

# Настраиваем Port Knocking в Linux (Debian / Ubuntu)

 [interface31.ru/tech\\_it/2021/02/nastraivaem-port-knocking-v-linux-debian-ubuntu.html](https://interface31.ru/tech_it/2021/02/nastraivaem-port-knocking-v-linux-debian-ubuntu.html)

## Записки IT специалиста

Технический блог специалистов ООО "Интерфейс"

- [Главная](#)
- Настраиваем Port Knocking в Linux (Debian / Ubuntu)

Безопасность - это всегда компромисс между многими показателями, включающими, в том числе, удобство пользования. Современные реалии предусматривают широкое применение удаленного доступа, но при этом не все служебные сервисы администраторы хотят открывать наружу. Одним из вариантов решения проблемы является VPN, но это применимо не всегда и не везде. Хорошей альтернативой в этом случае может являться Port Knocking - специальная технология, позволяющая буквально "постучать" в порты особым образом, после чего вы сможете получить доступ к системе. Стучите и вам откроют.



### Онлайн-курс по устройству компьютерных сетей

На углубленном курсе "[Архитектура современных компьютерных сетей](#)" вы с нуля научитесь работать с Wireshark и «под микроскопом» изучите работу сетевых протоколов. На протяжении курса надо будет выполнить более пятидесяти лабораторных работ в Wireshark.

В переводе с английского **Port Knocking** буквально означает стук в порт, по аналогии стука в дверь. В детстве многие из нас использовали подобный "секретный стук" приходя в гости к другу, домофонов тогда еще не было, и мы просто особым образом стучали или звонили в дверной звонок. Друг сразу понимал, что пришли к нему и бежал открывать.

Подобный принцип используется и здесь - нужно за определенное время в нужном порядке обратиться к определенным портам системы, после чего вам на некоторое время будет открыт доступ к служебному порту, чтобы вы могли подключиться. Для всех остальных порт будет закрыт.

### Предварительная настройка брандмауэра

Чтобы Port Knocking работал в системе должен быть правильно настроен брандмауэр. В рамках данной статьи мы не будем касаться подробностей, всю необходимую теоретическую часть вы можете найти в нашей статье: [Основы iptables для начинающих. Часть 2. Таблица filter](#). В частности, это касается цепочки INPUT, которая отвечает за фильтрацию входящих подключений. Она обязательно

должна начинаться с правила, разрешающего уже установленные и связанные соединения, а завершаться правилом, запрещающим любые входящие подключения, между ними должны располагаться правила, разрешающие доступ к отдельным сервисам.

В любом случае в минимальной конфигурации должны присутствовать следующие два правила, где **ens33** - имя внешнего интерфейса:

```
# Разрешаем установленные и связанные соединения
iptables -A INPUT -i ens33 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
#Запрещаем входящие извне
iptables -A INPUT -i ens33 -j DROP
```

Данные команды можно прямо ввести в консоли, и они начнут действовать сразу. Но будьте внимательны, если вы выполните данные команды удаленно, подключаясь через внешний интерфейс - то **полностью потеряете связь** с сервером до его перезагрузки. В этом случае добавьте между этими правилами еще одно и не удаляйте до тех пор, пока не убедитесь, что Port Knocking работает как требуется.

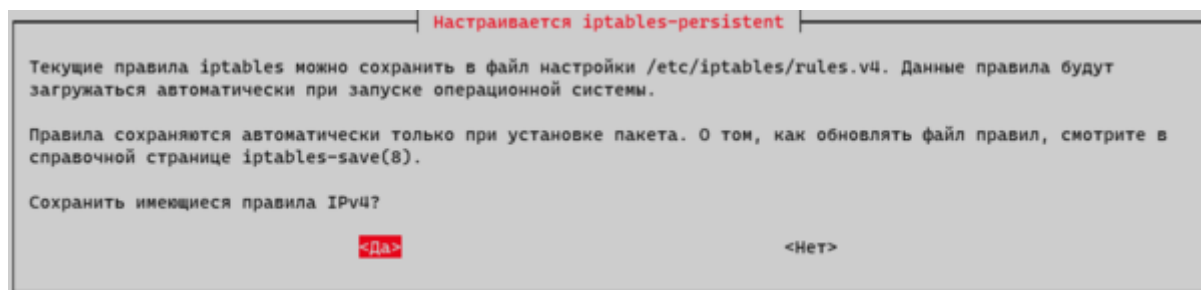
```
# Разрешаем подключения по SSH
iptables -A INPUT -i ens33 -p tcp --dport 22 -j ACCEPT
```

Если вы вводите указанные выше команды в консоли, то разрешающее правило для SSH следует вводить вторым.

После чего установим пакет **iptables-persistent** для автоматической загрузки правил **iptables**:

```
apt install iptables-persistent
```

В процессе установки пакета будет предложено сохранить текущие правила, с этим следует согласиться



В дальнейшем, если вы внесли какие-либо изменения в правила iptables через консоль их можно будет сохранить командой:

```
iptables-save > /etc/iptables/rules.v4
```

либо непосредственно откорректировать файл **/etc/iptables/rules.v4**.

## Установка knockd

---

Для реализации технологии Port Knocking нам потребуется пакет **knockd**, установим его:

```
apt install knockd
```

Затем откроем файл **/etc/default/knockd** и установим следующую опцию:

```
START_KNOCKD=1
```

Затем создадим файл юнита **systemd**:

```
touch /etc/systemd/system/knockd.service
```

И внесем в него следующие строки:

```
[Unit]
Description=Port-Knock Daemon
After=network.target
Requires=network.target
Documentation=man:knockd(1)

[Service]
EnvironmentFile=-/etc/default/knockd
ExecStartPre=/usr/bin/sleep 1
ExecStart=/usr/sbin/knockd $KNOCKD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=mixed
Restart=always
SuccessExitStatus=0 2 15
ProtectSystem=full
CapabilityBoundingSet=CAP_NET_RAW CAP_NET_ADMIN

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

Затем перечитаем изменения и добавим сервис **knockd** в автозагрузку:

```
systemctl daemon-reload
systemctl enable knockd
```

Но не спешите запускать саму службу, с большой долей вероятности вы получите ошибку, связанную с неправильными настройками.

## Настройка knockd

---

Откроем файл **/etc/knockd.conf** и удалим оттуда все, кроме первой секции **options**, которую приведем к следующему виду:

```
[options]
    UseSyslog
    Interface = ens33
```

Первая опция указывает записывать события в системный лог, вторая задает интерфейс, на котором будет слушать **knockd**.

Теперь создадим секцию для доступа к SSH:

```
[SSH]
    sequence = 7000,9000,8000
    seq_timeout = 5
    tcpflags = syn
    start_command = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
    stop_command = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
    cmd_timeout = 60
```

Опция **sequence** задает последовательность "стуков", по умолчанию используется протокол **tcp**, если вы хотите использовать **udp**, то следует указать его через двоеточие, например:

```
sequence = 7000,9000:udp,8000
```

**seq\_timeout** - время в секундах за которое клиент должен совершить всю последовательность "стуков", следует сделать его минимальным, но при этом следует учитывать возможные задержки каналов связи как сервера, так и клиента, **tcpflags** - определяет TCP-флаги пакетов, которые будут участвовать в последовательности, **syn** рекомендуется использовать совместно с SSH, так как SSH-трафик может мешать knockd, делая последовательность недействительной.

Опции **start\_command** и **stop\_command** определяют команды которые должен выполнить knockd с интервалом указанным в опции **cmd\_timeout**. Принцип действия здесь прост - получив указанную последовательность "стуков" сервис выполняет команду из опции **start\_command**, а по истечении заданного таймаута команду **stop\_command**.

Наша задача - открыть доступ к порту SSH на время достаточное для установления подключения. Поэтому первой командой мы добавляем самым первым правилом в цепочку INPUT разрешение подключиться к SSH с адреса источника "стука". Обратите внимание, что для этого мы используем ключ **-I**, который без указания номера добавляет правило первой строкой, а также подстановочную конструкцию **-s %IP%**, которая подставит в правило текущий адрес "стучавшего". Вторая команда удаляет уже добавленное правило.

Таким образом порт будет открыт на подключение только для адреса клиента и только на время, указанное в таймауте. Этот момент обычно вызывает у начинающих некоторое недоумение: а каким образом будет работать соединение, если мы закрыли порт? Все очень просто, доступ к порту 22 нам нужен только для

установления соединения, после чего оно перейдет в состояние ESTABLISHED и будет разрешено первым заданным нами правилом в цепочке INPUT, которое разрешает установленные и связанные подключения.

Сохраним конфигурацию и запустим сервис:

```
systemctl start knockd
```

Теперь постучимся, в Linux для этого можно использовать команду **knock**, а для Windows можно скачать официальный клиент со [страницы разработчика](#). Перед этим проверим состояние брандмауэра. Мы использовали самый простой набор из двух правил: разрешающего установленные соединения и запрещающего остальное.

```
root@debian-uefi:~# iptables -L -vn
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination          state
 5064 433K ACCEPT     all  --  ens33  *      0.0.0.0/0            0.0.0.0/0            RELATED,ESTABLISHED
 299 33182 DROP      all  --  ens33  *      0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 4477 packets, 1412K bytes)
 pkts bytes target     prot opt in     out     source               destination
root@debian-uefi:~#
```

Как видим - все закрыто, подключиться к серверу невозможно. А теперь стучим:

```
knock IP_сервера 7000 9000 8000
```

И снова проверяем брандмауэр, самым первым в цепочке INPUT появилось правило, разрешающее доступ к SSH с адреса "стучавшего" клиента.

```
root@debian-uefi:~# iptables -L -vn
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination          tcp dpt:22
 0      0 ACCEPT     tcp  --  *      *      192.168.233.132      0.0.0.0/0            state RELATED,ESTABLISHED
 5087 434K ACCEPT     all  --  ens33  *      0.0.0.0/0            0.0.0.0/0
 302 33362 DROP      all  --  ens33  *      0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 4498 packets, 1415K bytes)
 pkts bytes target     prot opt in     out     source               destination
root@debian-uefi:~#
```

Ровно через указанное в таймауте время, в нашем случае 60 секунд, правило исчезнет, поэтому выбирайте данное значение небольшим, но достаточным для выполнения всех действий по подключению.

Но возможности **knockd** не исчерпываются управлением брандмауэром, сама возможность выполнить произвольную команду открывает довольно широкие перспективы, ограниченные фантазией и здравым смыслом. Например, мы можем запустить произвольный скрипт, скажем, для резервного копирования.

Снова откроем файл **/etc/knockd.conf** и добавим в него еще одну секцию:

```
[MyScript]
sequence = 9000,7000,8000
seq_timeout = 5
tcpflags = syn
command = /path/to/myscript.sh
```

Основным отличием от предыдущей секции является опция **command**, при верной последовательности стуков служба просто выполняет указанную в ней команду. Таким образом в руки администратора попадает достаточно мощный и удобный инструмент, который может оказаться полезным в некоторых ситуациях.

Но есть у Port Knocking и существенный недостаток: если кто-то имеет возможность прослушивать ваш трафик (например, на публичной точке доступа), то вполне может перехватить вашу последовательность и чем чаще вы "стучитесь", тем быстрее это можно сделать. Как быть? В этом случае следует воспользоваться еще одной возможностью knockd - одноразовыми последовательностями.

Для этого замените опцию sequence в нужной секции опцией **one\_time\_sequences** с указанием пути к файлу одноразовых последовательностей:

```
one_time_sequences=/root/ssh-sequences
```

**Важно!** Не следует располагать данный файл в директории **/etc**, в этом случае knockd будет выдавать ошибку при запуске службы.

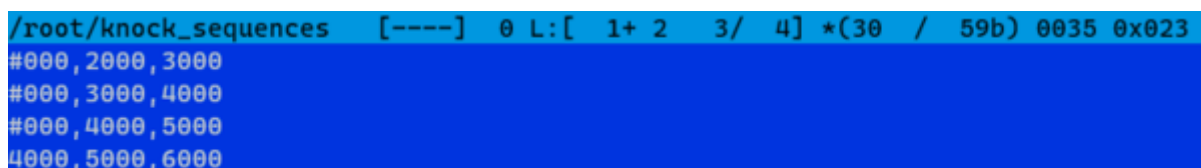
В сам файл внесите выбранные вами последовательности по одной на строку, например:

```
7000,9000,8000
9000,7000:udp,8000
...
8500,7500,9500:udp
```

Затем перезапустите службу:

```
systemctl restart knockd
```

И попробуйте постучаться, после каждого успешного "стука" **knockd** будет ставить комментарий в начале сработавшей строки.



```
/root/knock_sequences  [----]  0 L:[  1+ 2  3/  4]  *(30 /  59b) 0035 0x023
#000,2000,3000
#000,3000,4000
#000,4000,5000
4000,5000,6000
```

Но будьте внимательны, используя данный метод, если у вас закончатся одноразовые последовательности, то вы рискуете остаться полностью без доступа к серверу.

## **Онлайн-курс по устройству компьютерных сетей**

На углубленном курсе "Архитектура современных компьютерных сетей" вы с нуля научитесь работать с Wireshark и «под микроскопом» изучите работу сетевых протоколов. На протяжении курса надо будет выполнить более пятидесяти лабораторных работ в Wireshark.

Помогла статья? Поддержи автора и новые статьи будут выходить чаще:

---