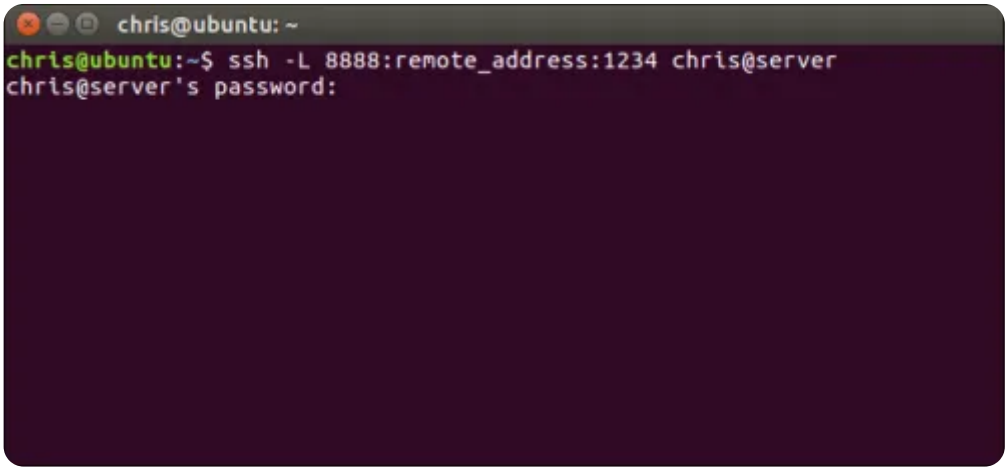
Учимся использовать SSH туннелирование для безопасного серфинга

СайтостроениеСтатьиИнтернетБезопасность2 года назад 6.1K

Существует три различных типа туннелирования, и все они используются для решения разных задач. Каждая задача предполагает использование **SSH-сервера** для перенаправления трафика из одного сетевого порта в другой. Трафик передается по зашифрованному **SSH-соединению**, поэтому его нельзя отследить или изменить в процессе передачи:



Туннелирование можно реализовать с помощью ssh-команды в **Linux**, **Mac OS** и операционных системах семейства **UNIX**. Для пользователей **Windows**, где нет встроенной **ssh-команды**, мы предлагаем бесплатный инструмент **PuTTY**. Он умеет подключаться к **SSH-серверам**. Он также поддерживает **SSH-туннелирование**.

Содержание

Локальное перенаправление портов (port forwarding): получаем доступ к удалённым ресурсам на локальной системе

«*Локальное перенаправление портов*» позволяет осуществлять доступ к ресурсам, находящимся внутри локальной сети. Предположим, что нужно попасть на офисный сервер БД, сидя дома. В целях безопасности этот сервер настроен так, чтобы принимать подключения только с ПК, находящихся в

локальной сети офиса. Но если у вас есть доступ к **SSH-серверу**, находящемуся в офисе, и этот **SSH-сервер** разрешает подключения из-за пределов офисной сети, то к нему можно подключиться из дома. Затем осуществить доступ к БД. Проще защитить от атак один **SSH-сервер**, чем защищать каждый ресурс локальной сети по отдельности.

Чтобы сделать это, вы устанавливаете **SSH-соединение** с **SSH-сервером** и говорите клиенту передать трафик с указанного порта на локальном ПК. Например, с **порта 1234** на адрес сервера базы данных и его порт внутри офисной сети. Когда вы пытаетесь получить доступ к БД через **порт 1234** на вашем ПК («***localhost***») трафик автоматически «*туннелируется*» по **SSH-соединению** и отправляется на сервер БД.

SSH-сервер выступает посредником, пересылая трафик туда-сюда. При этом можно использовать любую командную строку или графический инструмент для осуществления доступа к базе данных, как вы обычно делаете это на локальном ПК.

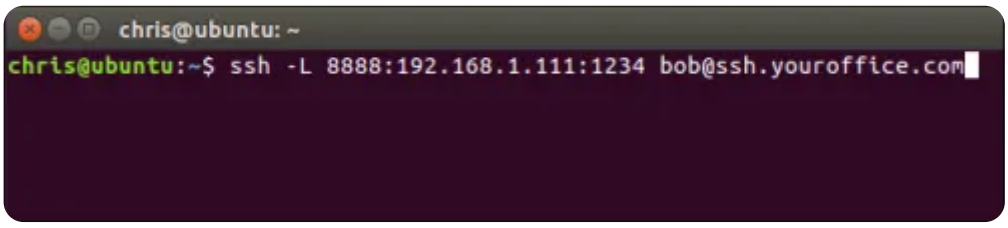
Чтобы использовать локальное перенаправление, подключитесь к **SSH-серверу** с использованием вспомогательного аргумента **-L**. Синтаксис для туннелирования трафика будет следующим:

```
ssh -L local_port:remote_address:remote_port username@server.com
```

Предположим, что офисный сервер находится по адресу **192.168.1.111**. У вас есть доступ к **SSH-серверу** через адрес **ssh.youroffice.com**, и имя вашего аккаунта на **SSH-сервере — bob**. В таком случае необходимая команда будет выглядеть следующим образом:

```
ssh -L 8888:192.168.1.111:1234 bob@ssh.youroffice.com
```

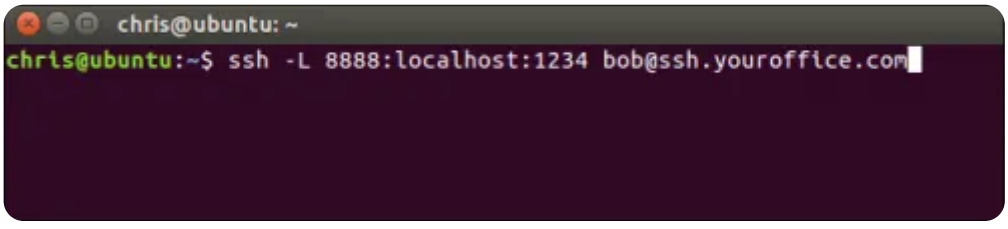
Запустив эту команду, вы попадете на офисный сервер баз данных через **порт 8888** на **localhost**. Если у СУБД есть веб-интерфейс, можно вписать в адресную строку браузера **http://localhost:8888**. Если у вас инструмент командной строки, которому необходим сетевой адрес базы данных, то направьте его на **localhost:8888**. Весь трафик, отправленный на **порт 8888** на ПК, будет перенаправлен на **192.168.1.111:1234** внутри офисной сети:



Это слегка сбивает с толку, если надо подключиться к серверному приложению, запущенному в той же системе, где и сам **SSH-сервер**. К примеру, есть **SSH-сервер**, работающий на порте 22 на офисном ПК. Но у вас также есть сервер баз данных, работающий на **порте 1234** в той же системе по тому же адресу. Вам нужно подключиться к БД из дома, но система принимает только **SSH-подключение** через **22 порт**, и сетевой экран не пропускает любые внешние подключения. В таком случае можно запустить следующую команду:

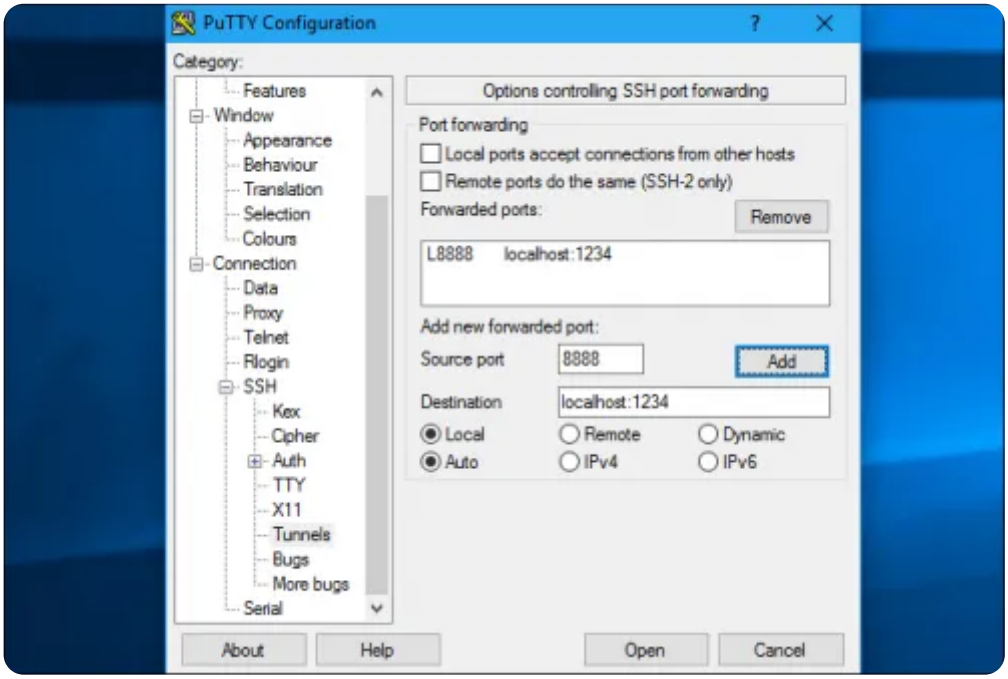
```
ssh -L 8888:localhost:1234 bob@ssh.youroffice.com
```

При попытке подключиться к БД через **8888 порт** на вашем ПК, трафик будет передаваться с помощью **SSH-подключения**. Когда он достигнет системы, в которой работает **SSH**, **SSH-сервер**отправит его на **порт 1234** на «***localhost***», принадлежащий тому же ПК, на котором запущен **SSH-сервер**. То есть, «***localhost***» в приведённой выше команде означает «***localhost***» с перспективы удалённого сервера:



Чтобы сделать это в **PuTTY** на **Windows**, выберите опцию **Connection > SSH > Tunnels**. Далее опцию «***Local***». В поле «***Source Port***» укажите локальный порт. В поле «***Destination***» введите целевой адрес и порт в формате **удалённый_адрес:удалённый_порт**.

Например, если нужно настроить **SSH-тоннель**, как это сделано выше, то введите **8888** в качестве порта-источника и **localhost:1234** в качестве целевого адреса. После этого нажмите «***Add***» и затем «***Open***», чтобы открыть **SSH-подключение**. До подключения **SSH**туннелирования нужно ввести адрес и порт самого **SSH-сервера** в разделе «***Session***»:



Дистанционное перенаправление портов: открываем доступ к локальным ресурсам на удалённой системе

«*Дистанционное перенаправление портов*» — ситуация, противоположная локальному перенаправлению, и используется не так часто. Она позволяет открывать доступ к ресурсам на локальном ПК через **SSH-сервер**. Предположим, что на локальном ПК настроен веб-сервер. Но ваш ПК защищён сетевым экраном, который не пропускает входящий трафик на сервер.

Если есть доступ к удалённому **SSH-серверу**, можно подключиться к этому **SSH-серверу** и использовать дистанционное перенаправление портов. Ваш **SSH-клиент** укажет серверу перенаправлять трафик с определённого порта – скажем, **1234** – на **SSH-сервере** на указанный адрес и порт на вашем ПК или внутри локальной сети. Когда кто-то подключается к **порту 1234**на **SSH-сервере**, этот трафик автоматически «*туннелируется*» по **SSH-соединению**. Любой, кто подключается к **SSH-серверу**, сможет получить доступ к серверу, запущенному на вашем ПК. Это достаточно эффективный способ обхода фаерволов.

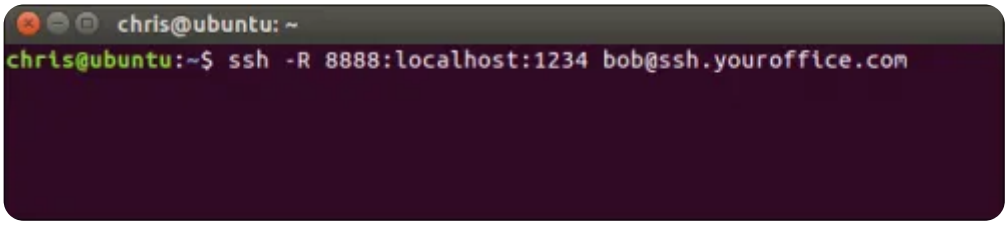
Чтобы воспользоваться дистанционным туннелированием **IP**, используйте **ssh-команду** с аргументом —**R**. Синтаксис здесь будет практически таким же, как и в случае с локальным перенаправлением:

```
ssh -R remote_port:local_address:local_port username@server.com
```

Предположим, что нужно создать серверное приложение, прослушивающее **порт 1234** на вашем ПК. Оно доступно через **порт 8888** на удалённом **SSH-сервере**. Адрес **SSH-сервера ssh.youroffice.com**, а ваше имя пользователя на **SSH-сервере bob**. Значит, команда будет следующей:

```
ssh -R 8888:localhost:1234 bob@ssh.youroffice.com
```

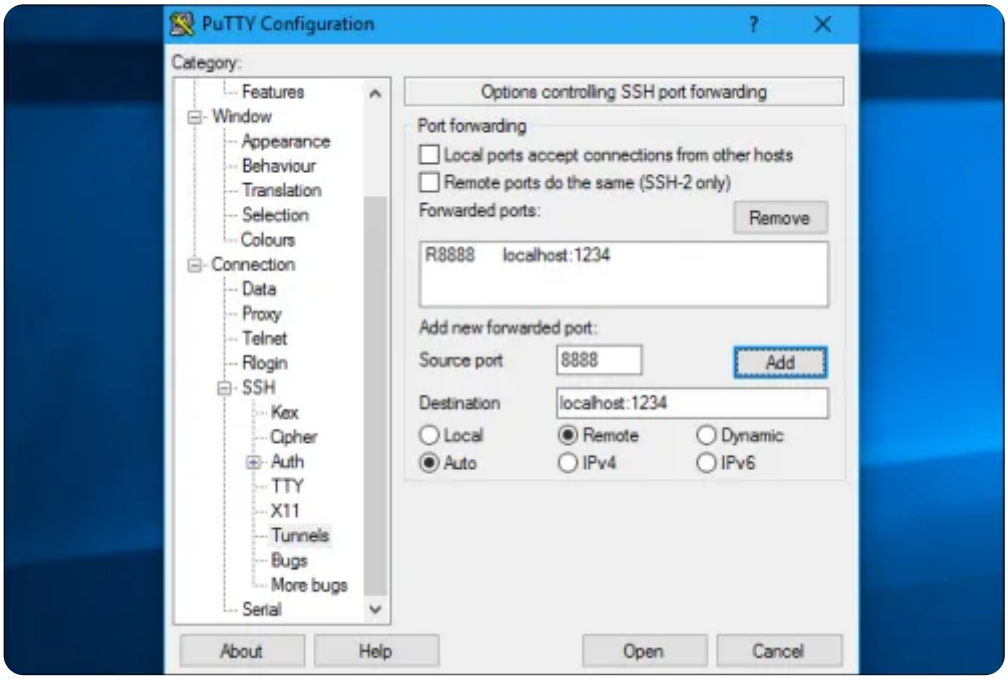
Затем кто-то может подключиться к **SSH-серверу** через **порт 8888**, и это подключение будет туннелировано на серверное приложение, запущенное на порте **1234 ПК**, с которого вы подключались:



Чтобы сделать это в **PuTTY** для **Windows**, выберите опцию **Connection > SSH > Tunnels**. Далее – опцию «***Remote***». В поле «***Source Port***» укажите удалённый порт. В поле «***Destination***» введите целевой адрес и порт в формате **локальный_адрес:локальный_порт**.

Например, если нужно настроить **SSH-тоннель**, как это сделано выше, то укажите **8888** в качестве порта-источника и **localhost:1234** в качестве целевого адреса. После этого нажмите «***Add***» и затем «***Open***», чтобы открыть **SSH-подключение**. До подключения нужно будет ввести адрес и порт самого **SSH-сервера** в разделе «***Session***».

После этого пользователи смогут подключаться к **порту 8888** на **SSH-сервере** и их трафик будет передаваться на **порт 1234** на вашей локальной системе:



По умолчанию, удалённый **SSH-сервер** будет слушать только подключения с того же хоста. Другими словами, доступ будет только у людей из той же системы, на которой запущен **SSH-сервер**. При туннелировании трафика так делается в целях безопасности.

Нужно включить опцию «***GatewayPorts***» в sshd_config на удалённом **SSH-сервере**, если хотите изменить эти настройки.

Динамическое перенаправление портов: используем SSH-сервер в качестве прокси

Также существует «*динамическое перенаправление портов*», которое работает по тому же принципу что прокси или **VPN-сервер**. **SSH-клиент** создаёт **SOCKS-прокси**, который можно настраивать под собственные приложения. Весь трафик, отправляемый через прокси, будет отправляться через **SSH-сервер**. Принцип здесь схож с локальным перенаправлением – берётся локальный трафик, отправленный на определённый порт на вашем ПК, и перенаправляется через **SSH-соединение** на удалённый адрес.

Предположим, что вы используете общедоступную **Wi-Fi** сеть. Но хочется делать это безопасно. Если у вас есть доступ к **SSH-серверу** из дома, то можно подключиться к нему и использовать динамическое перенаправление. **SSH-клиент** создаст **SOCKS-прокси** на вашем ПК. Весь трафик, отправленный на этот прокси, будет отправляться через подключение к **SSH-серверу**. Никто из тех, кто использует общедоступную **Wi-Fi** сеть, не сможет отслеживать ваши перемещения в сети или закрывать доступ к сайтам. С перспективы сайтов, которые посещаете, будет казаться, что вы заходите на них с домашнего ПК.

Или же может понадобиться подключиться к медиа-серверу, находящемуся в вашей домашней сети. В целях безопасности, к интернету подключен только ваш **SSH-сервер**. При этом вы не разрешаете подключаться к медиа-серверу через интернет. В таком случае можно включить динамическое перенаправление портов, настроить **SOCKS-прокси** в браузере и затем подключаться к серверам, работающим в домашней сети, через браузер, как будто вы сидите дома.

Например, если медиа-сервер находится по адресу **192.168.1.123** в вашей домашней сети, то можно добавить адрес **192.168.1.123** в любое приложение при помощи **SOCKS-прокси** и получить доступ к медиа-серверу, как будто вы находитесь внутри домашней сети.

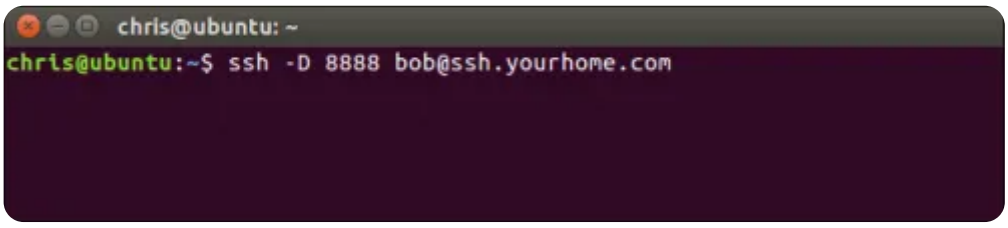
Чтобы воспользоваться динамическим перенаправлением, запустите **ssh-команду** с аргументом —**D**:

```
ssh -D local_port username@server.com
```

Предположим, что у вас есть доступ к **SSH-серверу** по адресу **ssh.yourhome.com**, а ваш логин на **SSH-сервере** – **bob**. Нужно использовать динамическое перенаправление для того, чтобы открыть **SOCKS-прокси** по **порту 8888** на текущем ПК. Тогда команда для **SSH** туннелирования будет выглядеть следующим образом:

```
ssh -D 8888 bob@ssh.yourhome.com
```

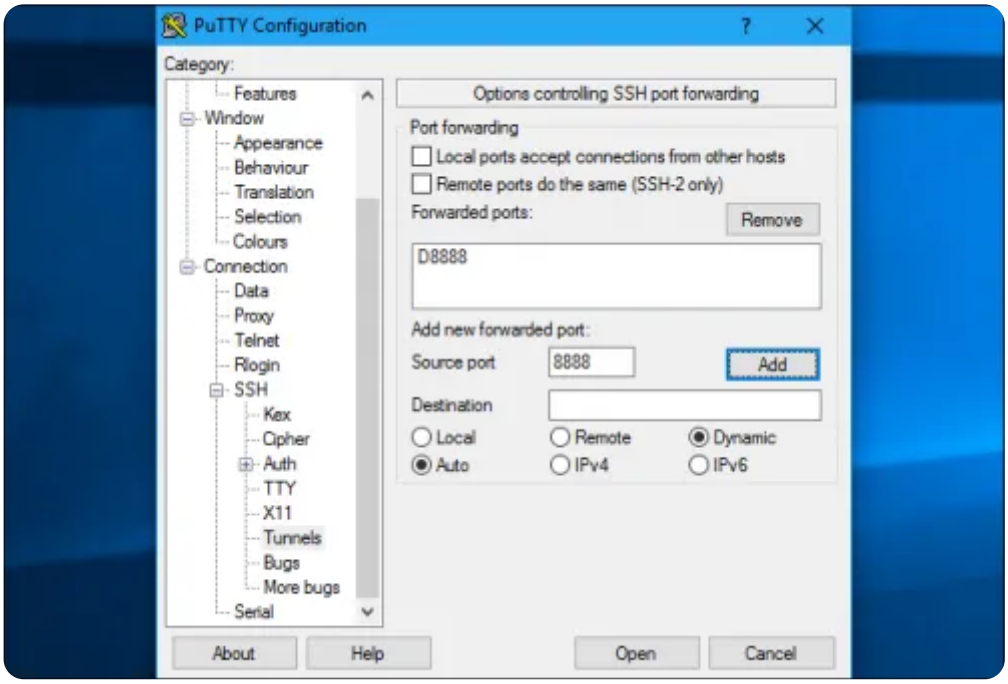
После этого можно настроить браузер или другое приложение на использование локального **IP-адреса (127.0.0.1)** и **порта 8888**. Весь трафик этого приложения будет перенаправляться через туннель:



Чтобы сделать это в **PuTTY** для **Windows**, выберите опцию **Connection > SSH > Tunnels**. Далее – опцию «***Dynamic***». В поле «***Source Port***» укажите локальный порт.

Например, если вам нужно настроить **SOCKS-прокси** на **порт 8888**, то введите **8888** в качестве порта-источника. После этого нажмите «***Add***» и затем «***Open***», чтобы открыть **SSH-подключение**.

После этого можно настроить приложение на подключение через **SOCKS-прокси** на вашем локальном ПК (*то есть, по **IP-адресу 127.0.0.1**, который ведёт на ваш локальный ПК*) и указать корректный порт для работы:



К примеру, можно настроить браузер **Firefox** на использование **SOCKS-прокси**. Это удобно, так как у **Firefox** могут быть отдельные настройки прокси, и поэтому не обязательно использовать базовые параметры для всей системы. **Firefox** будет отправлять трафик через **SSH туннелирование**, а другие приложения будут использовать интернет-подключение в обычном режиме.

Туннель будет оставаться активным и открытым до тех пор, пока открыта сессия **SSH-соединения**. Когда вы завершите **SSH-сессию** и отключаетесь от сервера, туннель тоже закроется. Чтобы снова открыть туннель, переподключитесь при помощи соответствующей команды или нужной функции в **PuTTY**.