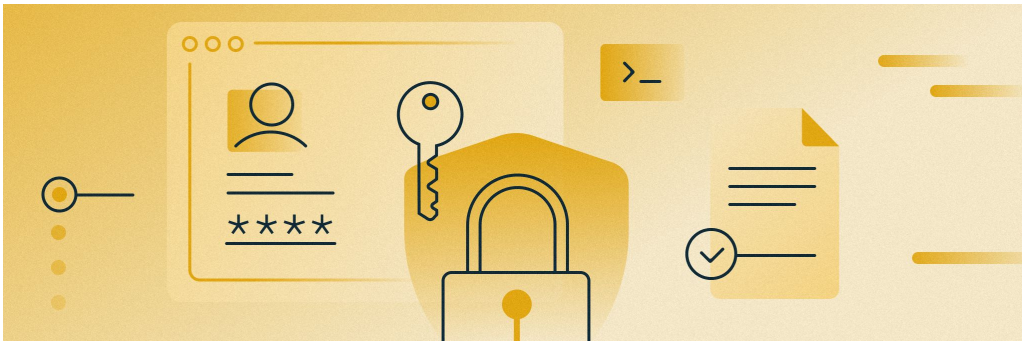
Авторизация на базе ключей SSH

Тирекс Самый зубастый автор 20 марта 2023

Рассматриваем процесс настройки SSH-авторизации по ключу и разбираем некоторые ошибки.



SSH-ключи для авторизации — это простой и надежный способ получения доступа к удаленному узлу. В статье мы рассмотрим процесс настройки SSH-авторизации по ключу, а также покажем способы устранения некоторых известных ошибок.

Что такое SSH-ключ

Аббревиатура SSH означает Secure Shell, что дословно переводится как «безопасная оболочка». Если говорить точнее, SSH — это защищенный сетевой протокол с проверкой подлинности и шифрованием. Он используется для передачи файлов, управления сетью и удаленного доступа к операционной системе. Аббревиатура SSH также используется для описания набора инструментов, используемых для взаимодействия с одноименным протоколом.

Подключение через SSH можно просто охарактеризовать как подключение к командной строке удаленного узла. То есть любые команды, которые будут вводиться в терминал на основной машине будут работать так же, будто вы вводите их напрямую на удаленном узле, сидя с клавиатурой около него.

Так как SSH работает по системе «клиент-сервер», обязательное условие для работы этого протокола — наличие на удаленном узле демона SSH. Демон SSH — это ПО, которое прослушивает определенный сетевой порт и при прохождении аутентификации другим узлом создает необходимую среду для работы с ним.

В свою очередь на локальной машине должно быть установлено соответствующее ПО — SSH-клиент. Он взаимодействует с удаленным хостом и передает ему необходимые данные, которые нужны для прохождения аутентификации.

Пользователи Linux могут установить OpenSSH с помощью команды:

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

В некоторых ОС компоненты OpenSSH можно установить отдельно для клиента *openssh-client* и отдельно для сервера *openssh-server*.

Структура ключа

Можно сказать, что определение «SSH-ключ» составное, так как на самом

Ключи могут быть сгенерированы с помощью различных алгоритмов, которые поддерживает текущая версия протокола SSH. Например, если использован тип шифрования RSA, то файлы будут именоваться следующим образом:

- id_rsa — закрытый ключ,
- id_rsa.pub — публичный (открытый) ключ.

В чем же разница открытого и закрытого ключа?

Открытые и закрытые SSH-ключи

Открытый (он же публичный) ключ используется для шифрования данных при обращении к удаленному узлу. Проще говоря, это набор символов, при помощи которых мы шифруем информацию. Он доступен всем. Не стоит бояться того, что открытый ключ может попасть в чужие руки, так как наличие одного лишь публичного SSH-ключа не дает злоумышленнику никаких преимуществ. Открытый SSH-ключ хранится на удаленном узле.

Закрытый (приватный) SSH-ключ — это ключ к данным. Он расшифровывает сами сообщения. Хранится он на устройстве, которое будет подключаться к удаленному узлу (на котором находится открытый ключ). Приватный ключ ни в коем случае нельзя передавать в чужие руки, в том числе через мессенджеры или файлообменники, во избежание утечки информации и персональных данных. Также рекомендуем сразу установить пароль на закрытый ключ, чтобы обеспечить ему дополнительную защиту.

Как работает SSH-авторизация?

Давайте представим, что Selectel — это сервер, а вы — клиент. Вы хотите подключиться к нам с использованием SSH-ключа. Предварительно вы уже создали пару ключей и передали публичный ключ нам. Алгоритм взаимодействия будет следующим:

1. Вы должны изъявить свое желание подключиться к нам, то есть отправить запрос на подключение по TCP-порту.

2. В случае установки TCP-соединения мы бмениваемся информацией о версиях наших SSH-протоколов. С помощью этой информации можно понять, какую именно конфигурацию (версию протоколов и алгоритмы работы) использовать. Самостоятельно узнать версию OpenSSH можно с помощью команды *ssh -V*.

3. После согласования мы (сервер) направляем вам (клиенту) открытый ключ. Теперь уже вы решаете, доверять такому ключу или нет. В случае положительного ответа мы с вами генерируем сеансовый ключ, который будет использоваться для симметричного шифрования канала. Этот ключ существует, только пока существует канал (текущая сессия).

4. Теперь следует аутентифицировать вас. Для этого вы отсылаете нам свой открытый ключ. Мы в свою очередь проверяем его со своим списком открытых SSH-ключей. Если совпадение найдено, мы генерируем случайное число, шифруем его открытым ключом и отправляем обратно. Вы как клиент расшифровываете сообщение закрытым ключом и отправляете полученные данные нам. В случае совпадения присланного числа с первоначальным аутентификация признается успешной.

Поздравляем! Теперь вам открыт доступ на сервер.

Типы ключей SSH

Как вы уже знаете, существуют два основных типа ключей – открытый и закрытый. Но также ключи можно разделить по типу шифрования.

Если мы введем в терминал команду *ssh-keygen ?*, то увидим справку, где у параметра *-t* указаны алгоритмы, которые можно использовать в данной системе:

```
~# ssh-keygen ?
…
[-t dsa | ecdsa | ecdsa-sk | ed25519 | ed25519-sk | rsa]
```

RSA – по умолчанию ssh-keygen использует в качестве параметра -t именно RSA, так как этот алгоритм обеспечивает наилучшую совместимость из всех, но требует большего размера ключа для обеспечения достаточной безопасности. Длина ключа по умолчанию составляет 3072 бит, но вы можете самостоятельно задать его размер от 1024 до 16384 бит с помощью опции -b команды ssh-keygen.

Имейте в виду, что, чем больше ключ, тем больше вычислительных мощностей и трафика будут задействовать устройства. Поэтому если вам нужна усиленная защита и для этого вы хотите увеличить длину ключа RSA, возможно, вам стоит рассмотреть алгоритмы на основе эллиптических кривых. При таких условиях они будут работать быстрее.

DSA – криптографический алгоритм с использованием открытого ключа для создания электронной подписи, но не для шифрования. Это значит, что только один узел сможет подписать сообщение, а другие смогут только проверить ее на корректность. Алгоритм основан на вычислительной сложности взятия логарифмов в конечных полях. Алгоритм DSA в сравнении с RSA показывает лучшие показатели при генерации подписи, но уступает по времени при ее проверке. Также отметим, что максимальная длина ключа данного типа – 1024 бит. Это не самый безопасный показатель в защите от взлома.

ECDSA – это реализация схемы цифровой подписи, основанная на использовании эллиптических кривых и модульной арифметики. Производительность данного алгоритма быстрее, чем у алгоритма RSA, так как для обеспечения шифрования требуются ключи гораздо меньшего размера. Однако у него есть минус: он более уязвим перед взломом с помощью квантовых вычислений, чем RSA.

ED25519 – это схема подписи на эллиптической кривой, которая обеспечивает лучшую безопасность, чем ECDSA и DSA, и хорошую производительность. Его главными преимуществами являются скорость. Однако данный алгоритм поддерживается только в версиях от OpenSSH 6.5.

ECDSA-SK и **ED25519-SK** – это те же ECDSA и ED25519, но поддерживающие аппаратный аутентификатор. Данные алгоритмы появились не так давно в версии OpenSSH 8.2. Они позволяют реализовать двухфакторную аутентификацию с помощью устройств, которые поддерживают протокол FIDO/U2F. Существует множество разновидностей физической аутентификации, например USB-токен, или другие виды устройств, подключаемые к оборудованию с помощью Bluetooth или NFC.

Создание SSH-ключа

Параметры утилиты ssh-keygen

В Linux для создания SSH-ключей используется команда ssh-keygen. Далее приведены наиболее популярные параметры для этой команды и их описание.

-C comment

Создание нового комментария. Например, одной из самых известных команд является добавление комментария с информацией о том, кто, на какой машине и когда создал ключ:

```
ssh-keygen -C "$(whoami)@$(uname -n)-$(date -I)"
```

-p

Если указать данный параметр при вводе *ssh-keygen*, то вам будет предложено изменить старую секретную фразу на новую. Ввод команды будет выглядеть следующим образом:

```
ssh-keygen -p
```

Также вы можете задать все параметры для изменения секретной фразы сразу:

```
ssh-keygen -p [-P old_passphrase] [-N new_passphrase] [-f keyfile]
```

-t type

Задает тип создаваемого ключа. Чтобы узнать какие типы доступны для вашей версии, введите:

```
ssh-keygen -t ?
```

-v

Подробный режим. ssh-keygen будет печатать отладочные сообщения о ходе выполнения. Несколько опций -v увеличивают степень подробности информации (максимум 3).

Генерация SSH-ключей в Linux

Процесс создания SSH-ключей на базе Linux предельно прост. Вам необходимо лишь указать некоторые параметры, которые мы опишем далее.

Мы будем создавать ключ RSA. При вводе команды ssh-keygen опцию *-t RSA* можно не указывать явно, так как RSA является алгоритмом по умолчанию.

```
~# ssh-keygen
Generating public/private RSA key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

В последней строке вам предлагается выбрать путь, куда будут сохранены ключи. Если оставить поле пустым, будут созданы файлы с именами id_rsa.pub и id_rsa.

В данной строке предлагается создать кодовую фразу. Оставив строку пустой, дополнительной защиты не будет.

С одной стороны, не вводить кодовую фразу опасно. Если закрытый ключ попадет в руки злоумышленников, то у них может появиться доступ к серверу. В то же время, если вы введете кодовую фразу, в будущем это может доставить вам неудобства. Пароль придется вводить каждый раз при использовании SSH-подключения с этим ключом. Поэтому использование кодовой фразы остается на ваше усмотрение.

Как мы упоминали ранее, кодовую фразу можно будет установить (или заменить) самостоятельно, введя *ssh-keygen -p*, когда ключ уже будет создан.

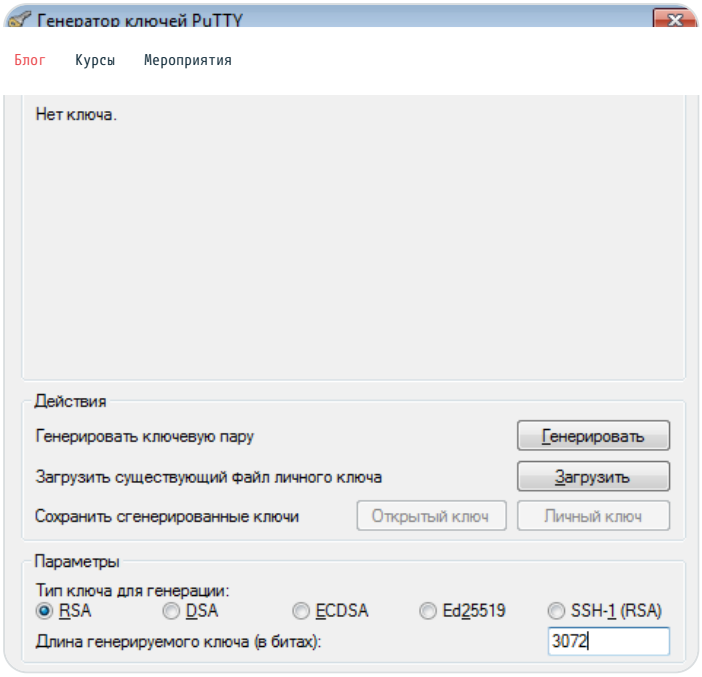
```
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Uh6CVy4Voz0/70Am8j+hXOLBV21l4rMmiMLG5BTz3cE root@trexc
The key's randomart image is:
+---[RSA 3072]----+
|       =. .      |
|       .o* .  E . o|
|       . =+*. . + + |
|       .o=.+. o =  |
|       *o.S•=o . o |
|       *+=+=o. o  |
|       . +.*...o   |
|       +..o      |
|       ...       |
+----[SHA256]-----+
```

Созданные файлы хранятся в директории /home/*SystemName*/.ssh/.

Генерация SSH-ключей в Windows с помощью Putty

Если вы хотите создать SSH-ключи на базе ОС Windows, самым популярным решением для этого будет использование программного обеспечения Putty. Скачать его можно с официального сайта по ссылке (актуальная версия на момент написания статьи – Putty 0.78). После установки программы вам будет доступно несколько .exe файлов.

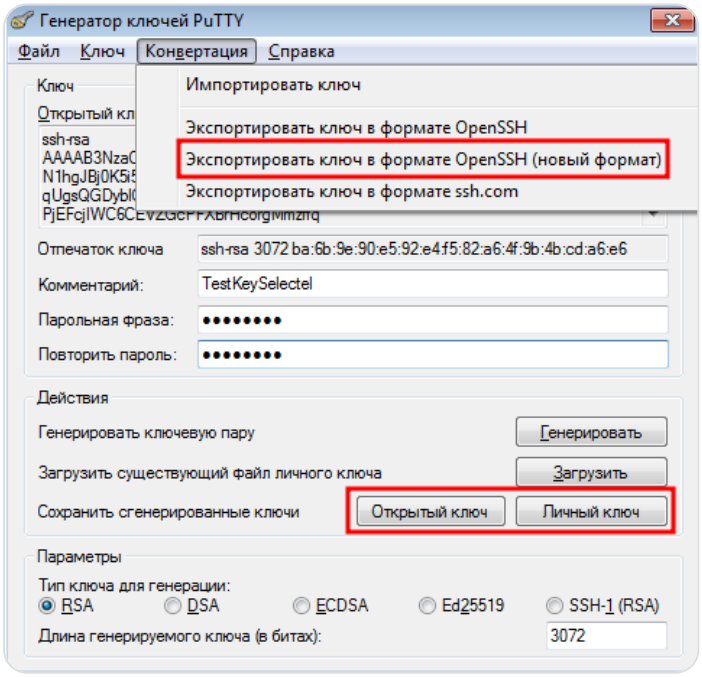
Для создания SSH-ключей откройте puttygen.exe, выберите необходимый тип ключа и задайте его размер. Мы будем использовать RSA с длиной 3072.



Далее нажмите **Генерировать** и водите мышкой по экрану в свободной зоне до тех пор, пока генерация SSH-ключей не будет завершен. Это необходимо делать для того, чтобы задать псевдослучайность при создании ключей.

После генерации ключа вы можете вписать комментарий и кодовую фразу.

Далее необходимо сохранить на диск открытый и закрытый SSH-ключи, для этого нажмите на соответствующие кнопки в окне программы. Вы можете самостоятельно выбрать путь, куда сохранять данные ключи.



Putty сохраняет закрытые ключи с разрешением .ppk, что означает, что такой ключ можно будет использовать только с Putty. Чтобы сохранить секретный ключ в формате, пригодном для OpenSSH, нужно во вкладке **Конвертация** выбрать пункт «Экспортировать ключ в формате OpenSSH (новый формат)» и указать расположение нового файла (старый формат при конвертации из .pkk в OpenSSH сохраняет не все поля – например, комментарий).

Генерация SSH-ключей в Windows с помощью OpenSSH

Не так давно в ОС Windows добавили возможность использования

Установить клиент OpenSSH можно с помощью следующей команды в PowerShell:

```
Add-WindowsCapability -Online -Name OpenSSH.Client
```

Или же с помощью графического интерфейса: Параметры → Приложения → Дополнительные возможности → Добавить компонент. В списке предложенных компонентов необходимо найти **Клиент OpenSSH** и нажать кнопку **Установить**.

Проверить статус компонента можно командой:

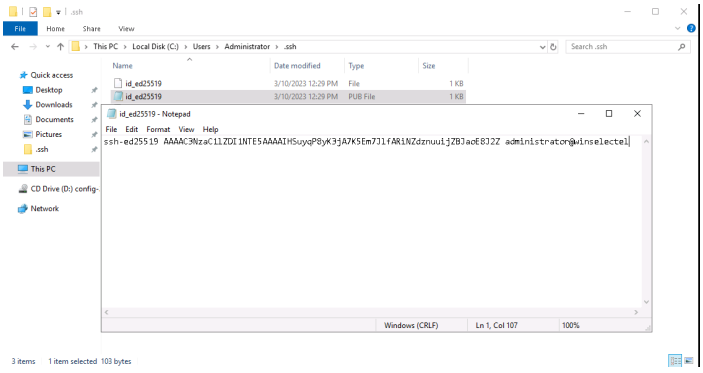
```
Get-WindowsCapability -Online | ? Name -like 'OpenSSH.Client*'
```

Если в ответе команды статус Installed, то загрузка выполнена успешно.

Генерация SSH-ключей происходит таким же образом, как и в ОС Linux, с помощью *ssh-keygen:*

```
PS C:\Users\Administrator> ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\Administrator/.
Created directory 'C:\Users\Administrator/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Administrator/
Your public key has been saved in C:\Users\Administrator/.ssh
The key fingerprint is:
SHA256:1qBO0MIgYM+A80G1zZgUzW4QCYHKNuLgwaIR77sY2/U administra
The key's randomart image is:
+--[ED25519 256]--+
|═══+=*           |
|*•=+oBo          |
|+= +Bo+ .        |
|**o  oo. o       |
|Ooo  .o S .      |
|.o.  o .         |
|.  .. .          |
| =.. .           |
|o o.  E          |
+----[SHA256]-----+
```

Созданные ключи можно найти в папке C:\Users\Administrator\.ssh, если она не была изменена при генерации.



Копирование открытого ключа на сервер

В данном разделе мы рассмотрим общий вариант копирования открытого

Копирование при помощи SSH-Copy-ID

Данный метод подойдет тем, чья ОС поддерживает команду SSH-Copy-ID, и удаленный сервер имеет доступ по SSH без ключа. Если это не так, попробуйте использовать второй или третий способ.

Синтаксис команды выглядит следующим образом:

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

Обратите внимание на последние строки вывода:

```
-# ssh-copy-id root@5.159.102.80
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
The authenticity of host '5.159.102.80 (5.159.102.80)' can't
ED25519 key fingerprint is SHA256:KSvVyjbijk+LQ7n3cEQd94qcyIc
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerp
```

Наш клиент запрашивает разрешение на продолжение подключения к удаленному узлу, так как встречает его впервые. Необходимо ввести *yes*. После успешного подключения ключи будут добавлены и мы увидим соответствующий вывод:

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -
root@5.159.102.80's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@5.159.10
and check to make sure that only the key(s) you wanted were a
```

Чтобы проверить, действительно ли скопировался открытый ключ, найдем искомый файл authorized_keys на удаленном узле и посмотрим его содержимое, так как именно в него добавляются открытые SSH-ключи.

```
~# cat /root/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCoQ/EcDWqYKTKCLkd3gZcz9
```

Как мы видим, публичный SSH-ключ клиента был успешно добавлен на сервер.

Примечание. Если на удаленном узле вы используете не стандартный порт в качестве порта для подключения по SSH, а любой другой, то необходимо при вводе команды ssh-copy-id указать дополнительные параметры:

```
ssh-copy-id ‘-p *port* username@remote_host’
```

После внесенных изменений необходимо перезапустить службы SSH.

Копирование ключа при помощи SSH

Данный метод подойдет тем, чья клиентская машина не поддерживает команду SSH-Copy-ID, но сервер имеет доступ по SSH.

В таком случае вы можете воспользоваться командой:

```
cat ~/.ssh/id_rsa.pub | ssh username@remote_host "mkdir -p ~/.ssh && cat >>
```

, где

- *cat ~/.ssh/id_rsa.pub* – вывод содержимого файла *id_rsa.pub,*
- *ssh username@remote_host* – подключение к удаленному серверу,
- *«mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys»* – проверка наличия директории */.ssh* и перенаправление вывода первой команды в файл *authorized_keys*.

Чтобы проверить, действительно ли скопировался открытый ключ, точно так же, как в первом варианте, найдем искомый файл authorized_keys на удаленном узле и посмотрим его содержимое.

```
~# cat /root/.ssh/authorized_keys
```

После внесенных изменений необходимо перезапустить службы SSH.

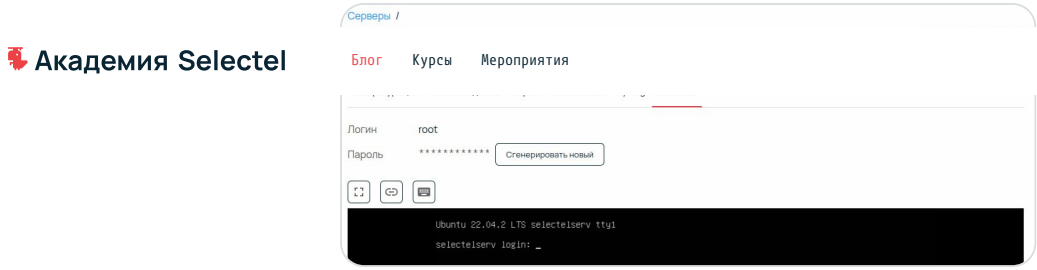
```
sudo service ssh restart (для Ubuntu / Debian / Mint Linux)
sudo service sshd restart (для CentOS / RHEL / Fedora Linux)
```

Копирование пользователем вручную

Данный метод следует использовать тем, у кого нет SSH-доступа на удаленный сервер. Вам необходимо найти файл *id_rsa.pub* (при использовании Linux) или *public* (при использовании Windows) на клиентской машине, открыть его и скопировать все содержимое. Через консоль Linux это можно сделать, используя команду *cat*:

```
cat ~/.ssh/id_rsa.pub
```

Затем следует любым доступным вам способом подключиться к удаленному серверу, куда необходимо скопировать открытый SSH-ключ. В нашем случае это будет консольный доступ к облачному (виртуальному) серверу через панель my.selectel.ru.



Найдя все тот же файл *authorized_keys*, необходимо просто вставить в него то, что мы скопировали из файла *id_rsa.pub* на клиентской машине ранее. В случае, если вы генерировали SSH-ключи в ОС Windows с помощью Putty, необходимо так же скопировать публичный ключ в файл *authorized_keys*. Либо вы можете открыть консоль и использовать команду для загрузки ключа:

```
echo *Text* >> ~/.ssh/authorized_keys
```

Где **Text** — это скопированные вами данные из *id_rsa.pub*.

После внесенных изменений необходимо перезапустить службы SSH.

```
sudo service ssh restart (для Ubuntu / Debian / Mint Linux)
sudo service sshd restart (для CentOS / RHEL / Fedora Linux)
```

Использование SSH-ключей

Настройка аутентификации на сервере с помощью SSH-ключей из ОС Linux

Чтобы воспользоваться подключением по SSH с помощью ключей, необходимо из-под клиентской машины, на которой мы создавали ключи, ввести команду:

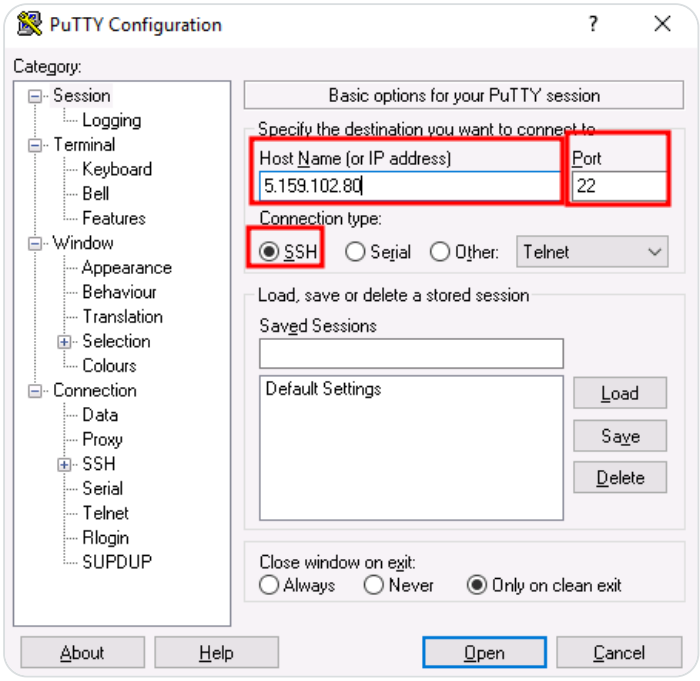
```
~# ssh username@remote_host

root@trexclient:~# ssh root@5.159.102.80
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-60-generic x8

 * Documentation:  https://help.ubuntu.com
 * Management:  https://landscape.canonical.com
 * Support:     https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sun Mar  5 12:31:34 2023 from 152.89.133.14
root@selectelserv:~#
```

Если имя пользователя на локальной машине и удаленном узле совпадают, можно сократить вводимую команду до:

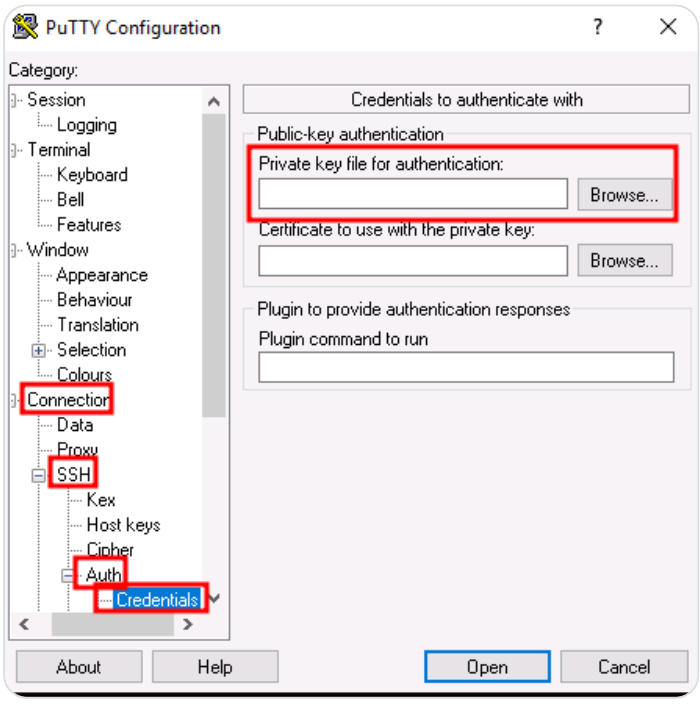
Настройка аутентификации на сервере с помощью SSH-ключей из ОС Windows

Ранее мы уже скопировали открытый SSH-ключ, сгенерированный с помощью Putty, на удаленный узел. Теперь необходимо произвести подключение.

Для этого откройте putty.exe и введите IP-адрес удаленного узла и порт для подключения, а также проверьте, что в качестве типа подключения выбран SSH.

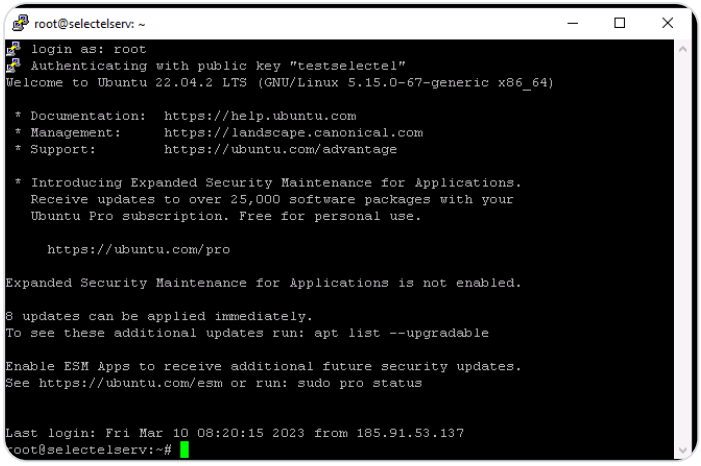


Далее пройдите по пути, указанному на скриншоте (Connection → SSH → Auth → Credentials):



В поле **Private key file for authentication** вставьте ссылку на приватный ключ, который необходимо использовать в данном подключении, и нажмите **Open**.

Перед вами откроется окно, в котором нужно ввести логин пользователя



Подключение из ОС Windows с помощью OpenSSH аналогично подключению по SSH на сервере Linux:

```
C:\Users\Administrator> ssh hostname@remote_host
```

Если при генерации SSH-ключей была создана секретная фраза, то удаленный хост запросит ее подтверждение:

```
Enter passphrase for key 'C:\Users\Administrator/.ssh/id_ed25519':
```

Выполнение одной команды на удаленном узле

Если вам необходимо ввести лишь одну команду, можно не создавать целый сеанс, а ввести нужную команду после *ssh hostname@remote_host*.

```
ssh hostname@remote_host command
```

Технически сессия все-таки создается, и происходит аутентификация на основе введенных данных. Затем передается нужная команда command, и сеанс сразу же закрывается. Поэтому визуально кажется, что мы просто передали команду на удаленный узел.

Отпечаток SSH-ключа

Каждая пара SSH-ключей использует один криптографический отпечаток. Его можно использовать для идентификации ключей.

```
~# ssh-keygen -l
Enter file in which the key is (/root/.ssh/id_rsa):
```

Здесь вы можете оставить поле пустым, если автоматически выбран нужный вам ключ. Если это не так, введите местоположение интересующего вас файла. После этого на экран будет выведена подобная запись (длина ключа, отпечаток ключа, пользователь и хост, для которого был создан ключ, используемый алгоритм шифрования):

Смена порта SSH

Дополнительной защитой доступа к вашему серверу может быть смена стандартного порта, на котором работает SSH. Перед сменой порта подключения необходимо открыть данный порт в файерволе с помощью команды:

```
sudo ufw allow *port*
```

Теперь приступим к изменению порта. В первую очередь, находясь на удаленном узле, необходимо открыть файл sshd_config:

```
sudo nano /etc/ssh/sshd_confog
```

Внутри этого файла найдите строку *Port* и укажите там свое значение, например, 4312.

```
Port 4312
```

После этого необходимо перезагрузить демон SSH:

```
sudo service ssh restart (для Ubuntu / Debian / Mint Linux)
sudo service sshd restart (для CentOS / RHEL / Fedora Linux)
```

Теперь при подключении к этому удаленному узлу необходимо указывать новый порт для подключения по SSH с помощью ключа -p:

```
ssh -p 4312 hostname@remote_host
```

Если вы хотите, чтобы -p 4312 не нужно было вводить каждый раз, можно изменить файл конфигурации SSH на локальной машине следующим образом:

```
nano ~/.ssh/config
Host *alias_name*
HostName *remote_host*
Port *port*
User *username*
```

Теперь при подключении к удаленному узлу с помощью команды ssh *AliasName* порт будет выбран автоматически согласно соответствующей записи в /config.

Проброс авторизации

Проброс авторизации – это аутентификация на другом сервере через сервер, к которому вы подключены, используя ключ на вашем локальном компьютере. Иными словами, мы с узла A подключаемся к узлу B, а узел B, используя данные узла A, присоединяется с их помощью к узлу C.

На первом узле мы ввели команду подключения с ключом -A, тем самым

```
ssh -A root@*host_B*
```

На узле host_B мы ввели следующую команду и присоединились к узлу host_C под учетными данными первого узла.

```
root@host_B:~# ssh root@*host_C*
```

Таким образом, вы можете подключаться по SSH к любому другому устройству, к которому разрешен доступ с помощью вашего SSH-ключа, через промежуточное устройство.

Увеличение времени простоя

Время ожидания подключения может истечь – придется подключаться к системе заново. Чтобы этого избежать, можно настроить конфигурацию так, чтобы удаленный узел проверял активное SSH-соединение, отправляя клиенту echo-запросы каждые *ServerAliveInterval* секунд. Если клиент не ответит на запрос *ServerAliveCountMax* раз, то соединение будет разорвано.

Данные изменения нужно внести в конфигурацию файла /etc/ssh/sshd_config на сервере.

```
ServerAliveInterval *count*
ServerAliveCountMax *count*
```

Отключение проверки пароля

Ранее мы с вами настроили доступ по SSH с использованием SSH-ключей, но альтернативный вход по паролю по-прежнему включен. Чтобы обезопасить себя от возможного брутфорса (взлома пароля простым перебором), необходимо отключить возможность такого входа. Для этого на сервере нам необходимо открыть на редактирование файл конфигурации SSH под названием sshd_config:

```
sudo nano /etc/ssh/sshd_config
```

В нем найдем строку *PasswordAuthentication*, изменим ее значение с *yes* на *no* и сохраним изменения.

```
PasswordAuthentication no
```

Для того, чтобы изменения вступили в силу, следует перезагрузить службу SSH:

```
sudo service ssh restart (для Ubuntu / Debian / Mint Linux)
sudo service sshd restart (для CentOS / RHEL / Fedora Linux)
```

Теперь ваш сервер не будет рассматривать в качестве возможного варианта подключения по SSH использование пароля.

Устранение неполадок

Ключ игнорируется сервером

В редких случаях можно столкнуться с проблемой, что вы настроили все корректно, но не можете подключиться к серверу с использованием SSH-ключей. Видя следующий вывод, можно подумать, что сервер не видит ключ:

```
Permission denied (publickey).
```

Вероятнее всего, сервер SSH считает, что установлены неподходящие разрешения к некоторым каталогам. Для решения этой проблемы следует изменить права доступа.

Настройка удаленного узла

Полный доступ к папке .ssh только владельцу, остальным — полный запрет:

```
chmod rwx------ ~/.ssh
```

Права на запись и чтение для файла authorized_keys только владельцу, остальным — полный запрет:

```
chmod rw------- ~/.ssh/authorized_keys
```

Отмена записи группой и остальными пользователями:

```
chmod go-w ~/
```

Настройка локальной машины

Полный доступ к папке .ssh только владельцу, остальным — полный запрет:

```
chmod rwx------ ~/.ssh
```

Права на запись и чтение для файла ключа только владельцу, остальным — полный запрет:

```
chmod rw------- ~/.ssh/*key*
```

После применения этих политик ошибка должна исчезнуть.

Зашифрованный домашний каталог

Если у вас есть зашифрованный домашний каталог, то SSH не сможет получить доступ к файлу authorized_keys, пока вы не пройдете аутентификацию. Поэтому SSH по умолчанию будет использовать вход по паролю. Чтобы решить эту проблему, создайте папку вне домашнего каталога с именем /etc/ssh/*username*. Этот каталог должен иметь права доступа rwxr-xr-x и принадлежать пользователю. Переместите в

него файл authorized_keys (authorized_keys должен иметь права доступа

запись:

```
AuthorizedKeysFile /etc/ssh/%u/authorized_keys
```

Перезагрузите службу SSH.

```
sudo service ssh restart (для Ubuntu / Debian / Mint Linux)
sudo service sshd restart (для CentOS / RHEL / Fedora Linux)
```

При следующем подключении по SSH вам не нужно будет вводить пароль.

Анализ логов подключения

При возникновении ошибок во время подключения вы можете проанализировать файл /var/log/auth.log, в котором будут указаны все попытки подключения к системе, а также механизмы, использованные для аутентификации.

Также детальную информацию о подключении можно получить с помощью опций -v, -vv или -vvv. Чем больше ключей -v (но не более 3), тем подробнее будет лог.

```
ssh -vvv hostname@remote_host
```

Вызов справки

Подробное описание команды (ее параметров, значений и дрю.) вы можете найти в документации, которая вызывается командой *man*, например:

```
man ssh
man sshd
```

Заключение

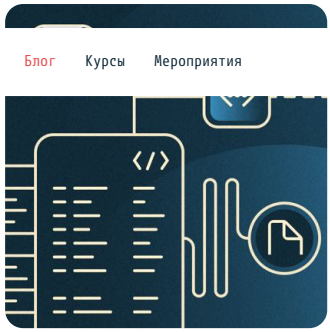
В тексте мы рассмотрели создание и авторизацию с помощью SSH-ключей на базе ОС Linux и Windows, а также разобрали некоторые ошибки, которые могут возникать при использовании такого способа авторизации. Использование SSH-ключей не только упрощает способ авторизации, но и увеличивает степень защиты вашего сервера.

Информационная безопасность и 152-ФЗ Сети

Читайте также:

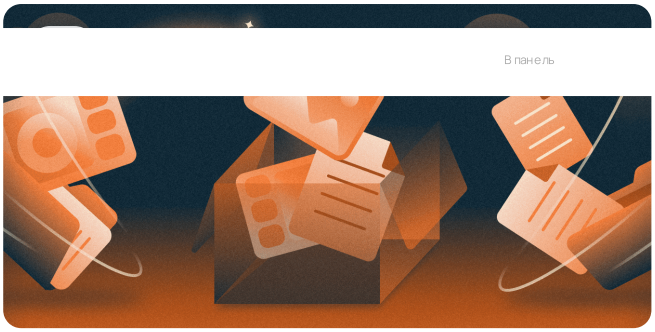
Новинки серверного железа за декабрь: процессоры Intel и видеокарты от AMD

Сергей Ковалев
Продакт-менеджер



«Хватит программировать в YAML и JSON!»: проблемы шаблонизирования

Анастасия Ербанова
Технический писатель



5 идей для pet-проектов, которые помогут прокачать навыки в Python

Анастасия Ербанова
Технический писатель

Блог Курсы Мероприятия

selectel.ru

Инвесторам О компании

Прессе Документация

Юристам Карьера

Email

Соглашаюсь на получение новостной рассылки в соответствии с Политикой