Удаленный доступ SSH

SSH (англ. *Secure SHell* — «безопасная оболочка») – протокол передачи данных, позволяющий производить безопасное и защищенное управление операционной системой и данными. Это сетевой протокол прикладного уровня, который дает возможность шифрования передаваемых данных и паролей. К тому же позволяет передавать любой другой протокол. Эта служба была создана в качестве замены не зашифрованному Telnet и использует криптографические техники, чтобы обеспечить, что всё сообщение между сервером и пользователем было зашифровано. SSH предоставляет механизм для авторизации удаленного пользователя, передавая команды клиента хосту и возвращая ответ обратно клиенту.

Возможности SSH

- Позволяет удаленно работать на компьютере через командную оболочку.
- Позволяет осуществлять шифрование с помощью различных алгоритмов.
- Так как SSH позволяет безопасно передавать практически любой сетевой протокол, это позволяет передавать по шифрованному каналу звуковые и видео файлы.
- Производит сжатие файлов для их последующего шифрования и передачи.
- Защищает передачу данных по каналу и предотвращает возможность включения в установленную сессию и перехватить данные.

Необходимое ПО для работы с SSH

Для работы с SSH необходим *SSH-сервер* и *SSH-клиент*.

SSH-сервер принимает соединение от клиентских машин и производит аутентификацию. Аутентификация на SSH производится тремя способами:

- По IP адресу клиента – при этом SSH использует несколько методов проверки. Способ не очень безопасный, так как существует возможность подмены IP адреса.
- По публичному ключу клиента – схема почти такая же, как при проверке IP адреса клиентской машины, только в данном случае проверяется ключ клиента и имя пользователя.
- По паролю клиента – часто используемый метод проверки. Пароль в данном случае передается также в зашифрованном виде.

Основными программными платформами, выступающими в роли **SSH-сервера** являются:

- BSD: OpenSSH
- Linux: dropbear, lsh-server, openssh-server, ssh
- Windows: freeSSHd, copssh, WinSSHD, КруМ Telnet/SSH Server, MobaSSH, OpenSSH

SSH-клиент используется для непосредственного входа на удаленный сервер и выполнения различных команд:

- Работа с файлами и директориями
- Работа по просмотру или редактированию файлов
- Отслеживание процессов работы
- Работа с архивами
- Работа с базами данных MySQL

SSH-клиенты и их программные оболочки:

- GNU/Linux, BSD: kdessh, lsh-client, openssh-client, putty, ssh, Vinagre
- MS Windows и Windows NT: PuTTY, SecureCRT, ShellGuard, Axessh, ZOC, SSHWindows, ProSSHD, XShell
- MS Windows Mobile: PocketPuTTy, mToken, sshCE, PocketTTY, OpenSSH, PocketConsole
- Mac OS: NiftyTelnet SSH
- Java: MindTerm, AppGate Security Server
- iPhone: i-SSH, ssh (в комплекте с Terminal)
- Android: connectBot

Главным преимуществом SSH над его предшественниками является использование шифрования для защиты передачи данных между хостом и клиентом. Хост – это удаленный сервер к которому необходимо получить доступ, тогда как клиент – это компьютер с которого вы пытаетесь получить доступ к хосту. Существует три различных технологий шифрования, используемых SSH:

- Симметричное шифрование
- Асимметричное шифрование
- Хеширование

Симметричное шифрование

Симметричное шифрование – это форма шифрования, где **секретный ключ** используется для шифрования и дешифровки сообщения как клиентом, так и хостом. Стоит отметить, что любой клиент имеющий ключ, может дешифровать передаваемое сообщение.

В симметричном шифровании обычно используется один ключ или пара ключей, где один ключ может быть легко вычислен с помощью другого.

Симметричные ключи используются для шифрования всего сообщения в течение SSH сессии. И клиент, и сервер получают ключ согласованным методом, и данный ключ никогда не разглашается третьим лицам. Процесс создания симметричного ключа осуществляется при помощи **алгоритма обмена ключами**. Что делает этот способ шифрования в некотором смысле безопасным, так это то, что ключ никогда не передаётся от клиента к хосту. Наоборот, оба компьютера делят части публичной информации и используют её для вычисления ключа. Даже если другая машина сможет перехватить информацию, она не сможет вычислить ключ, потому что алгоритм обмена ключом будет неизвестен.

Для каждой SSH сессии создается свой секретный токен (ключ), который генерируется до авторизации клиента. Как только ключ был сгенерирован, все пакеты должны быть зашифрованы приватным ключом. Это включает в себя пароль вписанный пользователем в консоль, таким образом личные данные всегда защищены от анализаторов трафика (снифферов).

Существует разные шифры симметричного шифрования, включая AES (Advanced Encryption Standard), CAST128, Blowfish и т.д. Перед установкой защищённого соединения, клиет и хост решают какой из шифров использовать, создавая список поддерживаемых шифров в порядке их предпочтительности. Самый предпочтительный шифр из списка клиента, который присутствует в списке хоста будет использоваться в качестве двунаправленного шифра.

К примеру, если две машины Ubuntu 14.04 LTS сообщаются друг с другом с помощью SSH, то они будут использовать **aes128-ctr** в качестве шифра по умолчанию.

Асимметричное шифрование

В отличии от симметричного шифрования, асимметричное использует два отдельных ключа для шифрования и дешифровки. Эти два ключа также известны как **приватный** и **публичный** ключи. Вместе они формируют **пару публичных-приватных ключей**.

Публичный ключ свободно распространяется между всеми группами. Однако, приватный ключ не может быть математически вычислен с помощью публичного. Сообщение, которое зашифровано публичным ключом машины может быть расшифровано лишь приватным ключом той же машины. Эти односторонние отношения означают, что публичный ключ не может дешифровать свои же сообщения, также как не может расшифровать ничего зашифрованного приватным ключом.

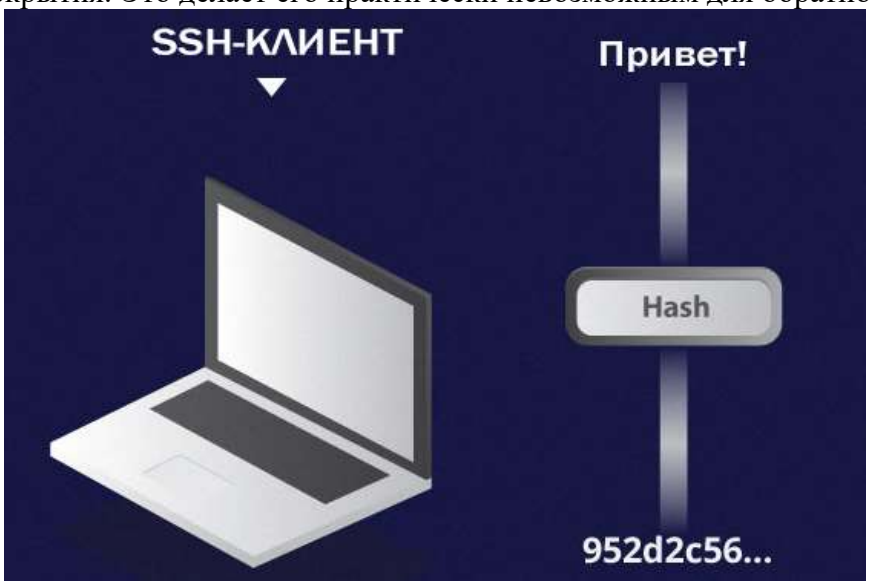
Приватный ключ должен оставаться приватным для сохранения защищённости соединения, никакие третьи лица не должны его знать. Надёжность всего этого зашифрованного соединения заключается в том, что приватный ключ никогда не показывается, так как это единственный компонент, который способен расшифровать сообщения зашифрованные публичным ключом. Поэтому любая группа имеющая возможность расшифровать сообщение в таком типе шифрования должна обладать соответствующим приватным ключом.

Асимметричное шифрование не используется для шифрования всей SSH сессии. Вместо этого оно используется только в процессе алгоритма обмена ключами симметричного шифрования. Перед налаживанием защищённого соединения, обе группы генерируют временные пары публично-приватных ключей и делятся приватным ключами для создания общего секретного ключа.

После того как симметричное соединение будет установлено, сервер использует публичный ключ и передаёт его клиенту для авторизации. Если клиент может успешно расшифровать сообщение значит это означает, что он обладает приватным ключом, который необходим для подключения. После этого начинается SSH сессия.

Хеширование

Одностороннее хеширование – это еще одна форма криптографии, которая используется в SSH. Хеширование не предназначено для дешифровки. Оно создает уникальное значение фиксированной длины для каждого ввода, которое не показывает никакого общего поведения для его раскрытия. Это делает его практически невозможным для обратного преобразования.



Создать криптографический хеш из заданного значения довольно легко, но получить значение из хеша невозможно. Это означает, что если клиент обладает нужным значением, он

может генерировать криптографический хеш и сравнить его значение со значением сервера, чтобы проверить совпадают ли они.

Исходные данные называются входным массивом, «ключом» или «сообщением». Результат преобразования (выходные данные) называется «хешем», «хеш-кодом», «хеш-суммой», «сводкой сообщения».

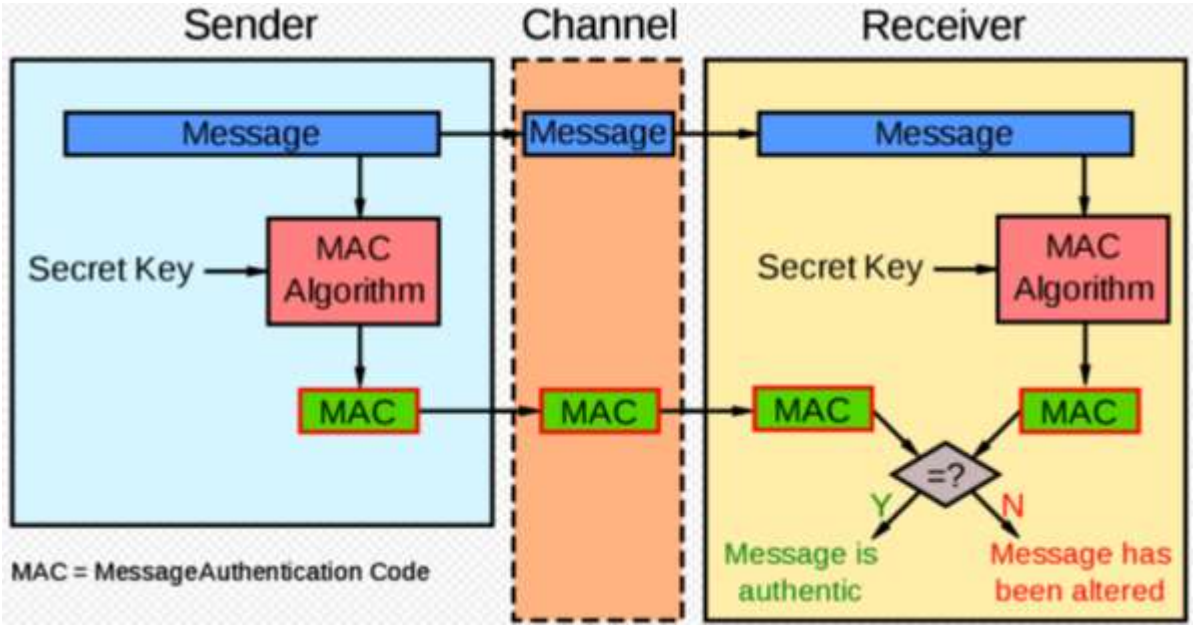
При изменении исходного текста даже на один знак результат хеш-функции полностью меняется.

- Это свойство хеш-функций позволяет применять их в следующих случаях:
- при построении ассоциативных массивов;
- при поиске дубликатов в сериях наборов данных;
- при построении уникальных идентификаторов для наборов данных;
- при вычислении контрольных сумм от данных (сигнала) для последующего обнаружения в них ошибок (возникших случайно или внесённых намеренно), возникающих при хранении и/или передаче данных;
- при сохранении паролей *в системах защиты* в виде хеш-кода (для восстановления пароля по хеш-коду требуется функция, являющаяся обратной по отношению к использованной хеш-функции);
- при выработке электронной подписи (на практике часто подписывается не само сообщение, а его «хеш-образ»);
- и др.

SSH использует хеши для подтверждения сообщений об аутентификации. Это делается при помощи HMACs или **H**ash-based **M**essage **A**uthentication **C**odes (Кода Аутентификации Сообщений, Использующего Хеш-функции). Это обеспечивает подлинность полученной команды.

Если выбран необходимый алгоритм симметричного шифрования, это означает, что также выбран подходящий алгоритм аутентификации сообщений.

Каждое переданное сообщение должно содержать **MAC**, который рассчитывается с помощью симметричного ключа, порядкового номера пакета и содержимого сообщения. Оно отправляется извне симметрично зашифрованных данных в качестве итогового раздела пакета связи.



Имитовста́вка (MAC, англ. *message authentication code* — код аутентификации сообщения) — средство обеспечения имитозащиты в протоколах аутентификациисообщений с доверяющими друг другу участниками — специальный набор символов, который добавляется к сообщению и предназначен для обеспечения его целостности и аутентификации источника данных.

Имитовста́вка обычно применяется для обеспечения целостности и защиты от подделки передаваемой информации.

Для проверки целостности (но не аутентичности) сообщения на отправляющей стороне к сообщению добавляется значение хеш-функции от этого сообщения, на приемной стороне также вырабатывается хеш от полученного сообщения. Выработанный на приёмной стороне и полученный хеш сравниваются, если они равны, то считается, что полученное сообщение дошло без изменений.

Для защиты от подделки (имитации) сообщения применяется имитовставка, выработанная с использованием секретного элемента (ключа), известного только

Как работает SSH с техниками шифрования

SSH работает используя модель клиент-сервер для обеспечения аутентификации двух удаленных систем и шифрования данных, которые проходят между ними.

SSH по умолчанию работает через TCP порт 22 (это может быть изменено, если необходимо). Хост (сервер) ожидает на порте 22 (или любом другом порте назначенном SSH) входящие подключения. Далее он организует защищённое соединение, проводя аутентификацию клиента и открывая необходимую оболочку, если проверка успешна.

Клиент должен начать SSH подключение инициируя соединение с сервером через TCP, обеспечив защищенное симметричное соединение, подтвердив отображенный идентификатор на соответствие с предыдущими записями (обычно записанными в RSA файле) и предоставив необходимые личные данные для аутентификации подключения.

Есть две стадии установки подключения: первое, обе системы должны договориться о стандартах шифрования для защиты их будущих соединений, и второе, пользователи должны пройти проверку подлинности. Если данные верны, то пользователь получит доступ.

Процесс выбора шифрования

Когда клиент пытается установить соединение с сервером через TCP, то сервер предоставит протоколы шифрования и их версии, которые он поддерживает. Если клиент имеет такой же подходящий протокол и версию, достигается соглашение и соединение устанавливается с выбранным протоколом. Сервер также использует асимметричные публичные ключи, которые клиент может использовать для проверки аутентификации.

Как только оно установлено, стороны используют алгоритм обмена Диффи—Хеллмана для создания симметричных ключей. Этот алгоритм позволяет и клиенту, и серверу получить общий код шифрования, который они будет использовать для дальнейших сессий подключения.

Вот как работает алгоритм на самом простом уровне:

- И клиент, и сервер генерируют очень большое число - **стартовое значение**.
- Далее, обе стороны договариваются об общем механизме шифрования для генерации другого набора значений
- Обе стороны отдельно друг от друга генерируют другие числа. Они используются в качестве приватных ключей для сообщений.
- Эти только что сгенерированные приватные ключи, с общим числом и алгоритмом шифрования (AES) используются для вычисления публичного ключа, который доставляется на другой компьютер.
- Затем стороны используют их персональные приватные ключи, общий публичный ключ другой машины и изначальное число, чтобы создать финальный общий ключ. Этот ключ будет вычислен обоими компьютерами независимо друг от друга, но он будет одинаковым.
- Теперь, когда обе стороны имеют общий ключ, они могут симметрично шифровать всю SSH сессию. Тот же ключ может быть использован для шифрования и дешифровки сообщений (читайте раздел о симметричном шифровании).

Теперь, когда сессия защищенная симметричным шифрованием была установлена, пользователь должен быть авторизован.

Авторизация пользователя

Финальный этап перед тем как пользователь получит доступ к серверу, это его авторизация. Для этого большинство пользователей используют пароль. Пользователя попросят вписать имя и затем последует пароль. Эти данные будут переданы через защищённый симметричным шифрованием туннель, поэтому они не могут быть перехвачены третьими лицами.

Однако даже если пароли зашифрованы, их не рекомендуют использовать для защищённого соединения. Причиной тому является относительная простота с которой боты могут подобрать пароль к вашему аккаунту. Поэтому рекомендуется использование пары ключей SSH.

Это набор асимметричных ключей, которые используются для авторизации пользователей без введения каких-либо паролей.

Способы аутентификации

Парольная аутентификация

Разрешенная по умолчанию парольная аутентификация является практически самым примитивным способом авторизации в sshd. С одной стороны это упрощает конфигурацию и подключение новых пользователей (пользователю достаточно знать свой системный логин/пароль), с другой стороны пароль всегда можно подобрать, а пользователи часто пренебрегают созданием сложных и длинных паролей. Специальные боты постоянно сканируют доступные из интернета ssh сервера и пытаются авторизоваться на них путем перебора логинов/паролей из своей базы. Настоятельно не рекомендуется использовать парольную аутентификацию.

Аутентификация по ключу

Несмотря на то, что пароль шифруется при передаче, этот метод обычно не рекомендуется использовать из-за ограничений сложности пароля. Автоматизированные сценарии очень легко подбирают пароли обычной длины и могут навредить серверу.

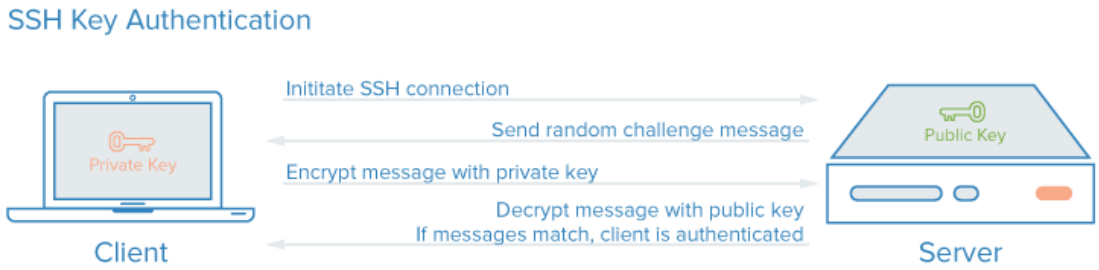
Самой популярной и рекомендуемой альтернативой парольной аутентификации является аутентификация на основе ключей SSH. Пары SSH-ключей являются асимметричными ключами, что означает, что два связанных ключа служат для разных функций.

Закрытый ключ должен храниться в секрете. Если злоумышленник получит закрытый ключ, он сможет войти на серверы, связанные с этой парой ключей, без дополнительной аутентификации. В качестве дополнительной меры предосторожности ключ можно защитить парольной фразой.

Открытый ключ можно свободно распространять. Он позволяет зашифровывать сообщения, которые можно расшифровать только с помощью соответствующего закрытого ключа. Аутентификация по ключам основана на этом принципе.

Открытый ключ загружается на удаленный сервер, на котором нужно настроить аутентификацию по ключам SSH. Ключ добавляется в специальный файл ~/.ssh/authorized_keys в учетной записи пользователя, с помощью которого нужно войти на сервер.

Когда клиент пытается аутентифицироваться с помощью SSH-ключей, сервер может проверить клиента на наличие у закрытого ключа. Если клиент может доказать, что ему принадлежит закрытый ключ, сервер запускает сеанс оболочки или выполняет требуемую команду.



Настройка доступа по SSH

Парольная аутентификация

1. Создать на сервере пользователя, под которым будет осуществляться аутентификация

```
useradd user
passwd user
задать пароль
повторить пароль
```

2. Открыть порт для ssh (по-умолчанию 22) и разрешить парольную аутентификацию

```
nano /etc/ssh/sshd_config
Port 22
PasswordAuthentication yes
```

3. Перезагрузить демон sshd

```
systemctl restart sshd
```

Настройка аутентификации на основе SSH – ключей

1. Сгенерировать пару ключей SSH на локальном компьютере:

```
ssh-keygen
```

По умолчанию создается пара 2048-битных ключей RSA, которые подходят в большинстве случаев. По умолчанию ключи будут храниться в каталоге **~/.ssh** в домашнем каталоге текущего пользователя. Закрытый ключ будет называться **id_rsa**, а связанный с ним открытый ключ – **id_rsa.pub**. В процесе создания будет предложено ввести парольную фразу (для большей безопасности), можно ее не вводить.

2. Добавить открытый ключ на удаленный сервер:

```
ssh-copy-id username@remote_host
```

При первом подключении будет возвращено:

```
Are you sure you want to continue connecting (yes/no)? yes
```

Затем утилита просканирует локальный аккаунт, найдет id_rsa.pub и запросит пароль удаленного пользователя.

3. Введите пароль (при вводе он не будет отображаться из соображений безопасности) и нажмите Enter.

Утилита подключится к учетной записи на удаленном хосте, используя предоставленный пароль. Затем он скопирует содержимое ключа **~/.ssh/id_rsa.pub** в файл **authorized_keys** в домашнем каталоге **~/.ssh** удаленной учетной записи.

Аутентификация через SSH-ключи

Базовая команда выглядит так:

```
ssh username@remote_host
```

Если не настраивалась парольная фраза для закрытого ключа, сразу же будет рсуществлен вход в систему. Если парольная фраза для закрытого ключа есть, необходимо будет ввести ее сейчас. После этого должен быть создан новый сеанс оболочки для удаленного пользователя.

Отключение парольной аутентификации

Однако механизм парольной аутентификации все еще включен, что означает, что сервер по-прежнему уязвим к brute-force атакам.

Необходимо убедится, что у учетной записи root или у пользователя с доступом sudo на удаленном сервере настроена аутентификация на основе ключей SSH. При отключении парольной аутентификации все пароли будут заблокированы, поэтому важно обеспечить себе административный доступ в случае неполадок.

1. Открыть конфигурационный файл демона SSH

```
nano /etc/ssh/sshd_config
PasswordAuthentication no
```

2. Перезагрузить демон sshd

```
systemctl restart sshd
```

Безопасное использование SSH

Необходимо соблюдать несколько правил пользования SSH, чтобы обезопасить свои данные:

- Запретить возможность удалённого root-доступа.
- Запретить подключение с пустым паролем или отключение входа по паролю.
- Необходимо выбирать нестандартный порт для SSH-сервера.
- Использовать длинные SSH2 RSA-ключи.
- Необходимо строго ограничить количество IP-адресов, с которых разрешён доступ.
- Запретить доступа с опасных адресов.
- Регулярно отслеживать сообщения об ошибках аутентификации.
- Установить системы обнаружения вторжений (IDS — Intrusion Detection System).
- Использовать специальные ловушки, подделывающих SSH-сервис (honeypots).