Аутентификация, авторизация и управление доступом в операционных системах Windows и Unix (полная версия)

Системы аутентификации, авторизации и управления доступом операционных систем

Система аутентификации пользователей является своего рода передовой линией фронта в борьбе за безопасность информации. Ошибки системы аутентификации исправить невозможно, какими бы совершенными не были остальные средства безопасности.

Существует две принципиально отличные схемы аутентификации, реализуемые операционными системами и специальными сетевыми службами. В одной из них, которую мы будем называть ***локальной системой аутентификации***, операционная система работает в пределах одного компьютера: она использует базу аутентификационных данных пользователей компьютера, на котором эта ОС установлена, и результаты аутентификации могут быть использованы только для доступа к ресурсам этого компьютера.

По другой схеме работает так называемая ***система аутентификации домена***: она базируется на центральной базе аутентификационных данных пользователей группы компьютеров (***домена аутентификации),*** хранящейся на одном из серверов сети, и результаты аутентификации используются для доступа к ресурсам данного домена.

Мы остановимся сейчас только на локальных системах аутентификации ОС, а к системам аутентификации домена вернемся в разделе «Централизованные системы аутентификации и авторизации».

Локальная система аутентификации ОС работает при логическом входе пользователя как *с терминала* компьютера, так и через *сеть*. Первый вариант называют ***интерактивным*** логическим входом, а второй – ***удаленным*** (*сетевым* или *неинтерактивным)* логическим входом.

Понятно, что при удаленном логическом входе риски безопасности выше, так как аутентификационные данные передаются через сеть – корпоративную или Интернет – и их легче перехватить. Перехват данных процедуры аутентификации представляет собой угрозу даже в случае строгой аутентификации, когда пароль не передается в открытом виде по сети или же не передается вовсе – при наличии большого массива аутентификационных данных, то есть данных перехватов большого количества процедур входа одного и того же пользователя, пароль может быть вычислен по имеющимся результатам его применения.

Локальные системы аутентификации поддерживают все распространённые методы аутентификации, описанные выше: на основе многоразовых и одноразовых паролей (аппаратных и программных), биометрических данных и цифровых сертификатов.

Основным методом аутентификации пользователей ОС является метод, основанный на использовании *многоразового пароля*. Практически все универсальные ОС, такие как MS Windows, Unix/Linux и MAC OS X, используют этот метод по умолчанию.

Одноразовые пароли, обеспечивающие более надёжную аутентификацию, чем многоразовые, чаще используются при удалённом логическом входе через соединения VPN с шифрованием информации, где передача аутентификационной информации идёт через Интернет и, следовательно, риск ее

перехвата и взлома особенно велик. Одноразовые пароли могут сочетаться с многоразовыми при двух-факторной аутентификации.

Аутентификация на основе сертификатов используется чаще всего для удаленно работающих пользователей, которые предъявляют сертификаты, выданные сервером сертификации организации, к которой принадлежит пользователь.

Аутентификация на основе биометрических данных штатными средствами универсальных ОС обычно не поддерживается, так как их повышенная надежность нужна только в особо защищённых системах и, кроме того, для поддержки биометрической аутентификации требуется приобрести и установить соответствующее специальное программное обеспечение и специальные устройства.

Необходимо отличать процедуру аутентификации пользователя операционной системы от *процедуры аутентификации пользователя серверной части некоторого приложения*. Многие серверные приложения имеют собственную систему аутентификации пользователей, никак не связанную с системой аутентификации ОС, под управлением которой они работают. Например, так работают многие реализации FTP-сервера, сервера баз данных.

Независимость системы аутентификации сервера приложений имеет как свои положительные, так и отрицательные стороны. Преимуществом здесь является разграничение по умолчанию пользователей ОС, которым потенциально может понадобиться доступ к любому ресурсу компьютера, и пользователей некоторого сервиса, которым нужен доступ только к ресурсам, относящихся к данному сервису, например только к файлам, хранящимся в корневом каталоге FTP-сервера. К недостаткам же можно отнести следующее:

- Поддержка двух или более (по числу сетевых приложений) баз данных пользователей может приводить к дополнительным сложностям для пользователей, например к необходимости запоминания двух различных имен и паролей для одного и того же пользователя, если он является пользователем ОС и пользователем сервиса; ошибкам администраторов ОС и сервиса из-за дублирования учетных записей и т.п.
- Низкая защищенность протокола аутентификации некоторых приложений. Классическим примером является сервер FTP, который использует протокол аутентификации с открытой передачей многоразового пароля по сети. Из-за этого сервис FTP не рекомендуется использовать при доступе к нему через Интернет, заменяя его более защищенным сервисом scp или SFTP.

Что касается систем управления доступом в универсальных ОС, то там доминирует дискреционная модель управления доступа, согласно которой владелец ресурса (пользователь, который его создал или которому передано владение) самостоятельно определяет, кто имеет доступ к этому ресурсу и какие операции с ним он может выполнять. Практически все популярные сегодня семейства универсальных ОС – Unix/Linux/CentOS/Ubuntu, Mac OS X, MS Windows – используют дискреционную модель доступа как основную. Мандатная модель – это особенность специализированных ОС, рассчитанных на применение в среде с повышенными требованиями к безопасности. Тем не менее существует набор модулей ядра Linux под названием SELinux (Security Enhanced Linux), который реализует многие свойства мандатной модели в среде Linux.

Модель ролевого управления доступом используется в универсальных ОС частично, в виде механизма встроенных групп с предопределенными правами, сосуществуя с моделью дискреционного доступа для индивидуальных пользователей и групп.

В далее мы ограничимся рассмотрением особенностей реализации систем аутентификации и управления доступом в ОС семейства Unix и Windows.

Аутентификация пользователей ОС Unix

Семейство ОС Unix выросло из многотерминальной многопользовательской операционной системы для мини-компьютера PDP-11, и ее традиционный интерактивный вход с алфавитно-цифрового терминала (или программы, его эмулирующей) с аутентификацией по паролю по-прежнему является основным способом логического входа в систему. Утилита ***login*** поддерживает процедуру *интерактивного входа* пользователей в текстовом[1] режиме. Эта процедура включает локальную аутентификацию пользователей на основе их учетных данных, хранящихся в особом файле `на диске`.

Удаленный вход пользователей Unix во времена сравнительно безопасного Интернета выполнялся с помощью протокола ***telnet***, который является протоколом эмуляции текстового терминала поверх транспортных средств стека TCP/IP. Протокол `telnet` передает символ, набираемые пользователем, и ответы ОС в открытом виде, поэтому перехватить пароль, набираемый клиентом `telnet`, не составляет труда. Из-за незащищенности `telnet` его строго не рекомендуется использовать для аутентификации пользователей Unix через Интернет; да и крупная корпоративная сеть также может представлять опасность для такой передачи.

Основным современным средством ОС Unix для *удаленного входа* является многофункциональный пакет программ ***SSH (Secure Shell)***, который поддерживает различные методы защищенной аутентификации, а также некоторые виды защищённых операций с файлами через сеть. За годы его существования было разработано много как открытых, так и коммерческих версий. Сегодня наиболее распространенной является версия ***Open SSH*** v.2, клиент и сервер которой включены практически во все свободно распространяемые версии Unix/Linux: Fedora, CentOS, Ubuntu и другие.

Рассмотрим более подробно два аспекта локальной системы аутентификации Unix: безопасное хранение паролей и аутентификацию с помощью протокола SSH.

Начиная с самых ранних версий и до последних реализаций ОС семейства Unix, пароли пользователей (точнее – их хеши) хранились в файле `passwd` (находящего в каталоге административных утилит `/etc`) вместе с учетными данными пользователей. Доступ к файлу `passwd` для чтения был разрешен всем, а запись в него – только суперпользователю `root`. Формат файла `passwd` не изменился с тех пор, и это удобно для многочисленных приложений, которые его используют. Рассмотрим следующий фрагмент файла `passwd`:

```
tomcat:x:91:91:Apache Tomcat:/usr/share/tomcat6:/sbin/nologin

webalizer:x:67:67:Webalizer:/var/www/usage:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash

tcpdump:x:72:72::/:/sbin/nologin

victoro:x:500:500:Victor Olifer:/home/victoro:/bin/bash
```

Каждая запись файла `passwd` состоит из следующих 10 полей:

- логическое имя пользователя (в последней записи это victoro);

---

[1] С появлением графических интерфейсов пользователя, таких как GNOME. KDE, текстовый режим работы утилиты login был дополнен ее графическим вариантом, однако ее суть работы от этого не изменилась.

- хеш пароля (здесь он представлен символом x) ;
- идентификатор пользователя (500);
- идентификатор группы пользователя (500);
- полное имя пользователя (Victor Olifer);
- домашний каталог (/home/victoro);
- версия оболочки Unix (shell), которая запускается для данного пользователя после его успешного логического входа (/bin/bash).

К сожалению, право непривилегированных пользователей читать файл passwd делает возможным злоумышленникам восстановливать паролей по их хешам с помощью многочисленных программ, имеющихся в свободном доступе в Интернете. Однако лишение непривилегированных пользователей этого права может нарушить работу тех утилит Unix, которые выступают от имени непривилегированного пользователя и читают из файла `passwd` не пароли, а такие атрибуты пользователя, как его домашний каталог или его полное имя. В некоторых версиях Unix эту проблему решают сохранением свободного доступа к файлу `passwd`, с переносом хешей паролей в отельный файл /etc/`shadow`, доступ к которому как по чтению, так и по записи имеет только суперпользователь `root`. Записи файла `shadow`, соответствующие записям файла passwd из приведенного выше примера, будут такими:

```
tomcat:!!:15866::::::

webalizer:!!:15866::::::

sshd:!!:15866::::::

postgres:!!:15866::::::

tcpdump:!!:15866::::::

victoro:$6$rzn7RYTyEWi3nKFS$gqXtf73qUuNVLZNR2jkoKu4lM9qOnwkL5Vmqyy/Zr1q68poIdDWik72
HWLKhuWDx4VJ3/1yyCfnpTQ0pLOkcn0:15866:0:99999:7:::
```

Формат записи файла `shadow (на примере последней записи)` следующий:

- логическое имя пользователя (victoro);
- поле хэша пароля, которое состоит из трех подполей, разделённых знаком «$»: подполя условного номера односторонней функции (в примере – 6), подполя криптографической «соли», добавленной к паролю при его хешировании (rzn7RYTyEWi3nKFS), и подполя собственно хэша (gqXtf73qUuNVLZNR2jkoKu4lM9qOnwkL5Vmqyy/Zr1q68poIdDWik72HWLKhuWDx4VJ3/1yyCfnpTQ0pLOkcn0);
- количество дней, прошедших с 1 января 1970 года (момент начала отсчета времени Unix – Unix epoch) до последнего изменения пароля (15866);
- количество дней до разрешенной смены пароля (минимальный срок пароля, здесь -0);
- количество дней до необходимой смены пароля (максимальный срок пароля, здесь - 99999);
- количество дней до предупреждения об истечении пароля (7);
- количество дней до неактивности учетной записи (не задано);
- количество дней с 1 января 1970 года до истечения действия учетной записи (не задано).

Заметим, что в записях файла `shadow` хеш пароля имеется только для записи victoro, а у остальных записей, которые соответствуют системным сервисам, пароль отсутствует, о чем говорит значение ”!!” вместо хэша.

Протокол SSH работает в архитектуре клиент-сервер, серверная часть представлена демоном sshd, клиентская часть - утилитой ssh, которая выполняет запрос пользователя на логический вход в удаленный хост. Например, для логического входа в хост ganimede.co.uk пользователь victor выполняет команду

```
ssh victor@ganymede.co.uk
```

Протокол SSH обеспечивает создание защищенного канала между клиентом и сервером, а также их взаимную аутентификацию.

Аутентификация с помощью SSH может осуществляться различными способами.

- *Аутентификация на основе имен хостов.* Удаленный хост успешно аутентифицирует пользователя, если выполняется два условия: (1) имя пользователя и хоста, с которого этот пользователь осуществляет логический вход, содержится в определенных файлах на удаленном хосте, и (2) удаленный хост успешно верифицировал открытый ключ клиентского компьютера, предъявленный в процессе данной сессии аутентификации. Верификация выполняется путем сравнения этого открытого ключа с храняшимися на удаленном хосте ключами, которые передавались клиентским хостом во время других, предыдущих успешных логических входов данного пользователя, в том числе использовавших другие методы аутентификации (из перечисленных ниже). Если же предъявленный ключ клиента не совпадает ни с одним из имеющихся у удаленного хоста ключей, то он выдает предупреждение о возможной фальсификации и аутентификация не считается успешной[2]. Ключи хостов для этого вида аутентификации генерируются автоматически при установке пакета SSH в операционной системе хоста, их основным назначением является создание защищенного канала. Поэтому сам по себе ключ клиента не может однозначно его аутентифцировать - для этого и требуется дополнительная информация его аутентичности в виде факта успешной аутентифкации другим способом.

- *Аутентификация с помощью публичных ключей.* Используя утилиту ssh-keygen, пользователь должен вручную сгенерировать открытый и закрытый ключи, поместить каждый из них в отдельный файл на своем клиентском хосте, а затем скопировать файл с открытым ключом в свой домашний каталог на хосте-сервере, в который он хочет удаленно входить. После этого пользователь может выполнять удаленный вход в этот сервер по протоколу SSH без использования пароля. В целях аутентификации клиента сервер с помощью известного ему открытого ключа данного клиента предпринимает попытку расшифровать информацию, поступающую к нему в ходе процедуры установления соединения. Поскольку эту информацию клиент шифрует своим соответствующим закрытым ключом, то успех расшифровки означает успех аутентификации.
- *Аутентификация на основе слова-вызова.* Этот вариант аутентификации уже был рассмотрен ранее (см. раздел «Строгая аутентификация в компьютерной сети на основе многоразовых паролей»). Передача слова-вызова от удаленного сервера и хеш-результата от хоста-клиента происходит по защищенному каналу, который образуется на основе автоматически сгенерированных ключей хоста и сервера.

[2] Проверка ключа предохраняет от атак типа IP-спуфинга или DNS-спуфинга, о которых будет сказано далее.

- *Аутентификация на основе пароля.* Отличается от аутентификации на основе слова-вызова тем, что клиент передает по сети пароль, а не хеш от него. Поскольку пароль и в этом случае передается по защищенному каналу, то этот способ также считается защищенным, хотя и менее защищенным, чем предыдущие.

Клиент SSH пытается использовать различные способы аутентификации в том порядке, в котором они описаны выше, то есть начиная с аутентификации на основе хостов и кончая аутентификацией на основе паролей. Переход к более низкоприоритетному способу аутентификации происходит в том случае, если очередной способ аутентификации не поддерживается обеими сторонами. В то же время пользователь может задать предпочтительный способ аутентификации принудительно, использовав переменную `PreferredAuthentications` в файле конфигурации клиента `ssh_config` или же задавая эту переменную как опцию команды SSH.

Кроме аутентификации во время логического входа программы пакета SSH могут использоваться для таких полезных операций, как защищенное копирование файлов между удаленными хостами (с помощью утилиты scp) или запуск приложения на удаленном хосте с отображением графического интерфейса пользователя этого приложения на хосте клиента (при логическом входе с ключом –X).

Семейство ОС Unix позволяет использовать для аутентификации не только локальную схему на основе файла паролей, хранящегося на диске данного компьютера, но и централизованные системы, такие как NIS, NIS+, Open Directory, Kerberos.

Контроль доступа в ОС Unix

В ОС Unix реализована дисреционная модель управления доступом к ресурсам, при которой права доступа назначаются децентрализовано пользователями системы.

Основой абстрактного представления ресурсов всех типов - файлов, каталогов, принтеров, устройств ввода-вывода - является файловая модель.

Все объекты хранятся в древовидных иерархических структурах, элементами которых являются ***объекты-ветви*** (каталоги) и ***объекты-листья*** (файлы). Для объектов файловой системы такая схема отношений является прямым отражением иерархии каталогов и файлов. Для объектов других типов иерархическая схема отношений имеет свое содержание, например для процессов она отражает связи «родитель-потомок», а для устройств отражает принадлежность к определенному типу устройств исвязи устройства с другими устройствами, например SCSI-контроллера с дисками.

В ОС Unix права доступа к файлу или каталогу определяются для следующих трех типов субъектов:

- владельца файла;
- членов группы, к которой принадлежит владелец;
- всех остальных пользователей системы.

Всем пользователям и группам назначены уникальные идентификаторы, обозначаемые UID (User ID) для пользователя и GID (Group ID) для группы.

В ОС Unix определен особый вид субъекта – ***суперпользователь*** – он имеет имя `root`, ему всегда позволены все виды доступа, поэтому его идентификатор (он имеет значение 0) никогда не фигурирует в списках управления доступом. Суперпользователь `root` можно рассматривать как элемент ролевой модели управления доступом, включенный в доминирующую в ОС Unix дискреционную модель.

В Unix определены всего три операции над объектами: чтение (**R**ead), запись (**W**rite) и выполнение (e**X**ecute), условно обозначаемые R, W и X соответственно.

Права доступа к объекту описываются ***списком доступа***, включающим девять признаков, каждый из которых определяет возможность выполнения одной из трех операций для каждого из трех типов субъектов доступа. Например (рис. Список доступа), если владелец файла (его идентификатор UID равен 54) разрешил себе выполнение всех трех операций, для членов своей группы (ее идентификатор GID=47) — чтение и выполнение, а для всех остальных пользователей — только выполнение, то девять характеристик безопасности файла выглядят следующим образом RWX R-X R--, где первые три символа RWX – это разрешение чтения, записи и выполнения для владельца, следующая «тройка» R-X представляет разрешение чтения и выполнения для членов группы владельца и, наконец, последние символы R - -описывают права для всех остальных - только чтение.

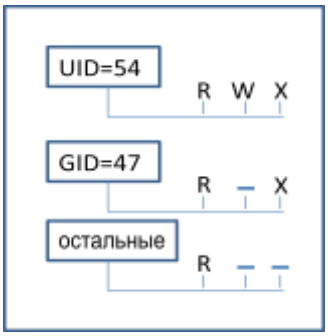

Рис. Пример списка доступа к объекту.

Процессы в ОС Unix также как и пользователи являются субъектами. С каждым процессом Unix связаны две пары идентификаторов – реальные и эффективные. ***Реальные идентификаторы*** принимают значение идентификаторов UID и GID пользователя, который создал этот процесс. Реальные идентификаторы пользователя и группы обозначаются как ***RUID*** (*Real User ID*), и ***RGID*** (*Real Group ID*) соответственно. ***Эффективные идентификаторы*** процесса ***EUID*** *(Effective User ID*) и ***EGID*** **(***Effective Group ID***)** могут определенным образом менятся на протяжении его «жизни». В момент создания процесса реальные и эффективные идентификаторы совпадают. Механизм эффективных идентификаторов позволяет процессу выступать в некоторых случаях от имени пользователя и группы, отличных от тех, которые ему достались "при рождении".

Именно эффективные, а не реальные идентификаторы, используются при проверке прав доступа процесса к объекту. Если эффективный идентификатор процесса EUID совпадает с идентификатором UID в списке доступа, то ему разрешены все права владельца объекта владельца, аналогично, если совпадают идентификаторы групп EGID и GID, то процесс получает права пользователей данной группы. Если такого совпадения нет, то процесс получает права доступа, определенные в списке для всх остальных.

В ОС Unix процесс, выполняющий системный вызов `exec file_program`, переключается на исполнение другого кода, хранящегося в файле с именем `file_program`. В момент смены исполняемого кода эффективные идентификаторы процесса могут быть заменены на идентификаторы пользователя и группы, указанные в новом программном файле. Так происходит, если в характеристиках безопасности файла, хранящего новый код, установлены признаки смены идентификаторов - SUID (Set User ID on execution), разрешающий смену идентификатора пользователя, и SGID (Set Group ID on execution) – смену группы.

Механизм эффективных идентификаторов позволяет пользователю ОС Unix получать некоторые виды доступа, которые явно ему не разрешены. Для этого могут быть использованы определенные системные приложения, исполняемые коды которых хранятся в файлах с установленными признаками разрешенной смены идентификаторов. Пример такой ситуации приведен на рис. 4.3.

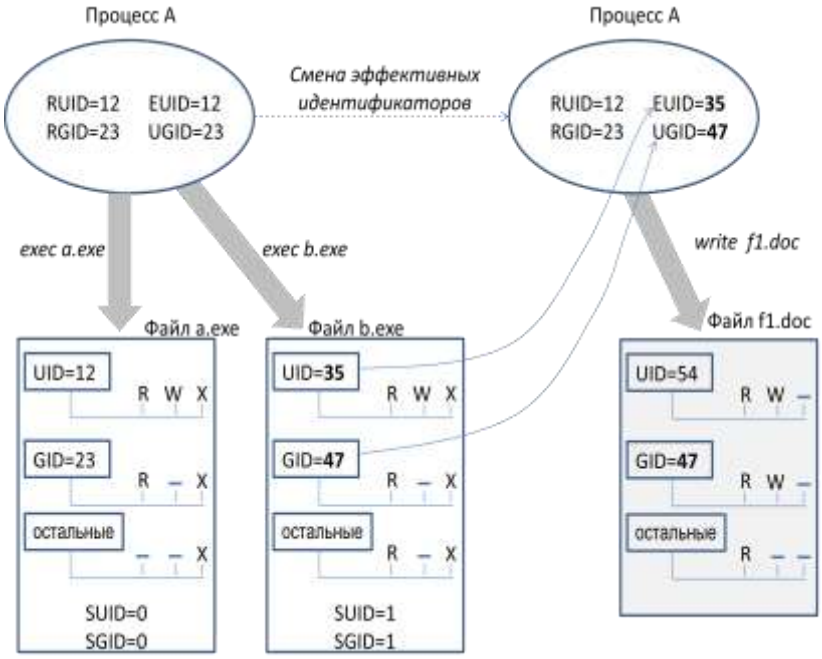

Рис. 4.3. Смена эффективных идентификаторов процесса

Первоначально процесс `A` выполнял код из файла a.exe и имел эффективные идентификаторы пользователя и группы (12 и 23 соответственно), совпадающие с реальными. На каком-то этапе работы процесс выполнил системный вызов `exec b.exe` по которому он переключился на исполняемый код из файла `b.exe`. (Заметим, что процесс А может выполнить файл `b.exe`, так как его выполнение разрешено всем пользователям.)

Файл `b.exe` имеет установленные признаки смены идентификаторов SUID и SGID, поэтому одновременно со сменой кода процесс А меняет значения эффективных идентификаторов (12 на 35 и 23 на 47). Благодаря этому при последующей попытке записать данные в файл `f1.doc` процессу `A` это удается, так как его новый эффективный идентификатор группы EGID=47 совпадает с идентификатором группы файла `f1.doc`. GID=47. Без смены идентификаторов эта операция для процесса `A` была бы запрещена.

Примером использования такого механизма является утилита /usr/bin/passwd, которая позволяет пользователю изменять свой собственный пароль, хранящийся в файле /etc/passwd . Пользовательский процесс не имеет права записи в файл /etc/passwd, но имеет право запустить на выполнение утилиту /usr/bin/passwd. При этом происходит смена эффективного идентификатора пользователя на эффективный идентификатор утилиты /usr/bin/passwd, у которого имеется разрешение на запись в файл /etc/passwd. (На рисунке 4.3 программный код из файла b.exe роль соответствует утилите /usr/bin/passwd , а файл /etc/passwd - файлу f1.doc)

Использование модели файла как универсальной модели разделяемого ресурса позволяет в Unix применять одни и те же механизмы для контроля доступа к файлам, каталогам, принтерам, терминалам и разделяемым сегментам памяти.

Особое внимание должно уделяться использованию учетной записи `root`. Из-за того, что `root` может выполнять в Unix любые операции без каких-либо ограничений, злоумышленник, завладевший паролем суперпользователя, может нанести очень серьезный ущерб системе, читая любые данные, изменяя ее конфигурацию или разрушая ее элементы. Кроме того, даже если вход с именем `root` сделал легальный администратор системы, то далеко не все операции, которые ему необходимо выполнять в

процессе работы, требуют полномочий суперпользователя. А вот последствия ошибок администратора будут намного более серьезными, если во время всего сеанса своей работы он будет обладать полномочиями суперпользователя. Поэтому считается нормальной практикой свести до минимума работу пользователей Unix от имени `root`, то есть следовать принципу минимальных прав.

Для реализации концепции минимального использования прав суперпользователя в системе Unix предусмотрены две команды: `su` и `sudo`. Обе они предполагают, что пользователи, которым нужно и разрешено выполнять некоторые административные привилегированные действия, входят в систему не под именем `root`.

Команда `su –username` используется для временной передачи пользователю, выполнившему эту команду, статуса другого пользователя, имя которого указывается в параметре `username`. **Ч**аще всего в качестве такого пользователя указывается суперпользователь `root`, именно он подразумевается по умолчанию в том случае, когда команда `su` вводится без параметра `username`. Для того чтобы вернуться к своей начальной учетной записи, пользователь должен выполнить команду `exit`. Применение команды `su` не так уж сильно повышает защищенность системы, так как пароль суперпользователя все равно вводится, а пользователь, принявший статус `root`, может находится в этом состоянии произвольное время, например, просто забыв выполнить команду `exit`, а значит, вероятность совершения им ошибок с тяжелыми последствиями для системы остается высокой.

Команда `sudo` гораздо надежнее защищает систему от случайных ошибок администрирования или перехвата пароля `root` злоумышленником. Эта команда избирательно позволяет некоторым пользователям или группам пользователей выполнять некоторые привилегированные команды.

Например, создание нового пользователя является привилегированной операцией, разрешенной только суперпользователю. Однако с помощью `sudo` пользователь может обойти этот запрет, выполнив следующие команды:

- `sudo /usr/sbin/useradd katerina`
- `password: ******`

Здесь `/usr/sbin/useradd` – утилита создания новых пользователей запускать программу /usr/sbin/useradd, запускать которую может только пользователь root и члены группы root;

`katerina` - параметр утилиты – имя нового пользователя,

строка `password: ******` требует ввода собственного пароля пользователя (а не суперпользователя).

Очевидно, что система должна каким-то образом ограничивать круг пользователей, которым разрешено использовать такое мощное средство доступа. Для этого администратор ОС создает и поддерживает системный конфигурационный файл `/etc/sudoers`, записи которого и определяют, каким пользователям позволено выполнять те или иные привилегированные команды.

Например, если администратор решил дать обычному пользователю victoro право создавать новых пользователей, то для этого достаточно добавить к файлу `sudoers` такую строку:

```
victoro ALL=/usr/sbin/useradd
```

Ключевое слово ALL означает, что пользователь victoro может применять команду `sudo` для выполнения утилиты `/user/sbin/useradd` на любых компьютерах, на которых он имеет учетную запись (для того, чтобы органичить пользователя только некоторыми компьютерами, администратор должен вместо ключевого слова ALL явно указать их IP-адреса).

Как мы видим, механизм команды `sudo` (вместе с конфигурационным файлом `sudoers`) является гибким средством предоставления пользователям доступа к выполнению некоторого набора привилегированных команд , обеспечивающим высокий уровень защищенности за счет отказа от использования пароля суперпользователя и его безграничных прав. Применение команды `sudo` вместо `su` или работы под именем root является настоятельной рекомендацией для повышения защищенности ОС семейства Unix.

Аутентификация пользователей в ОС Windows

В ОС семейства Windows свойства системы аутентификации пользователей в значительной степени зависят от того, какая модель аутентификации применяется: ***локальная система аутентификации*** или ***доменная система аутентификации.***

Доменная система аутентификации является предпочтительным вариантом, он рекомендуется компанией Microsoft даже в том случае, если сеть состоит из нескольких рабочих станций и одного сервера. В этом разделе мы сосредоточимся на свойствах локальной системы аутентификации, а доменную аутентификацию рассмотрим позже в отдельном разделе.

В ОС семейства Windows принят модульный подход к построению системы аутентификации, позволяющий использовать различные протоколы аутентификации. Этот подход иллюстрируется рис. 4.4. На нем показаны основные модули этой системы, причем также и те, которые относятся к доменной системе аутентификации по той простой причине, что без них трудно понять работу системы в целом.

Winlogon является сервисом Windows, который отвечает за логический вход пользователя в систему, в том числе за локальный интерактивный вход, который выполняется с помощью модуля GINA (Graphical Identification and Autgentication). Модуль GINA отвечает за показ на экране графической панели, с помощью которой пользователь вводит свое имя и пароль. Сервис Winlogon также ответственен за локальную неинтерактивную (удаленную) аутентификацию пользователей, когда доступ к графической оболочке ОС выполняется по сети с помощью программы Remote Desktop. В обоих случаях – интерактивной и удаленной локальной аутентификации – используются ученые данные пользователей, хранящиеся локально в базе учетных данных ***SAM*** (***Security Accounts Manager database***).

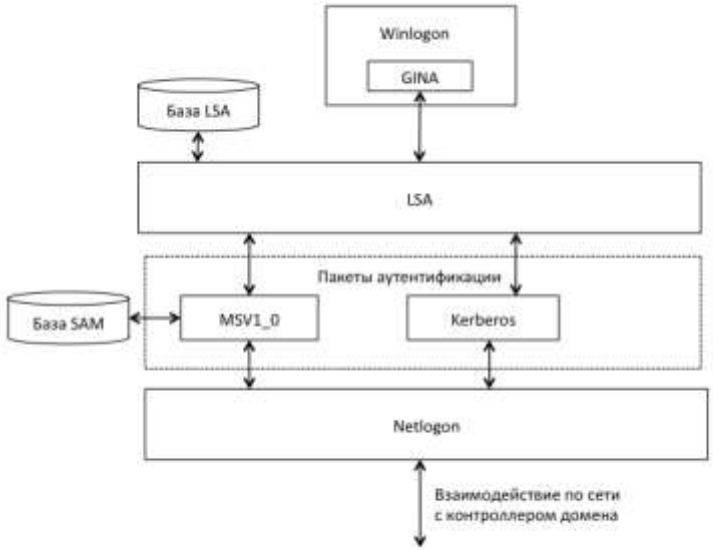

Рис. 4.4. Структура модулей системы аутентификации Windows

Сервис локальной безопасности LSA (Local Security Authority) координирует работу остальных модулей системы аутентификации при выполнении интерактивного логического входа. Этот сервис получает от сервиса Winlogon имя и пароль пользователя и передает их для аутентификации одному из пакетов аутентификации, установленных в системе. Однако перед этим сервис LSA проверяет свою базу данных, в которой содержатся правила политики логического входа, для проверки ограничений, например может ли данный пользователь выполнить интерактивный вход в данный компьютер в данный день недели и данное время? В базе данных LSA хранятся также и другие правила политики безопасности, а также некоторые права пользователя на выполнение системных действий.

В ОС Windows может быть установлено несколько пакетов аутентификации, в штатную конфигурацию входят пакеты Kerberos и MSV1_0. Пакет Kerberos используется только для доменной системы аутентификации, о которой будет сказано позже.

Пакет MSV1_0 может поддерживать как локальную, так и доменную систему аутентификации. При локальной аутентификации (как интерактивной, так и удаленной) пакет MSV1_0 вычисляет хеш введенного пользователем пароля и сравнивает его с хэшем пароля, хранящемся в локальной базе данных учетных записей пользователей *SAM*.

Пакет MSV1_0 работает с различными функциями хеширования паролей: LM, NTLMv1 и NTLMv2. Функция хеширования LM не обладает высокой криптостойкостью и довольно легко взламывается существующими программами взлома паролей, она поддерживается для обеспечения совместимости с более ранними версиями ОС. По умолчанию для интерактивной аутентификации используется ее усовершенствованная версия - NTLMv2.

В том случае, когда пакет MSV1_0 работает с доменной системой аутентификации, он обращается не к локальной базе SAM, а вызывает сервис Netlogon, который получает доступ к серверу-контроллеру домена, где хранятся учетные данные пользователей домена. Сервис Netlogon устанавливает в сети защищенный канал, по которому передается слово-вызов, используемое при аутентификации.

Политика управления паролями в ОС Windows является достаточно гибкой. Правила этой политики можно задавать локально, и тогда они хранятся в базе LSA, а также в пределах домена – тогда они хранятся в базе Active Directory.

Например, в ОС Windows с помощью утилиты администрирования Local Security Policy могут быть заданы следующие правила локальной политики паролей.

Правило ведения журнала паролей: позволяет задать число уникальных новых паролей (от 0 до 24) перед тем, как пользователь может использовать старый пароль вновь.

Правило, устанавливающее максимальный срок действия пароля: период времени в днях (от 1 до 999), в течение которого пароль может быть использован.

Правило, устанавливающее минимальный срок действия пароля: период времени в днях (от 0 до 998), в течение которого пользователь не может сменить действующий пароль; должен быть меньше, чем максимальный срок действия.

Правило, устанавливающее минимальную длину пароля: значение должно быть от 1 до 14, 0 означает отсутствие пароля. Рекомендуется использовать пароли не менее 12 символов длиной.

Правило, устанавливающее требования к сложности пароля:

- не содержать имени пользователя или частей его полного имени, превышающих два символа
- содержать комбинацию как минимум из трех указанных ниже четырех категорий символов: прописные буквы, строчные буквы, цифры, специальные символы (например. !, $, #, %).

Правило хранения паролей, которое может потребовать хранить пароли в виде, зашифрованном обратимыми (а не односторонними) функциями шифрования. Такой способ хранения позволяет при необходимости расшифровывать пароли пользователей. Применение этой опции ведет к снижению защищенности ОС, поэтому использовать ее нужно очень осторожно.

Контроль доступа в ОС семейства Windows

Система управления доступом в ОС семейства Windows реализует дискреционный алгоритм, она отличается высокой степенью гибкости, которая достигается за счет большого разнообразия типов субъектов и объектов доступа, а также детализации операций доступа.

Для разделяемых ресурсов в ОС семейства Windows применяется общая модель объекта, который содержит такие характеристики безопасности, как набор допустимых операций, идентификатор владельца, список управления доступом. Объекты создаются для любых ресурсов в том случае, когда они являются или становятся разделяемыми — файлов, каталогов, устройств, секций памяти, процессов.

Для системы безопасности ОС семейства Windows характерно наличие большого количества различных предопределенных (встроенных) субъектов доступа — как отдельных пользователей, так групп. Так, в системе всегда имеются пользователи *Administrator, System и Guest,* а также группы *Users, Administrators, Account Operators, Server Operators, Everyone* и другие. Смысл этих встроенных пользователей и групп состоит в том, что они изначально наделены некоторыми правами, облегчая администратору работу по созданию эффективной системы разграничения доступа. При добавлении нового пользователя администратору остается только решить, к какой группе или группам отнести этого пользователя. Конечно, администратор может создавать новые группы, а также добавлять права к встроенным группам для реализации собственной политики безопасности, но во многих случаях встроенных групп оказывается вполне достаточно.

ОС семейства Windows поддерживает три класса *операций доступа*, которые отличаются типом субъектов и объектов, участвующих в этих операциях:

- ***Разрешения*** (*permissions*) — это множество операций, которые могут быть определены для субъектов всех типов по отношению к объектам любого типа: файлам, каталогам, принтерам, секциям памяти и т. д. Разрешения по своему назначению соответствуют правам доступа к файлам и каталогам в ОС Unix.
- ***Права*** (*user rights*) определяются для субъектов типа группа на выполнение некоторых системных операций: установку системного времени, архивирование файлов, выключение компьютера и т. п. В этих операциях участвует особый объект доступа — операционная система в целом.
- ***Возможности пользователей*** (*user abilities*) определяются для отдельных пользователей на выполнение действий, связанных с формированием их операционной среды, например, изменение состава главного меню программ, возможность пользоваться пунктом меню `Run` (`Выполнить`) и т. п. За счет уменьшения набора возможностей (доступных пользователю по умолчанию) администратор может «заставить» пользователя работать с той операционной средой, которая наилучшим образом соответствует политике безопасности.

Права и разрешения, данные группе, автоматически предоставляются всем ее членам, позволяя администратору рассматривать большое количество пользователей как единицу учетной информации и минимизировать свои действия.

В основном именно права, а не разрешения отличают одну встроенную группу пользователей от другой. Права у встроенных групп могут быть встроенными либо изменяемыми. Встроенные права являются неотъемлемыми атрибутами встроенных групп, администратор не может ими распоряжаться . Изменяемые права можно удалять или добавлять к правам встроенной группы из общего списка изменяемых прав.

Например, встроенная группа Users не имеет никаких изменяемых или встроенных прав, в то время как группа *Administrators* наделена широким набором изменяемых прав (интерактивный и удаленный логический вход, установление прав собственности на файлы, управление аудитом событий, связанных с безопасностью, изменение системного времени, останов системы, инициирование останова с удаленной системы, резервное копирование файлов и каталогов, восстановление файлов и каталогов со стримера, загрузка и выгрузка драйверов устройств и др.) и встроенных прав (создание и управление пользовательской учетной информацией, назначение прав для пользователей, управление аудитом системных событий, блокирование и преодоление блокировки сервера, форматирование жесткого диска сервера и др.)

Результатом успешной аутентификации пользователя в ОС Windows является создание для него системой так называемого ***токена доступа*** (*access token*). Токен доступа привязывается ко всем процессам, которые данный пользователь создает в течении сеанса работы.

Токен доступа который включает:

- идентификатор пользователя;
- идентификаторы всех групп, в которые входит пользователь;
- список ACL по умолчанию; этот список может быть приписан объектам, создаваемым процессами пользователя;
- список прав пользователя на выполнение системных операций.

Так же как и в ОС Unix в некоторых ситуациях некоторые процессы в ОС Windows могут изменять свои идентификаторы. Смена идентификаторов разрешена процессам, которые работают в качестве серверов, то есть обслуживает запросы своих клиентов (например, процесс файлового сервера). Процессу-серверу разрешается получить токен доступа (а значит, и содержащиеся в нем идентификаторы) у процесса-клиента, запросившего у сервера выполнение некоторого действия, и использовать его при доступе к объектам.

Доступ к объекту описывается списком ACL. В отличие от Unix, здесь в элементах ACL могут присутствовать как списки разрешенных, так и списки запрещенных для пользователя (процесса) операций. Владелец объекта, обычно пользователь, который его создал, обладает правом управлять доступом к объекту и может изменять ACL объекта, чтобы позволить или не позволить другим осуществлять тот или иной вид доступа к объекту. Встроенный пользователь `Administrator` в ОС семейства Windows в отличие от суперпользователя в ОС Unix может не иметь некоторых разрешений на доступ к объекту. Для реализации этой возможности идентификаторы администратора и группы администраторов могут входить в ACL наряду с идентификаторами остальных пользователей. Однако у администратора имеется принципиальная возможность выполнять любые операции с любыми объектами, поскольку он наделен встроенным правом становиться владельцем любого объекта, а затем уже как владелец он может назначить себе полный набор разрешений по отношению к этому объекту.

В ОС семейства Windows однозначно определены правила, по которым вновь создаваемому объекту назначается список ACL.

- Если вызывающий код во время создания объекта явно задает все права доступа к вновь создаваемому объекту, то система безопасности приписывает объекту этот список ACL.
- Если же вызывающий код не снабжает объект списком ACL, а объект имеет имя, то применяется ***принцип наследования*** разрешений. Система безопасности просматривает ACL того каталога объектов, в котором хранится имя нового объекта. Некоторые из входов ACE каталога объектов могут быть помечены как наследуемые. Это означает, что они могут быть приписаны новым объектам, создаваемым в этом каталоге.
- В случае, когда процесс не задал явно список ACL для создаваемого объекта и объект-каталог не имеет наследуемых элементов ACE, используется список ACL по умолчанию из токена доступа процесса.

Чаще всего при создании новых объектов (особенно файлов) применяется наследование разрешений.

Проверка прав доступа к объектам любого типа выполняется *централизованно* с помощью модуля ОС - ***монитора безопасности*** (*Security Reference Monitor*). Монитор безопасности сравнивает идентификаторы пользователя и групп пользователей из токена доступа процесса с соответствующими идентификаторами, хранящимися в элементах ACL объекта.

Система безопасности могла бы осуществлять проверку разрешений каждый раз, когда процесс использует объект. Но список ACL состоит из многих элементов, процесс в течение своего существования может иметь доступ ко многим объектам, и количество активных процессов в каждый момент времени также велико. Поэтому для снижения издержек проверка выполняется только при первом обращении процесса к объекту[3], а не при каждом использовании объекта.

Для ОС семейства Windows характерна высокая степень детализации операций доступа. Поясним ее на примере доступа к каталогам и файлам.

В ОС семейства Windows предусмотрено два типа разрешений:

- ***индивидуальные разрешения*** относятся к элементарным операциям с каталогами и файлами;
- ***стандартные разрешения*** являются объединением нескольких индивидуальных разрешений.

В табл. 4.1 показано шесть индивидуальных разрешений (элементарных операций), смысл которых различается для каталогов и файлов.

Таблица 4.1. Индивидуальные разрешения для каталогов и файлов

Разрешение	Для каталога	Для файла
Read (R)	Чтение имен файлов и каталогов, входящих в данный каталог, а также атрибутов и владельца каталога	Чтение данных, атрибутов, имени владельца и разрешений файла
Write (W)	Добавление файлов и каталогов, изменение атрибутов каталога, чтение владельца и разрешений каталога	Чтение владельца и разрешений файла, изменение атрибутов файла, изменение и добавление данных файла
Execute (X)	Чтение атрибутов каталога, выполнение изменений в каталогах, входящих в данный каталог, чтение имени владельца и разрешений каталога	Чтение атрибутов файла, имени владельца и разрешений. Выполнение файла, если он хранит код программы
Delete (D)	Удаление каталога	Удаление файла

[3] Более точно – права доступа проверяются при каждом открытии объекта.

Change Permission (P)	Изменение разрешений каталога	Изменение разрешений файла
Take Ownership (O)	Вступление во владение каталогом	Вступление во владение файлом

Для файлов определено четыре стандартных разрешения: *No Access, Read, Change* и *Full Control*, которые объединяют индивидуальные разрешения, перечисленные в табл. 4.2.

Таблица 4.2. Стандартные разрешения, объединяющие индивидуальные разрешения

Стандартное разрешение	Индивидуальные разрешения
No Access	Ни одного
Read	RX
Change	RWXD
Full Control	Все

Для каталогов определено семь стандартных разрешений: *No Access, List, Read, Add, Add&Read, Change* и *Full Control*. В табл. 4.3 показано соответствие стандартных разрешений индивидуальным разрешениям для каталогов, а также то, каким образом эти стандартные разрешения преобразуются в индивидуальные разрешения для файлов, входящих в каталог, в том случае, если файлы наследуют разрешения каталога.

Таблица 4.3. Соответствие стандартных разрешений индивидуальным разрешениям при наследовании

Стандартные разрешения	Индивидуальные разрешения для каталога	Индивидуальные разрешения для файлов каталога при наследовании
No Access	Ни одного	Ни одного
List	RX	Не определены
Read	RX	RX
Add	WX	Не определены
Add & Read	RWX	RX
Change	RWXD	RWXD
Full Control	Все	Все

При создании файла он наследует разрешения от каталога указанным способом только в случае, если у каталога установлен признак наследования его разрешений. ОС управляет наследованием по принципу «все или ничего», не позволяя установить такой признак для каждого разрешения отдельно.

Гибкость системы безопасности ОС семейства Windows во многом определяется большим разнообразием прав на выполнение системных действий, высокой степенью детализации операций доступа к объектам, а также существованием встроенных групп, позволяющих администратору эффективно реализовывать политику безопасности данной информационной системы.

Классификация систем единого логического входа

Описанная схема допускает различные реализации в зависимости от применяемых технологий и протоколов аутентификации, типов операционных систем и сервисов, а также организационной принадлежности ее элементов.

В зависимости от применяемых криптографических технологий аутентификации системы единого логического входа делятся на два класса:

- *системы на основе разделяемого секрета*: многоразовых и одноразовых паролей, биометрических данных и другой информации, которая имеется как у пользователя, так и в базе учетных данных провайдера идентичности;
- *системы на основе технологии открытых и закрытых ключей*, использующей цифровые сертификаты; в этом варианте провайдером идентичности является центр сертификации, выдавший цифровой сертификат пользователю, а сам сертификат используется в качестве токена доступа.

В свою очередь системы на основе разделяемого секрета могут далее подразделяться в зависимости от применяемого протокола аутентификации. Наиболее популярными протоколами этого типа являются протоколы Kerberos, RADUIS, TACACS.

В зависимости от типа сервиса, к которому пользователь получает доступ, системы единого логического входа делятся на:

- *системы входа на основе веб-сервисов*. Эти системы рассчитаны на широкий класс веб-сервисов, к которым доступ осуществляется с помощью веб-браузера. Эти системы используют специфику веб-сервисов – протокол HTTPS, языки XML, SAML. Примером SSO этого типа является система *Shibboleth*, разработанная сообществом Internet2;
- *системы входа на основе корпоративных сервисов*. Здесь под корпоративными сервисами понимаются все сервисы, не использующие веб-браузер в качестве пользовательского интерфейса, например сервисы мейнфреймов, баз данных и других корпоративных приложений. В частности к этому типу систем единого логического входа относится система Kerberos.

В зависимости от организационной принадлежности системы единого логического входа делятся на корпоративные и федеративные:

- в *корпоративной системе* все элементы – пользователи, провайдеры идентичности и сервис провайдеры - принадлежат одной организации, возможно – разным ее структурным отделениям;
- *федеративную систему* составляют различные организации, которые договорились доверять друг другу в отношении аутентификации своих пользователей. В общем случае каждая организация выполняет функции как провайдера идентичности, так и сервис- провайдера. Каждая организация поддерживает базу учетных записей своих пользователей, которая не реплицируется в другие организации. При необходимости доступа пользователя одной организации к сервису другой организации выполняется вторичная аутентификация с помощью определенного протокола, например RADIUS или Shibboleth. Федеративная аутентификация достаточно популярна в академической среде, примером федерации такого типа является федерация Eduroam, членами которой являются многие университеты и исследовательские центры Европы и Америки. В Eduroam используется иерархия серверов RADIUS, а доступ ограничивается только доступом пользователей к Интернет через беспроводные сети.

Существуют также ряд систем манипулирования паролями, которые лишь очень условно могут быть отнесены к системам единого входа. Эти системы, представляющие собой надстройку над традиционной децентрализованной инфраструктурой локальных баз учетных записей, помогают пользователю управляться с большим количеством паролей для разных сервисов. Рассмотрим три типа таких систем:

- системы кэширования паролей на стороне клиента:
- системы кэширования паролей на стороне сервера;

- системы синхронизации паролей между серверами.

Система кэширования паролей на стороне клиента хранит в надежном зашифрованном виде все пароли, которые были использованы пользователем при входе в тот или иной сервер. При повторном обращении к этому серверу она сама автоматически выполняет логический вход от имени пользователя. Вы, конечно, сталкивались с такой функцией, встроенной во многие браузеры, когда браузер предлагает вам запомнить пароль. Считается, что кэширование паролей на стороне клиента является весьма опасной практикой, так как злоумышленник в случае получения доступа к вашему компьютеру, сразу получает доступ к всем серверам и сервисам, которыми вы пользуетесь, в том числе и вашему банковскому счету.

Кэширование на стороне сервера означает, что все пароли пользователя хранятся централизованно, например на выделенном сервере предприятия. При логическом входе пользователя в корпоративную сеть кэш паролей временно загружается в его клиентский компьютер, а после окончания сессии работы с ним кэш уничтожается. Считается, что это более защищенный способ по сравнению с постоянным хранением кэша паролей на клиентском компьютере.

Программы синхронизации паролей между серверами помогают уменьшить разнообразие паролей пользователей в корпоративной сети и свести их в крайнем случае к одному, действительному для всех серверов сети.

Все три разновидности программ манипулирования паролями не ликвидируют главную причину необходимости выполнения многочисленных логических входов в различные системы – наличие большого количества локальных баз учетной информации пользователей. Из-за этого обстоятельства такие системы и не относят к системам единого логического входа специалисты-пуристы.

Аутентификация на основе справочной службы Active Directory

Процедуры авторизации и аутентификации в крупных организациях часто реализуются на основе распределенной справочной службы (см. главу . Это делает их надежными, производительными и удобными в использовании и управлении.

Как было отмечено выше, Windows обладает модульной системой аутентификации, при этом доменную аутентификацию обеспечивают два типа модулей - Kerberos и MSV_1.

Модуль MSV_1 поддерживает протокол аутентификации NTLM (версии 1 и 2), который использовался в доменах Windows NT и считается теперь устаревшим и используемым только для обратной совместимости с доменами, построенными на контроллерах или рабочих станциях Windows NT. Мы не будем рассматривать этот способ доменной аутентификации подробно, отметим только, что он использует так называемую «транзитную» (pass-through) аутентификацию, когда ресурсный сервер обменивается с аутентифицируемым пользователем словом-вызовом, но передает ответ пользователя для проверки аутентичности контроллеру домена, так как только контроллер домена знает пароли пользователей. Этот способ вносит задержки в процесс аутентификации, так как каждое обращение клиента к ресурсному серверу требует обращения к контроллеру домена. Механизм аутентификации Kerberos свободен от этого недостатка, так как квитанция на доступ к ресурсному серверу может использоваться многократно.

Kerberos- аутентификация в пределах одного домена не требует больших пояснений, так как ее схема в точности совпадает с описанной выше в разделе «Система Kerberos». Kerberos KDC в этом случае работает на контроллере домена, там же располагается база учетных данных пользователей и ключи ресурсных серверов. При логическом входе пользователя в домен происходит интерактивная аутентификация, при этом в качестве ресурсного сервера выступает компьютер пользователя, который

должен быть членом домена. При успешной аутентификации пользователь получает доступ к своему компьютеру как пользователь домена, а также как член одной или нескольких групп домена – при условии, что данному пользователь и/или группам, в которые он входит, дано право логического входа в данный компьютер.

Сетевая (неинтерактивная аутентификация) при доступе пользователя к ресурсному серверу, отличному от компьютера, на котором он интерактивно работает, также происходит в точности со схемой, описанной в разделе «Система Kerberos».

В случаях, когда компьютер пользователя или ресурсный сервер находятся в домене, отличном от того домена, где был создана учетная запись пользователя, схема Kerberos-аутентификации немного усложняется. Как мы помним, аутентификационная информация пользователя – пароли – никогда не копируется в копии глобального каталога, а всегда хранится только в базе пользовательских данных того домена, к которому пользователь относится. Поэтому Kerberos-серверы различных доменов должны взаимодействовать в том случае, когда пользователь и ресурс находятся в разных доменах.

В качестве примера мы рассмотрим простой случай, когда имеется три домена, входящие в одно дерево. Эти домены показаны на рис. 16.12: корнем дерева является домен abc.ru, а листьями – домены moscow.abc.ru и orel.abs.ru. Между доменами существуют доверительные отношения «родитель-потомок», установленные автоматически при создании доменов-потомков. Важным фактом для рассмотрения междоменной Kerberos-аутентификации является то, что при установлении доверительных отношений создается разделяемый междоменный ключ, который хранится в базе Kerberos KDC каждого домена.

Рассмотрим отдельно случаи интерактивной и неинтерактивной многодоменной аутентификации.

Интерактивная аутентификация

Боб является пользователем домена moscow.abc.ru, о чем говорит его имя bob@moscow.abc.ru. В рассматриваемом случае Боб выполняет логический вход с компьютера `dell12.orel.abc.ru`, являющегося членом домена `orel.abc.ru`, то есть чужого домена (см. рис.5.14).

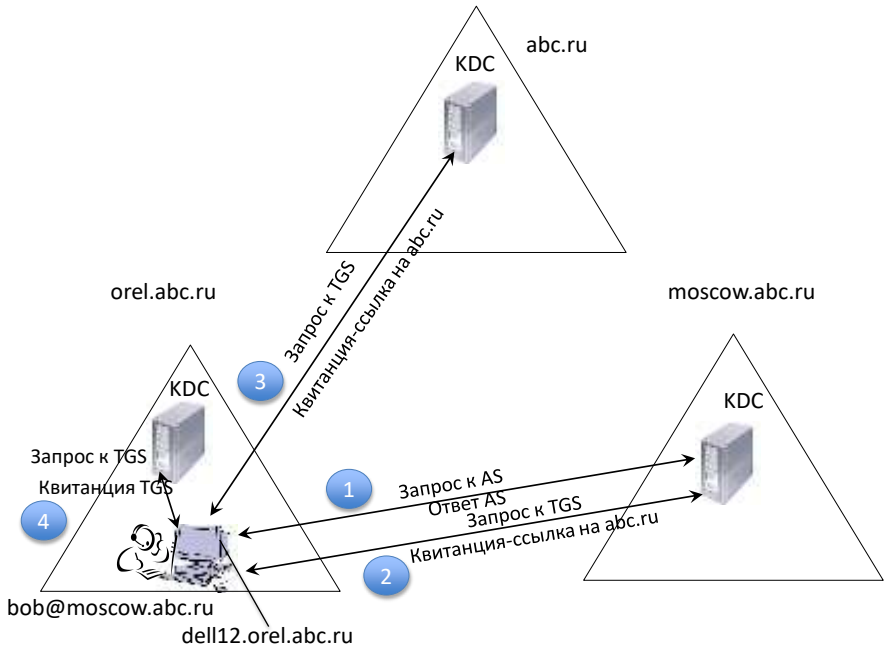

Рис. 5.14. Интерактивная многодоменная Kerberos-аутентификация

После ввода имени bob@moscow.abc.ru модуль Kerberos-аутентификации компьютера `dell12.orel.abc.ru` отправляет запрос на аутентификацию AS серверу домена `moscow.abs.ru`, так как только этот сервер хранит пароли пользователей этого домена `moscow.abc.ru`. Получив этот запрос, сервер AS направляет обычный ответ, в котором содержатся квитанция на доступ к серверу квитанций TGS и ключ сеанса, зашифрованные паролем пользователя.

Получив ответ AS, модуль Kerberos компьютера `dell12.orel.abc.ru` расшифровывает квитанцию и ключ сеанса, используя пароль, введенный Бобом.

Второй этап состоит в отправке компьютером `dell12.orel.abc.ru` запроса к серверу квитанций домена `moscow.abc.ru` за квитанцией доступа Боба к компьютеру `dell12.orel.abc.ru`. Сервер TGS домена `moscow.abc.ru` не может выдать такую квитанцию, так как ресурс находится не в его домене. Вместо этого он отправляет так называемую квитанцию-ссылку, которая указывает на домен `abc.ru` как на ближайший домен, с которым у домена `orel.abc.ru`, которому принадлежит ресурс, есть непосредственные доверительные отношения (а не транзитные). Квитанция-ссылка шифруется междоменным ключом, разделяемым доменами `abc.ru` и `moscow.abc.ru`.

Третий этап состоит в обращении компьютера `dell12.orel.abc.ru` с квитанцией-ссылкой к серверу TGS домена `abc.ru`. Этот сервер расшифровывает квитанцию-ссылку междоменным ключом и генерирует квитанцию доступа к компьютеру `dell12.orel.abc.ru`, зашифрованную ключом этого компьютера, вместе с новым ключом сеанса, а затем зашифровывает эту квитанцию старым ключом сеанса пользователя.

Четвертый этап является обычным для однодоменной схемы работы Kerberos – модуль Keberos-аутентификации пользователя расшифровывает квитанцию доступа старым ключом сеанса и передает ее вместе с аутентификатором пользователя, зашифрованным новым ключом сеанса, модулю Kerberos-аутентификации ресурсного сервера, в качестве которого выступает Kerberos-модуль компьютера `dell12.orel.abc.ru`. Последний расшифровывает квитанцию ключом, разделяемым с сервером

квитанций TGS, и извлекает из нее новый ключ сеанса, с помощью которого расшифровывает аутентификатор.

Неинтерактивная аутентификация

Этот случай иллюстрируется рис. 5.15, на котором представлены те же три домена, что мы рассматривали в предыдущем разделе, но на этот раз пользователь Алиса из домена `orel.abc.ru` работает за компьютером `com33.orel.abc.ru`, который также принадлежит этому домену.

Первый этап. Алиса уже аутентифицировалась для работы с этим компьютером (этот этап мы опускаем) и получила квитанцию на доступ к серверу TGS. Теперь ей нужен доступ к почтовому серверу `mail.moscow.abc.ru`, который находится в домене `moscow.abc.ru`. Сначала Алиса обращается за квитанцией доступа к почтовому серверу к серверу квитанций своего домена. Однако этот сервер не может ей выдать искомую квитанцию, так как ресурс находится за пределами его пространства имен. Поэтому он возвращает квитанцию-ссылку на сервер `abc.ru`, с которым у домена, которому принадлежит ресурс, есть прямые доверительные отношения.

Второй этап. Сервер TGS домена `abc.ru` возвращает квитанцию-ссылку на сервер TGS домена `moscow.abc.ru`, которому принадлежит ресурс `mail.moscow.abc.ru`.

Третий этап. Модуль Kerberos компьютера `com33.orel.abc.ru` отправляет запрос серверу TGS домена `moscow.abc.ru` на доступ к серверу `mail.moscow.abc.ru`. Сервер TGS может обслужить этот запрос и отправляет квитанцию на доступ.

Четвертый этап. Модуль Kerberos компьютера `com33.orel.abc.ru` отправляет серверу `mail.moscow.abc.ru` квитанцию доступа вместе с аутентификатором Алисы. Сервер `mail.msocow.abc.ru` проверяет аутентичность Алисы обычным для Kerberos способом.

При описании неинтерактивной аутентификации мы опустили подробности использования ключей, но читатель может дополнить этот пробел сам, пользуясь схемой интерактивного доступа.

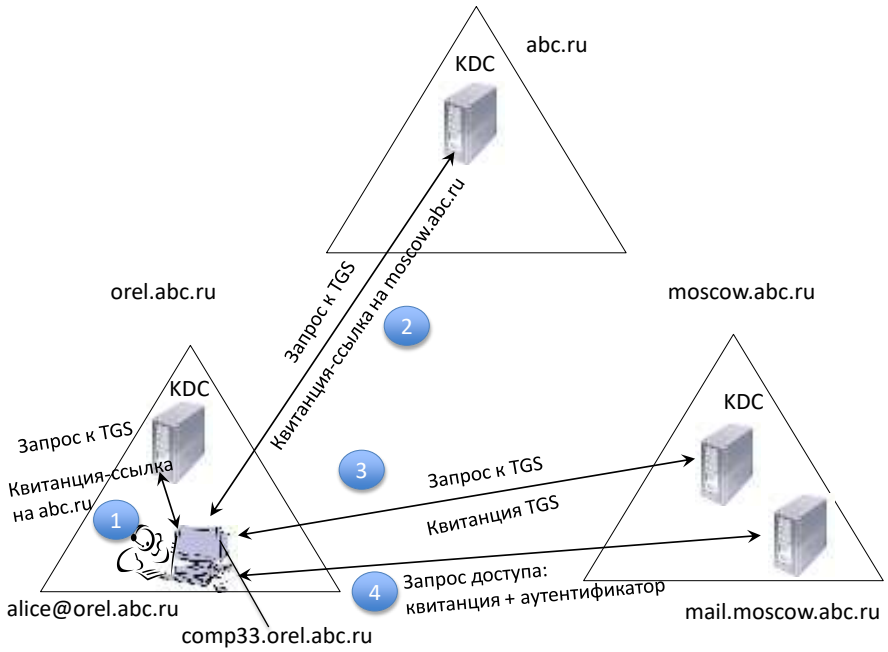

Рис. 5.15. Неинтерактивная многодоменная Kerberos-аутентификация

Реализация Kerberos в системе Active Directory использует фирменное расширение Micorsoft этого протокола, помогающее авторизации пользователей ресурсными серверами. Для этого квитанция на доступ к ресурсу включает ***сертификат привилегий пользователя (Privilege Attribute Certificate, PAC)***, который содержит идентификаторы групп, которым принадлежит пользователь, а также права пользователя.

Контроль доступа в сетевых файловых системах (ФС)

Например, при обращении пользователя к локальным файлам проверка его прав доступа (то есть авторизация) происходит на том же компьютере, на котором была выполнена проверка его легальности (то есть аутентификация). А при работе в сети возможна ситуация, когда аутентификация выполняется на одном компьютере, а авторизация на другом. Пространственное разнесение этих двух процедур на клиентский и серверный компьютеры создает дополнительные уязвимые этапы в общем процессе обработки данных, которые должны каким-то образом учитываться протоколом взаимодействия клиентов и серверов файловой службы. Мы еще вернемся к обстоятельному обсуждению этих вопросов в следующей главе, а сейчас коротко рассмотрим варианты возможного взаимодействия сетевой и локальной файловых систем при реализации контроля доступа к удаленным файлам и каталогам.

Во многих ФС с каждым разделяемым файлом связывается **список управления доступом** (Access Control List, ACL), обеспечивающий защиту данных от несанкционированного доступа по сети. В том случае, когда локальная файловая система поддерживает механизм ACL для файлов и каталогов при локальном доступе, сетевая файловая система использует этот механизм и при доступе по сети. Если же механизм защиты в локальной файловой системе отсутствует, то сетевой файловой системе приходится поддерживать его самостоятельно, иногда — упрощенным способом, защищая разделяемый каталог и входящие в него файлы и подкаталоги как единое целое.

Рассмотрим, например, как решила задачу контроля сетевого доступа к файлам компания Novell в ОС NetWare. Как уже было сказано, компания NetWare вынуждена была разработать собственную версию локальной файловой системы, так как наиболее популярная файловая система для персональных компьютеров тех лет — FAT — не хранила в своих служебных структурах данных о правах пользователей. В то же время протокол взаимодействия с локальной файловой системой NetWare, названный NCP, сохранил привычный для приложений персональных компьютеров интерфейс доступа к FAT, расширив его передачей атрибутов, необходимых для удаленной проверки прав доступа. Протокол NCP является хорошим примером зависимости между свойствами интерфейса, предоставляемого приложениям на клиентских машинах, и свойствами локальной ФС сервера.

Совершенно другой прием для разграничения прав доступа для удаленных пользователей был применен в файловых системах компаний Microsoft и IBM, построенных на основе протокола SMB. В качестве локальной ФС на сервере была оставлена неизмененная файловая система FAT, но сами серверы стали хранить в ней дополнительные служебные файлы с указанием прав пользователей на доступ к разделяемым каталогам. Эти права проверялись сервером при поступлении запроса из сети, локальные же запросы обслуживались в FAT по-прежнему без проверки прав доступа. Естественно, средства защиты каталогов нашли отражение в командах и ответах протокола SMB, а в качестве сетевого интерфейса на стороне клиентов стал использоваться расширенный интерфейс FAT. Позже протокол SMB был применен и для доступа к локальным файловым системам HPFS и NTFS.

В ОС семейства Windows NT существует два механизма защиты — на уровне разделяемых каталогов и на уровне локальных каталогов и файлов. Последний работает только в том случае, когда в качестве локальной файловой системы используется система NTFS, поддерживающая механизм ACL. Механизм

защиты разделяемых каталогов нужен для того, чтобы защищать данные, хранящиеся в локальной файловой системе FAT, не имеющей механизмов защиты. В том случае, когда работают оба уровня защиты, у пользователей и администраторов могут иногда возникать некоторые логические сложности, связанные с определением реальных прав доступа как комбинации нескольких правил.

В современных ОС контроль доступа ко всем типам разделяемых ресурсов — файлам, каталогам, принтерам, секциям памяти и др. — осуществляется с единых позиций. Основную роль в решении этой задачи играет сетевая справочная служба, которая рассматривается в этой главе далее.