Как использовать SSH-ключи для авторизации

Огромное количество облачных приложений построены на популярном протоколе SSH — он широко используется для управления сетевой инфраструктурой, передачи файлов и выполнению удаленных команд.

Название расшифровывается как Secure Socket Shell или *Безопасная оболочка сокета*. Это означает, что протокол предоставляет оболочку (интерфейс командной строки) вокруг соединения между несколькими удаленными хостами, которая является безопасной (зашифрованной и аутентифицированной).

SSH-соединение доступно во всех популярных операционных системах: Linux, Ubuntu, Windows, Debian и так далее. Благодаря закрытому и открытому ключу протокол устанавливает зашифрованный канал связи внутри незащищенной сети.

Ключи — основа SSH

Начнем с того, что протокол работает по клиент-серверной модели. Это значит, что на стороне пользователя есть SSH клиент (терминал в Linux или оконное приложение в Windows), а на стороне сервера активен так называемый *демон* (*daemon*), который принимает подключения от клиентов.

На практике общение по каналу SSH представляет собой управление терминалом удаленного сервера. Иными словами, после успешного подключения все, что вводится в консоль на локальном компьютере, тут же выполняется на удаленном сервере.

В наборе SSH есть два базовых ключа, необходимые для шифровки и дешифровки информации:

- открытый
- закрытый

Ключи связаны между собой некой математической информацией. Открытый ключ является публичным (можно передавать кому угодно), расположен на сервере и нужен для шифровки

данных. Закрытый ключ непубличный (держится в тайне), размещен на клиенте и предназначен, соответственно, для дешифровки данных.

Разумеется, ключи генерируются не вручную, а с помощью специальных программ— *keygen*-ов. Такие утилиты формируют новые ключи, задействуя основные алгоритмы шифрования, применяемые в технологии SSH.

Подробнее о работе SSH

Обмен публичными ключами

SSH основывается на так называемом симметричном шифровании. Два хоста, желающих вести защищенный обмен сообщениями друг с другом, формируют уникальный сеансовый ключ, криптографически основанный на публичных и непубличных данных каждого из хостов.

Например, хост *A* создает открытый и закрытый ключи. Открытый ключ отправляется хосту *B*. Хост *B* проделывает то же самое — свой открытый ключ он посылает хосту *A*.

Благодаря алгоритму Диффи-Хеллмана можно создать ключ из закрытого ключа хоста *A* и публичного ключа хоста *B*. Полученный новый ключ будет идентичен другому ключу, который генерируется на основе публичного ключа хоста *A* и закрытого ключа хоста *B*.

Таким образом оба хоста независимо друг от друга получают симметричный (одинаковый) ключ шифрования, после чего начинается защищенный обмен сообщениями. Отсюда и само название — *симметричное шифрование*.

Верификация сообщений

Для верификации сообщений хосты используют хеш-функцию, которая выдает строку фиксированного размера на основе следующих данных:

- Симметричного ключа.
- Номера пакета.
- Текста зашифрованного сообщения.

Результат хеширования этих данных называется HMAC — *код аутентификации сообщения на основе хэша*. Клиент создает свой HMAC и отправляет его серверу. Сервер получает HMAC клиента, создает свой HMAC (на основе имеющихся у него данных) и сравнивает их. Если они совпадают, значит верификация прошла успешно — никто никого не обманул.

Аутентификация хостов

Защищенное соединение между хостами — лишь полдела. Необходимо также выполнить аутентификацию пользователя, который подключается к удаленному хосту — вполне возможно, у него нет прав доступа для выполнения команд.

Вариантов много.

Например, пользователь может отправить на сервер некий пароль в зашифрованном виде. Если он верный, сервер будет «откликаться» на команды клиента.

Другой способ — сертификат. Пользователь первично отправляет серверу пароль и открытую часть сертификата, после чего взаимодействие между хостами функционирует без регулярного ввода паролей.

Алгоритмы шифрования

Ключевой элемент стойкости SSH — расшифровка симметричного ключа возможна только за счет закрытого ключа, а не открытого. Даже несмотря на то, что симметричный ключ состоит из обоих.

Для получения такого свойства необходим соответствующий алгоритм.

Существует три класса таких алгоритмов: RSA, DSA и алгоритмы на основе эллиптических кривых. У каждого есть свои особенности.

- RSA. Алгоритм, основанный в 1978 году на так называемой *целочисленной факторизации*. Учитывая, что разложение на множители больших полупростых чисел само по себе довольно сложно, существует зависимость между размером выбранных множителей и временем, необходимым для вычисления решения. Длина ключа варьируется от 1024 до 16384 бит.
- DSA. Алгоритм основан на дискретном логарифмировании и модульном возведении в степень. DSA похож на RSA, однако несколько иначе математически связывает открытый и закрытый ключи между собой. Длина ключа не превышает 1024 бит.
- ECDSA и EdDSA. Эти алгоритмы используют эллиптические кривые (в отличие от DSA, который основан на модульном возведении в степень), предполагая, что не существует эффективного решения дискретной логарифмической задачи. Ключи этих алгоритмов короче, но при этом уровень безопасности не хуже.

Так или иначе, подробное объяснение математических основ работы алгоритмов SSH останется за пределами этой статьи — такая информация в практике прикладной разработки используется довольно редко.

Генерация ключей

В каждой операционной системе есть свои утилиты, позволяющие оперативно сгенерировать SSH-ключ.

Например, в UNIX-подобных системах для получения пары ключей есть соответствующая команда — ssh-keygen:

```
ssh-keygen -t rsa
```

Обратите внимание, что тип (тот самый алгоритм шифрования) указывается после специального флага `-t`. Другие типы называются так: `dsa`, `ecdsa`, `ed25519`.

Аналогично можно конкретизировать длину ключа с помощью флага `-b`. Однако, с этим стоит быть осторожным — от длины ключа зависит степень безопасности будущего соединения.

```
ssh-keygen -b 2048 -t rsa
```

После ввода команды терминал выдаст предложение ввести полный путь и названия файлов, в которых сохранятся генерируемые ключи. Однако, вы можете принять указанный путь по умолчанию, нажав *Enter* — тогда будут установлены стандартные названия `id_rsa` (секретный) и `id_rsa.pub` (открытый).

Иными словами, открытый ключ окажется в файле `название.pub`, а закрытый в файле `название` без расширения.

Далее команда предложит ввести парольную фразу. И хотя она не является обязательной (она не имеет никакого отношения непосредственно к SSH-протоколу), рекомендуется ее указать для защиты от несанкционированного использования ключа внутри системы Linux неким сторонним пользователем. Во втором случае потребуется постоянно вводить пароль при установлении соединения.

Кстати, пароль можно в любой момент изменить:

```
ssh-keygen -p
```

Или можно даже указать все предварительные параметры целиком одной командой:

```
ssh-keygen -p password_old -N password_new -f path_to_files
```

Что касается Windows, есть два пути.

В первом используется аналогичная команда `ssh-keygen` из OpenSSH клиента, повторяющая ту же самую последовательность действий из Linux.

Во втором используется оконное приложение PuTTY, где с помощью нажатия одной кнопки получается открытый и закрытый ключ.

Установка клиентской и серверной части

Для SSH-соединения на платформе Linux (как на клиенте, так и на сервере) есть соответствующий инструмент под названием OpenSSH. И хотя, как правило, он уже установлен в операционной системе, в некоторых случаях его требуется установить самостоятельно (например, в Ubuntu).

Как правило, для установки SSH используется следующая универсальная команда с последующим введение пароля суперпользователя:

```
sudo apt-get install ssh
```

Тем не менее во многих операционных системах SSH иногда поделен на отдельные компоненты как для клиента, так и для сервера.

Для клиента

Чтобы узнать, есть ли клиент SSH на вашей локальной машине, просто введите `ssh` в консоль:

```
ssh
```

Если система поддерживает SSH, то терминал выведен описание этой команды. Если этого не произошло, тогда выполните ручную установку:

```
sudo apt-get install openssh-client
```

При установке также потребуется ввести пароль суперпользователя. После чего SSH-подключение станет доступно.

Для сервера

Аналогично клиенту, на сервере необходима серверная часть программного инструментария OpenSSH.

Чтобы проверить, доступен ли сервер SSH на используемом вами удаленном хосте, можно попытаться *локально* подключиться через `ssh`:

```
ssh localhost
```

Если SSH-демон действительно включен, то выйдет сообщение об успешном подключении. Если нет — демон нужно установить:

```
sudo apt-get install openssh-server
```

Как и на клиенте, терминал потребует ввести пароль суперпользователя. После установки можно снова проверить активность SSH. Кстати, помимо подключения к `localhost` есть и другой способ проверки:

```
sudo service ssh status
```

Кстати, после подключения можно произвольно менять настройки SSH. Для этого потребуется отредактировать конфигурационный файл `./ssh/sshd_config`.

Например, можно поменять стандартный порт на свой. Только не забудьте, что для применения новой конфигурации службу SSH нужно вручную перезапускать:

```
sudo service ssh restart
```

Копирование SSH-ключа на сервер

Специальная команда копирования

После создания публичного ключа его можно использовать в качестве авторизованного ключа на сервере. Это необходимо, чтобы быстро устанавливать соединение с сервером без регулярного ввода пароля.

Наиболее распространенный способ — использовать специализированную команду `ssh-copy-id`:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub name@server_address
```

В данном случае предполагается, что при генерации ключа использовались пути и имена по умолчанию. Если нет — имя `~/.ssh/id_rsa.pub` нужно просто заменить на свое.

Вместо `client` пишется имя пользователя, которое используется на удаленном сервере, а вместо `server_address` — адрес хоста. При этом команду можно сделать несколько короче,

если пользовательские имена на клиенте и сервере совпадают:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub server_address
```

Если вы предварительно устанавливали секретный пароль на SSH-ключ, то в консоли выведется сообщение, предлагающее его ввести. Если перед этим вы не устанавливали пароль, то сразу выполнится копирование.

Иногда реализация удаленного сервера предполагает SSH-подключение по нестандартному (стандартный — 22) порту. В таком случае его потребуется явно указать с помощью дополнительного флага `-p`:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub -p 8129 name@server_address
```

Полуручное копирования

Есть операционные системы, в которых по тем или иным причинам команда `ssh-copy-id` может не поддерживаться вообще, хотя подключение по SSH к серверу возможно. В таком случае операция копирования разбивается на несколько связанных между собой команд.

```
ssh name@server_address 'mkdir -m 700 ~/.ssh; echo ' $(< ~/.ssh/id_rsa.pub) ' >> ~/.ssh/authoriz
```

Эта последовательность команд создаст специальную папку протокола на сервере (в случае, если отсутствует) с соответствующими разрешениями (700) на запись и чтение. Далее создастся файл `authorized_keys` — в нем хранятся публичные ключи всех разрешенных пользователей. Поэтому именно в него будет записана информация из локального файла (со стандартным названием) с открытым ключом клиента. Если `authorized_keys` уже есть на удаленном сервере, то он будет просто дополнен. По итогу эта команда дополнительно устанавливает соответствующее разрешение на файл `authorized_keys` внутри системы удаленного хоста.

В дальнейшем соединение с сервером происходит с помощью той же команды, но уже с проверкой пользователя по открытому ключу, который был добавлен в `authorized_keys`.

```
ssh name@server_address
```

Ручное копирование

Некоторые хост-платформы предлагают управление сервером через другие каналы связи. Например, через пользовательский веб-интерфейс в личном кабинете. В таком случае, в них должен присутствовать функционал для добавления публичного ключа на сервер. Иногда в веб-интерфейсе имитируется терминал серверной консоли.

Так или иначе, удаленный хост обязательно будет содержать соответствующий файл `~/.ssh/authorized_keys` — внутри него содержатся все разрешенные открытые ключи пользователей.

Логично, что именно в этот файл нужно скопировать открытый ключ клиента, размещенный в `~/.ssh/id_rsa.pub` (при стандартном названии).

В том случае, когда генерация ключей выполнялась в оконном приложении (обычно это PuTTY в Windows), открытый ключ копируется из него напрямую — то есть скопированный текст просто добавляется к тексту, который уже размещен в файле `authorized_keys`.

Подключение к серверу

Для подключения к удаленному серверу в операционной системе Linux нужно ввести в консоль:

```
ssh name@server_address
```

Либо, если наименование локального пользователя идентично наименованию удаленного:

```
ssh server_address
```

Далее система попросит вас ввести пароль. При этом ввод пароля не будет графически отображаться в терминале — его нужно просто набрать на клавиатуре и нажать *Enter*.

Как и в случае с командой `ssh-copy-id` при подключении к удаленному серверу можно явно указать порт:

```
ssh client@server_address -p 8129
```

Теперь вам доступен контроль над удаленной машиной через терминал — все вводимые команды будут выполняться на стороне сервера.

Заключение

Сегодня SSH — один из самых популярных протоколов в разработке и системном администрировании. Именно поэтому так важно иметь базовое представление о его функционировании.

Задача данной статьи — в общих чертах описать устройство SSH-соединения, коротко рассказать об алгоритмах шифрования (RSA, DSA, ECDSA и EdDSA) и продемонстрировать, как через применение публичного и секретного ключей можно устанавливать безопасное соединение с собственным сервером, обмениваясь сообщениями, которые будут оставаться

недоступны третьим лицам.

Были показаны основные команды UNIX-подобных операционных систем, с помощью которых можно генерировать оба ключа и давать клиенту разрешение на SSH-доступ с помощью копирования открытого ключа на сервер с последующим подключением.

Revision #1
Created 2023-10-24 17:29:14 UTC by odiljonov
Updated 2023-11-25 17:58:18 UTC by odiljonov