

SSH

Для доступа по ssh нужно знать к кому подключаться (ip или host name), порт (по умолчанию 22), имя пользователя и пароль. Также нужен клиент, вообще все протоколы из стека TCP/IP это клиент-серверные протоколы, т.е. есть сервер, ожидающий подключения от клиента, и есть клиент, иницирующий подключение к серверу. Почти во всех дистрибутивах линукс клиент ssh идет в комплекте и доступен по команде «ssh»

```
$ ssh host -p 22 -l user
```

где -p это номер порта, -l — имя пользователя. Можно подключаться и так:

```
$ ssh user@host -p22
```

а если порт 22 то

```
$ ssh user@host
```

Все три команды идентичны и приведут к одному и тому же результату.

После успешного подключения тебе предоставляется доступ к shell`у удаленной машины. С этого момента все команды будут отправлены на сервер, выполнены и их результат будет отправлен обратно. Выглядит это так же как и при работе с эмулятором терминала на своей машине.

Конфигурационный файл сервера sshd находится по пути "/etc/ssh/sshd_config", доступ к которому имеет только администратор. А каждый пользователь, имеющий доступ по ssh, в домашнем каталоге хранит папку .ssh, в которой располагаются ключи аутентификации, файл с хешем открытых ключей клиентов и хеши открытых ключей серверов.

SSH поддерживает разные алгоритмы аутентификации мы разберем 2 основных - это авторизации по паролю и по ключу. С паролем все ясно, а вот с ключами интереснее. Для начала нам нужна своя пара ключей.

Очень часто администраторы забывают о том, что ssh поддерживает безпарольную аутентификацию по ключам. Если раз добавить на сервер свой ключ, то доступ сохранится даже после смены пароля.

Наверное тут надо сказать, что алгоритмы шифрования делятся на 2 типа: симметричные и асимметричные; симметричные алгоритмы шифрования (AES, Blowfish, DES) используют один ключ для шифрования и дешифрования информации (типа пароль), асимметричные алгоритмы шифрования (RSA) используют пару ключей: открытый для шифрования и закрытый для дешифрования. Например Оля и Вася решили обмениваться зашифрованными с помощью RSA сообщениями. Для этого Вася генерирует себе пару ключей (открытый и закрытый) и Оля делает тоже самое. Чтобы начать общение Вася должен передать Оле свой открытый ключ и Оля должна передать Васе свой открытый ключ. Теперь Вася шифрует свое сообщение открытым ключом Оли, а Оля может прочитать это сообщение дешифровав своим закрытым ключом. По открытому ключу невозможно расшифровать сообщение или восстановить закрытый ключ, в свою очередь зная закрытый ключ можно восстановить открытый. Т.е. для обеспечения защиты передаваемой информации оба должны обменяться открытыми ключами и обеспечить сохранность своих закрытых ключей.

Для того чтобы создать пару ключей для аутентификации по ssh нужно ввести команду

```
$ ssh-keygen
```

Менеджер спросит куда сохранить ключи и парольную фразу. Если ввести пароль то для работы с ключами надо будет его вводить. Если не менять путь сохранения ключей, то лежать они будут по пути "~/.ssh/", там "id_rsa" - приватный ключ, "id_rsa.pub" - публичный ключ. Теперь нужно передать этот ключ на ssh сервер. Можно воспользоваться командой

```
$ ssh-copy-id user@host
```

Синтаксис тут такой же как и у "ssh". Будет запрошен пароль сервера. После этого можно коннектиться к серверу как обычно, так как файлы ключей лежат по пути по умолчанию то

```
$ ssh user@host
```

если же нет то во-первых: приватный ключ должен иметь права 600, во-вторых: публичный - 644; и нужно добавить ключ "-i"

```
$ ssh user@host -i /path/to/id_rsa
```

При этом при подключении публичный ключ уже не нужен, хотя я настоятельно рекомендую хранить оба ключа в одной папке и настроить на нее правильные права.

Существует и другой способ загрузки публичного ssh ключа.

К слову сказать, что один чел оказывал услугу по загрузке ssh ключей на сервера с веб шелом. Только представь, ему платили за то что я сейчас рассказываю. Платили потому, что он прикрыл лавочку по неизвестным причинам.

Как я уже сказал у пользователя есть директория .ssh, кстати ее можно и создать, так вот в этой директории есть файл "authorized_keys" в котором и хранятся публичные ключи загружаемые через "ssh-copy-id" и ничего не мешает тебе в ручную просто вставить свой публичный ключ в новую строку в этот файл. Результат будет одинаковый.

```
$ echo "your pub key" >> /home/user/.ssh/authorized_keys
```

Все, теперь даже если кто то сменит пароль пользователю, то мы все равно сохраним доступ, пока наш ключ не будет удален из файла authorized_keys.

Через ssh можно не только выполнять команды, а еще и серфить интернет, для этого трафик между локальной машиной и сервером туннелируется и во вне трафик идет уже с сервера. Для проброса одного порта (создания прокси) используется такая команда

```
$ ssh -T RPM@72.192.91.13 -D 8080
```

Параметр -T отключит псевдо терминал, а -D прибиндит порт. Таким образом на локал хосте у тебя будет висеть прокси на порту 8080. Пару скринов для РОС

Открывает доступ

mr. WHOER

Мой IP VPN Тест скорости Пинг Whois Статьи

Мой IP: 80.240.23.126 [Whois](#)
[Скрыть IP](#)

Местоположение: Germany (DE), Frankfurt Am Main
Провайдер: Choopa, LLC
Хост: 80.240.23.126.vultr.com
ОС: Mac OS X
Браузер: Firefox 56.0

Ваша анонимность: 48% О вас известно слишком много!

DNS: 74.125.46.8 Finland
Прокси: Нет
Анонимайзер: Нет
Черный список: Нет

Вот такой ип у меня сейчас. Теперь включаю ссш туннель и биндую его на порт 44444

```
ssh -T RPM@72.192.91.13 -i /home/unit/Projects/ipcheck/extra/id_rsa -D 44444
```

```
unit@unit-computer ~ [19:30:42]  
> $ ssh -T RPM@72.192.91.13 -i /home/unit/Projects/ipcheck/extra/id_rsa -D 44444  
bind: Cannot assign requested address
```

тут я еще указал путь до ключа. Теперь надо прописать прокси в браузер. Для менеджринга прокси я юзаю FoxyProxy

Add Proxy

Proxy Type ★
SOCKS5

Title or Description (optional)
SSH 127.0.0.1:44444

Color
#8636cc

IP address, DNS name, server name ★
127.0.0.1

Port ★
44444

Username (optional)

Password (optional)

Add whitelist pattern to match all URLs ☒ On

Do not use for localhost and intranet/private IP addresses ☒ On

[Help](#)

Send DNS through SOCKS5 proxy? ☒ On

Cancel Save & Add Another Save & Edit Patterns Save

Теперь мой внешний ип равен ип ссш сервера

Мой IP: **72.192.91.13** [Whois](#) [Скрыть IP](#)

Местоположение: United States (US), Broken Arrow

Провайдер: Cox Communications

Хост: ip72-192-91-13.tu.ok.cox.net

ОС: Mac OS X

Браузер: Firefox 56.0

Ваша анонимность: 63% Серьезные провалы в вашей безопасности и анонимности

DNS:	68.105.28.139 ²	United State
Прокси:	Да	
Анонимайзер:	Нет	
Черный список:	Нет	

Также поменялся провайдер и имя хоста и днс тоже, последнее потому что я указал что это SOCKS5 прокси, которые в свою очередь поддерживают форвардинг днс.

Еще один пример создания туннеля с порта на порт. К примеру на удаленном сервере запущен какой то сервис, например IP телефонии, который работает на порту 5901. И тебе нужно соединиться с сервером так чтобы трафик между клиентом и сервером был зашифрован, то ты будешь пробрасывать туннель на конкретный порт.

```
$ ssh user@host -L 5901:127.0.0.1:5901 -N
```

Теперь указываешь в клиенте SIP адрес хоста как 127.0.0.1 и он заработает. Магия! Такая практика используется для проброса порта сервера базы данных, когда настраивают горизонтальное масштабирование для балансировки нагрузки или когда 2 сервера юзают один сервер с БД или просто когда сервер с БД хостится отдельно, в общем в любой ситуации когда необходимо получить доступ к серверу БД удаленно. Это лучше чем светить порт сервера БД в интернет, по ряду причин, одна из очевидных - это безопасность.

Если и этого недостаточно и хочется большего, то встречайте sshuttle. Утилита легко и просто настроит тебе глобальное туннелирование всего системного трафика. Делается это так:

```
unit@unit-computer ~ [19:32:20]
> $ sshuttle --dns -r RPM@204.148.103.158 -e 'ssh -i /home/unit/Projects/ipcheck/extra/id_rsa'
0/0
[local sudo] Password:
client: Connected.
```

параметр --dns включит еще и форвардинг днс запросов, -e позволяет указать свою команду запуска ssh, в данном случае я указал путь до ключа. Выключаю прокси в браузере и результат, как говорится, на лицо ;-)

The screenshot shows the WhoerVPN website interface. At the top, there's a navigation bar with a language selector set to 'Русский' and social media links. The main header features the 'WhoerVPN' logo and a list of features: 'Безопасный', 'Быстрый', 'Анонимный', and 'Открывает доступ'. A price tag of '4\$ в месяц при подписке на год' is also visible. Below the header, there are icons for 'Мой IP', 'VPN', 'Тест скорости', 'Пинг', 'Whois', and 'Статьи'. The main content area is divided into two sections. The left section, titled 'Мой IP:', displays the IP address '204.148.103.158' with a 'Whois' button and a 'Скрыть IP' link. It also shows location details: 'United States (US), New York', provider 'ANS Communications', host 'tbd-gw.customer.alter.net', OS 'Mac OS X', and browser 'Firefox 56.0'. The right section, titled 'Ваша анонимность: 60%', includes a warning about security and anonymity. It lists various services and their status: 'DNS: 74.125.44.88 United State', 'Прокси: Нет', 'Анонимайзер: Нет', and 'Черный список: Нет'.

И как я уже сказал, трафик туннелируется весь

```
unit@unit-computer ~ [19:37:30]
> $ curl http://ifconfig.me/ip -A "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:56.0) Gecko/20100101 Firefox/56.0"
204.148.103.158

unit@unit-computer ~ [19:37:37]
> $ _
```

ifconfig.me упорно отказывается отвечать если юзер агент не указывать

Заметил, что при использовании sshuttle на вхуере не светится, что я использую прокси? Чем же туннелирование лучше ТОРа в котором тоже невозможно перехватить и дешифровать трафик между тобой и выходной нодой? SSH туннели не поднимают куча правительственных организаций, чтобы мониторить пользователей ssh - это раз, быстрее чем ТОР - это два. И все это исключительно мое мнение!

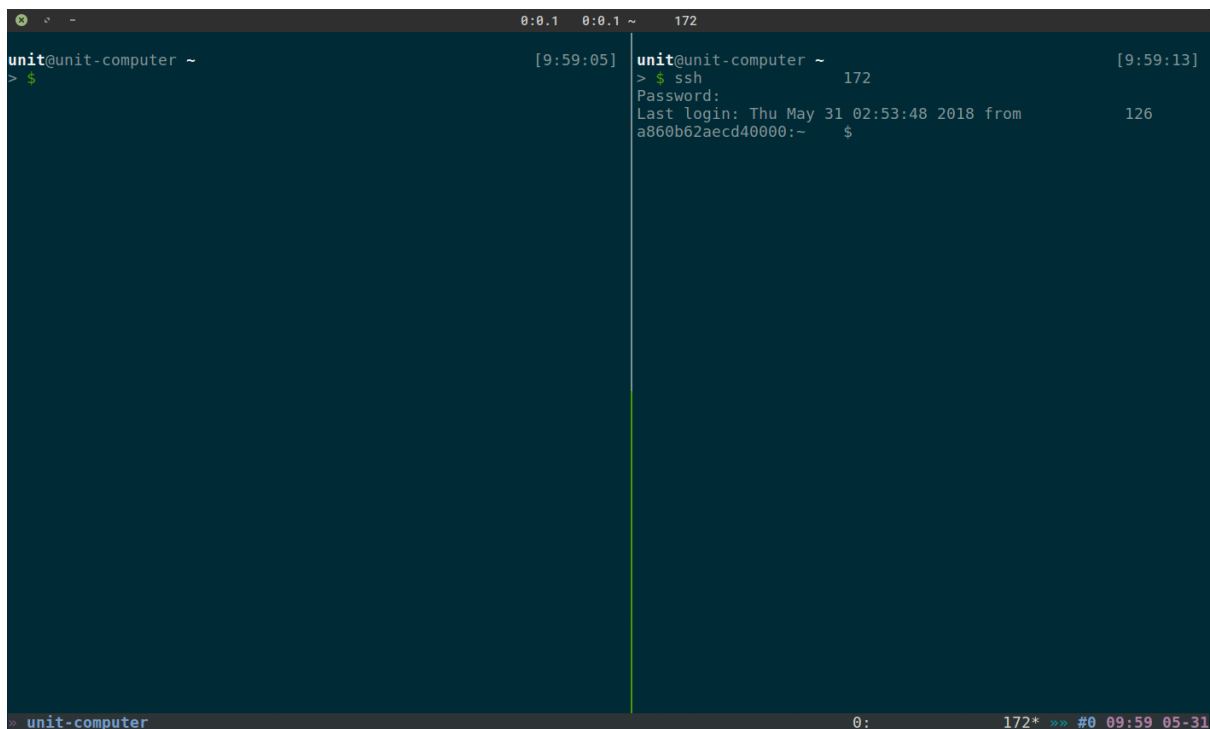
Если ты полюбишь ссш, то придется полюбить и брутить ссш любишь кататься люби и с санками ебаться. А для начала вспомним, как можно узнать кто работает на машине? Правильно w/who. А что же делать, спросишь ты? Все достаточно просто. SSH позволяет последним параметром указать команду, которая будет выполнена на удаленном сервере (такой тип параметров называется позиционным). Если сделать вот так:

```
$ ssh user@host "uname -a"
```

то ты увидишь информацию о системе на удаленном сервере и будешь отключен от него. Ну а чтобы не палить свою сессию можно подключиться к серверу вот так:

```
$ ssh user@host "/bin/sh -i"
```

и получить сессию sh в интерактивном режиме.



```
unit@unit-computer ~ [9:59:05]
> $

unit@unit-computer ~ [9:59:13]
> $ ssh 172
Password:
Last login: Thu May 31 02:53:48 2018 from 126
a860b62aec40000:~ $
```

Видно, что я подключился к серверу дважды, а в списке w видно только одну сессию, это как раз потому что вторую сессию я запускал с указанием "/bin/sh -i" последним

параметром. Таким нехитрым способом можно скрыть свое присутствие. Хотя тебя все равно можно обнаружить, просто для этого придется затратить больше усилий. Кстати пробрасывать туннель тоже лучше с указанием `"/bin/sh -i"`.

Теперь о брUTE ssh. Брутить можно любым удобным способом, например гидрой. Тут ничего сложного нет, составить список для брута, взять пачку диапазонов и запустить софт. Можно уведичить конверсию запуская брут с захваченных серверов. Тут возникает несколько трудностей: отсутствие прав на установку гидры на сервер - это раз, наличие самой гидры уже палево - это два, если сервер потерять, то теряются все набрученные хосты - это три, ну и так далее.