитают использовать в качестве паролей осмысленные слова. При этом количество возможных вариантов пароля сравнительно невелико. Например, английский язык содержит всего около 100 000 слов (не считая научных, технических, медицинских и других терминов), что в 6,5 раз меньше количества всех комбинаций из четырех английских букв.

При использовании описываемого метода подбора паролей нарушитель вначале опробует в качестве паролей все слова из словаря, содержащего наиболее вероятные пароли. Такой словарь нарушитель может составить сам, а может взять, например, в Internet, где имеется огромное количество подобных словарей, адаптированных для различных стран мира и различных субкультур. Существуют, например, словари паролей толкиенистов, словари паролей, набранных русскими буквами в латинской раскладке клавиатуры (наподобие «gfhjkm»), и т. п. Если подбираемый пароль отсутствует в словаре, нарушиьель опробует всевозможные комбинации слов из словаря, слова из словаря с добавленными к началу и/или к концу одной или несколькими буквами, цифрами и знаками препинания и т. д.

Обычно данный метод применяется в комбинации с предыдущим.

Подбор пароля с использованием знаний о пользователе. Выше уже говорилось, что пользователи стараются использовать легкозапоминаемые пароли. Многие пользователи, чтобы не забыть пароль, выбирают в качестве пароля свое имя, фамилию, дату рождения, номер телефона, номер автомобиля и т. д. В этом случае, если нарушитель хорошо знает пользователя, ему может потребоваться всего лишь 10–20 опробований.

Подбор образа пароля. Если подсистема аутентификации атакуемой системы устроена так, что образ пароля существенно короче самого пароля, нарушитель может подбирать не сам пароль, а его образ. Однако в этом случае нарушитель, подобрав образ паро-

ля, должен получить сам пароль, соответствующий подобранному образу, а это возможно только в том случае, если хеш-функция, применяемая в системе, не обладает достаточной стойкостью.

Пожалуй, самой известной программой подбора паролей является John the Ripper. Эта программа изначально разработана для взлома UNIX-систем, но в дальнейшем в нее были внесены функции подбора паролей для Windows. Программа свободно распространяется вместе с исходным текстом, загрузить программу можно с сайта http://www.openwall.com/john/

Список паролей, поставляемый вместе с программой, включает в себя 3546 строк (версия 1.7.9 от 17.12.2011), это простой текстовый файл, пользователь программы может дописывать в него другие варианты паролей, которые он считает вероятными. Первая десятка самых вероятных паролей, по мнению разработчиков программы, имеет вид:

1. 123456
2. 12345
3. password
4. password1
5. 123456789
6. 12345678
7. 1234567890
8. abc123
9. computer
10. tigger

Интересно, что пустой пароль присутствует в списке лишь на двадцать втором месте.

Правила перебора паролей, используемые John the Ripper, берутся из текстового файла конфигурации, каждая строка этого файла соответствует одному правилу. Пользователь программы может менять заданные по умолчанию правила перебора паролей, добавлять свои правила, удалять правила, менять порядок применения правил (чем ближе строка к началу списка, тем раньше применяется данное правило). Если алгоритм хеширования, применяемый в атакуемой системе, имеет известные слабости (не различаются заглавные и строчные буквы, хеш-образ состоит из нескольких фрагментов, вычисляемых независимо один от другого и т. п.), John the Ripper позволяет описать особые правила для таких случаев.

Существует целый ряд методов, позволяющих несколько уменьшить угрозу компрометации и подбора паролей пользователей. Рассмотрим наиболее распространенные из этих методов.

Ограничение срока действия пароля. При применении данного метода каждый пользователь защищаемой системы обязан менять пароль через определенные промежутки времени. Максимальный срок действия пароля целесообразно ограничить 1-3 месяцами. Менее сильные ограничения не дают желаемого эффекта, а при использовании более сильных ограничений резко повышается вероятность того, что пользователь забудет свой пароль. После того, как срок действия пароля истек, пользователь должен сменить свой пароль в течение некоторого времени (обычно не более суток) после первого входа в систему по истечении этого срока. Если пользователь не сменил пароль за отведенное время, операционная система запрещает ему входить в систему до тех пор, пока это явно не разрешит администратор системы.

Срок действия пароля желательно ограничивать не только сверху, но и снизу. В противном случае пользователь, сменив пароль, может немедленно вернуться к старому паролю, сменив пароль еще один раз.

Также целесообразно проверять при каждой смене пароля уникальность нового пароля. Для этого система должна хранить не только хеш-образ текущего пароля пользователя, но и хеш-образы последних 10–20 паролей, им применявшихся.

Ограничения на пароль. Данный метод заключается в том, что пользователь может выбрать себе в качестве пароля не произвольную строку символов, а только строку, удовлетворяющую определенным условиям. Обычно используются следующие условия:

- длина пароля не должна быть меньше некоторого количества символов. В литературе и в документации по операционным системам обычно рекомендуется запрещать использование паролей короче 6–8 символов, но, с учетом быстрого прогресса вычислительной техники, в настоящее время целесообразно ограничивать длину паролей 10–14 символами;
- в пароль должно входить по меньшей мере 5–7 различных символов;
- в пароль должны входить как строчные, так и заглавные буквы;
- пароль пользователя не должен совпадать с его именем;
- пароль не должен присутствовать в списке «плохих» паролей, хранимом в системе.

Как правило, администраторы могут варьировать эти ограничения как в пределах всей системы, так и для отдельных пользователей. Например, если какое-то имя пользователя используется

для гостевого входа, устанавливать ограничения на используемый пароль нецелесообразно.

При выборе ограничений на пароли не следует забывать, что введение чрезмерно жестких ограничений на пароли приводит к тому, что пользователи начинают массово забывать свои пароли.

Блокировка терминала. При использовании данного метода, если пользователь несколько раз подряд ошибся при вводе имени и пароля, терминал, с которого пользователь входит в систему, блокируется и пользователь не может продолжать дальнейшие попытки входа в систему. Параметрами метода являются следующие значения:

- максимально допустимое количество неудачных попыток входа в систему с одного терминала;
- интервал времени, после которого счетчик неудачных попыток входа обнуляется;
- продолжительность блокировки терминала (как правило, может быть сделана неограниченной, в этом случае блокировка терминала может снять только администратор системы).

Блокировка учетной записи пользователя. Данный метод отличается от предыдущего только тем, что блокируется не терминал, с которого пользователь входит в систему, а учетная запись пользователя.

Ограничения на режимы входа в систему пользователей, пользующихся плохими паролями. В некоторых операционных системах пользователь, назначивший себе явно плохой пароль (например, пустой), может работать в системой только с локальной консоли, но не через сеть.

Генерация паролей системой. В этом случае пользователи не могут самостоятельно придумывать себе пароли — это делает за них сама операционная система. Когда пользователю нужно сменить пароль, он вводит соответствующую команду и получает от системы новый пароль. Если предложенный вариант пароля не устраивает пользователя, он может потребовать другой вариант. Основным преимуществом данного метода является то, что система генерирует пароли случайным образом, подобрать такой пароль практически невозможно. С другой стороны, такие пароли обычно очень трудны для запоминания, это вынуждает пользователей записывать их на бумаге. Если создание текстовой копии паролей не является угрозой безопасности операционной системы (например, если пользователь входит в систему только через Internet со своего домашнего

компьютера), данная модель аутентификации близка к идеальной. В противном случае применять ее нецелесообразно.

Некоторые из перечисленных методов могут применяться в совокупности.

Если предположить, что в качестве пароля равновероятно выбирается любая последовательность букв латинского или русского алфавита, то при длине пароля в 8 или 10 знаков общее число вариантов ключей будет равняться для латинского алфавита соответственно $26^8 = 2 \cdot 10^{11}$ и $26^{10} = 1{,}4 \cdot 10^{14}$, а для русского — $32^8 = 1{,}1 \cdot 10^{12}$ и $32^{10} = 1{,}1 \cdot 10^{15}$. Для того чтобы выработать ключ, обеспечивающий стойкость на уровне стойкости стандарта шифрования DES, понадобится использовать в пароле не менее 11–12 букв алфавита, а для обеспечения стойкости на уровне российского стандарта ГОСТ 28147-89 — не менее 55–56 букв латинского алфавита и соответственно 51–52 русского алфавита. Придумать и запомнить такой пароль типичному пользователю практически невозможно.

На практике максимальная стойкость парольной защиты в первую очередь определяется способностью пользователя запоминать длинные бессмысленные последовательности символов текста. Если предполагается, что пользователи запоминают свои пароли, не записывая их на внешние носители, и что долговременная память пользователей не обладает никакими необычными особенностями, то максимально достижимая криптографическая стойкость парольной защиты составляет не более $10^9 \ldots 10^{12}$ опробований. На современной вычислительной техники перебор такого объема паролей является вполне реальной задачей. Фактически, любой пароль, который может удержать в памяти средний человек, может быть подобран квалифицированным и обеспеченным техникой нарушителем за приемлемое для него время. В связи с этим, парольная защита в настоящее время применяется лишь в тех системах, для которых действующая модель нарушителя исключает из рассмотрения высококвалифицированных и высокомотивированных нарушителей. Другими словами, парольная аутентификация может рассматриваться как эффективная защитная мера для тех систем, для которых актуальны угрозы со стороны только лишь малоквалифицированных и недостаточно мотивированных нарушителей. Однако поскольку подавляющее большинство современных компьютерных систем именно таковы, парольная аутентификация применяется очень широко и фактически является как системой аутентификации по умолчанию во всех распространенных современных операционных системах. В обозримом будущем эта ситуация вряд ли изменится.

Если в системе действует парольная аутентификация, задача выбора стойкого пароля, как правило, возлагается на самого пользователя. Эта задача весьма непроста. Наблюдается четкая закономерность — чем проще пользователю запомнить пароль, тем быстрее этот пароль подбирается программами, подобными John the Ripper. Человеческая память устроена так, что запоминать последовательности символов, являющиеся простыми функциями от осмысленных слов, гораздо проще, чем бессмысленные последовательности символов, которые трудно даже произнести вслух. Например, пароль «baraban2012» запомнить гораздо проще, чем пароль «;F34_iUvSI5». Однако первый из приведенных паролей подбирается практически мгновенно, в то время как подбор второго пароля требует от вычислительной системы нарушителя существенных затрат процессорного времени. Но если пользователь выбрал себе настолько сложный пароль, что сам не может его запомнить, пользователь вынужден записывать его на бумагу или иной внешний носитель информации, что существенно повышает риск компрометации пароля. Очевидно, весьма актуальной является задача построения метода генерации пароля, который, с одной стороны, был бы прост для запоминания, и, с другой стороны, обеспечивал бы приемлемую стойкость парольной защиты.

В разное время предлагалось множество методов генерации паролей, позволяющих добиться соблюдения вышеприведенных требований, хотя и с большой натяжкой. Приведем несколько примеров:

- составлять пароль из двух или трех осмысленных слов, разделенных случайно выбранными знаками препинания (например, «elves!trance&beside»);
- придумывать пароль на основе географического названия, относящегося к стране, в которой статистика естественного языка сильно отличается как от русского, так и английского языка (например, «e/1rbIrbItrbIH», на основе названия чукотского озера Эльгыгытгын);
- вместо географического названия можно использовать запоминающиеся слова из фантастических книг (например, rhabaar drobdt — гномье ругательство из романа Перумова) или с этикеток распространенных товаров (например, дандырылган туз — сухое обезжиренное молоко по-казахски);
- вводить пароль в латинской раскладке клавиатуры, глядя на буквы, соответствующие этим клавишам в русской раскладке (например, «gfhjkm»);

- запомнить несколько строк стихотворения или песни и составить пароль из первых букв каждого слова (например, «IfaqIfafIesaomcswswsfw»).

К сожалению, эффективность подобных методов оставляет желать лучшего. Как только тот или иной метод придумывания паролей становится более-менее популярным, немедленно появляются специализированные словари, ориентированные на подбор паролей, сгенерированных с использованием данного метода, и дальнейшее его применение становится нецелесообразным.

3.1.2. Аутентификация с использованием внешних носителей информации

При использовании данной схемы аутентификации аутентификационная информация хранится на внешнем носителе информации, который может представлять собой пластиковую карту, таблетку touch memory, электронный ключ и т. п. При входе в систему пользователь подключает к компьютеру этот носитель, и система считывает с него идентификационную и аутентификационную информацию пользователя. Далее аутентификация осуществляется, как было описано выше.

Поскольку аутентификационный ключ, хранящийся на внешнем носителе, может быть сделан гораздо более длинным, чем пароль, подобрать такой ключ практически невозможно. Однако угроза компрометации аутентификационных данных по-прежнему остается актуальной. Если процедура аутентификации не предусматривает дополнительных мер защиты, любой обладатель носителя аутентификационной информации, в том числе нарушитель, укравший носитель у легального пользователя системы, может войти в систему с правами пользователя, которому принадлежит этот носитель.

Описываемый механизм аутентификации, как правило, используется в совокупности с предыдущим. При этом пользователь, входя в систему, должен не только предъявить компьютеру носитель аутентификационных данных, но и ввести соответствующий этому носителю пароль (например, числовой пин-код). Формат хранения аутентификационной информации на носителе не должен позволять воспользоваться этой информацией случайному обладателю данного носителя.

Основной угрозой при использовании описываемого механизма аутентификации является угроза кражи носителя аутентификационных данных с последующим его копированием и подбором пароля

на доступ к ключу. Если аутентификационные данные выбираются случайно и формат их хранения на носителе не содержит проверочных полей (контрольных сумм и т. д.), оффлайн-подбор пароля на доступ к носителю аутентификационных данных невозможен — нарушитель просто не сможет сформулировать критерий, позволяющий отличать правильно расшифрованные аутентификационные данные от неправильно расшифрованных, и, следовательно, правильный пароль от неправильного.

Во многих реализациях аутентификации с использованием внешних носителей применяются следующие дополнительные меры защиты:

- защита ключевого носителя от копирования;
- блокировка или уничтожение аутентификационной информации после определенного количества неудачных попыток ввода пароля на доступ к ключу.

При технически грамотной реализации механизма хранения аутентификационных данных эти меры никак не влияют на стойкость реализуемой системы аутентификации, но они существенно повышают привлекательность данной системы для потенциальных заказчиков и потому применяются очень широко.

Если в качестве носителя ключевой информации применяются электронные ключи Touch Memory или пластиковые карты Memory Card, перечисленные меры защиты неприменимы. Хотя существующие средства защиты от копирования и позволяют несколько затруднить копирование носителя информации, любой из перечисленных носителей может быть скопирован за считанные минуты. Поскольку проверку правильности пароля на доступ к ключу осуществляет защищаемая операционная система, то, если нарушитель подбирает пароль с помощью специальной программы, подсчитывать количество неудачных попыток также невозможно.

В отличие от перечисленных носителей информации, интеллектуальные пластиковые карты Smart Card содержат, помимо энергонезависимой оперативной памяти, микропроцессор, способный выполнять криптографические преобразования информации. Поэтому интеллектуальные карты способны самостоятельно проверять правильность пароля на доступ к ключевой информации, и при аутентификации пользователя с использованием интеллектуальной карты проверку пароля на доступ к карте производит не операционная система, а сама карта. Интеллектуальная карта может быть запрограммирована на стирание хранимой информации после превышения максимально допустимого количества неправильных попыток ввода

пароля, что не позволяет подбирать этот пароль без частого копирования карты, что весьма дорого.

В целом использование для аутентификации пользователей не только паролей, но еще и внешних носителей информации позволяет заметно повысить защищенность операционной системы. Но, с другой стороны, при использовании в защищаемой системе аутентификации с использованием внешних носителей информации у администраторов и пользователей возникает целый ряд проблем.

Проблема генерации ключей

Ключ аутентификации должен быть достаточно длинным и абсолютно случайным. Генераторы псевдослучайных последовательностей не могут применяться для генерации ключей аутентификации, поскольку все современные генераторы псевдослучайных последовательностей работают по линейной конгруэнтной схеме:

$$r_{n+1} = ar_n + b(\mathrm{mod}\, m),$$

где r_0, r_1, ... — генерируемая псевдослучайная последовательность; a, b, m — параметры алгоритма.

Обычно m выбирается равным максимальному целому числу для данной аппаратной платформы, a и b — специально подобранные числа, обычно простые. Как правило, a и b выбираются из списков «хороших» a и b, приведенных в знаменитой монографии Кнута [5].

Нетрудно видеть, что значение каждого следующего члена псевдослучайной последовательности однозначно определяется значением ее предыдущего члена и, следовательно, количество разных псевдослучайных последовательностей, которые способен выдать генератор, в точности совпадает с количеством возможных начальных заполнений линейной рекурренты. Другими словами, стойкость защиты, реализуемой с использованием данного ключа, не зависит от того, сколько членов псевдослучайной последовательности входит в состав ключа, и всегда равна m. Обычно m выбирается равным 2^{32} или 2^{64}, что в современных условиях явно недостаточно.

В некоторых программах генерации ключей в качестве случайных данных берется содержимое случайных областей оперативной памяти, точное значение текущего времени, статистическая информация о функционировании системы в данный момент (например, количество байт, прочитанных каждым процессом за последний час) и т. д. Однако статистические эксперименты показывают, что все эти величины далеко не случайны и распределение ключей, генерируемых с их помощью, далеко от дискретного равномерного.

Идеальным средством генерации истинно случайных ключей является аппаратный генератор случайных чисел. Это устройство может быть сделано довольно простым, однако на сегодняшний день оно пока не находит широкого применения. Имеющиеся реализации, даже разработанные крупными компаниями, имеют кустарный или полукустарный характер, качество выдаваемых ими случайных последовательностей не всегда подвергается достаточно подробному анализу. При использовании аппаратных генераторов случайных чисел следует иметь в виду, что последовательность, выдаваемая генератором, должна обязательно проверяться статистическими критериями, например, описанными в вышеупомянутой монографии Кнута. Известны случаи, когда генераторы случайных ключей из-за неисправности или необычных условий эксплуатации (например, при очень низкой температуре) выдавали последовательности, далекие от случайных (например, последовательность, составленная из одних нулей) и эти последовательности, не будучи проверены, реально использовались в качестве ключей аутентификации. Для примера приведем так называемую «батарею Кнута» — четыре теста, которые наиболее часто используются для оценки качества случайных последовательностей, имеющих дискретное равномерное распределение.

1. Вероятность встретить на i-м месте последовательности значение x должна быть одинакова для всех i и x.

2. Вероятность встретить на i-м и $(i+1)$-м местах последовательности пару значений (x_1, x_2) должна быть одинакова для всех i, x_1 и x_2.

3. Усеченный покер-тест. Вероятности встретить в случайно выбранной четверке подряд идущих членов случайной последовательности все нижеперечисленные наборы комбинаций значений:

- четыре разных значения;
- одну пару одинаковых значений;
- две разные пары попарно одинаковых значений;
- три одинаковых значения и четвертое значение, отличное от этих трех;
- все четыре одинаковые значения,

должны соответствовать полиномиальной схеме независимых испытаний с соответствующими параметрами.

4. Вероятность встретить в любом месте анализируемой последовательности монотонно возрастающий (убывающий) участок длины n должна быть равна 2^{-n}.

При отсутствии в системе аппаратного генератора случайных чисел для генерации ключей аутентификации можно воспользоваться одним из двух методов, дающих удовлетворительные результаты:

1. **Метод обезьяны** — пользователю предлагается набрать на клавиатуре произвольный текст, который в дальнейшем рассматривается как случайная последовательность. Иногда этот метод дополняется требованием пользователю подвигать мышью произвольным образом. Распределение данных, полученных от пользователя при использовании данного метода, далеко от равномерного, случайность этих данных также оставляет желать лучшего. Поэтому в практических реализациях, данные, полученные от пользователя, предварительно прогоняются через функцию усложнения, например, симметричное криптографическое преобразование с обратной связью, которое несколько повышает качество ключа. Хорошей практикой при применении метода обезьяны является выбор вдвое большей длины ключа, чем необходимо для обеспечения требуемой стойкости. Это дает запас стойкости, компенсирующий недостаточное качество ключей. Метод обезьяны используется в большинстве современных программных генераторов ключей;

2. **Метод тетриса** — пользователю предлагается поиграть в компьютерную игру, в которой нужно регулярно нажимать на клавиши, при этом распределение интервалов времени между нажатиями на клавиши хорошо поддается аналитическому моделированию. Практически идеальным вариантом является игра «Тетрис». Пользователь играет в компьютерную игру, система фиксирует интервалы между нажатиями клавиш и проводит преобразование, приводящее распределение случайной величины к дискретному равномерному. Для генерации ключа обычно требуется около пяти минут. Данный метод позволяет генерировать ключи гораздо более высокого качества, чем метод обезьяны, но требует заметно больше времени для генерации одного ключа. Поэтому данный метод в настоящее время применяется редко.

При тестировании программного генератора ключа, построенного по одной из двух вышеописанных схем, желательно участие в экспериментах нескольких операторов разного пола и возраста, обладающих разным уровнем навыков работы с компьютером. Обязательным является участие в эксперименте оператора-нарушителя, сознательно стремящегося заставить генератор вырабатывать плохие ключи.

Если необходимо одномоментно сгенерировать несколько десятков или сотен ключей, могут применяться так называемые алгорит-

мы размножения ключей. На вход такого алгоритма поступает один ключ, а на выходе выдается N ключей, обладающих следующим свойством: если известно $N-1$ ключей, задача восстановления последнего N-го ключа не может быть решена быстрее, чем полным перебором всех возможных вариантов ключа. Естественно, если нарушитель знает первый ключ, поступивший на вход алгоритма, нарушитель легко сможет восстановить все N сгенерированных ключей. Поэтому по окончании процедуры генерации ключей первый ключ обязательно уничтожается.

Проблема рассылки ключей. Если защищаемая сеть является территориально распределенной, задача рассылки ключей из центрального офиса организации в территориальные подразделения решается весьма неочевидно. Руководителю службы безопасности организации приходится делать нелегкий выбор между двумя основными вариантами:

- разрешить каждому подразделению генерировать ключи самостоятельно, при этом центру трудно контролировать качество вырабатываемых ключей и отслеживать возможные организационные нарушения в ходе генерации ключей. Кроме того, в каждом подразделении организации приходится иметь свой комплект оборудования для прошивки аппаратных носителей информации;
- генерировать ключи централизованно и доставлять в региональные подраздделения курьерами, при этом возрастают расходы на командировки сотрудников службы безопасности.

Между этими двумя крайностями возможны промежуточные варианты, например, генерировать ключи централизованно, рассылать их в подразделения по защищенным каналам связи, а непосредственно прошивку носителей проводить уже на местах. Однако среди всех возможных вариантов нет ни одного однозначно лучшего, каждый вариант имеет свои недостатки.

Проблема смены ключей. Так же, как и пароли, ключи аутентификации время от времени должны подвергаться смене. Если ключи генерируются централизованно и количество ключей в организации достаточно велико, в деятельности службы безопасности время от времени происходят авралы, когда в течение короткого времени необходимо сгенерировать большое количество новых ключей. Еще тяжелее переносится ситуация, когда массовая смена ключей происходит незапланированно, при обнаружении факта массовой компрометации ключей в результате успешного взлома защищаемой сети. Помимо «естественных» авральных работ по лик-

видации уязвимостей защиты и локализации последствий ее взлома, на службу безопасности в этом случае возлагается дополнительная задача по срочной генерации полного набора ключей для всех пользователей организации и рассылке новых ключей в подразделения.

Проблема потерянных ключей. Пользователи, пользующиеся внешними носителями аутентификационных ключей, время от времени теряют эти носители. Пользователь, потерявший носитель с ключом, не может работать с компьютерными системами, при этом потеря ключа часто обнаруживается в момент, когда доступ данного пользователя в компьютерную сеть должен быть обеспечен незамедлительно. Поэтому в организации должна быть предусмотрена процедура быстрой внеплановой смены пользовательского ключа. Если ключи генерируются централизованно, в каждом подразделении должен иметься запас ключей для экстренной замены, должны быть предусмотрены механизмы блокирования потерянных ключей, не допускающие их использование нарушителями.

В целом аутентификация на основе внешних носителей ключа не является идеальной схемой аутентификации. Хотя данная схема может быть сделана гораздо более стойкой, чем парольная, оборотной стороной высокой стойкости являются проблемы, встающие перед администраторами при практической реализации политики безопасности, основанной на данной схеме аутентификации.

3.1.3. Биометрическая аутентификация

Каждый человек обладает своим неповторимым набором биометрических характеристик, к которым относятся отпечатки пальцев, рисунок сетчатки, рукописный и клавиатурный почерк и т. д. Эти характеристики могут быть использованы для аутентификации пользователя.

Если аутентификация пользователя осуществляется на основе биометрических характеристик, угрозы компрометации и подбора аутентификационных данных перестают быть актуальными — подделать биометрические характеристики человека, как правило, настолько сложно и дорого, что затраты злоумышленника на проникновение в защищенную систему превысят выгоды от такого проникновения. Таким образом, механизм аутентификации пользователя на основе биометрических характеристик создает практически непреодолимую защиту на этапе аутентификации.

С другой стороны, практическая реализация данного механизма аутентификации неизбежно создает ряд проблем, к основным из которых относятся следующие:

- поскольку псевдопользователи не являются людьми, и, следовательно, не имеют биометрических характеристик, для их аутентификации должен поддерживаться альтернативный механизм. При этом система должна гарантировать, что этот альтернативный механизм не будет применяться для аутентификации обычных пользователей;
- при двух последовательных входах в систему одного и того же человека его биометрические характеристики никогда в точности не совпадают. Поэтому в процессе аутентификации приходится использовать математический аппарат теории распознавания образов, при этом приходится мириться с неизбежностью ошибок как первого рода (успешный вход от чужого имени), так и второго рода (отказ в доступе легальному пользователю);
- большинство биометрических характеристик человека постепенно меняются со временем, что заставляет регулярно корректировать эталонный образ аутентифицирующей информации;
- биометрические характеристики человека могут испытывать резкие кратковременные изменения. Например, если пользователь оцарапал палец, система аутентификации, основанная на отпечатках пальцев, не сможет его аутентифицировать до тех пор, пока царапина не заживет;
- аутентификация пользователя на основе биометрических характеристик требует применения дорогостоящей аппаратуры для получения образа используемой характеристики и сложных вычислительных алгоритмов для сравнения этого образа с эталонным, что приводит к большим затратам финансовых средств на создание системы аутентификации и вычислительных ресурсов компьютера на ее поддержание.

Десять лет назад перечисленные недостатки делали биометрическую аутентификацию крайне неудобной для практического использования. Сейчас в данной области наблюдается большой прогресс и, вероятно, в ближайшее время, по мере повышения надежности и снижения стоимости устройств биометрической аутентификации, доля биометрических систем среди всех систем аутентификации заметно вырастет.

3.2. Аутентификация в UNIX

Список пользователей UNIX хранится в текстовом файле /etc/passwd, формат которого допускает его редактирование обычным текстовым редактором. В старых версиях UNIX зашифрованные

образы паролей хранились в том же файле, в большинстве современных UNIX-систем образы паролей хранятся в отдельном файле, также расположенном в каталоге /etc. Имя этого файла различается в разных версиях UNIX, чаще всего применяется имя /etc/shadow. В некоторых UNIX-системах пароль каждого пользователя хранится в своем отдельном файле, например, в ALT Linux пароль пользователя username хранится в файле /etc/tcb/username/shadow.

Каждая строка файла /etc/passwd соответствует одному пользователю или псевдопользователю. Она включает в себя следующие семь полей, разделенных двоеточиями:

- имя пользователя (login);
- хеш-образ пароля пользователя или признак того, что хеш-образ пароля хранится в другом файле, общем для всех пользователей (*) или признак того, что хеш-образ пароля хранится в отдельном файле (x);
- UID пользователя;
- GID первичной группы пользователя;
- комментарий к учетной записи пользователя (произвольная текстовая строка);
- домашняя директория пользователя;
- командный интерпретатор, который должен запускаться на терминалах, на которых работает данный пользователь (для псевдопользователей обычно указывается /dev/null).

Например:

```
root:x:0:0:System Administrator:/root:/bin/bash
bin:x:1:1:bin:/:/dev/null
    ...
_mpd:x:117:431:Music Player Daemon (MPD):/var/lib/mpd:/dev/null
user:500:500:user:/home/user:/bin/bash
```

Управление списком пользователей, как правило, осуществляется не путем ручного редактирования соответствующих файлов, а с использованием специальных утилит командной строки useradd, userdel, usermod, groupadd, groupdel, groupmod либо графических утилит, оснасток панели управления и т. п.

Начиная со второй половины 1990-х годов, в подавляющем большинстве UNIX-систем подсистема аутентификации строится в соответствии с архитектурой Pluggable Authentication Module (PAM), разработанной Open Software Foundation (OSF). Программные модули, входящие в подсистему аутентификации UNIX-системы, делятся на две основные группы:

- клиенты (приложения и демоны), пользующиеся услугами PAM (login, passwd, rlogin, telnetd, ftpd и т. п.);

- программные модули, предоставляющие услуги PAM. Обычно эти модули представляют собой библиотеки, размещаемые в директории /lib/security.

С точки зрения клиента аутентификации его взаимодействие с PAM реализуется посредством так называемых примитивов — системных вызовов, передающих управление соответствующим модулям PAM. Поддерживаются шесть основных примитивов, сгруппированных в четыре подсистемы:

- pam_authenticate (подсистема auth) — аутентифицировать пользователя;
- pam_setcred (подсистема auth) — авторизовать пользователя (установить UID, идентификаторы групп, квоты ресурсов и т. д.);
- pam_acct_mgmt (подсистема account) — проверить, доступна ли учетная запись пользователя для авторизации (не устарел ли пароль, не заблокирована ли учетная запись и т. п.);
- pam_open_session (подсистема session) — начать сеанс работы пользователя с операционной системой;
- pam_close_session (подсистема session) — завершить сеанс работы пользователя с операционной системой;
- pam_chauthtok (подсистема password) — назначить пользователю новые аутентификационные данные.

Клиентские программы, обращающиеся к PAM, могут не знать о том, какой метод был использован при аутентификации некоторого конкретного пользователя. Вся техническая реализация процедуры аутентификации пользователя инкапсулирована внутри PAM, клиенту выдается лишь самая общая информация о результатах выполнения PAM того или иного примитива.

Каждый модуль PAM должен содержать функции-обработчики примитивов хотя бы одной подсистемы. Когда клиентская программа вызывает тот или иной примитив, PAM обращается к функциям-обработчикам данного примитива одного или нескольких своих модулей. Важно отметить, что вся значимая функциональность PAM сосредоточена в модулях, сама система PAM не выполняет никаких действий, связанных с аутентификацией, а всего лишь вызывает в определенной последовательности заданные функции заданных модулей и принимает решение на основании возвращаемых ими результатов. Конкретный порядок того, какие функции каких модулей должны вызываться и как должны интерпретироваться результаты их вызова, определяется содержимым конфигурационных файлов PAM.

В ранних версиях PAM вся конфигурация PAM описывалась единственным файлом /etc/pam.conf. В более поздних версиях каждой клиентской программе сопоставляется индивидуальный файл конфигурации, расположенный в директории /etc/pam.d. Имя этого файла обычно совпадает с именем клиента, так, например, файл конфигурации службы SSH обычно имеет имя /etc/pam.d/sshd. В каждом файле конфигурации содержится последовательность строк, определяющих, какие PAM-модули должны использоваться клиентом и каким образом это должно осуществляться. Конфигурация PAM, используемая клиентскими программами по умолчанию, описывается в файле /etc/pam.d/other.

В качестве примера файла конфигурации PAM рассмотрим файл /etc/pam.d/login.conf из дистрибутива Simply Linux 5.0,1, описывающий порядок использования PAM программой login:

```
auth required pam_securetty.so
auth include system-auth
auth required pam_nologon.so
account include system-auth
password include system-auth
session required pam_loginuid.so
session include system-auth
session optional pam_lastlog.so nowtmp
session optional pam_motd.so
session optional pam_mail.so
session optional pam_console.so
```

Каждая строка файла соответствует одному модулю PAM и имеет вид

type control path arguments

Поле type описывает тип модуля. Каждой подсистеме примитивов PAM соответствует одноименный тип модуля PAM. При вызове клиентской программой примитива, принадлежащего определенной подсистеме, будут последовательно вызваны соответствующие функции-обработчики из всех модулей PAM соответствующего типа, при этом порядок вызова функций соответствует порядку перечисления модулей в конфигурационном файле.

Поле control описывает порядок интерпретации системой PAM результата обращения к данному модулю. Если функция-обработчик возвращает значение PAM_SUCCESS, считается, что модуль отработал успешно, в противном случае считается, что модуль сообщил об ошибке. Эти сведения обрабатываются в зависимости от значения поля control следующим образом:

- requisite — если модуль сообщает об ошибке, выполнение текущего примитива немедленно прерывается, клиентская программа получает сообщение об ошибке;
- required — если модуль сообщает об ошибке, клиентская программа получает сообщение об ошибке, но выполнение примитива продолжается (возможно, обнаружатся и другие ошибки, клиенту будет полезно знать обо всех);
- sufficient — если модуль отработал успешно, выполнение примитива считается успешно завершенным, функции-обработчики последующих модулей не вызываются;
- optional — модуль реализует второстепенные функции, не влияющие на общий статус выполнения запроса, статус обращения к модулю игнорируется.

Кроме перечисленных значений, поле control может также принимать два специальных значения include и substack, указывающих, что при интерпретации конфигурационного файла на место данной строки должны быть последовательно подставлены все строки типа type из конфигурационного файла path. Поле arguments в этом случае игнорируется. Значения include и substack различаются характером досрочного прерывания запроса при неуспешном обращении к requisite-модулю или успешном обращении к sufficient-модулю. Если дополнительный файл конфигурации был включен в текущий файл командой include, запрос в этих случаях прерывается окончательно, а если командой substack — отменяются лишь невыполненные обращения к модулям вложенного файла.

Поле path содержит путь к соответствующему модулю PAM, поле arguments — текстовую строку произвольного вида, которая будет передана данному модулю в качестве параметра.

Модули каждого типа вызываются в соответствующих ситуациях поочередно, соответственно порядку их перечисления в конфигурационном файле. Так, в вышеприведенном примере при аутентификации пользователя вначале выполняется модуль pam_securetty.so, затем выполняется процедура аутентификации по умолчанию, заданная в конфигурационном файле system-auth (учитываются только строки этого файла, описывающие модули типа auth), и если по завершении этой процедуры решение еще не принято — выполняется модуль pam_nologon.so.

Рассмотрим еще один пример конфигурационного файла PAM — файл /etc/pam.d/su, описывающий порядок взаимодействия с PAM утилиты su:

```
auth sufficient pam_rootok.so no_warn
```

```
auth sufficient pam_self.so no_warn
auth requisite pam_group.so no_warn group=wheel root_only\
fail_safe
auth include system
account include system
password required pam_deny.so
session include system
```

Нетрудно видеть, что примитивы подсистем account и session в данной конфигурации выполняются для команды su на общих основаниях, в соответствии с общесистемной политикой по умолчанию, описанной в файле system. Примитив pam_chauthtok (единственный прототип из подсистемы password) утилитой su в обычных условиях не вызывается, а если он все-таки будет вызван (например, в результате попытки нарушителя проэксплуатировать программную уязвимость), система PAM немедленно сообщит об ошибке (модуль pam_deny.so всегда сообщает об ошибке независимо от контекста обращения к нему).

Теперь рассмотрим порядок обработки примитивов из подсистемы auth, например, примитива pam_authenticate, непосредственно реализующего аутентификацию пользователя.

Модуль pam_rootok.so, указанный в первой строке, завершается успешно, если он выполняется от имени суперпользователя root, и неуспешно в противном случае. Таким образом, суперпользователь root в данной конфигурации PAM имеет возможность выполнять команду su, не проходя дополнительной аутентификации (поскольку модуль описан как sufficent, успешный результат проверки влечет за собой успех всей процедуры аутентификации).

Модуль pam_self.so, указанный во второй строке, завершается успешно тогда и только тогда, когда запущена процедура повторной аутентификации того же пользователя, от имени которого выполняется текущий процесс, и неуспешно в противном случае. Таким образом, если команда su user дана пользователем user (например, при исполнении скрипта), повторное подтверждение подлинности пользователя user не требуется.

Третья строка рассматриваемого конфигурационного файла задействует модуль pam_group.so, который в данном случае проверяет, что пользователь, давший команду su, является членом группы wheel. Поскольку данный модуль описан как requisite, любая попытка выдачи команды su пользователем, не входящим в группу wheel, завершится фатальной ошибкой.

Система PAM предоставляет мощные средства, позволяющие гибко настраивать подсистему аутентификации для самых различ-

ных ситуаций. Если, например, в некоторой конфигурации операционной системы необходимо полностью запретить выполнение команды su, достаточно вписать в начало файла /etc/pam.d/su строку

auth requisite pam_deny.so

А если администратор операционной системы считает целесообразным разрешить бесконтрольное выполнение команды su любыми пользователями без повторной аутентификации (например, на тестовой виртуальной машине) — вместо вышеприведенной строки следует вставить следующую:

auth sufficient pam_permit.so

Модули PAM позволяют реализовать в операционной системе не только парольную аутентификацию пользователей, но и аутентификацию с использованием внешних электронных носителей информации (например, pam_usb.so), а также биометрическую аутентификацию (например, pam_frprint.so). С помощью модуля pam_listfile.so можно реализовать в одной операционной системе различные схемы аутентификации для конкретных списков пользователей, групп пользователей, хостов и т. п. Например, следующая строка конфигурационного файла:

auth sufficient pam_listfile.so item=user sense=deny file=/etc/users.deny

запрещает вход в систему пользователям, имена которых перечислены в файле /etc/user.deny.

Из других часто употребляемых модулей PAM следует упомянуть:

- pab_abl.so — поддерживает автоматическую блокировку попыток аутентификации с удаленных хостов, с которых приходит слишком много неудачных попыток аутентификации (вероятна попытка подбора аутентификационных данных);
- pam_alreadyloggedin.so — позволяет пропускать повторную аутентификацию пользователя, уже аутентифицированного на другой локальной консоли;
- pam_chroot.so — помещает авторизуемого пользователя в «песочницу», предоставляя ему в качестве корневого каталога файловой системы какой-то другой каталог;
- pam_echo.so — выдает текстовое сообщение, заданное параметром модуля. Применяется главным образом для выдачи предупреждающих сообщений, правил пользования тем или иным сервисом и т. п.;
- pam_exec.so — запускает указанную программу. Может, применяться, например, для автоматического монтирования домаш-

него каталога в ходе авторизации пользователя, если этот домашний каталог физически расположен на удаленном сервере;
- pam_ldap.so — реализует схему аутентификации, совместимую со сквозной аутентификацией в доменах Windows;
- pam_lockout.so — запрещает вход в систему пользователю, имя которого указано в параметре модуля;
- pam_mkhomedir.so — автоматически создает для авторизуемого пользователя домашнюю директорию, если таковая не существует;
- pam_namespace.so — позволяет предоставлять авторизуемому пользователю индивидуальные образы некоторых системных директорий (чаще всего применяется для директории /tmp);
- pam_pwdfile.so — требует проводить дальнейшую аутентификацию с использованием не системного списка пользователей, а альтернативного списка, хранящегося в указанном файле;
- pam_unix.so — реализует UNIX-аутентификацию по умолчанию (вычисляет хеш-функцию пароля, заданную текущей конфигурацией операционной системы, и сравнивает с значением, хранящимся в /etc/shadow). Практически всегда включается в скрипт аутентификации по умолчанию, включаемый в конфигурационные файлы всех клиентов через директиву include.

Таким образом, архитектура PAM позволяет администратору UNIX-системы гибко настраивать подсистему аутентификации в соответствии с реальными потребностями конкретной вычислительной сети и конкретного экземпляра операционной системы. Все, что нужно сделать администратору, — установить в системе необходимые модули PAM и обеспечить корректное взаимодействие между ними. В результате подсистема аутентификации может быть легко адаптирована к самым разным технологиям аутентификации, принятым в конкретной организации. В частности, на базе PAM может быть реализована аутентификация с использованием смарт-карт или биометрическая аутентификация. Для этого достаточно, чтобы разработчик аппаратного устройства, используемого для аутентификации, поставлял вместе со своим устройством соответствующий модуль PAM.

3.3. Аутентификация в Windows

В Windows задачи идентификации, аутентификации и авторизации пользователей решаются специальной *подсистемой аутентификации*. Подсистема аутентификации Windows делится на три уровня — верхний, средний и нижний. Средний уровень подсистемы

аутентификации пользуется услугами нижнего уровня и предоставляет услуги верхнему.

Верхний уровень подсистемы аутентификации Windows включает в себя процесс аутентификации winlogon.exe и так называемые *провайдеры аутентификации* — заменяемые библиотеки, реализующие большую часть высокоуровневых функций процесса аутентификации.

Процесс Winlogon представляет собой обычный процесс, выполняющийся от имени псевдопользователя SYSTEM. Данный процесс автоматически запускается при старте операционной системы и остается активным до выключения питания или перезагрузки. При аварийном завершении Winlogon происходит аварийное завершение работы всей операционной системы («синий экран»). Таким образом, подменить Winlogon в процессе функционирования операционной системы практически невозможно.

При входе пользователя в систему с локального или удаленного терминала провайдер, обслуживающий данный терминал, получает от пользователя его имя и пароль. В Windows 2003 и более ранних версиях по умолчанию использовался единственный провайдер аутентификации — библиотека msgina.dll, которая осуществляет все взаимодействие между локальным пользователем и процессом аутентификации. Начиная с Windows Vista, в Windows реализован более сложный механизм взаимодействия провайдеров аутентификации и процесса Winlogon, основанный на COM-интерфейсах и позволяющий одновременно использовать несколько различных провайдеров аутентификации.

Вход локального пользователя в систему обычно выполняется в Windows следующим образом.

1. Провайдер аутентификации получает от пользователя идентификационную и аутентификационную информацию. В стандартной конфигурации операционной системы в качестве идентификационной информации используется текстовое имя, а в качестве аутентификационной информации — текстовый пароль. Также возможно применение для аутентификации внешних носителей ключевой информации или биометрических характеристик пользователя.

2. Провайдер аутентификации генерирует запрос на аутентификацию, передавая необходимые данные на средний уровень подсистемы аутентификации с помощью системного вызова LsaLogonUser или одной из более высокоуровневых программных оберток этого системного вызова. Если аутентификация прошла успешно, создается маркер доступа пользователя.

3. Если маркер доступа пользователя создан успешно, провайдер аутентификации осуществляет авторизацию пользователя, запуская процесс userinit.exe от имени аутентифицированного пользователя. Для этого используется системный вызов CreateProcessAs User, который отличается от вызова CreateProcess только тем, что запускаемому процессу назначается маркер доступа, отличный от маркера доступа процесса-родителя. В данном случае процессу userinit назначается только что созданный маркер доступа авторизуемого пользователя.

4. Процесс userinit загружает индивидуальные настройки пользователя из его профиля, запускает программу-оболочку пользователя (чаще всего это Проводник Windows) и после этого завершает работу.

В средний уровень подсистемы аутентификации Windows входит *локальный распорядитель безопасности* (*local security authority*, *LSA*) и так называемые *пакеты аутентификации* — заменяемые библиотеки, реализующие большую часть низкоуровневых функций аутентификации.

Локальный распорядитель безопасности представляет собой сервисный процесс lsass.exe, выполняющийся от имени псевдопользователя SYSTEM. Аварийное завершение LSA приводит к аварийному завершению работы всей операционной системы. Так же, как и Winlogon, LSA передоверяет большинство своих функций заменяемым библиотекам. Стандартная схема аутентификации Windows NT обслуживалась пакетом аутентификации MSV 1.0 (msv1_0.dll), а начиная с Windows 2000, стандартным является пакет аутентификации Kerberos.

Пакет аутентификации осуществляет аутентификацию пользователя в процессе обработки системного вызова LsaLogonUser. Аутентификация производится следующим образом.

1. Пакет аутентификации получает от верхнего уровня подсистемы аутентификации имя и пароль пользователя и генерирует образ пароля.

2. Используя услуги нижнего уровня подсистемы аутентификации, пакет аутентификации получает информацию, необходимую для проверки пароля, и проверяет пароль. Проверка пароля может вестись как путем простого сравнения хеш-образа введенного пароля с эталонным хеш-образом (протоколы LanManager, NTLM), так и путем более сложных криптографических процедур (Kerberos).

3. Если введенный пароль признан корректным, LSA получает от нижнего уровня подсистемы аутентификации информацию о том,

может ли данный пользователь начинать в данный момент работу с данной рабочей станцией (не устарел ли пароль, не заблокирован ли бюджет пользователя и т. д.).

4. В случае положительного результата проверки LSA формирует маркер доступа пользователя, получая необходимую информацию от нижнего уровня подсистемы аутентификации.

5. LSA передает сформированный маркер доступа верхнему уровню подсистемы аутентификации.

Нижний уровень подсистемы аутентификации Windows отвечает за хранение в системе учетной информации о пользователях, в том числе и эталонных образов паролей. При аутентификации пользователя нижний уровень подсистемы аутентификации передает среднему уровню эталонный образ пароля пользователя, а при авторизации — список групп и привилегий пользователя.

Аутентификация при удаленном входе в систему осуществляется в целом по той же схеме за исключением того, что на верхнем уровне вместо процесса Winlogon может выступать произвольная пара клиент + сервер. Существует специальный интерфейс SSPI (Security Support Provider Interface), обеспечивающий взаимодействие приложений Windows с LSA в ходе аутентификации. В Windows поддерживается пять стандартных провайдеров сетевой аутентификации.

NTLM (NT Lan Manager, поддерживается начиная с Windows NT 4.0)

Данный провайдер является наиболее универсальным, он может применяться практически в любой ситуации, когда необходимо осуществить удаленную аутентификацию. Алгоритм сетевого взаимодействия выглядит в общих чертах следующим образом.

1. Клиент направляет серверу имя пользователя в открытом виде (в NTLM идентификационная информация пользователя не считается секретом).

2. Сервер генерирует случайное число от 0 до 65535 и высылает его клиенту.

3. Клиент зашифровывает это число, используя в качестве ключа хеш-функцию пароля пользователя и высылает результат шифрования серверу. В качестве алгоритма шифрования используется модификация алгоритма DES.

4. Сервер проводит аналогичные вычисления и сравнивает результат с полученным от клиента. Если результаты совпали, аутентификация признается успешной, в противном случае — неуспешной. В домене Windows сервер может передоверить данный шаг

алгоритма контроллеру домена.

Kerberos (поддерживается начиная с Windows 2000)

Протокол аутентификации Kerberos весьма сложен, и детальное его рассмотрение выходит за рамки настоящего пособия. Отметим лишь основные его достоинства и недостатки.

Основным достоинством протокола Kerberos является его чрезвычайно высокая стойкость. Даже перехватив весь трафик информационного взаимодействия всех участников процесса аутентификации, получить несанкционированный доступ к ресурсам любого из участников информационного обмена практически невозможно. Особеннно повышают защищенность Kerberos жесткие ограничения, которые данный протокол устанавливает на время аутентификации. Большинство данных, которые могут быть перехвачены нарушителем, устаревают спустя считанные минуты, некоторые данные могут сохранять актуальность несколько часов. В любом случае, современная вычислительная техника, включая суперкомпьютеры, не позволяет осуществлять взлом используемых криптографических алгоритмов за приемлемое время.

Основным недостатком Kerberos является то, что аутентификация по этому протоколу требует некоторой подготовительной работы и не может быть выполнена произвольной парой клиент + сервер. Как минимум, клиент и сервер должны выбрать сервера-посредника, которому они оба доверяют и который заранее осведомлен о некоторых характеристиках клиента и сервера. Поэтому протокол Kerberos может эффективно применяться только в централизованно управляемых локальных сетях с априорно известными топологией и структурой. Существуют модификации Kerberos для работы в Internet и даже для локальной аутентификации, но это, фактически, профанация — в этих режимах Kerberos не имеет никаких преимуществ по сравнению с более примитивными протоколами типа NTLM, но вычислительная сложность криптографических преобразований Kerberos существенно выше.

Negotiate (поддерживается начиная с Windows 2000)

Этот провайдер обеспечивает автоматический выбор провайдера между NTLM и Kerberos. В современных версиях Windows Negotiate выбирает NTLM лишь в тех случаях, когда использование Kerberos невозможно по техническим причинам. Как правило, приложения обращаются не к NTLM и не к Kerberos, а именно к Negotiate.

Digest (поддерживается начиная с Windows XP)

Данный протокол аутентификации специально предназначен для веб-приложений. Подробно спецификации протокола изложены в RFC 2617. Функционально Digest похож на NTLM, для криптографических преобразований в Digest может использоваться поточный шифр RC4 с длиной ключа 40, 56 или 128 бит, а также DES либо Triple DES.

Schannel (поддерживается начиная с Windows NT 4.0 SP4)

Этот провайдер поддерживает протоколы сетевой аутентификации TLS 1.0 и SSL 3.0, а также устаревший протокол PCT 1.0. К криптографическим преобразованиям, используемым Schannel, относятся RC2, RC4, DES, Triple DES, RSA, DHE, MD5, SHA.

Помимо перечисленных стандартных провайдеров, Windows может работать и с нестандартными провайдерами, созданными вне Microsoft. Интерфейсы, используемые провайдерами аутентификации, практически полностью документированы.

Начиная с Windows 2000, Windows поддерживает специальный унифицированный интерфейс, обслуживающий внешние носители ключей аутентификации.

Подсистема аутентификации Windows обладает достаточно большой гибкостью и позволяет администраторам операционной системы настраивать различные параметры аутентификации как для отдельных пользователей системы, так и для всех пользователей в совокупности.

Администраторы Windows могут вводить следующие ограничения на пароли пользователей:

- минимальный и максимальный срок действия пароля;
- минимальную допустимую длину пароля;
- минимальное допустимое количество смен пароля до первого повторения;
- должна ли при смене пароля пользователем проводиться проверка качества нового пароля;
- разрешать ли хранение в системе образов паролей, допускающих обратное расшифрование (обычно это запрещено, но может потребоваться для некоторых сетевых сервисов);
- максимально допустимое количество неудачных попыток входа в систему;
- срок, по истечении которого счетчик неудачных попыток входа в систему обнуляется;
- срок, на который пользователю запрещается вход в систему в случае превышения максимально допустимого количества неудачных попыток входа в систему (может быть неограниченным,

в этом случае запрет на вход пользователя в систему может быть снят только администратором);
- могут ли пользователи самостоятельно менять пароль в случае истечения максимального срока его действия, или они должны уведомлять администратора о случившемся;
- могут ли использоваться пустые пароли при сетевой аутентификации;
- какие протоколы аутентификации могут использоваться программами, выполняющимися в данной системе;
- должно ли выдаваться пользователю, осуществляющему локальный вход в систему, имя пользователя, осуществлявшего локальный вход в систему в предыдущий раз;
- обязан ли пользователь нажимать Ctrl-Alt-Del перед вводом имени и пароля;
- какое сообщение должно выдаваться пользователю перед входом в систему;
- за какое время до истечения срока действия пароля пользователь начинает получать предупреждения от операционной системы;
- обязательно ли использование внешних носителей ключа при локальной аутентификации;
- как операционная система должна реагировать на извлечение внешнего носителя аутентификационной информации из соответствующего устройства (варианты: никак не реагировать, заблокировать консоль, завершить сеанс работы пользователя с операционной системой);
- через какое время неактивное сетевое соединение должно принудительно разрываться;
- должен ли принудительно завершаться сеанс работы пользователя с операционной системой по истечении разрешенного интервала времени;
- разрешено ли использовать при генерации образа пароля устаревшую хеш-функцию Lan Manager, обладающую низкой криптографической стойкостью;
- должен ли список зарегистрированных пользователей и групп считаться конфиденциальным.

Механизм автоматической блокировки (lock out) пользователя при превышении максимально допустимого количества неудачных попыток входа в систему не распространяется на пользователя Administrator.

Для каждого конкретного пользователя могут быть установлены следующие флаги:

- пользователь обязан сменить пароль при ближайшем входе в систему — обычно применяется для только что зарегистрированных пользователей;
- пользователь не может менять свой пароль — обычно применяется для «групповых» пользователей (например, Guest);
- на пользователя не распространяется ограничение максимального срока действия пароля — обычно применяется в совокупности с предыдущим требованием;
- пользователь не может работать в системе — применяется для временного блокирования учетной записи пользователя (например, на период отпуска или болезни пользователя).

Для пользователей домена могут быть введены следующие дополнительные требования к процедурам идентификации, аутентификации и авторизации:

- время работы пользователя с операционной системой может быть ограничено, в этом случае вход пользователя в систему разрешается только в отведенные для этого часы;
- количество компьютеров, с которых пользователь может входить в домен, может быть ограничено, администратор может явно перечислить компьютеры, с которых разрешен вход пользователя в домен;
- может быть установлена автоматическая блокировка учетной записи пользователя по истечении определенного времени;
- может быть указана программа или скрипт, автоматически выполняемая при входе пользователя в систему;
- может быть ограничена продолжительность терминальных сессий пользователя (подключений к терминальному серверу с удаленных компьютеров через Remote Desktop или другую подобную программу);
- может быть включена функция удаленного контроля («подсматривания») администратора за действиями пользователя в ходе работы с терминальным сервером. В зависимости от настроек данной функции, вмешательство администратора в сессию пользователя может происходить либо только с разрешения пользователя, либо без разрешения, незаметно для пользователя. Вмешательство администратора может быть ограничено просмотром пользовательского терминала либо ничем не ограничено — в этом случае администратор может управлять клавиатурой и мышью вместе с пользователем;

- могут быть установлены особые правила использования пользователем удаленного подключения к домену через модем или VPN.

Помимо вышеперечисленных требований и ограничений, при идентификации и аутентификации пользователя также осуществляется проверка одной из следующих пяти так называемых привилегий входа (привилегии входа, строго говоря, не являются привилегиями, поскольку никогда не добавляются в маркер доступа пользователя и, следовательно, не учитываются монитором безопасности объектов операционной системы):

- входить в систему интерактивно;
- входить в систему через сеть;
- входить в систему через терминальный сервер;
- запускать сервис от своего имени;
- запускать пакетное задание (batch job) от своего имени.

То, какая «привилегия» должна проверяться, определяется провайдером при вызове функции LogonUser. Например, если четвертый параметр этой функции равен LOGON32_LOGON_SERVICE, это означает, что пользователь входит в систему в качестве сервиса, т. е. запускает сервис от своего имени, и должна быть проверена «привилегия» запускать сервисы от своего имени.

Начиная с Windows 2000, для каждой привилегии входа поддерживается два списка — белый и черный. Чтобы субъект доступа получил некоторую привилегию входа, он должен быть прямо или косвенно упомянут в соответствующем белом списке и ни прямо, ни косвенно не упомянут в соответствующем черном списке.

В лесу доменов Windows все вышеперечисленные параметры интегрированы в групповую политику дерева доменов, что позволяет при необходимости централизованно управлять параметрами аутентификации всех компьютеров определенных подразделений корпоративной сети либо всей корпоративной сети в целом.

Выше была изложена стандартная схема идентификации и аутентификации пользователя в Windows, которая применяется при использовании стандартных провайдеров и пакетов аутентификации. Однако поскольку и провайдеры, и пакеты аутентификации являются заменяемыми компонентами подсистемы аутентификации, администратор операционной системы может, установив нестандартный провайдер или пакет аутентификации, реализовать в Windows любую другую схему аутентификации. Для этого необходимо всего лишь разместить в системной директории Windows необходимые библиотеки и внести изменения в соответствующие ключи реестра.

Вопросы для самопроверки

1. Что такое идентификация, аутентификация, авторизация?

2. Какие три основные схемы аутентификации вы знаете?

3. Каково важнейшее преимущество парольной аутентификации по сравнению с другими схемами?

4. Как должен храниться в операционной системе эталонный образ пароля, предназначенный для проверки пароля в ходе аутентификации?

5. Как можно построить криптографически стойкую хеш-функцию на основе практически стойкого симметричного криптографического преобразования?

6. Почему при хешировании паролей следует использовать маркант?

7. С какой целью применяемые при хешировании паролей хеш-функции иногда подвергаются искусственному усложнению, сильно замедляющему время вычисления функции?

8. Чем отличаются онлайн-подбор паролей от оффлайн-подбора?

9. Как строятся алгоритмы тотального перебора паролей?

10. Как строятся алгоритмы подбора паролей по словарю?

11. Какие пароли являются самыми распространенными в мире?

12. Зачем нужно ограничивать сроки действия паролей?

13. Какие ограничения обычно накладываются на содержание паролей?

14. В каких ситуациях целесообразно применять генерацию паролей не пользователем, а самой операционной системой?

15. Какова максимальная криптографическая стойкость системы парольной аутентификации, достижимая при условии, что пользователь не обладает феноменальной способностью запоминать длинные бессмысленные последовательности букв и цифр?

16. К каким негативным последствиям приводит неоправданное завышение требований к длине и качеству паролей пользователей операционной системы?

17. Какие способы придумывания трудно подбираемых, но легко запоминаемых паролей вы знаете?

18. Каково важнейшее преимущество схемы аутентификации, основанной на внешних электронных носителях аутентификационных данных?

19. Какие дополнительные преимущества дает применение в качестве носителя аутентификационных данных интеллектуальной пластиковой карты, содержащей бортовой микропроцессор?

20. Какие проблемы возникают при генерации ключей, предназначенных для аутентификации с использованием внешних электронных носителей аутентификационных данных?

21. Почему при генерации ключей для внешних электронных носителей аутентификационных данных нельзя применять стандартные алгоритмы программной генерации псевдослучайных последовательностей?

22. Какие тесты можно применять для оценки качества случайной последовательности?

23. Какие методы получения истинно случайных последовательностей с помощью программных генераторов вы знаете?

24. Какие проблемы возникают при рассылке ключей, предназначенных для аутентификации с использованием внешних электронных носителей аутентификационных данных?

25. Какие проблемы возникают при плановой и экстренной замене ключей, предназначенных для аутентификации с использованием внешних электронных носителей аутентификационных данных?

26. Что такое биометрическая аутентификация, каково ее важнейшее преимущество перед другими схемами аутентификации?

27. Какие проблемы возникают при практической реализации биометрической аутентификации?

28. Где и в каком формате обычно хранится список пользователей в операционных системах семейства UNIX?

29. В каких файлах хранится текущая конфигурация PAM?

30. Какие типы PAM-модулей вы знаете?

31. Как поле control конфигурационной записи PAM-модуля влияет на порядок интерпретации результата обращений PAM к этому модулю?

32. Как сконфигурировать PAM, чтобы команда su предоставляла полномочия суперпользователя без ввода соответствующего пароля?

33. Как с помощью PAM можно реализовывать в одной операционной системе разные схемы аутентификации для конкретных списков пользователей и групп пользователей?

34. Что необходимо для реализации на базе PAM аутентификации с использованием внешних электронных носителей аутентификационных данных или биометрической аутентификации?

35. На какие три уровня распадается подсистема аутентификации Windows? Какие основные задачи решаются на каждом из них?

36. Перечислите основные достоинства и недостатки протокола аутентификации Kerberos.

37. Какие ограничения на пароли пользователей могут применяться в Windows?

38. В каких ситуациях пользователь Windows обязан нажимать Ctrl-Alt-Del перед каждым вводом пароля на вход в систему?

39. Какие индивидуальные параметры аутентификации могут быть установлены для конкретного пользователя Windows?

40. Какие привилегии входа поддерживаются в Windows?

41. Как можно построить в Windows нестандартную схему аутентификации пользователя?

4 Аудит и обнаружение вторжений

4.1. Общие сведения

Процедура аудита применительно к защищенным компьютерным системам заключается в регистрации в специальном журнале, называемом *журналом аудита* или *журналом безопасности*, событий, которые могут представлять опасность для системы. Пользователи системы, обладающие правом чтения этого журнала, называются *аудиторами*.

Необходимость включения в защищенную систему функций аудита диктуется следующими обстоятельствами.

- Подсистема защиты компьютерной системы, не обладая интеллектом, неспособна отличить случайные ошибки пользователей от злонамеренных действий. Например, то, что пользователь в процессе входа в систему ввел неправильный пароль, может означать как случайную ошибку при вводе пароля, так и попытку подбора пароля. Но, если сообщение о подобном событии занесено в журнал аудита, администратор, просматривая этот журнал, возможно, сможет установить, что же имело место на самом деле — ошибка легального пользователя или атака злоумышленника. Если пользователь ввел неправильный пароль всего один раз — это явная ошибка. Если же пользователь пытался угадать собственный пароль 20–30 раз — это явная попытка подбора пароля.
- Администраторы защищаемой системы должны иметь возможность получать информацию не только о текущем состоянии системы, но и о том, как она функционировала в недавнем прошлом. Журнал аудита дает такую возможность, накапливая информацию о важных событиях, связанных с безопасностью операционной системы.
- Если администратор обнаружил, что против защищаемой системы проведена успешная атака, ему важно выяснить, когда она была начата и каким образом она осуществлялась. При наличии в системе подсистемы аудита не исключено, что вся необходимая информация содержится в журнале аудита.

Некоторые эксперты по компьютерной безопасности полагают, что привилегия работать с подсистемой аудита не должна предостав-

ляться администраторам операционной системы. Другими словами, множества администраторов и множество аудиторов не должны пересекаться. При этом создается ситуация, когда администратор не может выполнять несанкционированные действия без того, чтобы это тут же стало известно аудиторам, что повышает защищенность системы от несанкционированных действий администраторов.

Однако на практике отделение аудиторов от администраторов применяется редко, что обусловлено следующими причинами:

- обслуживание аудита на одном компьютере занимает существенно меньше времени, чем администрирование одного компьютера. В большой организации на одного аудитора должно приходиться 5–20 администраторов. Если организация невелика и в ней предусмотрены всего 1–2 штатные должности системных администраторов, создавать отдельную штатную должность аудитора нецелесообразно;
- администраторы и аудиторы часто вступают в приятельские отношения, в результате чего аудитор далеко не всегда докладывает начальнику об обнаруженных злоупотреблениях администраторов. Бывает, что аудиторы не только прикрывают злоупотребления администраторов, но и сами участвуют в них. Если отделение аудиторов от администраторов не подкреплено организационно-административными мерами, оно может оставаться чисто формальным — аудитор знает пароль администратора, администратор знает пароль аудитора, и оба они пользуются полномочиями друг друга по мере необходимости;
- при наличии в организации выделенного аудитора, недостаточно загруженного работой, аудитор часто вводит чрезмерно строгий контроль за действиями пользователей, негативно сказывающийся на моральном климате в коллективе. Пользователей сильно нервирует ситуация, когда малейшее нарушение правил работы с компьютерной системой (например, однократное посещение развлекательного сайта Internet) немедленно становится известным службе безопасности и влечет за собой неотвратимое наказание. Практика показывает, что выделенные аудиторы более склонны к реализации «параноической» политики безопасности, чем аудиторы, совмещающие свои обязанности с обязанностями системного администратора.

Обычно отделение аудиторов от администраторов практикуется только в крупных организациях (корпорациях, банках и т. п.). При этом обязанности аудитора часто берет на себя должностное лицо,

которому непосредственно подчиняются системные администраторы, либо его заместитель.

Подсистема аудита операционной системы должна удовлетворять следующим требованиям.

- Добавлять записи в журнал аудита могут только псевдопользователи, от имени которых выполняются системные процессы. Если предоставить эту возможность какому-то физическому пользователю, данный пользователь получит возможность компрометировать других пользователей, добавляя в журнал аудита соответствующие записи.
- Ни один субъект доступа, в том числе и сама операционная система, не имеет возможности редактировать или удалять отдельные записи в журнале аудита.
- Только пользователи-аудиторы, обладающие соответствующей привилегией, могут просматривать журнал аудита.
- Только пользователи-аудиторы могут очищать журнал аудита. После очистки журнала в него автоматически вносится запись о том, что журнал аудита был очищен, с указанием времени очистки журнала и имени пользователя, очистившего журнал. Система аудита должна поддерживать возможность сохранения журнала аудита перед очисткой в другом файле.

Для ограничения доступа пользователей к журналу аудита не всегда достаточно обычных средств разграничения доступа. В подавляющем большинстве систем администратор локальной или корпоративной сети, используя свои привилегии, может прочитать и изменить содержимое любого файла, хранящегося на любом компьютере сети. Поэтому, если принятая в системе политика безопасности предусматривает разделение администраторов и аудиторов, то для ограничения доступа к журналу аудита желательно применять дополнительные средства защиты, например, криптографические.

Политика аудита — это совокупность правил, определяющая то, какие события должны регистрироваться в журнале аудита. Для обеспечения надежной защиты операционной системы в журнале аудита должны обязательно регистрироваться следующие события:

- попытки входа/выхода пользователей из системы;
- попытки изменения списка пользователей;
- попытки изменения политики безопасности, в том числе и политики аудита.

При определении политики аудита не следует ограничиваться регистрацией событий только из перечисленных классов. Окончательный выбор того, какие события должны регистрироваться в жур-

нале аудита, возлагается на самих аудиторов. При этом политика аудита в значительной степени определяется спецификой информации, хранимой и обрабатываемой в операционной системе, и не зная этой специфики, давать какие-либо рекомендации бессмысленно.

При выборе оптимальной политики аудита следует учитывать ожидаемую скорость заполнения журнала аудита. Если политика аудита предусматривает регистрацию слишком большого числа событий, это не только не повышает защищенность операционной системы, но, наоборот, снижает ее. Если новые записи добавляются в журнал аудита слишком часто, аудиторам будет трудно выделить в огромном объеме малозначительной информации те события, которые представляют реальную угрозу безопасности системы. Кроме того, чем быстрее заполняется журнал аудита, тем чаще его нужно очищать и тем больше вероятность его переполнения.

Политику аудита не следует рассматривать как нечто неизменное, заданное раз и навсегда. Политика аудита должна оперативно реагировать на изменения в конфигурации операционной системы, в характере хранимой и обрабатываемой информации, и, особенно, на выявленные попытки атаковать защищаемую операционную систему. Если, например, с помощью аудита было обнаружено, что имела место попытка преодолеть защиту операционной системы, но основные принципы реализации этой атаки остались неясными, целесообразно изменить политику аудита таким образом, чтобы при дальнейших попытках осуществлять аналогичные атаки аудиторы получали более подробную информацию.

В целом политика аудита является своего рода искусством, и выбор оптимальной политики в значительной мере определяется опытом и интуицией аудитора.

В некоторых конфигурациях операционных систем подсистема аудита помимо записи информации о зарегистрированных событиях в специальный журнал предусматривает возможность интерактивного оповещения аудиторов об этих событиях. Когда аудитор начинает работу с операционной системой одного из компьютеров сети, операционные системы других компьютеров получают соответствующие сообщения, после чего при каждой регистрации события в журнале аудита одного из компьютеров копия информации об этом событии передается на терминал, с которым работает аудитор. Как правило, для реализации подобного механизма требуется установка дополнительного программного обеспечения.

Множество событий, регистрируемых в журнале аудита, не обязательно должен совпадать с множеством событий, информация о

которых передается аудиторам интерактивно. Целесообразно так организовать интерактивное оповещение аудиторов, чтобы аудиторы получали оповещение только о наиболее важных событиях — в противном случае аудиторам будет трудно выделить в сплошном потоке сообщений по-настоящему полезную информацию.

Данная дополнительная функция подсистемы аудита позволяет аудиторам более оперативно реагировать на попытки преодоления злоумышленниками защиты операционной системы и тем самым повышает общую защищенность системы.

4.2. Системы обнаружения вторжений

Логическим развитием концепции аудита является *система обнаружения вторжений* (СОВ), или *intrusion detection system* (IDS) — специализированное программное или программно-аппаратное средство, предназначенное для выявления успешных и неуспешных попыток осуществления несанкционированного доступа к ресурсам компьютерной системы или сети. Системы обнаружения вторжений обладают двумя характеристическими отличиями от обычных систем аудита:

- система обнаружения вторжений не просто регистрирует отдельные события, происходящие в системе, но и анализирует их в совокупности, пытаясь обнаружить в последовательности зафиксированных событий признаки атаки;
- система обнаружения вторжений может оперативно реагировать на обнаруженные атаки, самостоятельно блокируя соответствующие функции системы до того, как нарушитель успел им воспользоваться.

До середины 2000-х годов системы обнаружения вторжений применялись на практике крайне редко, некоторые специалисты и компании (например, Gartner) высказывали даже предположения, что к 2005 году системы обнаружения вторжений практически перестанут применяться на практике. Однако современные тенденции в области информационной безопасности таковы, что область применения систем обнаружения вторжений становится все шире и системы обнаружения вторжений постепенно перестают быть «экзотикой». Так, в конце 2004 года неизвестные хакеры получили несанкционированный доступ к базе данных персональной информации о сотрудниках и студентах Калифорнийского университета, а обнаружен этот факт был только в 21 ноября 2006 года, когда один из администраторов случайно обратил внимание на необычный характер сетевого трафика. При наличии системы обнаружения вторжений в сети атако-

ванного университета факт взлома системы был бы, скорее всего, обнаружен гораздо раньше.

Типичная архитектура системы обнаружения вторжений включает в себя следующие основные элементы:

- *сенсоры*, обеспечивающие сбор информации для последующего анализа;
- *анализаторы*, осуществляющие анализ полученной сенсорами информации;
- *хранилище* (как правило, базу данных), в которое помещаются результаты работы анализатора;
- *консоль управления*, обеспечивающую взаимодействие администратора безопасности с системой обнаружения вторжений.

По набору используемых сенсоров системы обнаружения вторжений классифицируются на:

- *узловые*, или *хостовые* (host IDS, HIDS) — берут информацию от подсистемы аудита защищаемой операционной системы или системы управления базами данных, а также дополнительных защитных подсистем (контроля целостности, антивирусного мониторинга и т. п.);
- *сетевые* (network IDS, NIDS) — анализируют сетевой трафик;
- гибридные или смешанные.

По способности предпринимать активные действия в ответ на выявленные угрозы безопасности системы обнаружения вторжений классифицируются на:

- активные;
- пассивные.

Сенсоры HIDS делятся на пять основных типов:

- сенсоры журналов;
- сенсоры признаков;
- сенсоры системных вызовов;
- сенсоры поведения приложений;
- сенсоры целостности файлов.

Сенсоры NIDS представляют собой программные или программно-аппаратные снифферы, осуществляющие перехват сетевого трафика одного компьютера или целого сегмента локальной сети.

Большинство IDS могут анализировать не только информацию, полученную непосредственно с сенсоров в реальном времени, но и работать с журналами, содержащими ранее собранную информацию. Это позволяет проводить при необходимости повторный «разбор полетов» для различных инцидентов, связанных с информационной безопасностью, например, проверять, способна ли перенаст-

роенная система обнаружения вторжений обнаружить атаку, ранее прошедшую незамеченной.

Анализаторы IDS анализируют собранную сенсорами информацию на предмет сходства с типичными атаками нарушителей. Современные системы обнаружения вторжений довольно надежно детектируют сканирование портов, с которого начинается большинство сетевых атак, а также попытки программных закладок, проникших внутрь защищаемой сети, связываться со своими хозяевами через Интернет. При этом, в отличие от традиционных систем разграничения доступа и аудита, системы обнаружения вторжений реагируют не на каждое зафиксированное событие в отдельности, а на всю последовательность событий в совокупности. Так, сетевая система обнаружения вторжений позволяет не просто блокировать трафик, идущий на определенные порты или исходящий от определенных процессов, но реализовывать более сложные правила фильтрации трафика, оценивая опасность не каждого пакета в отдельности, но целой последовательности пакетов в совокупности.

В некоторых системах обнаружения вторжений, например, в EMERALD, используется многоуровневая модель построения анализаторов. Анализаторы низшего уровня анализируют информацию от отдельных сенсоров, анализаторы более высокого уровня объединяют в одно целое информацию, полученную от больших групп сенсоров, и обрабатывают ее в совокупности, хотя и с меньшей детализацией. Анализаторы высоких уровней могут получать часть информации от анализаторов низких уровней.

Многоуровневая схема построения анализаторов позволяет системе обнаружения вторжений эффективно обнаруживать угрозы безопасности, затрагивающие сразу несколько компонент защищаемой системы.

Правила, реализуемые анализаторами IDS, бывают двух видов — сигнатурные и эвристические.

В первом случае в анализируемом потоке информации ищутся так называемые *сигнатуры* или *сценарии атак* — четкие и недвусмысленные признаки определенных атак. Примерами сигнатур могут служить:

- сигнатура подбора пароля — несколько неудачных попыток аутентификации с одного рабочего места;
- сигнатура сканирования портов — несколько попыток открытия различных портов защищаемого сервера одним и тем же клиентом;

- сигнатура эксплуатации уязвимости программного обеспечения — получение одного или нескольких определенных сетевых пакетов, пришедших на определенный порт.

Эвристические правила позволяют выявлять аномальную активность в защищаемой системе. Обычно в ходе функционирования системы активность отдельных ее компонент примерно одинакова и слабо меняется со временем. Если какая-то характеристика какой-то компоненты системы резко изменилась, это воспринимается как сигнал тревоги, например:

- если некий процесс начал потреблять заметно больше аппаратных ресурсов компьютера, чем раньше, возможно, в адресное пространство этого процесса внедрилась программная закладка;
- если некий компьютер генерирует необычно много исходящего SMTP-трафика, возможно, операционная система данного компьютера поражена сетевым вирусом, осуществляющим рассылки спам-почты.

Основным достоинством эвристических анализаторов является их способность более-менее адекватно реагировать на ранее неизвестные атаки злоумышленников. Сигнатурные анализаторы, напротив, способны реагировать лишь на те атаки, которые присутствуют в базе сигнатур системы обнаружения вторжений. Эта база должна регулярно обновляться.

Основным недостатком эвристических анализаторов является их склонность генерировать ложные тревоги. Пусть, например, на некотором компьютере развернут популярный веб-сервер, обеспечивающий большую нагрузку на процессоры компьютера. На том же компьютере развернут и FTP-сервер, однако он посещается пользователями гораздо реже и не создает существенной нагрузки на процессоры и оперативную память. Если в какой-то момент некий пользователь начнет скачивать с FTP-сервера большой файл, это может привести к генерации ложной тревоги. Для того чтобы, с одной стороны, избежать большого количества ложных тревог, а с другой стороны, своевременно обнаруживать большинство атак, система обнаружения вторжений, реагирующая на аномальную активность, нуждается в тонкой и нетривиальной настройке. Свежеустановленная и ненастроенная система обычно генерирует так много тревожных сигналов, что пользы от нее почти нет.

Эвристический анализатор системы обнаружения вторжений в общем случае не гарантирует обнаружение атаки. Большинство атак могут быть так модифицированы, что их реализация не будет

приводить к заметным всплескам активности тех или иных компонент атакуемой системы. Однако модифицированные атаки, как правило, менее эффективны, чем в оригинальном исполнении. Например, программная закладка может отслеживать загруженность процессоров и сетевого адаптера и так планировать свое функционирование, чтобы нагрузка на операционную систему, создаваемая закладкой, не превосходила некоторого порогового значения и не выделялась на фоне обычного функционирования системы. Но в этом случае закладка будет работать медленнее и менее стабильно, она может, например, досрочно прервать сеанс связи со своей клиентской программой, если текущее состояние операционной системы таково, что продолжение сеанса связи может быть обнаружено системой обнаружения вторжений.

В настоящее время сигнатурные и эвристические анализаторы часто используются в совокупности. В будущем, с ростом «интеллектуальности» систем обнаружения вторжений, эвристические анализаторы, вероятно, станут основным видом анализаторов, применяемых в системах обнаружения вторжений. Пока же в практической работе основная нагрузка ложится на сигнатурные анализаторы, а эвристические используются главным образом в пассивном режиме, когда решение о том, была ли выявлена атака или имела место ложная тревога, принимает человек.

Пассивные системы обнаружения вторжений просто записывают в специальный журнал результаты анализа информации, полученной от сенсоров. Реакция на зафиксированные атаки осуществляется администратором системы вручную.

Активные системы обнаружения вторжений (их часто называют *системами предотвращения вторжений*, *intrusion prevention systems*, *IPS*) отличаются от пассивных тем, что при обнаружении опасности самостоятельно предпринимают ответные действия. Иногда эти действия бывают довольно сложными, например, в ответ на зафиксированную сетевую атаку может осуществляться автоматическая перенастройка пакетного фильтра или маршрутизатора. В отдельных случаях активные системы обнаружения вторжений могут проводить сканирование или даже контратаку системнарушителей. Впрочем, последняя возможность на практике применяется очень редко, поскольку ее реализация может приводить к неспровоцированной атаке чужой системы из-за ошибки эвристического анализатора или даже к «битве титанов», когда две активные системы обнаружения вторжений устраивают настоящую дуэль в сетевом пространстве.

В любом случае, является ли система обнаружения вторжений активной или пассивной, узловой или сетевой, ее сопровождение требует от администраторов защищаемой сети довольно больших усилий и высокой квалификации. Малоквалифицированный администратор, обслуживающий систему обнаружения вторжений, как правило, постепенно отключает функции системы, смысл которых ему неясен, и со временем IDS приходит в состояние, когда анализатор игнорирует большую часть тревожных сигналов, поступающих от сенсоров. Такая система обнаружения вторжений не приносит никакой пользы и даже, напротив, приносит вред, поскольку создает у пользователей и администраторов системы ложное чувство защищенности.

Несмотря на свою высокую эффективность, системы обнаружения вторжений не являются панацеей, гарантированно пресекающей все атаки злоумышленников. Это обусловлено следующими факторами.

- Как и любое программное или программно-аппаратное средство обеспечения информационной безопасности, система обнаружения вторжений не обладает полноценным интеллектом и потому не всегда способна принимать адекватные решения в сложных ситуациях. Любая система обнаружения вторжений может быть обманута достаточно квалифицированным и удачливым злоумышленником.
- Как и любое программное или программно-аппаратное средство обеспечения информационной безопасности, система обнаружения вторжений нуждается в постоянном сопровождении администратора безопасности. Журналы, создаваемые системой обнаружения вторжений, должны регулярно просматриваться и анализироваться человеком, обладающим, в отличие от системы обнаружения вторжений, полноценным интеллектом. Настройки системы обнаружения вторжений должны регулярно корректироваться, не допуская, с одной стороны, фактов незамеченных вторжений злоумышленников в защищаемую систему, а с другой стороны, чрезмерного количества ложных тревог, затрудняющих функционирование защищаемой системы.
- Возможности системы обнаружения вторжений по автоматической реакции на зафиксированные угрозы весьма ограничены. «Чрезмерно активная» конфигурация системы обнаружения вторжений может приводить к тому, что нарушитель сможет спровоцировать систему обнаружения вторжений на несанкционированные действия, в лучшем случае парализующие ра-

боту защищаемой системы, а в худшем — негативно воздействующие на другие системы, не имеющие никакого отношения ни к защищаемой системе, ни к нарушителю.

- Ни одна система обнаружения вторжений не способна обнаруживать абсолютно любые атаки, для любой системы обнаружения вторжений существуют области защищаемой системы, находящиеся вне пределов видимости ее сенсоров. Так, в 2001 году с выходом новой версии Microsoft Internet Information Server обнаружилось, что ни одна из существовавших на тот момент систем обнаружения вторжений не способна правильно декодировать новый формат заголовков HTTP-пакетов, используемый данным сервером. В результате веб-серверы Microsoft IIS, обновленные до новой версии, на некоторое время оказались беззащитны для атак, защита от которых реализуется средствами систем обнаружения вторжений.
- Как и любой программный продукт, система обнаружения вторжений может иметь уязвимости, позволяющие нарушителю получать несанкционированный доступ к ресурсам компьютера, на котором установлено данное программное обеспечение.

4.3. Аудит в Windows

Для просмотра журнала аудита в Windows используется оснастка Event Viewer консоли администрирования, эту же оснастку можно использовать и для просмотра других системных журналов. Просматривать журнал аудита разрешено только пользователям, обладающим привилегией аудитора. Эти ограничения доступа действуют и в том случае, когда системный раздел жесткого диска отформатирован с использованием файловой системы, отличной от NTFS. Все пользователи, которые могут читать журнал аудита, могут и очищать его. Факт очистки журнала регистрируется сразу после очистки.

Размер журнала аудита ограничен, максимальный размер составляет по умолчанию от 512K в Windows NT 4 до 20M в Windows 7. Администратор операционной системы может менять это значение. Также администратор может определить поведение операционной системы при переполнении журнала аудита. По умолчанию события перезаписываются по мере необходимости, начиная с самых старых. В политике безопасности может быть выставлена опция «аварийно завершать работу операционной системы при переполнении журнала аудита». В этом случае после перезагрузки операционной системы работать с ней сможет только администратор. Чтобы вернуть

операционную систему в обычный многопользовательский режим, администратор должен очистить журнал аудита, сбросить в реестре соответствующий флаг и перезагрузить систему.

Добавлять записи в журнал аудита могут только субъекты доступа, обладающие соответствующей привилегией. По умолчанию эта привилегия предоставляется только псевдопользователю SYSTEM, менять данную установку не следует. Если предоставить данную привилегию какому-то физическому пользователю, он тем самым получит возможность записывать в журнал аудита произвольную информацию, в том числе и компрометирующую других пользователей.

Множество событий, информация о которых записывается в журнал аудита, определяется политикой аудита, которую определяют пользователи-аудиторы. Windows позволяет регистрировать в журнале аудита события следующих категорий:

- вход/выход пользователя в/из системы;
- аутентификация пользователя*;
- доступ субъектов к локальным объектам;
- доступ субъектов к объектам активного каталога;
- использование субъектами доступа опасных привилегий;
- изменения в списке пользователей;
- изменения в политике безопасности;
- системные события;
- запуск и завершение процессов.

Для каждого класса событий могут регистрироваться либо только успешные события, либо только неуспешные, либо и те, и другие, либо никакие.

По умолчанию в Windows, начиная с Windows Vista, реализуются следующие настройки политики аудита:

- на контроллерах доменов:
- вход/выход пользователя в/из системы — только успешные попытки;
- аутентификация пользователя — только успешные попытки;
- доступ субъектов к объектам активного каталога — только успешные попытки;

---

* В доменах Windows аутентификация пользователя может осуществляться на компьютере, отличном от того, с которым данный пользователь начинает сеанс работы. При аутентификации пользователя домена на рабочей станции подсистема аудита рабочей станции генерирует сообщение в категории «вход/выход пользователя», а подсистема аудита контроллера домена — сообщение в категории «аутентификация пользователя».

- изменения в списке пользователей — только успешные попытки;
- изменения в политике безопасности — только успешные попытки;
- системные события — только успешные попытки;
- на рабочих станциях:
- вход/выход пользователя в/из системы — только успешные попытки;
- аутентификация пользователя — только успешные попытки;

В доменах Windows политика аудита интегрирована в групповую политику и наследуется в соответствии с правилами наследования групповой политики.

Порядок регистрации событий при доступе субъектов к объектам определяется не только политикой аудита, но и атрибутами защиты объекта. Как уже упоминалось выше, в состав дескриптора защиты объекта может входить системный список контроля доступа (SACL), определяющий порядок регистрации событий аудита при доступе субъектов к данному объекту. Так же, как и DACL, SACL представляет собой список переменной длины, элементами которого являются ACE, имеющие точно такой же формат, как и ACE, входящие в состав DACL.

В отличие от ACE из DACL, ACE в SACL всегда имеют тип «регистрирующий ACE» (system audit ACE). Для объектов активного каталога также поддерживается тип «объектно-специфичный регистрирующий ACE». Кроме того, для всех объектов поддерживается (хотя эта возможность недокументированна) system audit compound ACE, позволяющий отдельно описывать параметры регистрации доступа к объектам для каждой пары «субъект-клиент + субъект-сервер».

Элементы контроля доступа, входящий в SACL, может иметь все флаги, которые может иметь ACE, входящий в DACL. Кроме того, ACE из SACL могут иметь еще два флага:

- SUCCESSFUL_ACCESS_ACE_FLAG (s) — если этот флаг установлен, будут регистрироваться в журнале аудита все успешные обращения к объекту субъекта, идентификатор которого записан в ACE, по любому из методов доступа, перечисленных в маске доступа ACE;
- FAILED_ACCESS_ACE_FLAG (f) — если этот флаг установлен, будут регистрироваться в журнале аудита все неуспешные обращения к объекту субъекта, идентификатор которого записан в ACE, по любому из методов доступа, перечисленных в маске доступа ACE.

Если в ACE установлены оба флага, регистрируются любые обращения субъекта к объекту по перечисленным методам доступа, как успешные, так и неуспешные. Если в ACE установлен флаг **i**, при доступе субъектов к объекту ACE игнорируется.

Поскольку все ACE в SACL однотипны, порядок их взаимного расположения не имеет значения.

Если в дескрипторе защиты объекта SACL отсутствует, обращения субъектов к этому объекту не регистрируются.

При создании нового объекта SACL назначается объекту по тем же правилам, что и DACL. При наследовании ACE флаги **s** и **f** остаются неизменными.

Для того чтобы событие, связанные с доступом субъекта к объекту, было зафиксировано в журнале аудита, необходимо одновременное выполнение следующих двух условий:

- Политика аудита операционной системы допускает регистрацию в журнале аудита событий, связанных с успешным (или, соответственно, неуспешным) доступом субъектов к объектам.
- SACL объекта содержит хотя бы один ACE, в котором:
- идентификатор субъекта относится к субъекту, открывающему объект;
- установлен флаг **s** (или, соответственно, **f**) и не установлен флаг **i**;
- после отображения отображаемых прав доступа пересечение маски доступа ACE и маски доступа, содержащей права, запрашиваемые субъектом, непусто.

Таким образом, глобальные настройки политики аудита в отношении доступа субъектов к объектам играют роль фильтра, позволяющего временно запретить регистрацию успешных или неуспешных попыток доступа всех субъектов ко всем объектам операционной системы.

Начиная с Windows Vista, администратор может привязать к каждому типу событий аудита одну или несколько задач следующего вида:

- вывести текстовое сообщение в текущую терминальную сессию;
- отправить электронное письмо на заданный адрес;
- запустить заданную программу.

К сожалению, во всех трех случаях подробные сведения о зарегистрированном событии не передаются задаче в качестве параметров. Фактически, данный механизм позволяет интерактивно оповещать администратора безопасности только о факте наступления

того или иного события (например, неудачной попытке входа пользователя в систему), но подробности события (с какой учетной записью и каким сетевым адресом оно связан) администратор должен будет выяснять самостоятельно, лично просмотрев журнал аудита.

Начиная с Windows Vista, в подсистеме аудита Windows поддерживается механизм, позволяющий автоматически перенаправлять записи о событиях определенных видов с одних компьютеров на другие. Перенаправление записей аудита реализуется посредством так называемых *подписок*, создаваемых с помощью оснастки Event Viewer консоли администрирования. Для управления подписками необходимо иметь полномочия администратора как в системе-источнике, так и в системе-сборщике. Кроме того, может потребоваться дополнительная настройка пакетного фильтра.

Средства автоматического обнаружения вторжений в современных версиях Windows отсутствуют. Microsoft ISA Server, согласно документации, включает в себя встроенную систему обнаружения вторжений, но она настолько примитивна, что вряд ли ее можно всерьез относить к данному классу средств защиты информации.

4.4. Аудит в UNIX

В современных операционных системах семейства UNIX реализуются два принципиально различных подхода к организации аудита. Первый из них основан на применении традиционных для UNIX демонов регистрации событий syslogd и klogd, изначально разработанных не столько для поддержания безопасности операционной системы, сколько для выявления и устранения ошибок конфигурирования, сетевых неполадок, взаимных несовместимостей пакетов программного обеспечения и т. п. Второй, альтернативный подход появился сравнительно недавно, он основан на переносе в среду UNIX архитектуры и интерфейсов подсистемы аудита, изначально свойственных операционным системам семейства Windows. Если UNIX-система включена в состав гетерогенной сети, большинство узлов которой работают под управлением Windows, единообразное построение подсистем аудита всех операционных систем становится серьезным преимуществом, поскольку позволяет, например, распространять политики аудита на большие подмножества узлов сети независимо от того, под управлением какой операционной системы работает каждый конкретный узел того или иного подмножества. В состав большинства современных UNIX-систем могут включаться обе подсистемы аудита, при этом они могут работать как независимо одна от другой, так и во взаимодействии.

4.4.1. syslogd, klogd

Вначале мы рассмотрим первый, более традиционный и более часто применяемый подход к организации аудита в UNIX, основанный на использовании демонов syslofd и klogd. Регистрация событий, связанных с безопасностью операционной системы, не отделяется этими демонами от регистрации других системных событий, поэтому при описании данного подхода мы будем использовать термин «аудит» расширенно — как процедуру регистрации любых событий, имевших место в ходе функционирования операционной системы и не обязательно непосредственно связанных с ее безопасностью.

Демон syslogd предоставляет услуги прикладным и системным программам, демон klogd реализует аудит операций, выполняемых ядром операционной системы. Основным назначением этих демонов является поддержка единой в масштабах операционной системы политики аудита, предписывающей выполнение определенных действий для каждого типа регистрируемых событий. Каждый раз, когда прикладная или системная программы выполняет потенциально аудируемое действие, информация об этом действии передается демону syslogd, тот сверяется с текущей политикой аудита и принимает решение о порядке регистрации данного события. Получение информации демоном syslogd обычно реализуется через сокет /dev/log (для локальных клиентов) или UDP-порт 514 (для удаленных клиентов). Демон klogd получает информацию через специальный файл /proc/kmsg или системный вызов ядра sys_syslog. Идентификаторы процессов демонов аудита хранятся в десятичном виде в текстовых файлах /var/run/syslogd.pid и /var/run/klogd.pid соответственно.

Действующая политика аудита описывается текстовым файлом /etc/syslog.conf. Каждая строка этого файла описывает одно элементарное правило политики аудита и включает в себя два основных поля:

- *селектор* — описывает условия применимости данного правила;
- *действие* — описывает действие, которое должно быть выполнено в результате применения данного правила.

Поле селектора в общем случае имеет следующий вид:

⟨источник⟩.⟨модификатор⟩ ⟨приоритет⟩

Подполе ⟨источник⟩ описывает подсистему операционной системы, которая обращается к демону syslogd с намерением зарегистрировать то или иное событие. Данное подполе может принимать следующие значения:

auth — клиентская часть подсистемы аутентификации;

authpriv — серверная часть подсистемы аутентификации;

cron — демон автоматического запуска заданных процессов в заданное время;

daemon — все другие демоны, кроме явно перечисленных в данном списке;

ftp — FTP-сервер;

kern — ядро операционной системы;

lpr — сервер печати;

mail — сервер электронной почты;

mark — предназначено для внутреннего использования;

news — NNTP-сервер;

security — синоним для auth, считается устаревшим;

syslog — сам демон аудита;

user — любая прикладная программа;

uucp — демон UUCP;

local0-local7 — зарезервированы;

* — любой источник.

В одном селекторе можно указывать несколько источников через запятую.

Подполе ⟨модификатор⟩ может принимать следующие значения:

(не заполнено) — данная строка описывает реакцию на события, имеющие приоритет, больший или равный указанному;

= — данная строка описывает реакцию на события, имеющие приоритет, точно равный указанному;

! — данная строка описывает реакцию на события, имеющие приоритет, меньший указанного;

!= — данная строка описывает реакцию на события, имеющие приоритет, отличный от указанного.

Подполе ⟨приоритет⟩ может принимать следующие значения (перечислены в порядке убывания значимости):

emerg, panic — операционная система вышла из строя, после регистрации данного события произойдет аварийное завершение ее работы;

alert — критический сбой, требующий немедленной реакции;

crit — критический сбой;

err, error — сбой;

warning, warn — предупреждение;

notice — важное информационное сообщение;

info — информационное сообщение;

debug — отладочное сообщение.

Также поддерживаются два специальных значения приоритета:
* — любой приоритет;
none — «никакой» приоритет, предназначено для описания отношений типа «все, кроме».

В одной строке файла syslogd.conf можно описывать несколько селекторов, они разделяются точкой с запятой (;).

Поле действия описывает конкретное действие, которое должен предпринять демон syslogd при регистрации события аудита, соответствующего указанному селектору. Поддерживаются следующие типы действий:

- запись информации о событии в файл — указывается имя файла. Обычно сразу после записи информации о событии выполняется принудительный сброс дискового кэша для данного файла, этим гарантируется сохранение информации в случае внезапного аварийного завершения работы операционной системы. Принудительный сброс кэша можно отменить для любого конкретного файла, указав перед именем файла символ – (минус);
- выдача информации о событии на терминал или консоль — указывается имя файла в директории /dev, соответствующее данному терминалу и консоли;
- запись информации о событии в указанный именованный канал — указывается имя канала, которому предшествует символ «|»;
- перенаправление информации о событии демону syslogd другого компьютера (514 порт UDP) — указывается имя компьютера, которому предшествует символ «@». При неаккуратном описании порядка перенаправления сообщений аудита с одних компьютеров на другие могут возникать циклы, что может приводить к перегрузке сети и переполнению файлов аудита;
- интерактивное оповещение о событии одного или нескольких пользователей — указываются имена пользователя через запятую;
- интерактивное оповещение о событии всех пользователей, работающих в данный момент с операционной системой — указывается одиночный символ «*».

Рассмотрим несколько примеров описания политик аудита в файле syslog.conf.

```
# Все критические сообщения, кроме исходящих от ядра,
# записывать в файл /var/adm/critical
*.crit;kern.none /var/adm/critical
# Все сообщения от ядра записывать в файл /var/adm/kernel
kern.* /var/adm/kernel
```

```
# Критические сообщения от ядра также выводить на консоль и
# перенаправлять на компьютер netaud
kern.crit /dev/console
kern.crit @netaud
# Не очень важные сообщения от ядра записывать в файл
# /var/adm/kernel-info
kern.!err /var/adm/kernel-info
# Информационные сообщения от почтового сервера выводить на
# двенадцатый терминал
mail.=info /dev/tty12
# Все остальные сообщения от почтового сервера записывать в файл
# /var/adm/mail
mail.*;mail.!=info /var/adm/mail
# Информационные сооообщения от почтового сервера и NNTP-сервера
# записывать в файл /var/adm/info
mail,news.=info /var/adm/info
# Все остальные информационные сообщения записывать в файл
# /var/log/messages
*.info mail,news.none /var/log/messages
# Все сообщения с приоритетами info и notice, кроме исходящих от
# почтового сервера, записывать в файл /var/log/messages
*.=info;*.=notice;mail.none /var/log/messages
# Об аварийном завершении работы операционной системы
# интерактивно оповещать всех пользователей
*.emerg *
# О критических сбоях, требующих немедленной реакции,
# интерактивно оповещать пользователей root и vadim
*.alert root,vadim
# Перенаправлять все сообщения на компьютер netaud
*.* @netaud
# Отладочные сообщения, кроме исходящих от серверной части
# подсистемы аутентификации, записывать в файл /var/log/debug,
# не заботясь о сохранности записываемой информации
*.debug;authpriv.none -/var/log/debug
# Отладочные сообщения ядра также направлять в именованный канал
# /usr/adm/debug
kern.debug |/usr/adm/debug
```

В большинстве UNIX-систем большая часть сообщений аудита традиционно записывается в файл /var/log/messages или /var/adm/log/messages. Для хранения сообщений о попытках аутентификации традиционно используется файл /var/log/auth.log или (например, в Solaris) /var/adm/loginlog.

При обновлении файла syslog.conf новая политика аудита вступает в силу только после перезагрузки демона syslogd или после того, как этот демон получит сигнал SIGHUP. Прикладные программы

UNIX интерпретируют данный сигнал как обрыв связи удаленного клиента с терминалом и обычно, получив его, аварийно завершают работу, но демон syslogd воспринимает сигнал SIGHUP иначе — как требование обновить политику аудита.

Журналы аудита UNIX представляют собой обычные текстовые файлы. Их просмотр и анализ может проводиться как обычными утилитами просмотра текстов, так и более продвинутыми программными средствами, облегчающими работу с аудитом. В отличие от Windows, в UNIX нет встроенных средств, позволяющих жестко ограничивать максимальные размеры журналов аудита. Неадекватная политика аудита в UNIX потенциально может приводить к исчерпанию свободного места на жестких дисках компьютера. Для предотвращения данной угрозы могут применяться следующие меры:

- ограничение доступа к 514 порту UDP пакетным фильтром;
- размещение файлов аудита на отдельном разделе жесткого диска;
- запуск демона syslogd от имени псевдопользователя с ограниченными полномочиями.

Удаление из файлов аудита устаревшей информации традиционно реализуется путем регулярного запуска демоном cron специальной утилиты logrotate, конфигурация которой описывается файлом /etc/logrotate.conf.

Серьезным недостатком подсистемы аудита UNIX является то, что доступ к демону syslogd в общем случае предоставляется любым программам, как системным, так и прикладным. При этом клиентская программа не только передает демону текст сообщения, но и самостоятельно указывает источник и приоритет регистрируемого события. Более того, существует специальная утилита командной строки logger, позволяющая обычному непривилегированному пользователю передать демону syslogd произвольную информацию от имени произвольной подсистемы операционной системы и присвоить этой информации произвольный уровень значимости, например:

```
logger -p kernel.emerg virus: hard drive formatting started
```

В приведенном примере сообщение, переданное демону syslogd, является очевидной шуткой. Однако ничто не мешает пользователю-нарушителю навязывать демону аудита не столь безобидную информацию, например, имитировать нарушение правил безопасности другим пользователем, к которому нарушитель испытывает неприязнь.

4.4.2. auditd

Механизм регистрации событий, реализуемый демонами syslogd и klogd, в основном предназначен не столько для поддержания безопасности системы, сколько для выявления программных и аппаратных неисправностей, тестирования и отладки новых компонент операционной системы и т. п. Применять данный механизм для регистрации событий, связанных с безопасностью системы, не очень удобно, и в некоторых операционных системах, принадлежащих к семейству UNIX, для этой задачи предназначен отдельный, дополнительный демон аудита, которому чаще всего назначается имя auditd. Концептуально реализации данного демона в разных UNIX-системах обычно очень похожи на реализацию подсистемы аудита в Windows. Однако технические детали разных реализаций могут очень сильно отличаться одна от другой.

В операционной системе AIX демон auditd предназначается главным образом для регистрации обращений пользователей к файлам. Конфигурация демона описывается в файлах config и objects, размещаемых в директории /etc/security/audit. Данные аудита записываются либо в псевдофайл /audit/trail, либо на логическое устройство /dev/audit, для считывания данных аудита уполномоченным пользователем используются специальные утилиты auditptr и auditstream. Формат данных аудита при этом выглядит примерно следующим образом:

```
event           login status time                    command
---------       ----- ------ -----                   -------
S_NOTAUTH_READ  root  OK     Thu Nov 1 14:07:05 2012 cat
S_NOTAUTH_READ  root  OK     Thu Nov 1 14:07:05 2012 cat
FILE_Unlink     root  OK     Thu Nov 1 14:07:09 2012 vi
S_NOTAUTH_READ  root  OK     Thu Nov 1 14:07:09 2012 vi
S_NOTAUTH_READ  root  OK     Thu Nov 1 14:07:09 2012 vi
S_NOTAUTH_READ  root  OK     Thu Nov 1 14:07:09 2012 vi
S_NOTAUTH_WRITE root  OK     Thu Nov 1 14:07:13 2012 vi
FILE_Unlink     root  OK     Thu Nov 1 14:07:13 2012 vi
FILE_Unlink     root  OK     Thu Nov 1 14:07:20 2012 vi
S_NOTAUTH_READ  ash   OK     Thu Nov 1 14:09:39 2012 cat
S_NOTAUTH_READ  ash   OK     Thu Nov 1 14:09:39 2012 cat
```

В Mac OS X демон audit записывает регистрируемые данные в один или несколько файлов, обычно расположенных в директории /var/audit. Политика аудита для данного демона описывается конфигурационными файлами audit_class, audit_event, audit_control и audit_user. Для каждого пользователя операционной системы порядок регистрации событий, связанных с деятельностью этого пользователя, задается индивидуально, кроме того, определен поря-

док регистрации событий по умолчанию. Механизм регистрации событий, реализуемый демоном audit в Mac OS, во многом похож на реализацию аудита в Windows. Аналогично категориям событий аудита в Windows, в Mac OS события аудита объединяются в так называемые классы, при этом порядок регистрации событий отдельно определяется для успешных и неуспешных событий каждого класса. Для каждого пользователя определяются так называемые флаги аудита — битовая маска, в которой каждый бит соответствует одному классу потенциально регистрируемых событий. По умолчанию в Mac OS задано 19 классов событий аудита, при этом система их классификации запутана и не слишком удобна для практического использования.

Для некоторых событий возможно интерактивное оповещение администратора с помощью автоматического запуска специального скрипта audit_warn, который в качестве параметра получает текстовую строку, содержащую описание зарегистрированного события. По умолчанию скрипт audit_warn просто записывает текущее время и переданный ему текст в конец файла /etc/security/audit_messages, но администратор операционной системы может менять код скрипта произвольным образом.

Похожим образом реализован дополнительный аудит в операционной системе Astra Linux. Здесь для каждого пользователя или группы пользователей может быть создан так называемый профиль аудита (фактически, те же самые флаги аудита), описывающий порядок регистрации событий аудита для следующих категорий:

open — открытие объекта доступа;

create — создание объекта доступа;

exec — запуск процесса;

delete — удаление объекта доступа;

chmod — изменение вектора доступа объекта;

chown — изменение владельца объекта доступа;

mount — монтирование или размонтирование файловой системы;

module — загрузка или выгрузка модуля расширения ядра операционной системы;

uid — запуск процесса с использованием механизма SUID;

gid — запуск процесса с использованием механизма SGID;

audit — изменение профиля аудита;

xattr — изменение списка доступа объекта;

mac — изменение мандатных меток объектов доступа;

cap — изменение привилегий субъекта доступа;

chroot — смена корневого каталога файловой системы;

rename — переименование объекта доступа;

net — изменение сетевых настроек.

Большинство перечисленных категорий фактически представляют собой системные вызовы, при каждом выполнении которых регистрируется либо не регистрируется соответствующее событие аудита.

Вопросы для самопроверки

1. В чем заключается процедура аудита применительно к защищенным операционным системам?

2. Должна ли предоставляться администраторам операционной системы привилегия работать с подсистемой аудита?

3. Какие основные требования предъявляются к подсистеме аудита операционной системы?

4. Что такое политика аудита?

5. Как часто должна корректироваться политика аудита?

6. Что представляет собой система обнаружения вторжений?

7. Какие основные элементы включает в себя архитектура системы обнаружения вторжений?

8. На какие основные типы делятся системы обнаружения вторжений?

9. Чем различаются сигнатурные и эвристические правила обнаружения вторжений?

10. Каков основной недостаток применения эвристических анализаторов в системах обнаружения вторжений?

11. Чем активные системы обнаружения вторжений отличаются от пассивных?

12. Каковы основные факторы, ограничивающие эффективность систем обнаружения вторжений?

13. К каким негативным последствиям может приводить установка чрезмерно активной конфигурации системы обнаружения вторжений?

14. Каким пользователям Windows разрешается работать с подсистемой аудита операционной системы?

15. Какие категории событий аудита поддерживаются в Windows?

16. Какие элементы входят в список SACL дескриптора защиты объекта доступа Windows, каково их назначение?

17. Какие дополнительные флаги должны быть установлены в ACE, входящих в состав SACL, каково назначение этих флагов?

18. Какие задачи можно привязывать к событиям аудита в современных версиях Windows?

19. Какие два механизма аудита поддерживаются в современных версиях UNIX?

20. Для чего предназначены UNIX-демоны syslogd и klogd?

21. Как описывается текущая конфигурация демона syslogd?

22. Какие источники и приоритеты событий аудита поддерживаются в современных версиях UNIX?

23. Какие действия могут связываться с событиями аудита в современных версиях UNIX?

24. Как можно интерактивно оповестить о том или ином событии, зарегистрированном подсистемой аудита, всех пользователей UNIX, работающих с операционной системой в данный момент?

25. В каком формате хранятся данные в журналах аудита UNIX?

26. Какие недостатки имеет стандартная подсистема аудита UNIX?

27. Какие функции выполняет демон audit в Mac OS?

28. Какие профили аудита поддерживаются в операционной системе Astra Linux?

5 Домены Windows

5.1. Общие сведения

Основной задачей, для решения которой предназначена доменная архитектура компьютерной сети, является упрощение администрирования и управления сетью.

Доменом называется совокупность компьютеров, объединенных в общую сеть и разделяющих общий список пользователей и общую политику безопасности. Учетная информация о пользователях, псевдопользователях и группах домена централизованно хранится в единой базе данных. Наличие общей базы учетных записей позволяет пользователю, единожды войдя в домен, осуществлять доступ к любому разделяемому ресурсу данного домена*. Если на пользователя не наложены специальные ограничения, он может подключаться к домену с любого компьютера сети.

Компьютеры, входящие в домен и работающие под управлением Windows, могут управляться централизованно. Большинство задач администрирования могут выполняться администратором домена со своего рабочего места в отношении любого компьютера домена. В частности, администратор домена может выполнять на любом компьютере домена следующие действия:

- создавать, удалять, переименовывать и менять атрибуты разделяемых каталогов и принтеров;
- просматривать и редактировать реестр;
- создавать, удалять, загружать и выгружать драйверы и сервисы;
- просматривать системные журналы, включая журнал аудита.

Одним из важнейших элементов доменной архитектуры Windows является база учетных записей, в которой централизованно хранится информация о пользователях, псевдопользователях и группах домена. Каждому компьютеру домена соответствует псевдопользователь с именем computer_name$, учетная информация

* Большинство возможностей, предоставляемых доменной архитектурой, могут быть отключены для отдельных компьютеров или пользователей. Далее везде под словами «пользователь может выполнить некоторое действие» подразумевается «пользователь может выполнить некоторое действие, если эта возможность не отключена явно».

компьютера хранится в базе в таком же формате, как и учетная информация пользователя.

В каждом домене Windows обязательно существует хотя бы один сервер, называемый *контроллером домена* (*domain controller, DC*). База учетных записей домена физически хранится на жестком диске этого компьютера: в Windows NT — в реестре, начиная с Windows 2000 — в специальной базе данных, называемой активным каталогом. В роли контроллера домена может выступать только Windows Server, рабочие станции контроллерами доменов быть не могут.

В домене может существовать более одного контроллера, в этом случае все контроллеры домена хранят идентичные копии (реплики) базы учетных записей домена. Время от времени в домене выполняется синхронизация реплик базы учетных записей, хранящихся на разных контроллерах домена. В Windows NT среди контроллеров домена выделялся один *первичный контроллер*, хранящий эталонную копию базы учетных записей. Начиная с Windows 2000, все контроллеры домена равноправны, для синхронизации разных копий базы учетных записей используются развитые средства репликации данных.

Членами домена могут быть любые компьютеры, операционные системы которых способны взаимодействовать с контроллерами домена. В полном объеме преимуществами доменной архитектуры могут пользоваться только компьютеры, работающие под управлением Windows 2000 или выше, однако отдельные функции доменной архитектуры могут использоваться любыми компьютерами, программное обеспечение которых способно осуществлять информационный обмен по сетевому протоколу SMB.

5.2. Сквозная аутентификация

Когда пользователь начинает работу с операционной системой Windows, входящей в состав домена, пользователь указывает в соответствующем поле формы ввода домен, в котором зарегистрирована учетная запись. Если пользователь указал в качестве домена имя локального компьютера, вход в домен не производится, аутентификация выполняется с использованием локальной базы учетных записей и пользователь работает с операционной системой, как если бы рабочая станция представляла собой изолированный компьютер. В дальнейшем всякий раз, когда пользователь захочет обратиться к ресурсам другого компьютера того же домена, пользователь должен будет пройти повторную аутентификацию.

Если же пользователь, входя в систему, указал, что его учетная запись зарегистрирована в домене, операционная система выполняет *сквозную* или *транзитную аутентификацию*. Сквозную аутентификацию может осуществлять не только Windows, но и любая другая операционная система, в состав которой входит соответствующий сетевой клиент.

Если пользователь не указал, где зарегистрирована его учетная запись, операционная система вначале пытается провести локальную аутентификацию, а если учетная запись пользователя отсутствует в локальной базе — сквозную.

Сквозная аутентификация выполняется следующим образом.

1. Рабочая станция устанавливает сетевое соединение с контроллером домена. Если ни один контроллер домена недоступен, сквозная аутентификация невозможна.

2. Осуществляется взаимная аутентификация рабочей станции и контроллера домена. Псевдопользователь, соответствующий рабочей станции, проходит аутентификацию на контроллере домена, а псевдопользователь, соответствующий контроллеру домена, проходит аутентификацию на рабочей станции. Если взаимная аутентификация компьютеров невозможна (например, если контроллер домена подменен нарушителем), сквозная аутентификация пользователя также невозможна.

3. Рабочая станция и контроллер домена договариваются о протоколе, по которому будет передаваться идентификационная и аутентификационная информация аутентифицирующегося пользователя. Если договориться невозможно (например, если в домене запрещена открытая передача по сети аутентификационной информации, а программное обеспечение рабочей станции не поддерживает шифрование паролей), сквозная аутентификация невозможна.

4. Идентификационная и аутентификационная информация пользователя, проходящего аутентификацию, пересылается контроллеру домена. Если в домене используется протокол аутентификации Kerberos (начиная с Windows 2000, он используется почти всегда), данный шаг алгоритма включает в себя длинную и нетривиальную последовательность запросов и ответов, при этом используется весьма сложное шифрование.

5. Контроллер домена проводит аутентификацию пользователя с использованием данных, хранящихся в базе учетных записей домена. Если аутентификация прошла неуспешно (например, пользователь ввел неверный пароль или пользователю запрещено входить в

домен с данного компьютера), рабочая станция получает от контроллера домена отрицательный ответ и аутентификация прерывается.

6. Если аутентификация прошла успешно, контроллер домена высылает рабочей станции учетную информацию пользователя, необходимую для формирования его маркера доступа. Рабочая станция формирует маркер доступа и на этом аутентификация завершается.

Из вышеприведенного алгоритма видно, что сквозная аутентификация в доменах Windows защищена от навязывания нарушителем ложного сервера. Если нарушитель каким-то образом отключит от сети контроллер домена и подключит вместо него свой компьютер, отвечающий на те же запросы, рабочая станция распознает подмену, поскольку компьютер нарушителя не сможет пройти взаимную аутентификацию компьютеров (чтобы это стало возможным, нарушитель должен иметь доступ к базе учетных записей домена, а тогда отпадает необходимость в данной атаке).

Единственное, что нарушитель может навязать рабочей станции и котроллеру домена, реализующим сквозную аутентификацию некоторого пользователя — выбор алгоритма, по которому аутентификационная информация пользователя будет передаваться по сети. Если в обеих операционных системах, участвующих в информационном обмене, нет никаких ограничений на алгоритм передачи аутентификационной информации, нарушитель может спровоцировать передачу пароля по сети в открытом виде, без шифрования. Однако начиная с Windows NT 4.0 SP4, открытая передача пароля по сети по умолчанию запрещена. Самое большее, чего может добиться нарушитель в таких системах — заставить рабочую станцию отправить аутентификационную информацию по устаревшему протоколу LanMan, более уязвимому в отношении подбора паролей. Однако начиная с Windows 2000 даже это, как правило, невозможно.

Если ни один из контроллеров домена не в состоянии обслужить запрос рабочей станции, сквозная аутентификация невозможна. Этот факт существенно снижает устойчивость работы сети. Например, при выходе из строя сетевого коммутатора ни один пользователь домена не может войти ни на один из компьютеров, входящих в состав данного сегмента, обслуживаемого данным коммутатором. Поскольку пользователи сети обычно не имеют локальных учетных записей на рабочих станциях домена (в обычном режиме работы сети это просто не нужно), данная ситуация фактически парализует работу сети.

Для исключения подобных ситуаций в Windows поддерживается кэширование аутентификационной информации. После каждого успешного входа пользователя в домен аутентификационная информация пользователя сохраняется в зашифрованном виде в локальном реестре рабочей станции. В дальнейшем, когда пользователь пытается войти в систему и ни один из контроллеров домена не в состоянии выполнить сквозную аутентификацию, аутентификация пользователя осуществляется с использованием кэшированных аутентификационных данных.

Кэширование аутентификационной информации может быть отключено администратором операционной системы. Некоторые эксперты в области компьютерной безопасности рекомендуют отключать кэширование аутентификационных данных, поскольку наличие зашифрованного образа аутентификационной информации в реестре рабочей станции упрощает злоумышленнику получение исходных данных для дальнейшего подбора аутентификационной информации пользователя. Однако в большинстве случаев снижение защищенности аутентификационных данных пользователей от подбора вполне компенсируется повышением устойчивости работы сети.

Так же, как и на отдельно стоящем компьютере Windows, в доменах Windows могут существовать группы пользователей. В группы пользователей домена (*глобальные группы*) могут входить только пользователи домена, пользователи отдельных компьютеров домена в группы домена входить не могут. Однако локальные группы отдельных компьютеров домена могут включать в себя пользователей домена и даже группы, зарегистрированные в домене. Если локальная группа, зарегистрированная на некотором компьютере домена, включает в себя в качестве подгруппы глобальную группу, зарегистрированную в домене, то считается, что на данном компьютере все пользователи домена, входящие в глобальную группу, тем самым входят и в соответствующую локальную группу данного компьютера. Список пользователей, входящих в локальную группу, составляется заново при каждой авторизации пользователя, этим гарантируется, что в маркер доступа авторизуемого пользователя будет внесена актуальная информация о его членстве в локальных и глобальных группах.

Когда пользователь домена авторизуется на компьютере, входящем в состав домена, пользователь получает права и привилегии, предоставленные ему как:

- пользователю домена;
- члену локальных групп данного компьютера;

- члену глобальных групп домена;
- члену глобальных групп домена, являющихся подгруппами локальных групп данного компьютера.

Если пользователю на данном компьютере не назначены никакие полномочия, пользователь получает полномочия, предоставленные группе Everyone.

Описанная схема назначения полномочий позволяет гибко и централизованно управлять политикой безопасности в рамках всего домена. Например, если внести некоторого пользователя в группу домена Domain Admins, пользователь тем самым включается в группу Administrators на всех компьютерах домена, на которых группа Domain Admins является подгруппой группы Administrators (данное включение устанавливается по умолчанию при включении рабочей станции Windows в домен).

С другой стороны, если имеется необходимость реализовать на некотором компьютере домена политику безопасности, существенно отличающуюся от политики безопасности домена, это легко может быть сделано путем изменения порядка включения глобальных групп пользователей домена в локальные группы пользователей данного компьютера. В вырожденном случае возможна ситуация, когда в формальном описании локальной политики безопасности компьютера не упоминается ни один из субъектов, зарегистрированных в домене, и, фактически, данный компьютер входит в домен чисто номинально — политика безопасности операционной системы данного компьютера никак не зависит от политики безопасности, принятой в домене.

Начиная с Windows 2000, в группы могут включаться не только пользователи, но и компьютеры домена (фактически — псевдопользователи, соответствующие компьютерам).

5.3. Отношения доверия

Между доменами Windows, функционирующими в одной физической сети, могут быть установлены *отношения доверия*. Если домен A доверяет домену B, это означает, что каждый пользователь домена B имеют доступ ко всем ресурсам домена A, за исключением тех ресурсов, доступ к которым явно запрещен данному конкретному пользователю.

В Windows NT отношения доверия были односторонними, т. е. из того, что домен A доверяет домену B, не следовало, что домен B доверяет домену A. Для того чтобы установить между доменами

двусторонние отношения доверия, требовалось создать пару односторонних отношений доверия, направленных навстречу друг другу. Отношения доверия в Windows NT не были транзитивными, т. е. если домен A доверяет домену B, а домен B доверяет домену C,. из этого не следовало, что домен A доверяет домену C.

Начиная с Windows 2000, домены могут быть объединены в единый лес, в котором все домены доверяют друг другу. Важно заметить, что Windows 2000 содержит развитые средства управления полномочиями пользователей в разных доменах и поэтому доверие всех всем вовсе не означает всеобщей вседозволенности, а означает лишь, что каждый пользователь имеет возможность обращаться ко всем ресурсам сети, к **которым ему явно разрешен доступ**.

При необходимости администратор леса доменов Windows 2000/2003/2008 может устанавливать и отношения доверия «в стиле NT», но на практике эта возможность почти не используется.

Каждому домену в лесу соответствует псевдопользователь, используемый для взаимной аутентификации контроллерами доверяющих друг другу доменов. Любое обращение контроллера одного домена к контроллеру другого домена начинается с взаимной аутентификации этих компьютеров, что делает практически невозможной подмену контроллера домена и навязывание неверной информации контроллерам других доменов. В маленьком лесу каждый контроллер домена хранит у себя эталонный образ аутентификационных данных всех других контроллеров домена. В большом лесу каждый контроллер домена хранит у себя лишь аутентификационные данные «ближайших соседей», а взаимная аутентификация с контроллером «далекого» домена осуществляется путем выстраивания *цепочки*, состоящей из контроллеров доменов, при этом каждая пара соседних в цепочке контроллеров способна выполнить взаимную аутентификацию, не прибегая к посредничеству других компьютеров.

Пусть пользователь домена A пытается войти в операционную систему рабочей станции, входящей в состав домена B, доверяющего домену A. Сквозная аутентификация в этом случае осуществляется следующим образом.

1. Рабочая станция устанавливает сетевое соединение с контроллером домена B.

2. Рабочая станция передает идентификационные и аутентификационные данные пользователя, входящего в систему, контроллеру домена B. Взаимная аутентификация рабочей станции и контроллера домена, а также выбор протокола передачи данных происходят

точно так же, как и в случае сквозной аутентификации в одном домене.

3. Контроллер домена B передает идентификационные и аутентификационные данные пользователя контроллеру домена A (если необходимо, перед этим выполняется взаимная аутентификация контроллеров доменов A и B).

4. Контроллер домена A проводит аутентификацию пользователя с использованием информации, хранящейся в его базе учетных записей. Результаты аутентификации передаются контроллеру домена B.

5. Контроллер домена B перенаправляет результат аутентификации пользователя на рабочую станцию, с которой непосредственно работает пользователь.

Если домен A доверяет домену B, то на каждом компьютере домена A права и привилегии могут назначаться следующим субъектам доступа:

- локальным пользователям, зарегистрированным на данном компьютере;
- локальным группам, зарегистрированным на данном компьютере;
- пользователям домена A;
- пользователям домена B;
- группам домена A;
- группам домена B.

5.4. Активный каталог

Начиная с Windows 2000, иерархическая база учетных записей SAM преобразована в полнофункциональную распределенную базу данных, основанную на модели данных X.500 и называемую *активным каталогом* (*active directory*, *AD*). Большинство операций над активным каталогом выполняются в контексте процесса-сервера lsass.exe, также отвечающего за аутентификацию, аудит и криптографические функции и являющегося ядром подсистемы безопасности Windows.

В общем случае каталог — это база данных, оптимизированная для ситуации, когда обновления базы происходят много реже, чем получение информации из базы. Активный каталог Windows представляет собой частный случай общего понятия каталога. На каждом контроллере домена хранится своя копия активного каталога, эти

копии регулярно синхронизируются между собой. Место физического хранения файлов активного каталога указывается администратором в ходе установки сервисов активного каталога, в большинстве конфигураций предлагается путь по умолчанию %windir%\SYSVOL.

Информация, хранящаяся в активном каталоге, не ограничивается одними только учетными записями пользователей. В общем случае в активном каталоге могут храниться любые данные, структурированные в соответствии с требованиями формата базы данных.

Данные, хранящиеся в активном каталоге, рассматриваются как совокупность объектов, имеющих атрибуты (подобъекты), при этом набор атрибутов объекта определяется типом объекта. Подобъекты объекта могут содержать вложенные подобъекты, всего поддерживается пять уровней вложенности подобъектов (включая сам объект), однако в текущих версиях Windows используются только три уровня:

- объект;
- набор свойств (property set) — подобъект первого уровня;
- свойство (property) — подобъект второго уровня;

Среди объектов выделяются *контейнеры* — объекты, содержащие другие объекты. Объекты активного каталога образуют древовидную иерархическую структуру, подобную структуре файловой системы, в роли каталогов выступают контейнеры, а в роли файлов — объекты других типов.

Каждый объект активного каталога уникально идентифицируется 128-битным числовым идентификатором GUID. Кроме того, каждый объект имеет уникальное текстовое имя в специальном формате DN (distinguished name). DN однозначно идентифицирует объект, в активном каталоге не могут существовать два разных объекта, имеющих общее DN. Для того чтобы обратиться к объекту, необязательно знать его DN, поиск объекта может быть осуществлен по части DN или по совокупности атрибутов объекта. DN имеет вид:

/атрибут=значение/атрибут=значение...

Некоторые клиентские программы используют альтернативную запись DN:

атрибут=значение,атрибут=значение...

Начиная с Windows 2000, учетные записи пользователей и псевдопользователей хранятся в активном каталоге. Имя пользователя Windows может представляться в одном из следующих форматов:

- внутренний (DN) — /O=организация/D = DNS-домен/CN = Users/CN=имя пользователя;
- основной — имя пользователя@DNS-домен;

- совместимый с Windows NT — NetBIOS-домен\имя_пользователя;
- сокращенный — имя_пользователя.

Последние два формата не обеспечивают однозначную идентификацию пользователя. При использовании этих форматов осуществляется поиск в каталоге учетной записи пользователя, имеющего заданное имя, причем результатом поиска всегда является первая найденная запись (после нахождения первой записи поиск прекращается).

Нетрудно видеть, что основной формат имени пользователя Windows в точности совпадает с форматом, принятом в современных системах электронной почты. Это не является случайным совпадением. Список пользователей Windows весьма тесно интегрирован с электронной почтой и системами электронного документооборота. Так, группы пользователей Windows могут использоваться как списки рассылки Microsoft Exchange. Отдельные пользователи или целые группы могут иметь атрибут «не использовать при управлении доступом к объектам», такие пользователи и группы не могут получать доступ к ресурсам сети иначе как через электронную почту или автоматизированные системы электронного документооборота.

Помимо локальных групп пользователей, зарегистрированных на отдельных компьютерах, и глобальных групп, зарегистрированных в отдельных доменах, в лесу доменов Windows могут существовать так называемые *вселенские*[*] группы. Вселенские группы определены в пределах всего леса, они могут использоваться при настройке политики безопасности на любом компьютере леса. Также вселенские группы могут выступать в роли глобальных списков рассылки. Вселенские группы могут включать в себя пользователей, глобальные группы и другие вселенские группы. Включаться вселенские группы могут только в другие вселенские группы.

Таким образом, система групп Windows (локальные, глобальные и вселенские группы) позволяет описывать любые иерархические отношения между пользователями на любом из трех уровней: компьютера, домена и леса в целом.

Наиболее важные объекты активного каталога помещаются в так называемый *глобальный каталог* (*global catalog*, *GC*), который автоматически реплицируются на все контроллеры всех доверяемых

---

[*] В русскоязычной литературе чаще встречается термин *универсальная группа*, являющийся результатом некорректного перевода английского термина universal group.

доменов. Данные, лежащие в глобальном каталоге, доступны из любой точки любого доверяемого домена в любой момент времени (естественно, исключая фатальные сбои в функционировании сети).

В сети Windows домены объединяются в единую структуру более высокого уровня, называемую *лесом*. Логическая структура леса представляет собой одно или несколько деревьев, узлами которого являются домены. Все домены, принадлежащие одному лесу, связаны между собой двусторонними и транзитивными отношениями доверия. В отличие от Windows NT, в доменах Windows 2000 и более поздних версий нет необходимости устанавливать отношения доверия вручную — при включении в лес нового домена автоматически устанавливаются отношения доверия со всеми другими доменами леса. Все деревья, составляющие лес, а также все их поддеревья являются организационными единицами. Это позволяет одной операцией назначать одни и те же настройки политики безопасности многим компьютерам сети.

Одним из важнейших элементов активного каталога является его *схема*, включающая в себя список поддерживаемых классов (типов) объектов и список поддерживаемых атрибутов для каждого класса. Все домены леса разделяют общую схему, вся информация о схеме активного каталога полностью реплицируется на каждый контроллер каждого домена. Контейнер, в котором хранится схема леса, всегда имеет DN вида /CN=Schema/CN=Configuration/DC=имя_домена, где имя_домена — имя любого домена данного леса.

Классы объектов активного каталога делятся на три категории:

- структурные (structural) — «нормальные» классы объектов, для которых могут существовать объекты, к ним относящиеся;
- абстрактные (abstract) — шаблоны для порождения других классов, например абстрактный класс Top используется как шаблон для порождения всех других классов;
- дополнительные (auxiliary) — используются для добавления к объектам других классов стандартных наборов атрибутов. Например, структурный класс User (пользователь) использует наборы атрибутов, определяемые дополнительными классами Mail-Recipient (получатель электронной почты) и Security-Principal (субъект доступа).

Каждому типу объектов активного каталога соответствует регистрационная запись в схеме леса, содержащая следующие основные поля:

cn — основное имя (common name) данного класса. Именно этот атрибут указывается в DN объектов, относящихся к данному классу;

lDAPDisplayName — отображаемое имя данного класса, используется клиентскими программами для вывода результатов запросов пользователя;

schemaIDGUID — GUID класса, 128-битный целочисленный идентификатор, уникальный в пределах всего леса;

mustContain — список обязательных атрибутов объекта данного класса;

mayContain — список необязательных атрибутов объекта данного класса;

possSuperiors — список классов контейнеров, в которых могут лежать объекты данного класса;

objectClassCategory — категория данного класса: структурный, абстрактный или дополнительный;

subClassOf — родительский класс, от которого порожден данный класс. Все атрибуты родительского класса наследуются дочерним классом;

auxiliaryClass — список дополнительных классов, использованных при регистрации данного класса;

defaultHidingValue — булевское значение, описывающее, должны ли объекты данного класса по умолчанию иметь атрибут «скрытый» (если TRUE, то должны). Скрытые объекты не отображаются при просмотре активного каталога с помощью стандартной консоли администрирования (MMC);

systemOnly — булевское значение, описывающее, может ли данный класс модифицироваться администраторами (если TRUE, то не может);

defaultSecurityDescriptor — дескриптор защиты по умолчанию для вновь создаваемых объектов данного класса;

isDefunct — булевское значение, описывающее, является ли данный класс отмененным. Для отмененных классов не могут создаваться новые объекты, однако уже существующие объекты автоматически не удаляются;

description — текстовое описание класса.

Каждому атрибуту объекта также соответствует регистрационная запись в схеме леса, она содержит следующие основные поля:

cn, lDAPDisplayName, schemaIDGUID, systemOnly, isDefunct, description — то же, что и для классов объектов;

attributeSyntax — тип атрибута. Поддерживаются один булевский тип, семь целочисленных типов и восемнадцать строковых и бинарных типов;

rangeLower, rangeUpper — минимальное и максимальное допустимые значения (для целочисленных атрибутов), минимальная и максимальная допустимые длины в байтах (для строковых и бинарных атрибутов);

isSingleValued — булевское значение, описывающее, содержит ли данный атрибут одно значение или список однотипных значений (если TRUE, одно значение).

В качестве примера приведем небольшое подмножество стандартных классов объектов в активном каталоге Windows 2003 с указанием иерархии.

Top (абстрактный класс)
Domain (абстрактный класс)
Domain-DNS (с участием дополнительного класса Sam-Domain)
Group (с участием дополнительных классов Mail-Recipient и Security-Principal)
Group-Policy-Container
Leaf (абстрактный класс)
Connection-Point (абстрактный класс)
Print-Queue
Service-Connection-Point
MS-SQL-SQLServer
MS-SQL-SQLDatabase
Volume
Domain-Policy
Secret
Organizational-Unit
Person (абстрактный класс)
Organizational-Person (абстрактный класс)
Contact (с участием дополнительного класса Mail-Recipient)
User (с участием дополнительных классов Mail-Recipient и Security-Principal)
Computer
inetOrgPerson
Security-Object (абстрактный класс)
Sam-Server

Вышеописанная архитектура активного каталога может показаться чрезмерно сложной, но на самом деле наличие абстрактных и дополнительных классов не усложняет, а упрощает создание новых структурных классов.

Например, абстрактный класс Connection-Point содержит в себе все атрибуты, общие для всех объектов, к которым может подключиться по сети удаленный пользователь. Производные от Connection-Point структурные классы Volume (логический диск), Print-Queue (сетевой принтер) и Service-Connection-Point (сетевой сервис) лишь детализируют свойства объекта класса Connection-Point. Так, например, из 97 атрибутов объекта класса Volume лишь три атрибута (Content-Indexing-Allowed — должно ли осуществляться индексирование диска, Last-Content-Indexed — время последнего индексирования и UNC-Name — сетевое имя в формате UNC) вводятся непосредственно в классе Volume. Еще три атрибута (Keywords — ключевые слова для поиска, Managed-By — пользователь, отвечающий за обслуживание объекта и ms-DS-Settings — строка, описывающая свойства объекта в произвольной форме) введены в классе Connection-Point, а все остальные атрибуты унаследованы от корневого класса Top.

Аналогично из 285 атрибутов класса Computer:

91 атрибут унаследован от абстрактного класса Top;

6 атрибутов унаследованы от абстрактного класса Person;

54 атрибута унаследованы от абстрактного класса Organizational-Person;

14 атрибутов унаследованы от дополнительного класса Mail-Recipient;

14 атрибутов унаследованы от дополнительного класса Security-Principal;

84 атрибута унаследованы от класса User;

и только лишь 22 атрибута специфичны для класса Computer.

Следует отметить, что в некоторых случаях дизайнеры Microsoft излишне увлеклись наследованием атрибутов от родительских классов. Например, каждый объект типа Computer имеет атрибуты «фамилия», «домашний адрес» и «домашний телефон», что вызывает недоумение. Впрочем, все эти атрибуты являются необязательными и вряд ли будут когда-либо применены к какому-то реальному компьютеру.

Не следует думать, что огромное количество атрибутов, определенных для объектов различных классов, приводит к столь же огромному расходу дисковой и оперативной памяти. Подавляющее большинство атрибутов являются необязательными и отсутствуют у подавляющего большинства объектов.

Для доступа к активному каталогу в Windows 2000 может использоваться либо специально разработанный в Microsoft протокол

ADSI, либо протокол LDAP (RFC 2251), использующийся для доступа к другим каталогам. Ряд функций активного каталога доступны также по протоколам MAPI-RPC и X.500.

В большом лесу запросы пользователей далекого домена могут проходить долгий путь. Для оптимизации путей прохождения запросов в лесах доменов Windows используется понятие *сайт*, или *узел* (*site*). В каждый сайт входят компьютеры, соединенные между собой высокоскоростными линиями связи. Компьютеры, входящие в одну подсеть протокола IP, всегда входят в один сайт. Внутри одного сайта репликация всегда осуществляется с использованием протокола RPC. Репликация между сайтами может происходить либо по RPC, либо с использованием одного из MAPI-протоколов (SMTP, X.400 и т. п.). Как правило, сайт объединяет все компьютеры одного или нескольких доменов. Ситуаций, когда домен разбивается на два или более сайтов, рекомендуется избегать, поскольку это заметно ухудшает репликацию внутри домена.

Для идентификации компьютеров в лесу используется протокол DNS*, гарантирующий уникальность имен компьютеров в отличие от протокола NetBIOS, применявшегося для этой цели в Windows NT. Например, DNS-именам server.filial.xxx.ru и server.filial.xxx.ua соответствует общее NetBIOS-имя filial/server. Сервер DNS входит в состав дистрибутива Windows Server, обычно сервер DNS устанавиливается на всех контроллерах доменов.

Любой объект активного каталога может быть защищен дескриптором защиты, имеющим такой же формат, как и дескриптор защиты локального объекта Windows.

Дескрипторы защиты объектов типа «организационная единица» позволяют *делегировать* пользователям полномочия на доступ к тем или иным ресурсам леса. Например, для того, чтобы делегировать некоторому пользователю полномочия управлять списком пользователей некоторого домена, достаточно добавить в DACL дескриптора защиты домена ACE, предоставляющий пользователю требуемые права доступа.

Дескрипторы защиты объектов активного каталога наследуются по тем же правилам, что и дескрипторы защиты локальных объектов операционной системы. Для объектов активного каталога поддерживается атрибут «автоматическое наследование», позволяющий рекурсивно распространять делегирование полномочий пользователей на домены низших уровней.

---

* Точнее, DDNS (RFC 2136).

5.5. Групповая политика

Одним из важнейших новшеств в подсистеме защиты Windows является *групповая политика* (*group policy*) — совокупность объектов активного каталога, описывающих те или иные аспекты конфигурации операционной системы, а также индивидуальных настроек отдельных пользователей и групп. Групповая политика включает в себя большинство элементов политики безопасности Windows, в частности:

- распределение привилегий между пользователями;
- параметры подсистемы аутентификации, включая параметры протокола Kerberos;
- политику аудита;
- параметры системных журналов, включая журнал аудита;
- параметры сервисов, включая режим запуска (автоматический/ручной/запуск запрещен) и дескриптор защиты сервиса;
- список агентов восстановления EFS;
- шаблоны настроек отдельных прикладных и системных программ (Internet Explorer, Task Scheduler, Windows Installer и т. п.).

В типичных конфигурациях Windows групповая политика содержит около 200-400 элементов, для удобства администрирования они объединены в древовидную иерархическую структуру контейнеров.

Каждый компьютер, работающий под управлением Windows, имеет собственную групповую политику. Исключение составляют контроллеры доменов, которые разделяют между собой общую групповую политику, единую для всех контроллеров одного домена. Также существует единая групповая политика для всего домена в целом.

В домене могут быть выделены особые группы пользователей и компьютеров, называемые *организационными единицами* (*organizational units*). Организационные единицы отличаются от обычных групп тем, что им могут назначаться собственные групповые политики. Это позволяет одномоментно назначать одинаковые настройки всем компьютерам, входящим в состав одной организационной единицы. Тем самым организационные единицы упрощают администрирование большой сети. Организационные единицы могут включаться одна в другую, групповые политики вышележащих организационных единиц наследуются нижележащими организационными единицами. Также групповые политики могут назначаться сайтам.

Каждый элемент групповой политики может быть либо не определен, либо иметь некоторое значение. Тип и диапазон возмож-

ных значений различаются для разных элементов групповой политики. Так, значение элемента Computer Configuration\Windows Settings\Security Settings\Account Policies\Password Policy\Password must meet complexity requirements может принимать значения «да» или «нет», значением элемента Computer Configuration\Windows Settings\Security Settings\Account Policies\Password Policy\Enforce password history является целое число от 0 до 24, а значение элемента Computer Configuration\Windows Settings\Scripts\Startup представляет список текстовых строк переменной длины.

Значение элемента групповой политики, определенное в некоторой организационной единице, автоматически наследуется всеми нижележащими организационными единицами. Например, если администратор домена присвоил элементу Computer Configuration\Windows Settings\Security Settings\Local Policies\Audit Policy\Audit logon events групповой политики домена значение «Failure», неудачные попытки входа пользователей в систему будут регистрироваться в журнале аудита каждого компьютера домена.

Если значение элемента групповой политики, унаследованное от групповой политики вышележащей организационной единицы, вступает в противоречие с значением того же элемента, определенного в нижележащей организационной единице, то конфликт разрешается по следующим правилам:

- если администраторы обеих организационных единиц не указали никаких особых правил разрешения данного конфликта, то действует унаследованное значение;
- если администратор нижележащей организационной единицы установил на данную групповую политику флаг «не наследовать сверху», а администратор вышележащей организационной единицы не установил порядок разрешения данного конфликта, то действует значение, определенное администратором нижележащей организационной единицы;
- если администратор вышележащей организационной единицы установил на данную групповую политику флаг «наследовать вниз в любом случае», то унаследованное значение данного элемента групповой политики действует в любом случае, независимо от того, что определил в отношении данного элемента администратор нижележащей организационной единицы.

Групповая политика позволяет администраторам организационных единиц централизованно управлять политикой безопасности большой сети. Если есть необходимость изменить некоторый аспект политики безопасности всего дерева, администратору корня дерева

достаточно всего лишь настроить соответствующим образом групповой политики корня дерева и внесенные им изменения будут автоматически реплицированы на все компьютеры дерева. При этом администраторы нижележащих организационных единиц, несогласные с решением более высокого администратора, могут (если это явно не запрещено администратором вышележащей организационной единицы) отменить это решение в части, касающейся подведомственных им организационных единиц леса.

Обычно администраторы организационных единиц высокого уровня оставляют большинство полей групповой политики незаполненными. Это дает администраторам организационных единиц низшего уровня свободу выбора в настройке политики безопасности своих организационных единиц.

Элементы групповой политики сгруппированы в древовидную иерархическую структуру, аналогичную структуре файловой системы или реестра. Групповая политика, назначенная компьютеру, домену, организационной единице или сайту, включает в себя два больших раздела:

- Computer Configuration (конфигурация компьютера) — содержит политики, действующие на всю операционную систему в целом;
- User Configuration (конфигурация пользователя) — содержит политики, действующие на индивидуальные настройки пользователей, работающих с данным компьютером.

Пользователям и группам, не являющимся организационными единицами, может назначаться «неполноценная» групповая политика, содержащая только раздел User Configuration. Эта групповая политика считается более высокоприоритетной, чем групповая политика компьютера.

Многие элементы групповых политик проецируются в реестр Windows, т. е. при изменении значения элемента групповой политики автоматически меняется соответствующее значение реестра Windows. При этом политики из раздела Computer Configuration проецируются в ключ реестра HKEY_LOCAL_MACHINE, а политики из раздела User Configuration — в HKEY_CURRENT_USER. Например, элемент групповой политики User Configuration\Windows Settings\ Internet Explorer Maintenance\Browser User Interface\Browser Title проецируется в значение Window Title ключа реестра HKEY_CURRENT_USER\Software\Microsoft\ Internet Explorer\Main. При каждом обновлении элемента групповой политики автоматически обновляется соответствующее значение реестра, это называется *при-*

менением групповой политики. Применение групповой политики происходит не только при внесении в нее изменений, но и автоматически, по умолчанию задан полуторачасовой интервал между обновлениями. Кроме того, политики из раздела Computer Configuration применяются при кажом старте операционной системы, а политики из раздела User Configuration — при каждом входе пользователя в систему. Впрочем, эксперименты показывают, что применение групповой политики часто происходит реже, чем заявлено в документации Microsoft.

В Windows 2000 применение групповых политик происходило в синхронном режиме — на компьютерах, входящих в состав домена, загрузка операционной системы не завершалась до применения всех политик из разделов Computer Configuration, вход пользователя в систему не завершался до применения всех политик из разделов User Configuration. Это приводило к неприятным задержкам, и, начиная с Windows XP, на рабочих станциях применение групповых политик проходит асинхронно. В результате в течение первых минут после авторизации пользователя конфигурация операционной системы может регулироваться устаревшими групповыми политиками. На серверах, начиная с Windows 2003, политики Computer Configuration применяются синхронно, а политики User Configuration — асинхронно.

Если прикладная или системная программа считывает свои конфигурационные данные точно в тот момент, когда происходит применение групповой политики, возможна ситуация, когда некоторые данные, считанные программой, относятся к обновленной групповой политике, а некоторые другие данные — к старой, необновленной политике. Для предотвращения таких ситуаций предназначены специальные системные функции EnterCriticalPolicySection и LeaveCriticalPolicySection, расположенные в библиотеке userenv.dll. Перед считыванием своей конфигурации из реестра программа должна вызвать функцию EnterCriticalPolicySection, а по завершении считывания — функцию LeaveCriticalPolicySection. Между этими двумя вызовами гарантируется, что групповая политика в это время применяться не будет. Интервал времени между этими вызовами не должен превосходить десяти минут, в противном случае операционная система ведет себя так, будто программа вызвала LeaveCriticalPolicySection.

Если пользователь, обладающий соответствующими полномочиями, вручную модифицирует значение реестра, на которое проецируется элемент групповой политики, это изменение будет действо-

вать только до следующего применения групповой политики, затем оно будет отменено.

Каждый из двух разделов групповой политики распадается на три подраздела:

- Software Settings — содержит политики, используемые сторонним программным обеспечением, не входящим в состав дистрибутива Windows. Чаще всего этот подраздел пуст;
- Windows Settings — содержит политики, описывающие настройки различных компонент Windows. Большинство политик, содержащихся в данном подразделе, связаны с безопасностью операционной системы;
- Administrative Templates — содержит политики, позволяющие автоматизировать управление большой сетью, применяя одни и те же настройки к разным компьютерам сети. В отличие от подраздела Windows Settings, в подразделе Administrative Templates по умолчанию все политики не определены. Администратор может определять эти политики либо вручную, с помощью консоли администрирования Windows (MMC), либо с использованием заранее подготовленных файлов административных шаблонов (ADM-файлов).

В качестве примера перечислим политики подраздела Administrative Templates, которые могут быть применены к подсистеме печати Windows (Computer Configuration/Administrative Templates/Printers):

- допустима ли печать документов веб-сервером по запросам пользователей Internet (не поддерживается начиная с Windows 2003);
- должны ли новые принтеры автоматически публиковаться в активном каталоге (не поддерживается начиная с Windows Vista, в более ранних версиях по умолчанию — да);
- какую веб-ссылку должен выводить Проводник Windows вместо заданной по умолчанию ссылки «Получение справки о выводе на печать», указывающей на сайт Microsoft;
- какое максимальное количество принтеров каких типов должен отображать Проводник Windows на странице сканирования сети, если установлена связь с контроллером домена (поддерживается начиная с Windows Vista). По умолчанию заданы следующие значения: 20 принтеров активного каталога, 10 принтеров Bluetooth, TCP/IP-принтеры и веб-принтеры не отображать;
- какое максимальное количество принтеров каких типов должен отображать Проводник Windows на странице сканирова-

ния сети, если не удается установить связь с контроллером домена (поддерживается начиная с Windows Vista). По умолчанию заданы следующие значения: 50 TCP/IP-принтеров, 50 веб-принтеров, 10 принтеров Bluetooth;

- следует ли формировать задание на печать на стороне клиента (по умолчанию) или на стороне сервера (поддерживается начиная с Windows Vista);
- следует ли показывать недоступные принтеры в списке принтеров активного каталога (по умолчанию — нет);
- разрешено ли использование на данном компьютере драйверов принтера, работающих в режиме ядра (т. е. именно драйверов, а не библиотек). По умолчанию в Windows XP и ранее драйверы принтера, работающие в режиме ядра, разрешены, в Windows 2003 — запрещены, а начиная с Windows Vista такие драйверы запрещены в любом случае, независимо от состояния данного элемента групповой политики;
- какое место размещения сетевых принтеров считать ближайшим (по умолчанию определяется автоматически на основе IP-адреса компьютера);
- следует ли учитывать атрибут «размещение» объекта «принтер» при поиске сетевых принтеров (по умолчанию — нет);
- следует ли включать сетевые принтеры в карту сетевого окружения, поддерживаемую сервисом Computer Browser (по умолчанию — тогда и только тогда, когда отсутствует связь с контроллером домена);
- в каких случаях следует удалять из активного каталога недоступные принтеры: никогда (по умолчанию), когда принтер недоступен, но доступен сервер печати, к которому этот принтер должен быть подключен, либо всегда;
- как часто следует проверять доступность сетевых принтеров (по умолчанию — каждые восемь часов);
- какой относительный приоритет должен присваиваться потоку сервера активного каталога, отслеживающему недоступные принтеры (по умолчанию — обычный относительный приоритет, THREAD_PRIORITY_NORMAL);
- сколько раз принтер должен не ответить на запрос, чтобы быть признанным недоступным (по умолчанию — три раза);
- регистрировать ли в журнале результаты проверки доступности сетевых принтеров (поддерживается начиная с Windows XP, по умолчанию — нет);

- можно ли публиковать информацию о сетевых принтерах в активном каталоге (по умолчанию — да);
- может ли сервер печати обслуживать удаленных клиентов (поддерживается начиная с Windows 2003, по умолчанию — да, если к компьютеру подключен хотя бы один сетевой принтер);
- как часто сервер печати должен проверять доступность сетевых принтеров, подключенных к данному компьютеру (по умолчанию — только один раз, при старте сервера печати).

Физически групповые политики хранятся в контейнере /CN=Policies/CN=System/DC=имя_домена активном каталога. Каждому объекту групповой политики соответствует объект класса groupPolicyContainer, имя (CN) которого представляет собой текстовое представление GUID данной групповой политики (этот GUID отличается от атрибута objectGUID объекта групповой политики). Групповая политика может содержать ссылки на ассоциированные с нею файлы (например, на скрипт, который должен автоматически выполняться перед завершением работы операционной системы), путь к месту хранения этих файлов хранится в атрибуте gPCFileSysPath объекта групповой политики. По умолчанию эти файлы хранятся на контроллере домена в директории sysvol\имя_домена\Policies\{GUID_групповой_политики}. Внутри каждого контейнера групповой политики располагаются подконтейнеры с именами (CN) Machine и User, в них хранятся, соответственно, разделы Computer Configuration и User Configuration данной групповой политики.

Вопросы для самопроверки

1. Какую основную задачу помогает решать доменная архитектура корпоративных сетей?

2. Что такое контроллер домена?

3. Что такое сквозная аутентификация в доменах Windows?

4. Как обеспечивается в Windows защита протокола сквозной аутентификации от навязывания нарушителем ложного сервера?

5. Для чего предназначено кэширование аутентификационной информации пользователей доменов Windows?

6. Какие субъекты доступа могут входить в глобальные группы доменов Windows?

7. Каким субъектам доступа могут назначаться права и привилегии на компьютерах, входящих в состав доменов Windows?

8. Как строятся отношения доверия между доменами Windows, входящими в один и тот же лес?

9. Какие особенности имеет сквозная аутентификация в ситуации, когда пользователь, зарегистрированный в одном домене леса, обращается к серверу, входящему в состав другого домена того же леса?

10. Что такое активный каталог?

11. Какова внутренняя структура объектов активного каталога?

12. В каких форматах может быть представлено в активном каталоге имя пользователя?

13. Что такое глобальный каталог?

14. Какие категории классов объектов активного каталога вы знаете?

15. Как описываются наборы обязательных и необязательных атрибутов объектов того или иного класса?

16. Как в активном каталоге описываются списки, составленные из однотипных значений?

17. Для решения каких задач в системе классов активного каталога применяются абстрактные и дополнительные классы?

18. Что в лесу доменов Windows называется сайтом?

19. Как реализуется делегирование полномочий пользователей в лесу доменов Windows?

20. Что называется групповой политикой в доменах Windows?

21. По каким правилам разрешаются конфликты, которые могут возникать при наследовании групповой политики?

22. Каково основное предназначение групповой политики?

23. В какие два основных раздела группируются значения групповой политики, входящие в состав одного объекта групповой политики?

24. На какие три подраздела распадается каждый из этих основных разделов?

25. Что такое применение групповой политики?

26. Где физически хранятся групповые политики?

6 Безопасность операционных систем мобильных устройств

Одной из наиболее заметных тенденций развития компьютерной техники в последние годы является неуклонный рост вычислительной мощности мобильных электронных устройств: мобильных телефонов, планшетных компьютеров, электронных книг и т. п. Начиная примерно с 2007 года, граница между персональными компьютерами и мобильными электронными гаджетами стала заметно стираться. Сегодня почти не вызывает удивления ситуация, когда мобильный телефон превосходит некоторые персональные компьютеры по вычислительной мощности процессора, объему оперативной и долговременной памяти. Современные мобильные устройства оснащаются полноценными операционными системами, близкими по своим возможностям к универсальным операционным системам персональных компьютеров. Неудивительно, что мобильные операционные системы заимствуют от универсальным операционных систем не только отдельные алгоритмические решения, но и целые программные компоненты. Так, самая распространенная на сегодняшний день мобильная операционная система Google Android использует то же ядро, что и универсальная операционная система Linux, а вторая по популярности мобильная операционная система Apple iOS построена на основе прикладных интерфейсов универсальных операционных систем семейства BSD UNIX.

Типичные угрозы безопасности операционной системы мобильного устройства существенно отличаются от аналогичных угроз для операционной системы персонального компьютера или сетевого сервера. Некоторые угрозы, малозначительные для обычных компьютеров, становятся очень опасными для мобильных устройств, и наоборот. Например, кража мобильного телефона карманным вором является намного более серьезной угрозой, чем кража сервера вором-домушником. Программная закладка, внедренная в операционную систему персонального компьютера и получившая доступ к электронным банковским счетам пользователя, обычно имеет весьма ограниченные возможности по несанкционированному переводу денежных средств с этих счетов. Но программная закладка, внедренная в операционную систему мобильного устройства, элементарно решает данную задачу путем несанкционированного заказа дорогостоящих SMS-услуг с телефонного номера, контролируемого нару-

шителем, либо (реже) путем имитации голосового звонка на платный номер. С другой стороны, угрозы, связанные с одновременным доступом нескольких пользователей к одному экземпляру операционной системы, для мобильных операционных систем, как правило, неактуальны.

К наиболее актуальным угрозам безопасности мобильных операционных систем обычно относят следующие:

- раскрытие конфиденциальной информации в результате утери или кражи мобильного устройства;
- несанкционированный заказ дорогостоящих услуг программной закладкой, внедренной в операционную систему мобильного устройства;
- раскрытие конфиденциальной информации в результате перехвата беспроводного сетевого трафика, генерируемого мобильным устройством;
- несанкционированный сбор программной закладкой персональных данных пользователя мобильного устройства;
- потеря данных, хранящихся на мобильном устройстве.

В настоящее время для мобильных операционных систем разрабатывается огромное количество вредоносного программного обеспечения. По данным [24], около четверти всех приложений, написанных для операционной системы Android, являются вредоносными. Количество вредоносных программ для Android растет экспоненциально, каждый год оно увеличивается в 2–16 раз. Такое большое различие в цифрах, даваемых разными источниками, объясняется тем, что граница между множествами вредоносных и невредоносных мобильных приложений весьма условна. Хорошим примером «пограничных» приложений являются программы, позволяющие легальному владельцу телефона обнаружить украденный у него телефон путем незаметного перехвата и доставки на заданный адрес электронной почты информации о сделанных звонках, отправленных и полученных SMS, географическом местоположении и т. п. Если такое приложение скрытно установлено на мобильное устройство, принадлежащее чужому человеку, оно, очевидно, является вредоносным. Но если оно скрытно установлено на телефон, подаренный жене или ребенку, вопрос о вредоносности приложения неочевиден. Другой пример «пограничных» приложений — навязчивые баннерные сети.

Одно из наиболее типичных вредоносных действий вредоносного мобильного приложения — отправка дорогостоящих SMS на

номер, контролируемый создателем приложения или его сообщниками. Реже встречаются автоматизированный сбор персональных данных, перехват и подмена сетевого и SMS-трафика при доступе клиента к банковских системам, подмена рекламных баннеров в веб-браузере на баннеры, предоставляемые сообщниками разработчика приложения. В конце 2011 года появилось несколько вредоносных программ, несанкционированно записывающих телефонные разговоры пользователя и отправляющих полученные звуковые файлы на заданный сервер Internet. Теоретически возможно распознавание вредоносным приложением перехваченной речевой информации и последующая передача на заданный адрес только тех фрагментов телефонных разговоров, которые содержат заранее заданные ключевые слова. Впрочем, пока такая функциональность реализовывалась только в опытно-демонстрационных приложениях [22], но не в «боевых» программных закладках.

Начиная с марта 2011 года, отмечаются единичные случаи появления вредоносных приложений на сайте Android Market. В середине июня 2011 года появилась модификация знаменитого бота Zeus, предназначенная для работы под управлением операционой системы Android. Кроме того, существуют версии этого бота под мобильные операционные системы Symbian, Windows Mobile и BlackBerry.

По данным «Лаборатории Касперского», по состоянию на август 2012 года вредоносное программное обеспечение, предназначенное для операционной системы Android, классифицируется следующим образом:

49 % — программные закладки, собирающие персональные данные пользователей для последующей рассылки спама;

25 % — программные закладки, несанкционированно заказывающие услуги путем отправки SMS;

18 % — многофункциональные боты, входящие в состав ботнетов;

2 % — программные закладки, специализирующиеся на сборе информации о банковских счетах;

6 % — другие вредоносные приложения.

Типичными симптомами заражения мобильного устройства вредоносными программами являются:

- необъяснимо быстрое уменьшение денежных средств на счету;
- появление новых ярлыков в списке установленных приложений;
- появление непонятных записей в папках полученных и отправленных SMS-сообщений или в списках входящих и исходящих голосовых вызовов;

- скачкообразное увеличение расхода электроэнергии без видимых причин.

Не менее серьезной проблемой, чем заведомо вредоносное программное обеспечение, являются некорректно работающие мобильные приложения, причиняющие ущерб пользователю непреднамеренно в результате допущенных разработчиками программных ошибок. Пользователи часто не понимают, что причина некорректной работы мобильного устройства кроется не в устройстве как таковом, а в недостаточно отлаженном приложении. В результате недоработанные приложения могут оказывать заметное негативное влияние на репутацию разработчиков операционной системы мобильного устройства и самого мобильного устройства как такового.

Компания Apple решает эту проблему путем тщательного тестирования всех мобильных приложений сторонних производителей, разработанных для iPhone и iPad. Такое тестирование может занимать несколько недель, более половины приложений в итоге получают отказ в размещении в официальном онлайн-магазине Apple App Store. По некоторым сведениям [18], причиной отказа часто является не недостаточно высокое качество приложения, а то, что данное приложение может составить конкуренцию собственным разработкам Apple. Кроме того, каждый производитель программного обеспечения для iOS должен быть предварительно зарегистрирован в Apple App Store, что стоит от 100 долларов США в год. Хотя эта сумма и является чисто символической, для многих вирусописателей даже такая сумма неприемлема.

В целом данная стратегия позволяет радикально сократить количество и разнообразие вредоносного программного обеспечения, разрабатываемого для операционной системы iOS. Пока известно только одно вредоносное приложение, сумевшее проникнуть на Apple App Store и получить заметное распространение — спам-бот Find and Call. Несколько других вредоносных приложений (в том числе и знаменитый вирус Ike) опасны только для тех мобильных устройств, на которых взломана программная блокировка, запрещающая установку приложений из источников, отличных от App Store (т. е. выполнен jailbreak).

С другой стороны, ограничение доступа сторонних фирм на рынок приложений для iOS заметно снижает популярность этой операционной системы среди конечных пользователей. Так, по данным [23], по состоянию на май 2012 года, в США под управлением iOS работало 29 % мобильных устройств, в то время как доля устройств, работающих под управлением Android, составляла 61 %.

Перечислим некоторые правила, которым должно удовлетворять каждое приложение, доступное через Apple App Store:

- приложение не должно содержать недокументированных функций, как и использовать недокументированные функции операционной системы;
- приложение не должно выполнять загрузку больших файлов из мобильной сети без уведомления пользователя;
- приложение не должно устанавливать или запускать другие приложения;
- приложение не должно обращаться к данным других приложений;
- приложение не должно собирать персональные данные пользователя (включая географическое местоположение) без явно выраженного согласия пользователя.

Компания Google, в отличие от Apple, декларирует максимальную открытость своей операционной системы Android. Каждое приложение для Android должно быть подписано разработчиком, но заверение этой подписи каким бы то ни было удостоверяющим центром не требуется. В каталог Google Play и интернет-магазин Android Market допускаются все приложения, кроме заведомо вредоносных и заведомо неработоспособных. Время от времени это приводит к скандальным ситуациям, негативно сказывающихся на репутации торговой марки Android. Например, Android-версия приложения eMobiStudio MemoryUp, хорошо зарекомендовавшего себя на платформах Symbian, BlackBerry и Windows Mobile, несанкционированно удалила из-за программной ошибки данные адресных книг нескольких тысяч пользователей, установивших эту программу.

Открытость операционной системы Android не следует преувеличивать. В отличие от большинства разработчиков универсальных операционных систем, компания Google сохраняет за собой контроль над всеми приложениями, работающими на всех экземплярах операционной системы Android. При необходимости компания Google, так же, как и Apple, может одновременно удалить все экземпляры любого заданного приложения, установленного на всех мобильных устройствах со своей операционной системой.

Долгое время обеспечение безопасности мобильных операционных систем рассматривалось их разработчиками как второстепенная, низкоприоритетная задача. Но ситуация постепенно меняется к лучшему, с каждой следующей версией мобильные операционные системы становятся все более защищенными. При этом разработчики мобильных операционных систем широко используют решения,

апробированные на универсальных операционных системах. Так, в версии 4.2 операционной системы Android начала поддерживаться функция доверенного подтверждения пользователем потенциально опасного действия, очень похожая на UAC в Windows. Эта функция используется по умолчанию при отправке SMS-сообщений — до тех пор, пока пользователь не подтвердит отправку сообщения, оно не будет отправлено, при этом прикладные программы не имеют технической возможности имитировать данное подтверждение. Для более ранних версий Android существуют приложения третьих фирм (например, LBE Security Master, LBE Privacy Guard), реализующие аналогичную функциональность.

В мобильных операционных системах широко применяется давно используемая в универсальных операционных системах идентификация программных файлов по цифровым подписям. Подобно тому, как пользователь Windows может запретить установку на свой компьютер драйверов, не имеющих цифровой подписи Microsoft, пользователь Android может запретить установку на свое устройство программных файлов, безопасность которых не подтверждена компанией Google.

Управление доступом в Android почти не использует унаследованный от Linux дискреционный механизм, основанный на субъектах, объектах, методах и правах доступа. Единственным инструментом управления доступом в Android являются привилегии (permissions*), которые назначаются, в отличие от универсальных операционных систем, не пользователям, а приложениям. Фактически, в операционной системе Android приложения рассматриваются в качестве пользователей. При установке приложения ему назначаются собственные UID и GID, добавляются все необходимые записи в системные файлы Linux, содержащие сведения о зарегистрированных пользователях. Исключением является ситуация, когда несколько приложений, изготовленных одним и тем же разработчиком, явно указывают операционной системе, что их следует выполнять от имени одной и той же учетной записи пользователя. Это может быть сделано посредством атрибута sharedUserId манифеста AndroidManifest.xml, включаемого в дистрибутив Android-приложения.

Поскольку каждое приложение Android в обязательном порядке снабжается цифровой подписью разработчика, теоретически воз-

* Заметим, что в Windows термин permission используется для обозначения не привилегий, а прав доступа, это создает терминологическую путаницу.

можно управление доступом приложений на основе информации об их разработчиках, например, разделение приложений на более доверенные и менее доверенные. Но на практике эта возможность пока не используется.

Перечислим некоторые привилегии Android, часто запрашиваемые приложениями:

- ACCESS_COARSE_LOCATION — позволяет приложению получать доступ к информации о местоположении мобильного устройства, предоставляемой сетями GSM и WiFi;
- ACCESS_FINE_LOCATION — позволяет приложению получать доступ к информации о местоположении мобильного устройства, предоставляемой модулем GPS;
- ACCESS_NETWORK_STATE — позволяет приложению получать информацию о GSM-сети, к которой подключено устройство;
- ACCESS_WIFI_STATE — позволяет приложению получать информацию о WiFi-сети, к которой подключено устройство;
- CALL_PHONE — разрешает приложению инициировать голосовые звонки, за исключением звонков на номера экстренной помощи;
- CAMERA — разрешает приложению доступ к фото/видеокамере;
- INTERNET — разрешает приложению доступ к Internet;
- RECEIVE_SMS — разрешает приложению обрабатывать входящие SMS-сообщения;
- RECORD_AUDIO — разрешает приложению доступ к микрофону;
- WAKE_LOCK — разрешает приложению предотвращать переход устройства в режим сна;
- WRITE_EXTERNAL_STORAGE — разрешает приложению записывать данные на карту внешней памяти устройства.

Операционная система Android не поддерживает динамическое включение и выключение привилегий. Все привилегии, требуемые приложением, предоставляются ему сразу при инсталляции и затем автоматически включаются при каждом запуске. Перечень требуемых приложению привилегий включается в манифест приложения (файл AndroidManifest.xml) в виде списка записей следующего вида:

```
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

Если в ходе выполнения той или иной операции обнаруживается, что привилегия, необходимая для завершения данной операции, не предоставлена приложению, в приложении возникает исключительная ситуация SecurityException. В некоторых случаях (например,

при использовании метода sendBroadcast) приложение не получает никакой информации о причине сбоя или даже вообще не получает информации о случившемся сбое.

Почти все случаи отказа в доступе из-за отсутствия необходимых привилегий регистрируются в системном журнале. Кроме того, операционная система может выдавать пользователю интерактивные сообщения об ошибках, связанных с попытками нарушения действующей политики безопасности теми или иными приложениями.

Помимо привилегий, стандартных для всех приложений Android, каждое конкретное приложение (или группа приложений, изготовленных одним и тем же разработчиком) может определять в AndroidManifest.xml свои собственные нестандартные привилегии примерно следующим образом:

```
<permission android:name="com.me.app.myapp.permission.DEADLY_
ACTIVITY"
android:label="@string/permlab_deadlyActivity"
android:description="@string/permdesc_deadlyActivity"
android:permissionGroup="android.permission-group.COST_MONEY"
android:protectionLevel="dangerous"/>
```

Атрибут protectionLevel является обязательным, он используется операционной системой при выборе способа реакции на исключительную ситуацию, связанную с определяемой привилегией. Атрибут permissionGroup является необязательным, единственное его назначение — выбор сообщения, разъясняющего пользователю смысл случившегося сбоя. Как правило, приложения используют одно из стандартных сообщений операционной системы, это упрощает локализацию таких сбоев. Атрибуты label и description являются обязательными, они содержат, соответственно, идентификаторы краткого и полного текстового описания данной привилегии. Идентификаторы этих описаний описываются примерно следующим образом:

```
<string name="permlab_callPhone"> directly call phone
numbers</string>
<string name="permdesc_callPhone"> Allows the application
to call phone numbers without your intervention.  Malicious
applications may cause unexpected calls on your phone bill.
Note that this does not allow the application to call
emergency numbers.</string>
```

Пользователь может просматривать список привилегий, поддерживаемых в текущей конфигурации операционной системы, с

помощью главного меню пользовательского интерфейса (пункт Settings/Application) или в командной строке командой adb shell pm list permissions.

Временные данные приложений, записываемые во внутреннюю память устройства, хранятся внутри домашних директорий соответствующих пользователей Linux. Поскольку в Linux пользователи по умолчанию не имеют доступа к содержимому домашних директорий друг друга, данные (в том числе и cookie посещенных интернет-сайтов) одного приложения Android обычно недоступны другим приложениям. Исключением является ситуация, когда приложение явно предоставляет доступ другим приложениям к своим данным.

В iOS система управления доступом в целом похожа на аналогичную систему Android, но устроена намного проще и примитивнее. Привилегий поддерживается всего пять:

- обращаться к входящей электронной почте;
- обращаться к входящим SMS;
- отсылать SMS;
- инициировать телефонные звонки;
- обращаться к подсистеме GPS.

Перед тем, как воспользоваться любой из трех последних привилегий, приложение iOS должно получить явно выраженное согласие пользователя на выполнение соответствующего действия.

Как правило, на устройствах, работающих под управлением iOS, выполняются только доверенные приложения, тщательно проверенные и одобренные специалистами Apple. Поэтому к подсистеме управления доступом iOS не предъявляется высоких требований. Таким образом, несмотря на свою крайнюю примитивность, подсистема управления доступом iOS представляется вполне удовлетворительной.

Принцип минимизации полномочий соблюдается в базовых конфигурациях как Android, так и iOS (впрочем, в последнем случае не вполне ясно, можно ли корректно говорить о принципе минимизации полномочий для такой примитивной системы разграничения доступа) — ни одно приложение не имеет технической возможности получить полномочия суперпользователя root. Однако многие пользователи Android и некоторые пользователи iOS отключают эту защиту. В результате приложения, в том числе и вредоносные, получают больше функциональных возможностей, в число которых входят возможности реализовывать стелс-технологии и активно противодействовать средствам администрирования и антивирусному программному обеспечению. Всякое приложение, выполняющееся на

«рутованном» телефоне с полномочиями суперпользователя, имеет ничем не ограниченный доступ ко всем объектам операционной системы телефона. Когда недостаточно отлаженный или заведомо вредоносный программный код получает такой доступ, последствия могут стать фатальными как для самой операционной системы, так и для хранимых и обрабатываемых в ней пользовательских данных.

Операционная система iOS хранит все данные в пользовательской части внешней памяти в зашифрованном виде. Наиболее конфиденциальные данные подвергаются дополнительному шифрованию. Используемые при этом ключи не предоставляются пользователем, а хранятся во внутренней памяти устройства, т. е. защита фактически реализуется только от нарушителя, пытающегося прочитать внешнюю память данного устройства, используя другое устройство. В операционной системе Android, в отличие от iOS, шифрование данных практически не применяется, хотя и поддерживается.

Практически все производители антивирусного программного обеспечения уже разработали версии своих программ для операционной системы Android. По архитектуре и функциональности мобильные антивирусные комплексы не имеют принципиальных отличий от антивирусных комплексов, предназначенных для применения на персональных компьютерах или серверах. Многие эксперты отмечают, что эффективность мобильных антивирусных комплексов существенно уступает эффективности тех же антивирусных комплексов для универсальных операционных систем. Так, по данным исследования, проведенного в марте 2012 года компанией AV-Test [19], две трети мобильных антивирусных комплексов практически бесполезны. Впрочем, для мобильных антивирусов Касперского и F-Secure данное исследование не выявило существенных слабостей реализуемой защиты.

Начиная с версии Android 4.2, поддерживается режим, когда каждое приложение, независимо от источника, из которого оно установлено, должно автоматически проверяться облачным антивирусным сканером Google.

Антивирусные приложения для мобильных операционных систем часто дополняются так называемой «противоугонной» функциональностью, позволяющей легальному пользователю устройства найти свое устройство по сигналам GPS, удаленно заставить устройство издать громкий звуковой сигнал, привлекающий внимание окружающих к вору, удаленно уничтожить персональные данные, хранящиеся на устройстве и т. п. Эффективность такой защиты оставляет желать лучшего. После выключения и полной перепрошив-

ки украденного устройства весь противоугонный функционал отключается. Кроме того, даже исправно работающая противоугонная программа не всегда позволяет вернуть украденное устройство. Так, некто Mystyes пишет в комментариях к [25]: «У меня украли айфон. Телефонная книжка злоумышленника через айклауд синхронизировалась с новым айфоном, но наша милиция найти его не может».

Некоторые мобильные антивирусы (например, Avast) включают в себя дополнительную функциональность, позволяющую пользователю осуществлять фильтрацию входящих и исходящих голосовых звонков и SMS-сообщений, вводить белые и черные списки телефонных номеров.

Как и у любых других операционных систем, программное обеспечение мобильных операционных систем может иметь уязвимости, позволяющие вредоносным приложениям несанкционированно повышать свои полномочия. Так, вредоносные приложения DreamExploid, Wukong и Gongfu эксплуатировали уязвимости старых версий операционной системы Android, чтобы несанкционированно получать полномочия суперпользователя root и устанавливать полный контроль над зараженной системой.

Интересно отметить, что программный код приложений iOS выполняется непосредственно на процессоре мобильного устройства, в то время как программный код каждого приложения Android работает в виртуальной Java-машине. Как и следовало ожидать из соображений здравого смысла, это различие не оказывает заметного влияния на сравнительную безопасность этих двух операционных систем.

Вопросы для самопроверки

1. Каковы наиболее актуальные угрозы безопасности мобильных операционных систем?

2. Какова доля вредоносного программного обеспечения среди всего программного обеспечения, разрабатываемого для операционной системы Android?

3. Какие вредоносные действия наиболее характерны для вредоносных мобильных приложений?

4. Каковы типичные симптомы заражения мобильного устройства вредоносной программой?

5. Какими мерами компания Apple пытается затруднять распространение вредоносного программного обеспечения на мобильных устройствах, произведенных этой компанией?

6. Чем отличается подход компании Google к взаимодействию с независимыми разработчиками приложений для операционной системы Android от аналогичного подхода компании Apple к взаимодействию с независимыми разработчиками приложений iOS?

7. Как в мобильных операционных системах используются цифровые подписи программных файлов?

8. Какие привилегии приложений поддерживаются в операционной системе Android?

9. Поддерживается ли в операционной системе Android динамическое включение и выключение привилегий?

10. Как разработчик приложения для операционной системы Android может определять нестандартные привилегии для использования своим приложением?

11. Какие привилегии приложений поддерживаются в операционной системе iOS?

12. Как в современных мобильных операционных системах реализуется принцип минимизации полномочий пользователей?

13. Какие особенности имеют мобильные антивирусные комплексы по сравнению с аналогичным программным обеспечением, разрабатываемым для персональных компьютеров и сетевых серверов?

7 Виртуализация операционных систем

Виртуализация операционных систем в последние годы стала одним из магистральных направлений развития информационных технологий. По данным [21], в 2011 году 39 % всех операционных систем в мире выполнялись на виртуальных машинах, к 2018 году ожидается, что эта доля достигнет 86 %. По данным [6], по состоянию на апрель 2012 года 59 % средних и крупных российских компаний уже внедрили либо собираются внедрить в ближайшем будущем виртуализацию корпоративных серверов, при этом для серверов баз данных эта доля составляет 80 %.

Основная причина, объясняющая рост интереса к виртуализации операционных систем, носит чисто экономический характер. До примерно 2006 года содержание виртуальной машины обходилось дороже, чем содержание реального физического компьютера с теми же характеристиками, другими словами, высокопроизводительный сервер приложений, способный поддерживать N виртуальных серверов меньшей мощности, обходился организации дороже, чем N физических серверов меньшей мощности. Но затем развитие информационных технологий (прежде всего, введение аппаратной поддержки гипервизоров в новых процессорах и чипсетах) привело к резкому снижению фактической стоимости виртуальных машин, в результате реальные компьютеры стали вытесняться виртуальными машинами, физически размещаемыми на высокопроизводительных серверах приложений. Важно отметить, что речь здесь идет не только и не столько о цене оборудования, сколько о значительном снижении эксплуатационных затрат. Так, в некоторых странах важнейшим преимуществом виртуализации серверов считается снижение потребления дорогостоящей электроэнергии.

Если два или более корпоративных серверов представляют собой виртуальные машины, работающие на одном физическом сервере, аппаратные ресурсы сервера распределяются между параллельно выполняющимися машинами-задачами более эффективно. Пусть, например, в некоторый момент времени почтовый сервер организации простаивает, а веб-сервер испытывает большой наплыв посетителей. Если каждый из этих серверов физически размещен на своем отдельном компьютере, то веб-сервер в такой ситуации будет работать с перегрузкой, в результате чего запросы клиентов к

нему будут выполняться с большими задержками. Если же оба сервера являются виртуальными машинами, работающими на одном и том же физическом компьютере, этот компьютер сможет динамически перераспределять свои аппаратные ресурсы, отдавая большую часть своего процессорного времени, оперативной памяти и пропускной способности сетевых интерфейсов тому виртуальному серверу, которому эти ресурсы нужнее в данный момент. Особенно большой выигрыш производительности достигается, когда на одном физическом сервере размещается несколько виртуальных серверов, активно взаимодействующих между собой по локальной сети. В таких случаях виртуализации могут подвергаться целые сегменты корпоративной сети, что не только многократно ускоряет обмен данными, но и приводит к заметному сокращению количества, ассортимента и суммарной стоимости сетевого оборудования, эксплуатируемого в организации.

В случае серьезного аппаратного сбоя на физическом сервере, обслуживающем несколько виртуальных машин, перенос этих машин на другой, исправный сервер, как правило, сводится к простому копированию файлов и последующему запуску скопированных виртуальных машин на новом сервере. Это позволяет сокращать время простоя системы при аппаратных сбоях, упрощает обновление аппаратного обеспечения корпоративных серверов. Создание резервных копий информации и восстановление с них информации для виртуальных машин реализуется намного проще, чем для реальных физических компьютеров, особенно если при этом применяется специально разработанное для таких случаев программное обеспечение.

На первый взгляд, виртуализация операционных систем может показаться новой революционной технологией, но на самом деле в ней нет ничего принципиально нового, это просто дальнейшее развитие концепций, характеризовавших развитие операционных систем на протяжении последних десятилетий. Первые виртуальные машины были реализованы еще в операционной системе IBM VM, разработанной для мейнфрейма IBM/370 (в России клон этого компьютера серийно производился под названием ЕС-1045). Фактически, IBM VM представляла собой не полноценную операционную систему, способную поддерживать функционирование собственных приложений, а специализированный гипервизор виртуальных машин, подобный современному VMWare ESX.

Начиная с первой половины 1990-х годов, в операционных системах персональных компьютеров практически не используется прямое обращение процессоров к оперативной памяти по явно указы-

ваемым физическим адресам. Вместо этого популярные системы машинных команд используют сложные алгоритмы многоуровневой трансляции адресов, по сути виртуализирующие оперативную память компьютера для всех видов прикладного и системного программного обеспечения за исключением небольшого фрагмента ядра операционной системы, выступающего в роли гипервизора оперативной памяти. Аналогично, драйверы, являющиеся обязательными посредниками при доступе прикладного программного обеспечения к аппаратным устройствам компьютера, могут рассматриваться как гипервизоры, виртуализирующие аппаратное обеспечение компьютера. Часто такая виртуализация оказывается настолько удобной, что многие начинают воспринимать виртуальное отображение того или иного аппаратного устройства как единственно верную объективную реальность. Например, мало кто знает, что PS/2-клавиатура и PS/2-мышь с точки зрения чипсета компьютера управляются одним и тем же контроллером и фактически представляют собой единое аппаратное устройство. Прикладные и системные программы, взаимодействующие с клавиатурой и мышью, реально взаимодействуют с драйверами логической клавиатуры и логической мыши, создающими для единого физического устройства «PS/2-порт» два виртуальных образа: «клавиатура» и «мышь», и тем самым виртуализирующими данное устройство.

Таким образом, полная виртуализация одной операционной системы в другой операционной системе является всего лишь расширением множества «малых виртуализаций», буквально пронизывающих современные информационные технологии. При этом виртуализация операционных систем не является конечной точкой на пути виртуализации, облачные технологии обработки данных могут рассматриваться как еще более далекое развитие данной концепции. Заметим, впрочем, что рассмотрение вопросов безопасности при облачной обработке данных выходит за рамки данного учебного пособия.

Распространение виртуализации породило целый ряд новых проблем в деле обеспечения безопасности операционных систем. Наиболее серьезной из них является проблема «погружения операционной системы в матрицу», когда нарушитель несанкционированно устанавливает на атакуемом компьютере программу-гипервизор, превращающую операционную систему реального компьютера в операционную систему виртуальной машины, работающей под управлением данного гипервизора. В этом случае несанкционированно установленный гипервизор получает полный контроль над всеми инфор-

мационными потоками внутри атакованной системы. Обнаружение такого гипревизора «изнутри матрицы» возможно только по ненадежным косвенным признакам. При этом код гипервизора может размещаться не только на жестком диске компьютера, но и в долговременной памяти одной или нескольких микросхем материнской платы. Известны опытные образцы таких гипервизоров, и есть основания полагать, что существуют и «боевые» версии программных закладок в флэш-память микросхем компьютера (в обиходе эту память часто называют BIOS, хотя базовая система ввода-вывода, как правило, занимает в ней ничтожную долю объема). Многим памятна история с так называемыми китайскими закладками [20] в южном мосту одного семейства материнских плат, производимых компанией Intel.

Разместить скрытый гипервизор в долговременной памяти микросхемы обычно возможно только в заводских условиях при активном участии разработчика этой микросхемы. Но программно реализованный гипервизор может быть внедрен в операционную систему с использованием тех же схем, по которой внедряются другие программные закладки режима ядра. Самая знаменитая из таких закладок-гипервизовров, известная под названием Blue Pill (голубая пилюля, очевидная аллюзия на фильм «Матрица»), была разработана Йоанной Рутковской и Александром Терешкиным и впервые продемонстрирована в 2006 году. Первоначально закладка работала только на процессорах AMD под операционной системой Windows Vista (возможно, правильнее говорить «над», а не «под») и отличалась от классических руткитов только принципом реализации, но не поддерживаемой функциональностью. Версия Blue Pill, работающая на процессорах Intel, появилась позже.

Существует ряд методов, позволяющих с некоторой долей уверенности определить, работает ли операционная система «на голом железе» или внутри виртуальной машины. Большинство этих методов основываются на том, что работа гипервизора несколько снижает быстродействие аппаратного обеспечения. Скрыть этот факт средствами гипервизора невозможно. Даже если гипервизор будет манипулировать скоростью хода системного таймера, то обращение к другому таймеру, внешнему по отношению к проверяемой системе, позволит обнаружить замедление хода времени внутри системы и тем самым демаскировать гипервизор.

В целом обнаружение руткитов, подобных Blue Pill, является намного более сложной задачей, чем обнаружение обычных руткитов. Подобная закладка может быть выявлена только лишь путем

анализа очень тонких нюансов работы компьютерной системы (точные времена выполнения определенных машинных команд, точные количества экземпляров определенных системных структур в физической памяти компьютера и т. п.). При этом разработчик алгоритма выявления закладки вынужден полагаться на тонкие недокументированные особенности конкретной версии защищаемой системы, которые могут быть изменены ее разработчиками в любой момент. Стоит отметить, что, вопреки распространенному мнению, детектор виртуализации Red Pill, разработанный Йоанной Рутковской, не обнаруживает факта собственного выполнения в виртуальной среде Blue Pill.

По состоянию на сегодняшний день несанкционированно устанавливаемые гипервизоры еще не получили широкого распространения. В основном это связано с техническими сложностями разработки и отладки данного программного обеспечения, а также с необходимостью достаточно жесткой его привязки к индивидуальным особенностям аппаратного обеспечения, установленного в атакуемой системе.

Наличие в защищаемой сети большого количества виртуальных машин предъявляет повышенные требования к качеству управления безопасностью корпоративной сети. В первую очередь речь здесь идет о повышенных требованиях к безопасности серверов приложений, на которых разворачиваются виртуальные машины. Если корпоративная сеть построена на обычных физических серверах, то взлом любого из них возможен только тогда, когда атака нарушителя направлена на данный конкретный сервер. Но в сети, построенной на небольшом количестве высокопроизводительных серверов, захват контроля над любым из них может предоставить нарушителю одномоментный несанкционированный доступ к десяткам критически важных виртуальных машин, что может иметь катастрофические последствия. При неблагоприятном стечении обстоятельств одна-единственная критическая уязвимость в гипервизоре сервера виртуальных машин может отдать под контроль нарушителя большой сегмент корпоративной сети или даже всю корпоративную сеть.

Другой серьезной проблемой безопасности является проблема неуправляемых виртуальных машин. В любой достаточно большой корпоративной сети рано или поздно появляются всеми забытые серверы, не удостаивающиеся внимания администраторов на протяжении многих месяцев или даже лет. Программное обеспечение таких серверов, как правило, никогда не обновляется, а политика безопасности, реализуемая операционной системой, может сильно расходи-

ться с корпоративными стандартами. В целом такие компьютеры обычно защищены хуже других, и часто становятся первой жертвой вирусной эпидемии или хакерской атаки.

Если корпоративная сеть построена традиционным образом, с использованием обычных физических компьютеров, вероятность появления в ней неуправляемого сервера сравнительно невелика. Но если в корпоративной сети одновременно работают сотни и тысячи виртуальных машин, появление среди них нескольких неуправляемых вполне вероятно. Кроме того, на неуправляемых виртуальных машинах гораздо чаще, чем на физических компьютерах, встречаются совершенно незащищенные операционные системы с зияющими брешами в политике безопасности. Чаще всего такие машины возникают из «несанкционированно оживленных» резервных копий и снапшотов других виртуальных машин. Иногда администратор, добавив в корпоративную сеть временный сервер для экспериментов, забывает его выключить, когда необходимость в таком сервере отпадает. Иногда неуправляемые машины возникают из-за «несанкционированного клонирования» виртуальной машины, когда она копируется с неисправного сервера на исправный, затем работоспособность неисправного сервера восстанавливается и его виртуальные машины автоматически перезапускаются. В результате в сети появляется второй клон виртуальной машины, на который никто не обращает внимания до тех пор, пока он не начнет упоминаться в отчетах корпоративной системы обнаружения вторжений. Кроме того, в корпоративной сети, включающей в себя много виртуальных машин, заметно сложнее решается задача своевременной установки обновлений программного обеспечения на все компьютеры защищаемой сети.

Виртуальная машина, находящаяся в выключенном состоянии, практически неспособна противодействовать несанкционированному доступу к своим ресурсам. Если, например, нарушитель начнет несанкционированно копировать файлы виртуальной машины за пределы защищаемой сети, операционная система виртуальной машины никак не сможет противодействовать этому. В сети, построенной традиционным образом, угроза кражи нарушителем физического сервера гораздо менее актуальна, чем угроза несанкционированного копирования виртуальной машины, а угроза несанкционированного копирования физического сервера практически нереализуема.

Файлы виртуальных машин могут копироваться администраторами с одних серверов приложений на другие, иногда эти действия

остаются незамеченными администраторами безопасности. В результате администратор безопасности не всегда адекватно представляет себе топологию защищаемой сети, порядок маршрутизации сетевого трафика внутри нее. Это предъявляет повышенные требования к политикам защиты сетевого трафика от перехвата, навязывания и блокирования. Дополнительные проблемы создает невозможность применения некоторых реализаций пакетных фильтров и систем обнаружения вторжений к виртуальному сетевому трафику, не выходящему за пределы единственного физического сервера. Подключение сенсоров систем обнаружения вторжений к физическому трафику, циркулирующему между хост-серверами, как правило, малоэффективно, поскольку в этом случае трафик выглядит как обезличенный поток разнородных сетевых пакетов, в котором крайне затруднительно определить принадлежность того или иного пакета к конкретной виртуальной машине и, тем более, к конкретной сетевой сессии.

При внедрении вредоносного программного обеспечения в корпоративную сеть, построенную на виртуальных машинах, программная закладка может интегрироваться внутрь особой виртуальной машины, специально предназначенной для того, чтобы быть контейнером для программной закладки или для временного хранения конфиденциальной информации, подготовленной к отправке нарушителю. Если виртуальных машин в сети много, выявить среди них одну вредоносную может быть непростой задачей.

Несмотря на все вышеизложенное, было бы неверно рассматривать виртуализацию как фактор, влияющий на безопасность операционных систем однозначно негативным образом. Существует целый ряд средств, методов и технологий обеспечения информационной безопасности, реализация которых без применения виртуализации операционных систем была бы серьезно затруднена или вообще невозможна.

При реализации антивирусной защиты операционных систем серьезной проблемой является то, что в случае успешного внедрения руткита в защищаемую систему его вредоносный код выполняется на том же самом уровне привилегированности, что и код ядра операционной системы, при этом данный уровень является наиболее привилегированным из всех поддерживаемых операционной системой. Но включение на процессоре режима аппаратной виртуализации дает возможность включать в операционную систему еще более привилегированный программный код. Традиционно уровни привилегированности процессоров Intel и AMD обозначаются целыми

числами от 0 до 3, при этом нулевому уровню, на котором должно выполняться ядро операционной системы, соответствует наивысший уровень привилегированности, предоставляющий программному коду наибольшие возможности. Программный код, выполняющийся на этом уровне, может, в частности, виртуализировать среду выполнения программного кода, выполняющегося на 1-3 уровнях привилегированности, и выступать для него в роли гипервизора. Режим гипервизора (VT-x для процессоров Intel и AMD-V для процессоров AMD) позволяет программному коду, выполняющемуся в этом режиме, выступать в роли гипервизора для программного кода нулевого кольца. Из-за этого режим гипервизора часто называют минус первым кольцом защиты. Режим системного управления (System Management Mode, SMM), в котором работает Blue Pill, фактически является минус вторым кольцом защиты, а режим vPro/AMT — минус третьим кольцом защиты. Заметим, что программный код, работающий в режиме SMM, должен размещаться в особой области оперативной памяти SMRAM, которая в подавляющем большинстве чипсетов, изготовленных после 2006 года, недоступна постороннему коду. Руткит Blue Pill при проникновении в SMRAM фактически эксплуатировал аппаратную уязвимость, повсеместно устраненную после того, как этот руткит стал широко известен. Начиная с 2009 года, достоверной информации об уязвимостях, позволяющих несанкционированно внедрять в SMRAM вредоносный код, в открытых источниках не обнаруживается.

Поддержка дополнительных колец защиты позволяет разработчику защищенной операционной системы в полной мере реализовать концепцию доверенного микроядра, когда суммарный объем доверенного кода, которому присваивается наивысший уровень привилегированности, делается минимальным. В такой операционной системе большая часть кода ядра, все модули расширения ядра и все драйверы устройств выполняются на уровне привилегированности, строго более низком, чем у доверенной части ядра. Это серьезно затрудняет искажение информационных потоков ядра вредоносным программным кодом, например, с целью сокрытия своего присутствия в системе. Впрочем, построение ядра операционной системы на базе концепции доверенного микроядра является весьма сложной задачей, имеющей на данным момент удовлетворительные решения только для простейших вырожденных случаев (встроенные операционные системы сетевых коммутаторов, аппаратных шифраторов и т. п.).

Гораздо более простым с технической точки зрения является ре-

шение, при котором в дополнительные кольца защиты помещается сравнительно компактный программный код, не реализующий основные функции подсистемы безопасности операционной системы, а всего лишь контролирующий корректность их функционирования. В этом случае ядро операционной системы по-прежнему работает в нулевом кольце защиты и по-прежнему является уязвимым для вредоносных драйвером и модулей расширения ядра. Но если вредоносный код начнет искажать информационные потоки ядра, это искажение, скорее всего, будет замечено контролирующим программным кодом, неуязвимым для программных закладок. Чуть более сложным для программной реализации является решение, когда программный код, выполняющийся в дополнительных кольцах защиты, автоматически получает управление при наступлении определенного события (например, при появлении в системе нового программного файла) и выполняет некоторые действия, связанные с наступившим событием (например, проверяет новый файл на наличие вредоносного кода). При этом программный код обработчиков таких событий может размещаться не непосредственно в гипервизоре, а внутри вспомогательной виртуальной машины, специально предназначенной для решения задач, связанных с безопасностью, и недоступной пользователям напрямую. Существует несколько программных продуктов (наиболее известны VMware VShield EndPoint и TrendMicro Deep Security), позволяющих размещать на сервере виртуальных машин особую виртуальную машину, внутри которой находится антивирусный комплекс, защищающий другие виртуальные машины, расположенные на том же сервере.

Рассматриваемая концепция позволяет включать в состав подсистемы безопасности сервера виртуальных машин готовые программные решения, изначально предназначенные для обеспечения безопасности операционных систем физических компьютеров. Фактически, здесь происходит виртуализация функций обеспечения безопасности виртуальных машин.

Серьезным препятствием на пути практической реализации данного подхода является то, что гипервизор обычно не имеет полной информации о высокоуровневом контексте действий, выполняемых в виртуальных машинах, работающих под его управлением. Гипервизор знает во всех подробностях, какие машинные команды выполняются на каких виртуальных процессорах, к каким областям виртуальной памяти и к каким виртуальным устройствам делаются обращения, но с какой целью и в каком контексте выполняются все эти операции — эта информация гипервизору, как правило, недос-

тупна. Некоторые атаки на виртуальные машины вообще не могут быть обнаружены гипервизором, не обладающим полноценным искусственным интеллектом. Поэтому перенос функций безопасности из ядра операционной системы в гипервизор нельзя рассматривать как панацею. Это решение повышает надежность многих функций подсистемы безопасности операционной системы, делает ее более устойчивой к применению вредоносным кодом стелс-технологий, но многие опасные действия из гипервизора попросту не видны. В целом функции безопасности, реализованные в гипервизоре, должны рассматриваться только как дополнение, но не как замена функций безопасности, реализованных в ядре операционной системы.

Кроме того, не вполне очевидным является то, каким образом виртуальная машина, обычно даже не знающая, что она работает под гипервизором, будет передавать гипервизору оповещения о потенциально опасных событиях, решения по которым должен принимать гипервизор. Существуют два основных подхода к решению этой задачи:

1) повышение интеллектуальности гипервизора, внесение в него элементов эвристического поведения, характерного для систем обнаружения вторжений. В этом случае гипервизор рассматривает элементарные события функционирования виртуальной машины не каждое по отдельности, а в совокупности, что позволяет ему обнаруживать попытки нарушения безопасности на основе сигнатурных или эвристических правил. Фактически, такой гипервизор эквивалентен системе обнаружения вторжений, перехватывающей все информационные потоки, связывающие защищаемую операционную систему с окружающим миром. Серьезным недостатком данного подхода является то, что против гипервизора, построенного на его основе, хорошо работают средства и методы, традиционно применяемые программными закладками для противодействия системам обнаружения вторжений. Кроме того, некоторые функции безопасности, реализуемые внутри защищаемой виртуальной машины (например, прозрачное шифрование тех или иных информационных потоков) могут серьезно затруднять работу такого гипервизора [16];

2) установка внутри защищаемой виртуальной машины специального программного агента, явно оповещающего гипервизор о необходимости выполнить ту или иную проверку безопасности. Данный подход лишен недостатков предыдущего, но имеет очень серьезную слабость — агент оповещения гипервизора уязвим для воздействий вредоносного кода, захватившего контроль над виртуальной машиной.

Когда технологии переноса инструментов обеспечения безопасности в гипервизор только начинали входить в обиход, некоторые эксперты высказывали предположения, что многократные передачи управления из виртуальной машины в гипервизор и обратно будут существенно ухудшать производительность операционной системы. Однако практический опыт показывает, что потери производительности в этом случае не превышают 20 %, а некоторые антивирусные комплексы на уровне гипервизора работают даже быстрее, чем внутри защищаемой виртуальной машины [13]. А если на одном сервере одновременно работает порядка ста виртуальных машин, затраты аппаратных ресурсов на антивирусное сканирование могут сокращаться в десятки раз [9].

Еше более радикальное улучшение результатов может быть достигнуто при переносе в гипервизор антивирусных мониторов, обнаруживающих вредоносный код в динамике путем отслеживания подозрительных информационных потоков. Выполняясь на уровне гипервизора, такой монитор может обнаруживать некоторые события, характерные для активизации эксплойтов, обнаружить которые изнутри защищаемой системы затруднительно. К таким событиям относятся, например, следующие:

- установка атрибута «разрешается выполнять программный код» на логическую страницу памяти, в которую ранее были считаны данные с внешнего носителя информации или сетевого устройства (за исключением легальной процедуры загрузки программного файла для исполнения);
- выполнение в режиме ядра программного кода, расположенного в младшей части адресного пространства, доступной в режиме пользователя;
- выполнение большого количества подряд идущих команд nop;
- выполнение программного кода, размещенного внутри логической страницы, внутри которой также находится вершина стека.

На протяжении всей эволюции операционных систем характерной тенденцией было стремление к все более полной изоляции выполняющихся в системе процессов друг от друга. Виртуализация позволяет довести эту тенденцию до логического завершения — выполнять каждый процесс в собственной виртуальной машине, имеющей не только свое собственное виртуальное адресное пространство, но и полный набор виртуальных внешних устройств. Взаимодействие процессов в такой системе может быть реализовано посредством интерфейсов, традиционно применяемых для организации взаимодействия процессов, выполняющихся на разных компьютерах:

именованных каналов, сокетов и т. п. Такая схема построения операционной системы затрудняет функционирование в ней программных закладок, но рассматривать ее как панацею не следует. Все трудности построения программной закладки, способной эффективно функционировать в такой среде, вполне преодолимы при наличии у разработчиков закладки соответствующих ресурсов и мотивации. Тем не менее, инкапсуляция недоверенных процессов в виртуальные машины является потенциально перспективным направлением развития тех операционных систем, к которым предъявляются повышенные требованиями в отношении безопасности хранимых и обрабатываемых данных. Уже существует несколько прототипов практической реализации данной концепции, одним из наиболее интересных является операционная система Qubes [26], основанная на ядрах Linux, работающих под управлением гипервизора Xen. В состав Qubes входит специализированный X-сервер, создающий у пользователя полную иллюзию того, что он работает с единственным экземпляром операционной системы Linux.

Вопросы для самопроверки

1. Почему виртуализация операционных систем получила в последние годы такое широкое распространение?

2. Как виртуализация операционных систем позволяет оптимизировать распределение нагрузки на используемое аппаратное обеспечение?

3. В какой операционной системе были разработаны первые виртуальные машины?

4. В чем заключается угроза несанкционированной установки гипервизора на атакуемом компьютере?

5. Какие программные закладки, реализующие функциональность гипервизора, вы знаете?

6. Какие дополнительные требования к качеству управления безопасностью предъявляются при наличии в защищаемой сети большого количества виртуальных машин?

7. В чем заключается угроза безопасности корпоративной сети от неуправляемых виртуальных машин?

8. В чем заключается угроза бесконтрольного копирования администраторами файлов выключенных виртуальных машин?

9. Какие дополнительные средства, методы и технологии обеспечения информационной безопасности могут реализовываться с использованием виртуализации операционных систем?

10. Как поддержка дополнительных колец защиты влияет на возможности реализации в операционной системе доверенного микроядра?

11. Как можно использовать для защиты виртуальных машин служебные виртуальные машины, работающие на том же самом сервере приложений, что и защищаемые виртуальные машины?

12. В чем заключается основная проблема, затрудняющая перенос в гипервизор части защитных функций операционной системы, предназначенной для

обслуживания виртуальной машины? Какие два основных подхода к решению этой проблемы применяются на практике?

13. Какие дополнительные возможности предоставляет антивирусному монитору полный или частичный перенос его функциональности в гипервизор?

14. Каких преимуществ позволяет добиться инкапсуляция недоверенных процессов операционной системы в отдельные виртуальные машины?

Приложение

Методические рекомендации по организации изучения дисциплины «Защита в операционных системах»

Анализ требований ФГОС ВПО

Рассмотрим требования ФГОС ВПО третьего поколения по направлению подготовки «Информационная безопасность» (квалификации бакалавр, специалист и магистр), на выполнение которых ориентировано изучение материала данного учебного пособия.

Наиболее глубокие знания вопросов защиты в операционных системах требуются при обучении по специальности **10.05.01 — «Компьютерная безопасность»** (квалификация — специалист). Закончив обучение по данной специальности, выпускник должен обладать, в том числе, следующими компетенциями:

- способностью применять современные методы и средства исследований для обеспечения информационной безопасности компьютерных систем;
- способностью проводить обоснование и выбор рационального решения по уровню обеспечения информационной безопасности компьютерной системы с учетом заданных требований;
- способностью проводить анализ проектных решений по обеспечению безопасности компьютерных систем;
- способностью участвовать в разработке системы защиты информации предприятия (ведомства, подразделения) и подсистемы информационной безопасности компьютерной системы;
- способностью к проведению экспериментального исследования компьютерных систем с целью выявления уязвимостей;
- способностью разрабатывать предложения по совершенствованию системы управления информационной безопасностью компьютерной системы;
- способностью производить установку, тестирование программного обеспечения и программно-аппаратных средств по обеспечению информационной безопасности компьютерных систем;

- способностью принимать участие в эксплуатации программного обеспечения и программно-аппаратных средств обеспечения информационной безопасности компьютерных систем.

Для реализации перечисленных компетенций в части, касающейся операционных систем, в результате изучения дисциплины «Защита в операционных системах» студент должен:

- знать защитные механизмы и средства обеспечения безопасности операционных систем;
- уметь формулировать и настраивать политику безопасности основных операционных систем, а также локальных компьютерных сетей, построенных на их основе;
- владеть навыками разработки программных модулей, реализующих задачи, связанные с обеспечением безопасности операционных систем распространенных семейств.

Закончив обучение по специальности **10.05.02 — «Информационная безопасность телекоммуникационных систем»**, выпускник должен обладать в том числе следующими компетенциями:

- способностью проводить анализ проектных решений по обеспечению безопасности телекоммуникационных систем;
- способностью оценивать эффективность систем защиты информации в телекоммуникационных системах;
- способностью осуществлять аудит уровня защищенности и аттестацию телекоммуникационных систем в соответствии с существующими нормами;
- способностью принимать участие в эксплуатации системы обеспечения информационной безопасности телекоммуникационных систем;
- способностью проводить мониторинг, техническую диагностику средств защиты, оценку эффективности информационной безопасности защищенных телекоммуникационных систем;
- способностью выявлять и прогнозировать угрозы информационной безопасности телекоммуникационных систем и разрабатывать меры противодействия.

Для их реализации в части, касающейся операционных систем, студент должен в результате изучения дисциплин профессионального цикла:

- знать программно-аппаратные средства обеспечения информационной безопасности в типовых операционных системах;
- уметь развертывать, конфигурировать, администрировать операционные системы вычислительных систем;

- уметь обеспечивать защиту от разрушающих программных воздействий.

Закончив обучение по специальности **10.05.03 — «Информационная безопасность автоматизированных систем»**, выпускник должен обладать в том числе следующими компетенциями:

- способностью проводить анализ защищенности автоматизированных систем;
- способностью разрабатывать политики информационной безопасности автоматизированных систем;
- способностью участвовать в проектировании средств защиты информации и средств контроля защищенности автоматизированной системы;
- способностью организовать эксплуатацию автоматизированной системы с учетом требований информационной безопасности;
- способностью обеспечить эффективное применение средств защиты информационно-технологических ресурсов автоматизированной системы;
- способностью администрировать подсистему информационной безопасности автоматизированной системы;
- способностью управлять информационной безопасностью автоматизированной системы.

Для их реализации в части, касающейся операционных систем, студент должен в результате изучения дисциплин профессионального цикла:

- знать принципы построения и функционирования, примеры реализаций современных операционных систем;
- знать программно-аппаратные средства обеспечения информационной безопасности в типовых операционных систем;
- владеть навыками установки и настройки операционных систем семейств Windows и UNIX с учетом требований по обеспечению информационной безопасности.

Закончив обучение по специальности **10.05.04 — «Информационно-аналитические системы безопасности»**, выпускник должен обладать в том числе следующими компетенциями:

- способностью применять основные защитные механизмы и средства обеспечения безопасности операционных систем;
- способностью выявлять основные угрозы безопасности информации, строить и исследовать модели нарушителя в компьютерных системах;
- способностью разрабатывать защитные механизмы и средства обеспечения информационной безопасности.

Для их реализации студент должен в результате изучения дисциплин профессионального цикла:

- знать принципы построения современных операционных систем и особенности их применения;
- знать основные виды и угрозы безопасности операционных систем;
- знать защитные механизмы и средства обеспечения безопасности операционных систем;
- владеть навыками разработки программных модулей, реализующих задачи, связанные с обеспечением безопасности операционных систем распространенных семейств.

Студенты, осваивающие квалификацию **бакалавр** направления подготовки **10.03.01 — «Информационная безопасность»**, должны реализовать в ходе обучения следующие компетенции:

- способность формировать комплекс мер (правила, процедуры, практические приемы и пр.) для управления информационной безопасностью;
- способность участвовать в работах по реализации политики информационной безопасности;
- способен выполнять работу по самостоятельному построению алгоритмов, проведению их анализа и реализации в современных программных комплексах;
- способен проводить экспериментальное исследование компьютерных систем с целью выявления уязвимостей.

При этом студенты, обучающиеся по профилю подготовки «Безопасность автоматизированных систем», должны, кроме того, реализовать компетенцию:

- способность выполнять комплекс задач администрирования подсистем информационной безопасности операционных систем, систем управления базами данных, компьютерных сетей.

Для реализации указанных компетенций студент (квалификации бакалавр) в результате изучения дисциплин профессионального цикла должен:

- уметь формулировать и настраивать политику безопасности распространенных операционных систем, а также локальных вычислительных сетей, построенных на их основе;
- владеть методами и средствами выявления угроз безопасности автоматизированным системам.

Обучающийся по уровню подготовки **магистр** должен реализовать, в том числе, следующие компетенции:

- способность анализировать фундаментальные и прикладные проблемы информационной безопасности в условиях становления современного информационного общества;
- способность анализировать угрозы информационной безопасности объектов и разрабатывать методы противодействия им.

Для реализации указанных компетенций студент должен:

- знать основные принципы организации технического, программного и информационного обеспечения защищенных информационных систем;
- уметь обосновывать принципы организации технического, программного и информационного обеспечения информационной безопасности.

Таким образом, изучение на основе предлагаемого пособия вопросов защиты информации в операционных системах способствует освоению знаний и умений, направленных на реализацию компетенций в соответствии с ФГОС ВПО третьего поколения по направлению подготовки «Информационная безопасность» (квалификации бакалавр, специалист и магистр). Закрепление теоретических знаний о средствах и методах заищты информации в операционных системах, практических навыков их использования и окончательное формирование компетенций предусматривается в процессе производственных и преддипломных практик, а также при выполнении дипломной работы.

Организация изучения дисциплины «Защита в операционных системах»

Изучение дисциплины «Защита в операционных системах» основано на дисциплинах «Информатика», «Аппаратные средства вычислительной техники», «Операционные системы».

Знания и практические навыки, полученные при изучении дисциплины «Защита в операционных системах», обеспечивают освоение дисциплин «Модели безопасности компьютерных систем», «Основы построения защищенных сетей», а также используются обучаемыми при разработке курсовых и дипломных работ.

Преподавателю важно учесть, что материал пособия имеет практическую направленность, при этом набольшее значение для его усвоения имеют практические занятия, на которых студенты непосредственно усваивают преподаваемые умения и навыки. Особенно полезны в этом плане лабораторные работы, при выполнении которых студенты вырабатывают практические навыки управления безопасностью современных операционных систем.

В теме «Понятие защищенной операционной системы» студенты должны твердо уяснить наиболее общие положения, на которых основывается весь дальнейший материал. Студенты должны четко понимать, что защищенность операционной системы находится в обратной зависимости с ее эксплуатационными качествами, что адекватная политика безопасности не всегда обеспечивает наиболее высокий уровень защищенности, что процесс формирования и сопровождения адекватной политики безопасности является весьма растянутым во времени и никогда полностью не завершается.

Тема «Управление доступом» является наиболее сложной в данном пособии. Это связано с тем, что здесь дается большой объем специальной терминологии (субъект, объект, метод, право, привилегия), рассматриваются сложные структуры данных (дескрипторы защиты, маркеры доступа), излагаются сложные алгоритмы (проверки прав доступа, динамического изменения полномочий субъекта, наследования атрибутов защиты, построения ограниченных маркеров доступа). Излагая материал данной темы, преподаватель должен постоянно отслеживать ответную реакцию аудитории и, при необходимости, излагать те или иные фрагменты курса менее подробно, а возможно, даже пропускать фрагменты, наиболее сложные для понимания. Особое внимание следует уделить освоению студентами на практических занятиях и лабораторных работах практических навыков управления разграничением доступа. Опыт преподавания данной дисциплины показывает, что после двух-трех практических занятий в компьютерном классе у большинства студентов наступает «просветление» и те концепции управления доступом, которые раньше казались им невероятно сложными, становятся простыми, понятными и «естественными». По окончании данной темы целесообразно провести контрольную работу.

Тема «Идентификация и аутентификация», напротив, не вызывает у студентов серьезных затруднений. Единственным проблемным местом является система библиотек PAM операционной системы Linux, устроенная весьма нетривиально и предполагающая довольно сложную процедуру настройки администратором. Если интеллектуальный уровень аудитории невысок, соответствующий материал можно опустить.

Тема «Аудит» также не вызывает у студентов серьезных затруднений. Для выработки понимания концепций аудита и навыков работы с аудитом вполне достаточно одной лекции и двух практических занятий, отведенных на данную тему тематическим планом.

В теме «Домены Windows» следует обратить особое внимание на

получение слушателями практических навыков администрирования доменов Windows и управления политикой безопасности в доменах Windows. При изложении материала данной темы целесообразно сконцентрироваться на рассмотрении типовых процедур администрирования, не вдаваясь в излишние подробности доменной архитектуры сетей Windows.

В последних двух главах данного учебного пособия описываются сравнительно новые аспекты обеспечения информационной безопасности операционных систем. Традиция преподавания этих аспектов пока еще не сформировалась, преподавателю предлагается разработать соответствующую методику самостоятельно. Если при изучении данных тем у студентов возникают затруднения, соответствующий материал можно опустить.

В соответствии со спецификой вуза, со специальностью и квалификацией, реализуемыми образовательной программой, в процессе изучения защиты в операционных системах методически целесообразно из каждого раздела дисциплины выделить наиболее важные подразделы и акцентировать на них внимание. При этом возможно сокращение времени на анализ рассмотренных в пособии вопросов и использование освободившегося времени на дополнительное изучение других вопросов защиты в операционных системах.

Литература

1. Безопасность информационных технологий. Операционные системы. Базовый профиль защиты. — Центр безопасности информации, 2002.

2. Бетелин В.Б., Галатенко В.А., Кобзарь М.Т., Сидак А.А., Трифаленков И.А. Профили защиты на основе «Общих критериев». Аналитический обзор. — www.citforum.urc.ac.ru/security/criteria/

3. Брэгг Р. Безопасность сетей на основе Microsoft Windows Server 2003. — М.: Русская редакция, 2006. — 672 с.

4. Девянин П.Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учебное пособие для вузов. — М.: Горячая линия — Телеком, 2011. — 320 с.

5. Кнут Д. Искусство программирования, т. 2. Получисленные алгоритмы: Учебное пособие. 3-е изд. — М.: Издательский дом «Вильямс», 2000. — 832 с.

6. Ледовской В. Виртуальным инфраструктурам — прогрессивная защита. — www.anti-malware.ru/analytics/Progressive_Defense_for_Virtual_Infrastructures

7. Майкл Х., Лебланк Д. Защищенный код для Windows Vista. — М.: Русская редакция, 2008. — 224 с.

8. Моримото Р., Гардиньер К., Ноэл М., Драуби О. Microsoft Windows Server 2003. Полное руководство. — М.: Вильямс, 2005. — 1312 с.

9. Прилепский А. Антивирусная защита в VMware View. — www.vsphere5.ru/doku.php?id=deep-security

10. Проскурин В.Г., Крутов С.В., Мацкевич И.В. Программно-аппаратные средства обеспечения информационной безопасности. Защита в операционных системах: Учеб. пособие для вузов. М.: «Радио и связь», 2000, 168 с.

11. Проскурин В.Г. Защита программ и данных: учеб. пособие для студ. учреждений высш. проф. образования. — 2-е изд. — М.: Издательский центр «Академия», 2012. — 208 с.

12. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows: Windows 2003 Server, Windows XP, Windows 2000. — СПб: Русская редакция, 2005. — 992 с.

13. Силаков Д.В. Использование аппаратной виртуализации в контексте информационной безопасности. — www.ispras.ru/ru/proceedings/docs/2011/20/isp_20_2011_25.pdf

14. Фленов М. Linux глазами хакера. — СПб: БХВ-Петербург, 2006. — 544 с.

15. Хорев П.Б. Методы и средства защиты информации в компьютерных системах: учеб. пособие для студ. высш. учеб. заведений. — М.: Издательский центр «Академия», 2006. — 256 с.

16. T. Garfinkel, M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. — www.suif.stanford.edu/papers/vmi-ndss03.pd

17. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам. Учебное пособие для вузов / А.А. Афанасьев, Л.Т. Веденьев, А.А. Воронцов и др.; Под ред. А.А. Шелупанова, С.Л. Груздева, Ю.С. Нахаева. — М.: Горячая линия — Телеком, 2009. — 522 с.

18. Безопасность Android под шквалом критики. — www.hi-tech.mail.ru/article/misc/bezopasnost_android_pod_shkvalom_kritiki.html

19. Более половины Android-устройств имеют серьезную брешь в безопасности. — www.androidstreet.net/2012/09/14/duo-security-report/

20. Китайские закладки: непридуманная история о виртуализации, безопасности и шпионах. — www.xakep.ru/post/58104/

21. Многоуровневая (сквозная) виртуализация с поддержкой на уровне ОС. Возможности применения в системах безопасности. — www.daily.sec.ru/publication.cfm?pid=36491

22. Эволюция угроз безопасности Android за последний год. — www.habrahabr.ru/post/130612/

23. Android & iOS: концепции распространения приложений и вопросы безопасности. — www.habrahabr.ru/company/drweb/blog/143971/

24. Android отметила 5-летие. — www.cybersecurity.ru/os/163693.html

25. iOS против Android в вопросах безопасности: Find My iPhone vs Cerberus. — www.iphones.ru/iNotes/230903

26. Qubes Architecture Specification. Version 0.3.2010. — www.qubes-os.org/files/doc/arch-spec-0.3.pdf

Оглавление