Secure Shell (Русский)

Ссылки по теме

- **Ключи SSH**
- **Pam abl**
- **fail2ban**
- **sshguard**
- **Sshfs**
- **Syslog-ng**
- **SFTP chroot**

Эта страница нуждается в сопроводителе

Статья не гарантирует актуальность информации. Помогите русскоязычному сообществу поддержкой подобных страниц. См. **Команда переводчиков ArchWiki**

Состояние перевода: На этой странице представлен перевод статьи **Secure Shell**. Дата последней синхронизации: 2014-10-18. Вы можете **помочь** синхронизировать перевод, если в английской версии произошли **изменения**.

Secure Shell (SSH) - это сетевой протокол, который позволяет двум компьютерам обмениваться данными по безопасному каналу. Шифрование обеспечивает конфиденциальность и неприкосновенность информации. SSH использует шифрование с открытым ключом для проверки подлинности удаленного компьютера и предоставляет ему аутентификацию пользователя, если это необходимо.

Обычно SSH используется для подключения (входа) к удаленной машине и выполнения команд, но он также поддерживает туннелирование (tunneling), проброс TCP-портов и передачу сеанса X11; передача файлов может быть осуществлена с использованием протоколов SFTP или SCP.

По умолчанию сервер SSH "прослушивает" стандартный порт TCP номер 22. Программа-клиент SSH обычно используется для установления связи с демоном *sshd*, который принимает удаленные подключения. И сервер, и клиент, как правило, присутствуют в современных операционных системах, в том числе в Mac OS X, GNU/Linux, Solaris и OpenVMS. Существуют проприетарные, freeware и open source (с открытым исходным кодом) версии различных уровней сложности и завершенности.

(Источник: **SSH**)

Contents

[hide]

OpenSSH

OpenSSH (OpenBSD Secure Shell) - это набор компьютерных программ, дающих возможность установления шифрованных сессий через компьютерную сеть с использованием протокола ssh. Он был создан как свободная (open source) альтернатива проприетарному программному обеспечению, предлагаемому SSH Communications Security. OpenSSH разрабатывается как часть проекта OpenBSD, возглавляемого Тео де Раадтом (Theo de Raadt).

OpenSSH иногда путают с похожим по звучанию OpenSSL, однако, у этих проектов разные цели, и они разрабатываются разными командами. Похожесть названий связана с использованием слова "Open", которое означает приверженность одним и тем же идеям.

Установка OpenSSH

Установите пакет **`openssh`** из **официальных репозиториев**.

Настройка SSH

Клиент

Файл конфигурации клиента SSH - `/etc/ssh/ssh_config` или `~/.ssh/config`.

Теперь нет необходимости явно указывать `Protocol 2`, по умолчанию он раскомментирован. Это значит, что `Protocol 1` не будет использоваться, пока вы явно не включите его (источник: **http://www.openssh.org/txt/release-5.4**).

Демон

Файл конфигурации демона SSH можно найти и отредактировать в `/etc/ssh/sshd_config`.

Чтобы предоставить доступ только некоторым пользователям, добавьте эту строку:

```
AllowUsers    пользователь1 пользователь2
```

Чтобы предоставить доступ только некоторым группам пользователей:

```
AllowGroups   группа1 группа2
```

Чтобы отключить вход через SSH под суперпользователем (root), измените строку PermitRootLogin:

```
PermitRootLogin no
```

Чтобы добавить приятное приветственное сообщение, отредактируйте файл `/etc/issue` и измените строку Banner:

```
Banner /etc/issue
```

Совет:

- Вы можете захотеть изменить порт, используемый по умолчанию, на другой (смотрите статью **Безопасность через неясность**). Для получения помощи в выборе портов смотрите статью **Список портов TCP и UDP**, также вы можете найти информацию о них на своей машине в `/etc/services`. Выберите желаемый порт, который **не** связан с другими сервисами во избежание конфликтов
- Полное отключение входа с использованием паролей значительно увеличит безопасность. Для получения дополнительной информации смотрите раздел **Ключи SSH#Отключение входа по паролю**

Управление демоном sshd

Демон SSH имеет различные файлы юнитов systemd.

Вы можете запустить демон для немедленного использования и/или включить его в автозагрузку при старте системы, как описано в разделе **Systemd (Русский)#Использование юнитов**:

```
# systemctl start sshd.service
# systemctl enable sshd.service
```

Важно: Systemd является асинхронной системой инициализации. Если вы привяжете демон SSH к определенному IP-адресу `ListenAddress 192.168.1.100`, он может не загрузиться, поскольку файл юнита sshd.service по умолчанию не имеет зависимости от того, включены ли сетевые интерфейсы. Если вам необходима привязка к определенному IP-адресу, добавьте строку `After=network.target` в свой файл юнита sshd.service. Смотрите раздел **Systemd (Русский)#Редактирование предоставленных пакетами файлов юнитов**

В качестве альтернативы демон SSH поддерживает активацию сокета. Это означает, что systemd будет прослушивать сокет SSH и запускать его только тогда, когда будет получено первое входящее подключение:

```
# systemctl start sshd.socket
# systemctl enable sshd.socket
```

Если вы используете отличный от порта по умолчанию (22) порт для сокета, необходимо задать "ListenStream" в файле юнита.
Скопируйте `/lib/systemd/system/sshd.socket`в `/etc/systemd/system/sshd.socket`, чтобы избежать его перезаписи во время обновлений. В файле `/etc/systemd/system/sshd.socket` измените значение строки "ListenStream" на необходимый порт.

Важно: Использование `sshd.socket` фактически сводит на нет настройку `ListenAddress`, так что умолчальный sshd.socket будет принимать подключения с любого адреса. Чтобы получить эффект от настройки `ListenAddress`, необходимо создать файл юнита и отредактировать строку ListenStream (например, `ListenStream=192.168.1.100:22`равнозначен `ListenAddress 192.168.1.100`). Также необходимо добавить `FreeBind=true` в секцию `[Socket]`, иначе настройка IP-адреса будет иметь тот же эффект, что и настройка `ListenAddress`: сокет не запустится, если сеть не поднимется вовремя

Совет: При использовании активации через сокет ни `sshd.socket`, ни обычный `sshd.service` не позволяют отслеживать подключение в логе, а при вызове `# journalctl /usr/bin/sshd` такая возможность есть

Подключение к серверу

Для подключения к серверу выполните:

```
$ ssh -p порт пользователь@адрес_сервера
```

Защита SSH

Предоставление удаленного входа в систему через SSH хорошо подходит для административных задач, но может угрожать безопасности вашего сервера. Часто являясь целью атак полного перебора (brute force), SSH-доступ нуждается в правильном ограничении для защиты от третьих лиц.

- Используйте нестандартные имена аккаунтов и пароли
- Допускайте входящие SSH-подключения только от проверенных машин
- Используйте **fail2ban** или **sshguard** для контроля подобных атак и запрещайте доступ с IP-адресов, которые их проводят

Защита от атак полного перебора (brute force)

"Brute forcing" - довольно простое понятие: кто-либо постоянно пытается авторизоваться на веб-странице или в командной строке сервера путем перебора большого количества комбинаций из имен пользователей и паролей. Вы можете защититься от подобных атак, используя автоматический скрипт, блокирующий их. Например, **fail2ban** или **sshguard**.

Также можно сделать подобные атаки невозможными отключив вход по паролю. Для этого добавьте следующую настройку в файл `sshd_config`:

```
PasswordAuthentication no
```

Перед тем, как добавлять эту настройку, убедитесь, что все учетные записи, которым требуется SSH-доступ с использованием аутентификации по открытому ключу, настроены в соответствующих файлах `authorized_keys`.

Ограничение входа от имени суперпользователя

Предоставление возможности входа через SSH от имени суперпользователя без какой-либо защиты считается плохой практикой. Существует два способа ограничения этой возможности для увеличения безопасности.

Отключение

Sudo выборочно предоставляет права суперпользователя для действий, которым они необходимы, без соответствующего входа в учетную запись root. Благодаря этому можно заблокировать аккаунт root, чтобы отключить возможность входа в него через SSH. Это потенциально является средством защиты от брут-форс (brute force) атак, поскольку в этом случае атакующему придется подбирать еще и имя учетной записи в дополнение к паролю.

Чтобы отключить вход от имени суперпользователя через SSH, получите его права и отредактируйте секцию "Authentication" файла `/etc/ssh/sshd_config`. Просто измените значение `#PermitRootLogin yes` на `no` и раскомментируйте строку:

```
/etc/ssh/sshd_config

PermitRootLogin no
...
```

Перезапустите демон SSH:

```
# systemctl restart sshd
```

Теперь вы не сможете войти в систему через SSH от имени суперпользователя, но по-прежнему будете иметь возможность входить от имени обычного пользователя и использовать команды **su** и **sudo** для администрирования.

Ограничение

Некоторые автоматические задачи, такие как удаленное создание резервной копии системы, требуют полного root-доступа. Чтобы получить безопасную возможность его использования, вместо отключения можно указать конкретные команды. Для этого отредактируйте файл `~root/.ssh/authorized_keys`, создав префиксы для соответствующих ключей, например:

```
command="/usr/lib/rsync/rrsync -ro /" ssh-rsa …
```

Благодаря этому при входе с использованием соответствующего ключа можно будет получить права суперпользователя лишь для выполнения тех команд, которые описаны между кавычками.

При этом остается возможность входа в систему с использованием имени суперпользователя. Это можно исправить, добавив следующую строку в файл `sshd_config`:

```
PermitRootLogin forced-commands-only
```

Эта настройка не только ограничит список команд, которые могут выполняться от имени суперпользователя через SSH, но и отключит использование паролей, оставляя возможность входа в аккаунт root лишь с использованием аутентификации по открытому ключу.

Есть альтернатива, вводящая меньше ограничений, которая позволит выполнять любые команды от имени суперпользователя, но сделает невозможным использование брут-форс (brute-force) атак благодаря отключению входа по паролю. Для этого пропишите:

```
PermitRootLogin without-password
```

Другие клиенты и серверы SSH

Помимо OpenSSH, существует большое количество **клиентов** и **серверов** SSH.

Dropbear

Dropbear - это клиент и сервер SSH-2. Пакет **dropbear** доступен для установки из **AUR**.

Клиент для командной строки называется dbclient.

Альтернатива SSH: Mobile Shell

С **веб-сайта Mosh**:

Remote terminal application that allows roaming, supports intermittent connectivity, and provides intelligent local echo and line editing of user keystrokes. Mosh is a replacement for SSH. It's more robust and responsive, especially over Wi-Fi, cellular, and long-distance links.

Установите пакет **mosh** из **официальных репозиториев** или самую свежую версию **mosh-git**[AUR] из **AUR**.

Советы и рекомендации

Шифрованный туннель SOCKS

Эта опция крайне полезна для пользователей портативных компьютеров (ноутбуков), подключающихся к различным небезопасным беспроводным сетям. Единственное, что вам необходимо, это сервер SSH, запущенный в каком-либо безопасном месте, например, дома или на работе. Может оказаться полезным использование сервиса динамического DNS, такого как **DynDNS**, чтобы не было необходимости знать ваш IP-адрес.

Шаг 1: установка соединения

Для запуска соединения необходимо выполнить лишь эту простую команду:

```
$ ssh -TND 4711 пользователь@хост
```

где `пользователь` - ваше имя пользователя на сервере SSH, запущенном на машине `хост`. Вас попросят ввести пароль, после чего соединение будет установлено! Флаг `N` отключает интерактивное приглашение командной строки, а флаг `D` указывает локальный порт для прослушивания (вы можете выбрать любой номер порта, если хотите). Флаг `T` отключает псевдо-tty распределение (pseudo-tty allocation).

Также хорошей идеей является использование флага verbose (`-v`).

Шаг 2: настройка браузера (или других программ)

Эта опция совершенно бесполезна, если вы не настроите ваш веб-браузер (или другие программы) на использование вновь созданного туннеля socks. Текущая версия SSH поддерживает как SOCKS4, так и SOCKS5, и вы можете использовать любой из них.

- Для Firefox: *Правка* > *Параметры* > *Дополнительно* > *Сеть* > *Соединение* > *Настройка*:
  Выберите пункт *Ручная настройка прокси* и введите `localhost` в поле *SOCKS host*, после чего введите ваш номер порта в следующее текстовое поле (в приведенном выше примере это `4711`)

Firefox не посылает автоматического запроса DNS через туннель socks. Эта потенциальная проблема безопасности может быть "смягчена" при помощи следующих действий:

1. Перейдите по адресу about:config, введя его в адресную строку Firefox
2. Найдите параметр network.proxy.socks_remote_dns
3. Установите его значение в true
4. Перезапустите браузер

- Для Chromium: вы можете установить настройки SOCKS через переменные окружения или опции командной строки. Я рекомендую добавить одну из следующих функций в ваш `.bashrc`:

```
function secure_chromium {
    port=4711
    export SOCKS_SERVER=localhost:$port
    export SOCKS_VERSION=5
    chromium &
    exit
}
```

ИЛИ

```
function secure_chromium {
    port=4711
    chromium --proxy-server="socks://localhost:$port" &
    exit
}
```

Теперь откройте терминал и просто выполните:

```
$ secure_chromium
```

Наслаждайтесь вашим защищенным туннелем!

Проброс X11

Проброс X11 - это механизм, который позволяет графическим интерфейсам программ, работающих на удаленной машине-сервере, отображаться на локальной машине-клиенте. При этом нет необходимости устанавливать на удаленном узле всю систему X11, но надо установить хотя бы *xauth*. *xauth* - это утилита, которая поддерживает конфигурации `Xauthority`, используемые сервером и клиентом для аутентификации сессии X11 (**источник**).

Важно: Использование проброса X11 имеет важные последствия для безопасности, о которых необходимо, по крайней мере, знать. Прочитайте соответствующие разделы страниц справочных руководств (man) `ssh`, `sshd_config` и `ssh_config`. Также смотрите **краткую рецензию**

Настройка

На удаленной системе:

- **Установите** пакеты **`xorg-xauth`** и **`xorg-xhost`** из **официальных репозиториев**
- В файле `/etc/ssh/sshd_config`:

- Удостоверьтесь, что опциям `AllowTcpForwarding` и `X11UseLocalhost` присвоены значения *yes*, а `X11DisplayOffset` - *10* (это значения по умолчанию, если ничего не менялось; смотрите **`sshd_config(5)`**)
- Установите для опции `X11Forwarding` значение *yes*
- **Перезапустите демон *sshd***

На клиенте включите опцию `ForwardX11` либо добавив флаг `-X` в командной строке для необходимых подключений, либо присвоив в **конфигурационном файле клиента openSSH** опции `ForwardX11` значение *yes*.

Совет: Если GUI отображается плохо или вы получаете ошибки, можно включить опцию `ForwardX11Trusted` (в командной строке это флаг `-Y`). Это предотвратит управление пробросом X11 со стороны **расширения безопасности X11**. Если вы будете использовать эту опцию, обязательно прочитайте **предупреждение** в начале этого раздела

Использование

Войдите в систему на удаленной машине, как обычно, добавив флаг `-X`, если опция *ForwardX11* не включена в конфигурационном файле клиента:

```
$ ssh -X пользователь@узел
```

Если вы получаете ошибки при попытке запуска графических приложений, попробуйте опцию *ForwardX11Trusted*:

```
$ ssh -Y пользователь@узел
```

Теперь вы можете запустить любую программу X (с графическим интерфейсом пользователя) на удаленном сервере, и ее вывод будет перенаправлен в вашу локальную сессию:

```
$ xclock
```

Если вы получите ошибки "Cannot open display", попробуйте выполнить следующую команду от имени обычного пользователя:

```
$ xhost +
```

эта команда позволит выполнять проброс приложений X11 любому пользователю. Чтобы ограничить проброс конкретным хостом:

```
$ xhost +имя_хоста
```

где имя_хоста - это имя конкретного хоста. Для получения дополнительной информации смотрите **`xhost(1)`**.

Будьте осторожны с некоторыми приложениями, так как они могут проверять локальную машину на предмет уже работающего приложения. Один из примеров - **Firefox**: либо закройте работающий Firefox, либо используйте следующий параметр запуска:

```
$ firefox -no-remote
```

Если вы получите ошибку "X11 forwarding request failed on channel 0" при подключении (и лог-файл сервера `/var/log/errors.log` будет содержать строку "Failed to allocate internet-domain X11 display socket"), удостоверьтесь, что пакет **`xorg-xauth`** установлен. Если его установка не поможет, попробуйте сделать одно из двух:

- Включить опцию `AddressFamily any` в `sshd_config` на *сервере*
- Присвоить опции `AddressFamily` в `sshd_config` на *сервере* значение inet

Присвоение значения inet может исправить проблемы с клиентами Ubuntu при использовании IPv4.

Для запуска приложений X от имени других пользователей на сервере SSH вам необходимо добавить (`xauth add`) строку аутентификации, взятую из `xauth list`пользователя, вошедшего в систему.

Проброс других портов

В дополнение к встроенной поддержке SSH X11, также можно установить безопасный туннель для любого соединения TCP с использованием локального или удаленного проброса.

Локальный проброс открывает на локальной машине порт, подключения к которому будут перенаправлены на удаленный хост, а оттуда - по заданному направлению. Очень часто этим направлением будет сам удаленный хост, предоставляющий secure shell и, например, безопасное соединение VNC для этой же машины. Локальный проброс осуществляется при помощи ключа `-L` и задания спецификации проброса в следующей форме: `<порт туннеля>:<адрес назначения>:<порт назначения>`.

Например:

```
$ ssh -L 1000:mail.google.com:25 192.168.0.100
```

будет использовать SSH для входа в систему и открытия шелла на 192.168.0.100, а также создаст туннель от порта 1000 локальной машины на порт 25 mail.google.com. В результате подключения к localhost:1000 будут перенаправлены на порт Gmail SMTP. To Google, it will appear that any such connection (though not necessarily the data conveyed over the connection) originated from 192.168.0.100, and such data will be secure as between the local machine and 192.168.0.100, but not between 192.168.0.100, unless other measures are taken.

Также:

```
$ ssh -L 2000:192.168.0.100:6001 192.168.0.100
```

будет принимать подключения к localhost:2000, которые будут перенаправлены на порт 6001 удаленного хоста. Этот пример хорош для установления подключений VNC с использованием vncserver utility.

Удаленный проброс позволяет удаленному хосту подключаться к произвольному хосту через туннель SSH и локальную машину, предоставляя функционал, обратный локальному пробросу. Это полезно в ситуациях, когда, например, удаленный хост ограничен фаерволлом. Он включается ключом `-R` и заданием спецификаций проброса в следующей форме: `<порт туннеля>:<адрес назначения>:<порт назначения>`.

Например:

```
$ ssh -R 3000:irc.freenode.net:6667 192.168.0.200
```

поднимет шелл на 192.168.0.200, и соединения из 192.168.0.200 к своему же порту 3000 (иначе говоря, localhost:3000) будут посланы через туннель на локальную машину, а затем на

irc.freenode.net, порт 6667, что в данном примере позволит использовать программы IRC на удаленном хосте, даже если обычно порт 6667 будет для них заблокирован.

Оба вида проброса могут быть использованы для предоставления безопасного "шлюза", позволяющего другим компьютерам получить преимущества туннеля SSH без непосредственно работающего SSH или демона SSH, при использовании bind-адреса в начале туннеля как части спецификации проброса, например, `<адрес туннеля>:<порт туннеля>:<адрес назначения>:<порт назначения>`. `<адрес туннеля>` может быть любым адресом на машине, `localhost`, `*` (или blank), который, соответственно, пропускает соединения через заданный адрес, интерфейс loopback или любой интерфейс. По умолчанию проброс ограничен соединениями от машины в начале туннеля, `<адрес туннеля>` установлен в `localhost`. Локальный проброс не требует дополнительной настройки, в то время как удаленный проброс ограничен конфигурацией демона SSH удаленного сервера. Смотрите опцию `GatewayPorts` на справочной странице `sshd_config(5)` для получения дополнительной информации.

Увеличение скорости SSH

Вы можете сделать так, чтобы все сессии использовали одно соединение, что значительно увеличит скорость последующих логинов. Для этого добавьте следующие строки под необходимым хостом в `/etc/ssh/ssh_config`:

```
Host examplehost.com
  ControlMaster auto
  ControlPersist yes
  ControlPath ~/.ssh/socket-%r@%h:%p
```

Смотрите справочную страницу `ssh_config(5)` для получения полного описания этих опций.

Другая возможность увеличения скорости - включение сжатия при помощи флага `-C`. Еще лучше добавить следующую строку под необходимым хостом в `/etc/ssh/ssh_config`:

```
Compression yes
```

Важно: В **ssh(1)** утверждается, что "*сжатие помогает в случае использования модемных линий и других медленных соединений, а в сетях с большой скоростью лишь замедляет работу*". В зависимости от конфигурации вашей сети этот совет может быть контрпродуктивен

Время, необходимое на вход в систему, может быть уменьшено при помощи флага `-4`, который отключает поиск через IPv6. Это может быть также достигнуто добавлением следующей строки под необходимым хостом в `/etc/ssh/ssh_config`:

```
AddressFamily inet
```

Изменение шифров, используемых SSH, на менее требовательные к процессору, также может увеличить скорость. С этой точки зрения наилучшим выбором станут arcfour и blowfish-cbc.

Важно: Пожалуйста, не делайте этого, если вы не знаете, к чему это приведет; у arcfour имеются слабые стороны

Для использования альтернативных шифров запустите SSH с флагом `-c`:

```
$ ssh -c arcfour,blowfish-cbc пользователь@адрес-пользователя
```

Чтобы использовать их постоянно, добавьте следующую строку под необходимым хостом в `/etc/ssh/ssh_config`:

```
Ciphers arcfour,blowfish-cbc
```

Монтирование удаленных файловых систем при помощи SSHFS

Обратитесь к статье **SSHFS**, чтобы использовать его для монтирования удаленных систем при помощи SSH в локальный каталог, с тем, чтобы у вас была возможность производить любые операции при помощи любых утилит (копировать, переименовать, отредактировать при помощи vim и т.д.). Использование sshfs вместо shfs является предпочтительным, поскольку новые версии shfs не выпускаются с 2004 г.

Поддержание подключения

Если вы не совершаете каких-либо действий, будет выполнен автоматический выход из сессии ssh. Для поддержания подключения в активном состоянии добавьте следующую строку в `~/.ssh/config` или `/etc/ssh/ssh_config` клиента:

```
ServerAliveInterval 120
```

В этом случае на сервер будет посылаться сигнал "keep alive" каждые 120 секунд.

С другой стороны, для поддержания в активном состоянии входящих подключений, вы можете прописать

```
ClientAliveInterval 120
```

(или любое другое число больше 0) в `/etc/ssh/sshd_config` на сервере.

Сохранение данных о подключении в конфигурационном файле ssh

Когда бы вы ни хотели подключиться к серверу ssh, обычно вам необходимо набирать его адрес и имя пользователя. Для сохранения данных о серверах, к которым вы регулярно подключаетесь, вы можете использовать персональные `~/.ssh/config` или глобальные `/etc/ssh/ssh_config` файлы, как показано в следующем примере:

```
~/.ssh/config
```

```
Host myserver
    HostName 123.123.123.123
    Port 12345
    User bob
Host other_server
    HostName test.something.org
    User alice
    CheckHostIP no
    Cipher blowfish
```

Теперь вы сможете легко подключиться к серверу, используя имя, которое вы указали:

```
$ ssh myserver
```

Чтобы увидеть полный перечень доступных опций, смотрите справочную страницу ssh_config вашей системы или **документацию ssh_config** на официальном сайте.

Autossh - автоматический перезапуск сессий и туннелей SSH

Если сессия или туннель не может поддерживаться в активном состоянии, например, из-за плохого подключения к сети и связанных с ним отключений, вы можете использовать **Autossh** для их автоматического перезапуска. Autossh может быть установлен из **официальных репозиториев**.

Примеры использования:

```
$ autossh -M 0 -o "ServerAliveInterval 45" -o "ServerAliveCountMax 2"
имя_пользователя@example.com
```

Совместно с **SSHFS**:

```
$ sshfs -o
reconnect,compression=yes,transform_symlinks,ServerAliveInterval=45,ServerAli
veCountMax=2,ssh_command='autossh -M 0' имя_пользователя@example.com:
/mnt/example
```

Подключение через SOCKS-прокси, сконфигурированный при помощи **настроек proxy**:

```
$ autossh -M 0 -o "ServerAliveInterval 45" -o "ServerAliveCountMax 2" -NCD
8080 имя_пользователя@example.com
```

При помощи опции `-f` autossh может быть запущен в качестве фонового процесса. Однако, в этом случае вы не сможете вводить пароль в интерактивном режиме.

Сессия будет завершена, как только вы введете команду `exit`, иначе процесс autossh получит сигнал SIGTERM, SIGINT of SIGKILL.

Автозапуск Autossh при загрузке системы при помощи systemd

Если вы хотите, чтобы autossh запускался автоматически, вы можете использовать systemd. Например, вы можете создать файл юнита, подобный этому:

```
[Unit]
Description=AutoSSH service for port 2222
After=network.target

[Service]
Environment="AUTOSSH_GATETIME=0"
ExecStart=/usr/bin/autossh -M 0 -NL 2222:localhost:2222 -o TCPKeepAlive=yes
foo@bar.com

[Install]
WantedBy=multi-user.target
```

Здесь `AUTOSSH_GATETIME=0` - это переменная окружения, указывающая, как долго ssh должен быть поднят, прежде чем autossh утвердит успешное подключение. Установка значения 0

укажет autossh игнорировать неудачное подключение ssh. Это может быть полезно при добавлении autossh в автозагрузку. Другие переменные окружения доступны на справочной странице. Конечно, вы можете сделать этот юнит более комплексным, если вам это необходимо (для получения дополнительных подробностей смотрите документацию systemd); очевидно, вы можете использовать ваши собственные опции для autossh, но учтите, что флаг `-f`, подразумевающий `AUTOSSH_GATETIME=0`, не работает с systemd.

Затем поместите все это, например ,в /etc/systemd/system/autossh.service. Теперь вы можете включить туннели autossh при помощи подобной команды:

```
$ systemctl start autossh
```

Если после этого autossh запустится, вы можете включить его в автозагрузку, выполнив

```
$ systemctl enable autossh
```

После этого autossh будет автоматически запускаться при старте системы.

Легко поддерживать несколько процессов autossh для разных туннелей. Просто создайте несколько файлов .service с разными именами.

Изменение номера порта SSH для активации через сокет (sshd.socket)

Создайте файл `/etc/systemd/system/sshd.socket.d/port.conf` со следующим содержанием:

```
[Socket]
# Disable default port
ListenStream=
# Set new port
ListenStream=12345
```

systemd будет автоматически прослушивать новый порт после перезапуска:

```
systemctl daemon-reload
```

Решение проблем

Проверка

Здесь представлен список вещей, которые необходимо проверить, прежде чем искать решение проблем. Рекомендуется проверить то, что здесь описано, прежде чем смотреть другие разделы.

1. Каталоги `~/.ssh` на клиенте и сервере, а также их содержимое, должны иметь соответствующие права доступа:

```
$ chmod 700 /home/ПОЛЬЗОВАТЕЛЬ/.ssh
$ chmod 600 /home/ПОЛЬЗОВАТЕЛЬ/.ssh/*
```

2. Удостоверьтесь, что всеми файлами в каталогах `~/.ssh` на клиенте и сервере владеют правильные пользователи:

```
$ chown -R ПОЛЬЗОВАТЕЛЬ: ~/.ssh
```

3. Убедитесь, что строка открытого ключа клиента (например, `id_rsa.pub`) есть в файле `authorized_keys` на сервере

4. Проверьте, не ограничивали ли вы доступ через SSH в строке `AllowUsers` файла `/etc/ssh/sshd_config` (разделяя имена пользователей пробелами)

Удаление устаревших ключей (необязательно)

5. Удалите строки, содержащие старые/неправильные ключи, из файла `~/.ssh/authorized_keys` на сервере

6. Удалите старые/неправильные закрытые и открытые ключи из каталога `~/.ssh` на клиенте

Рекомендации

7. В файле сервера `~/.ssh/authorized_keys` храните настолько мало ключей, насколько это возможно

Зависание подключения SSH после выключения/перезагрузки

Подключение SSH зависает после выключения или перезагрузки в том случае, когда systemd останавливает сеть раньше, чем sshd. Чтобы этого не происходило, закомментируйте и измените строку `After`:

```
/usr/lib/systemd/system/systemd-user-sessions.service

#After=remote-fs.target
After=network.target
```

Подключение отклонено или проблема тайм-аута

Ваш роутер настроен на проброс портов?

ПРОПУСТИТЕ ЭТОТ ШАГ, ЕСЛИ ВАША МАШИНА НЕ НАХОДИТСЯ ЗА МОДЕМОМ/РОУТЕРОМ NAT. Большинство домов и небольших предприятий имеют модем/роутер NAT.

Прежде всего необходимо убедиться, что ваш роутер пробрасывает любые входящие соединения ssh к вашей машине. Ваш внешний IP вам выдает ISP, и он ассоциируется со всеми запросами, идущими от вашего роутера. Поэтому ваш роутер должен знать, что все входящие соединения ssh, обращенные к вашему внешнему IP, должны быть перенаправлены на машину с работающим sshd.

Узнайте ваш адрес внутри сети:

```
ip a
```

Найдите ваш интерфейс (который используется для подключения к сети), а в нем - поле inet. Зайдите в веб-интерфейс настройки роутера, используя его IP. Задайте перенаправление на ваш inet IP. Перейдите по ссылке [1] для получения дополнительной информации о том, как это сделать для некоторых роутеров.

SSH запущен и прослушивает?

```
$ ss -tnlp
```

Если эта команда не показывает открытый порт SSH, SSH НЕ запущен.
Смотрите `/var/log/messages` на наличие ошибок.

Имеются ли правила фаерволла, блокирующие соединения?

Iptables может блокировать подключения к порту `22`. Проверьте это следующей командой:

```
# iptables -nvL
```

Просмотрите вывод на предмет правил, которые могут блокировать нужные вам пакеты (секция `INPUT`). Затем, если необходимо, разблокируйте порт командой вида:

```
# iptables -I INPUT 1 -p tcp --dport 22 -j ACCEPT
```

Для получения помощи по настройке фаерволлов, смотрите статью **firewalls**.

Трафик доходит до вашего компьютера?

Запустите дамп трафика на компьютере, с которым возникли проблемы:

```
# tcpdump -lnn -i any port ssh and tcp-syn
```

Будет показана некоторая базовая информация. Подождите совпадения. После этого попробуйте подключиться вновь. Если вы не видите никакого вывода команды, когда вы пытаетесь подключиться, это значит, что что-то вне вашего компьютера блокирует трафик (это может быть аппаратный фаерволл, роутер NAT и т.д.).

Ваш провайдер или кто-то еще блокирует нужный порт?
Примечание: Пробуйте этот шаг в том случае, если вы **знаете**, что у вас не запущено никаких фаерволлов, и что вы настроили роутер на DMZ или проброс портов на компьютер. Здесь вы найдете рекомендации по диагностике и возможные решения

В некоторых случаях ваш провайдер может блокировать порт по умолчанию (SSH порт 22). Чтобы это проверить, создайте сервер на всех интерфейсах (0.0.0.0) и подключитесь удаленно.

Если вы получите сообщение об ошибке вроде этого:

```
ssh: connect to host www.inet.hr port 22: Connection refused
```

это означает, что порт **не** был заблокирован провайдером: просто на сервере не запущен SSH для этого порта (смотрите статью **Безопасность через неясность**).

Однако, если вы получите сообщение об ошибке вроде этого:

```
ssh: connect to host 111.222.333.444 port 22: Operation timed out
```

это означает, что что-то отклоняет ваш трафик TCP, предназначенный для порта 22. Как правило, этот порт скрыт либо вашим фаерволлом, либо третьей стороной (например, провайдером, блокирующим и/или отклоняющим входящий трафик на порт 22). Если вы знаете, что фаерволл на вашем компьютере не запущен и Гремлины не размножаются на ваших роутерах и свитчах, это означает, что провайдер блокирует трафик.

Чтобы убедиться в этом, вы можете запустить Wireshark на сервере и "прослушать" трафик, предназначенный для порта 22. Поскольку Wireshark является утилитой анализа трафика на уровне 2 , а TCP/UDP используют уровень 3 и выше (смотрите статью **TCP/IP**), если вы ничего не получаете при создании удаленного подключения, вероятнее всего, что третья сторона блокирует трафик для этого порта на вашем сервере.

Диагностика при помощи Wireshark

Установите Wireshark из пакета **`wireshark-cli`**, доступного в **официальных репозиториях**.

Затем запустите его:

```
tshark -f "tcp port 22" -i NET_IF
```

где NET_IF - сетевой интерфейс для соединения WAN (для проверки смотрите `ip a`). Если вы не получаете никаких пакетов при попытке удаленного подключения, можете быть уверены, что ваш провайдер блокирует входящий на порт 22 трафик.

Возможное решение

Вы можете просто использовать другой порт, который провайдером не блокируется. Откройте `/etc/ssh/sshd_config` и укажите другой порт. Например, добавьте:

```
Port 22
Port 1234
```

Также удостоверьтесь, что другие строки "Port" закомментированы. Если просто закомментировать строку "Port 22" и прописать "Port 1234", проблема не будет решена, поскольку sshd будет прослушивать лишь порт 1234. Используйте обе строки для запуска сервера SSH на обоих портах.

Перезапустите сервер `systemctl restart sshd.service`. Готово! Теперь вам необходимо настроить ваш(и) клиент(ы) на использование другого порта.

Read from socket failed: connection reset by peer

Последние версии openssh иногда выдают подобное сообщение об ошибке из-за бага. В этом случае добавьте следующую строку в файл `~/.ssh/config`:

```
HostKeyAlgorithms ssh-rsa-cert-v01@openssh.com,ssh-dss-cert-
v01@openssh.com,ssh-rsa-cert-v00@openssh.com,ssh-dss-cert-
v00@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-
nistp521,ssh-rsa,ssh-dss
```

С openssh 5.9 это исправление не поможет. В этом случае добавьте в `~/.ssh/config` следующее:

```
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
MACs hmac-md5,hmac-sha1,hmac-ripemd160
```

Смотрите также **обсуждение** на форуме openssh.

"[ваша командная оболочка]: No such file or directory" / ssh_exchange_identification problem

Одна из возможных причин - необходимость найти абсолютный путь (который возвращает команда `whereis -b [ваша командная оболочка]`, например) в `$SHELL`, даже если бинарный пакет вашего интерпретатора находится в одной из записей `$PATH`.

Сообщение об ошибке "Terminal unknown" или "Error opening terminal "

В ssh возможно получение ошибок, вроде "Terminal unknown", во время входа в систему. Запуск приложений ncurses (например, nano) не удается, появляется ошибка "Error opening terminal". Есть два способа исправления этой проблемы: по-быстрому - используя переменную $TERM, и основательно с использованием файла terminfo.

Обходной путь с использованием переменной $TERM

После подключения к удаленному серверу задайте переменной $TERM значение "xterm" следующей командой:

```
TERM=xterm
```

Этот метод является обходным путем и должен использоваться на серверах ssh, к которым вы редко подключаетесь, поскольку он имеет нежелательные побочные эффекты. К тому же вам придется повторять ввод этой команды после каждого подключения, либо прописывать это значение в ~.bashrc.

Решение с использованием файла terminfo

Лучшее решение - передача файла terminfo с вашего клиента на сервер. В этом примере мы покажем, как настроить файл terminfo на терминал "rxvt-unicode-256color". Создайте каталог, содержащий файлы terminfo, на сервере ssh, когда вы подключены к нему:

```
mkdir -p ~/.terminfo/r/
```

Теперь скопируйте файл terminfo вашего терминала в новый каталог. Замените `rxvt-unicode-256color` на свое значение (клиентской машины) в следующей команде, а `ssh-server` - подходящими именем пользователя и адресом сервера.

```
$ scp /usr/share/terminfo/r/rxvt-unicode-256color ssh-server:~/.terminfo/r/
```

После входа и выхода из сервера ssh проблема должна быть решена.

Смотрите также

- **Защита от общих атак на логин SSH**
- **Защита против брут-форс (brute force) атак ssh**
- **Управление ключами OpenSSH, Часть 1** и **Часть 2** на сайте IBM developerWorks

Categories:

- Secure Shell (Русский)
- Servers (Русский)
- OpenBSD (Русский)
- Русский