Хакер - Твой тайный туннель. Детальный гайд по настройке OpenVPN и stunnel для создания защищенного канала

nopaywall


https://t.me/nopaywall

## Содержание статьи



У тебя могут быть самые разные мотивы, чтобы пользоваться VPN: недоверенные сети, разного рода ограничения или просто разумное желание не распространять лишний раз свои данные. В этой статье я расскажу, как сделать себе личный VPN на арендованном сервере и настроить OpenVPN и stunnel таким образом, чтобы даже глубокая инспекция пакетов ничего не давала.

# О сервисах и блокировках

Существует бесчисленное множество сервисов, которые предоставляют VPN, в том числе и бесплатные. Вот несколько причин, почему бесплатный VPN — это плохая идея.

1. Качество. Те, кто пользовался бесплатным VPN, знают, что в большинстве случаев сервис просто ужасен: низкая скорость, постоянные обрывы. Это и неудивительно, ведь, кроме тебя, им одновременно может пользоваться еще пара сотен человек.
2. Безопасность. Даже если качество более-менее сносное, ты не знаешь, что на самом деле происходит с твоим трафиком. Хранится и анализируется ли он, кто и в каких целях оперирует сервисом. Бесплатный сыр, как говорится…
3. Малое количество или полное отсутствие опций и настроек: нет возможности выбрать шифр, протокол и порт. Остается только пользоваться тем, что дали.

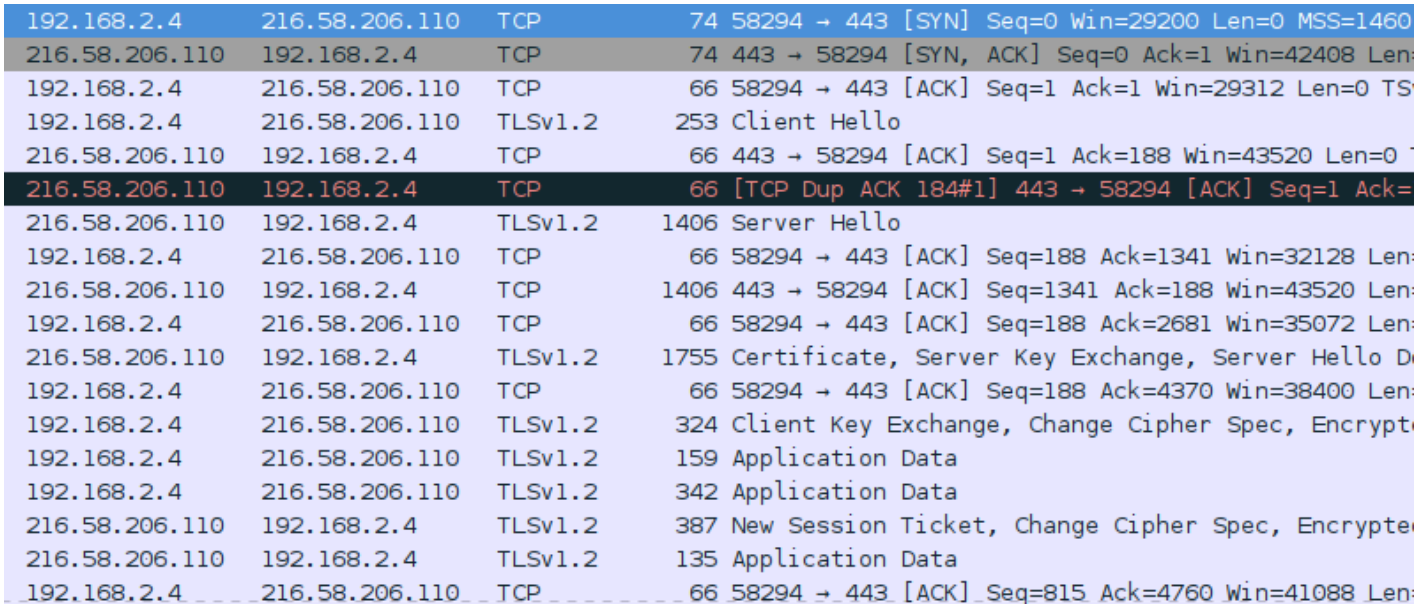
С платными сервисами дела обстоят лучше: можно ожидать какого-то гарантированного качества и наличия настроек. Но ты все еще не можешь знать наверняка, хранятся твои логи непосредственно на сервере или нет. К тому же твоего провайдера могут заблокировать.

Великий китайский файрвол, к примеру, научили определять и блокировать трафик OpenVPN при помощи техники Deep packet inspection (DPI). На какой бы порт ты его ни прятал, будь то UDP 53 или TCP 443, в Китае просто так OpenVPN не попользуешься. Дело в том, что в режиме TLS трафик OpenVPN отличается от обычного трафика HTTPS. Если под рукой есть сниффер, в этом несложно убедиться.

| | | | |
|---|---|---|---|
| 192.168.2.4 | 52.212.223.6 | TCP | 74 41740 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1 |
| 52.212.223.6 | 192.168.2.4 | TCP | 74 443 → 41740 [SYN, ACK] Seq=0 Ack=1 Win=26847 |
| 192.168.2.4 | 52.212.223.6 | TCP | 66 41740 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 |
| 192.168.2.4 | 52.212.223.6 | OpenVPN | 110 MessageType: P_CONTROL_HARD_RESET_CLIENT_V2 |
| 52.212.223.6 | 192.168.2.4 | TCP | 66 443 → 41740 [ACK] Seq=1 Ack=45 Win=26880 Len= |
| 52.212.223.6 | 192.168.2.4 | OpenVPN | 122 MessageType: P_CONTROL_HARD_RESET_SERVER_V2 |
| 192.168.2.4 | 52.212.223.6 | TCP | 66 41740 → 443 [ACK] Seq=45 Ack=57 Win=29312 Len |
| 192.168.2.4 | 52.212.223.6 | OpenVPN | 118 MessageType: P_ACK_V1 |
| 52.212.223.6 | 192.168.2.4 | TCP | 66 443 → 41740 [ACK] Seq=57 Ack=97 Win=26880 Len |
| 192.168.2.4 | 52.212.223.6 | TLSv1.2 | 270 Client Hello |
| 52.212.223.6 | 192.168.2.4 | TCP | 66 443 → 41740 [ACK] Seq=57 Ack=301 Win=28032 Le |
| 52.212.223.6 | 192.168.2.4 | TLSv1.2 | 1236 Server Hello |
| 192.168.2.4 | 52.212.223.6 | OpenVPN | 118 MessageType: P_ACK_V1 |
| 52.212.223.6 | 192.168.2.4 | TLSv1.2 | 560 Ignored Unknown Record |
| 192.168.2.4 | 52.212.223.6 | TLSv1.2 | 1236 |
| 52.212.223.6 | 192.168.2.4 | OpenVPN | 118 MessageType: P_ACK_V1 |
| 192.168.2.4 | 52.212.223.6 | TLSv1.2 | 451 Ignored Unknown Record |

TLS-трафик OpenVPN

А вот как выглядит обычный HTTPS.



| | | | | |
|---|---|---|---|---|
| 192.168.2.4 | 216.58.206.110 | TCP | 74 | 58294 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 |
| 216.58.206.110 | 192.168.2.4 | TCP | 74 | 443 → 58294 [SYN, ACK] Seq=0 Ack=1 Win=42408 Len |
| 192.168.2.4 | 216.58.206.110 | TCP | 66 | 58294 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TS |
| 192.168.2.4 | 216.58.206.110 | TLSv1.2 | 253 | Client Hello |
| 216.58.206.110 | 192.168.2.4 | TCP | 66 | 443 → 58294 [ACK] Seq=1 Ack=188 Win=43520 Len=0 |
| 216.58.206.110 | 192.168.2.4 | TCP | 66 | [TCP Dup ACK 184#1] 443 → 58294 [ACK] Seq=1 Ack= |
| 216.58.206.110 | 192.168.2.4 | TLSv1.2 | 1406 | Server Hello |
| 192.168.2.4 | 216.58.206.110 | TCP | 66 | 58294 → 443 [ACK] Seq=188 Ack=1341 Win=32128 Len |
| 216.58.206.110 | 192.168.2.4 | TCP | 1406 | 443 → 58294 [ACK] Seq=1341 Ack=188 Win=43520 Len |
| 192.168.2.4 | 216.58.206.110 | TCP | 66 | 58294 → 443 [ACK] Seq=188 Ack=2681 Win=35072 Len |
| 216.58.206.110 | 192.168.2.4 | TLSv1.2 | 1755 | Certificate, Server Key Exchange, Server Hello D |
| 192.168.2.4 | 216.58.206.110 | TCP | 66 | 58294 → 443 [ACK] Seq=188 Ack=4370 Win=38400 Len |
| 192.168.2.4 | 216.58.206.110 | TLSv1.2 | 324 | Client Key Exchange, Change Cipher Spec, Encrypt |
| 192.168.2.4 | 216.58.206.110 | TLSv1.2 | 159 | Application Data |
| 192.168.2.4 | 216.58.206.110 | TLSv1.2 | 342 | Application Data |
| 216.58.206.110 | 192.168.2.4 | TLSv1.2 | 387 | New Session Ticket, Change Cipher Spec, Encrypte |
| 216.58.206.110 | 192.168.2.4 | TLSv1.2 | 135 | Application Data |
| 192.168.2.4 | 216.58.206.110 | TCP | 66 | 58294 → 443 [ACK] Seq=815 Ack=4760 Win=41088 Len |

Трафик HTTPS

Некоторые популярные платные VPN предоставляют средства обхода DPI, но чем больше популярность, тем больше шанс, что провайдер узнает о сервисе и сможет полностью заблокировать доступ к нему. От полной блокировки не защищен никто, но, когда используешь публичный сервис, шанс всегда выше.

## Пара слов об OpenVPN

OpenVPN использует два канала: канал управления (control channel) и канал данных (data channel). В первом случае задействуется TLS — с его помощью ведется аутентификация и обмен ключами для симметричного шифрования. Эти ключи используются в канале данных, где и происходит само шифрование трафика. Существуют скрипты, которые автоматизируют установку, и процесс занимает меньше времени. Но, во-первых, эти скрипты подходят только для конкретных дистрибутивов, а во-вторых, они не предоставляют выбора. Например, используют RSA и AES-CBC, когда есть поддержка ECDSA и AES-GCM. Таким образом, без знания и понимания того, как это работает, ты не сможешь подправить скрипт, чтобы он исполнялся на других системах или делал то, что ты хочешь.

## Что такое stunnel

Stunnel — это утилита для обеспечения защищенного соединения между клиентом и сервером посредством TLS для программ, которые сами шифровать трафик не умеют. Например, можно туннелировать трафик для

`netcat`

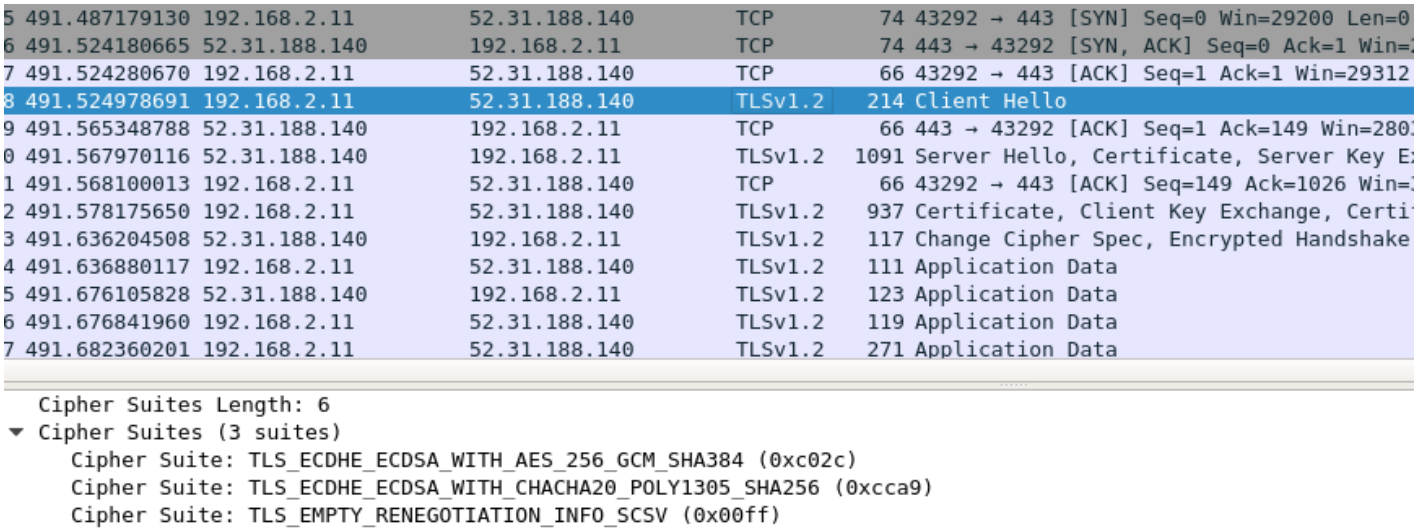
,

vnc

и даже

bash

. В нашем случае stunnel будет использоваться для маскировки трафика OpenVPN под «чистый» TLS, чтобы его было невозможно определить посредством DPI и, следовательно, заблокировать.




Трафик, туннелируемый через stunnel, ничем не отличается от обычного HTTPS
С учетом того что OpenVPN использует шифрование для своего канала данных, у нас есть два варианта настройки:

- использовать шифрование stunnel плюс шифрование канала данных OpenVPN;
- использовать шифрование stunnel, а шифрование канала данных OpenVPN отключить.

Таким образом, в первом варианте получается два слоя: один от stunnel, второй от OpenVPN. Этот вариант позволит использовать RSA вместе с ECDSA. Недостаток в том, что тратится больше ресурсов, и второй вариант позволит этого избежать. В любом случае настройка stunnel остается неизменной.

## Что нам понадобится

## Провайдер VPS

Первым делом нужно выбрать провайдера, который нам предоставит виртуальный выделенный сервер (VPS). Что выбирать — дело каждого и зависит от страны и от того, сколько ты готов платить. Главная рекомендация — выбирай страну, наиболее близкую по географическому расположению, это сведет задержку к минимуму. Но, конечно, живя в Китае, покупать сервис в Индии или Пакистане смысла мало.

### Выбор ОС

Я буду использовать RHEL 7.4. Для точного копирования команд из статьи годится и CentOS 7 (1708), так как это бесплатная и почти идентичная копия RHEL, основанная на его коде. Возможно, подойдут другие дистрибутивы, а также производные RHEL (Fedora), но пути конфигурационных файлов и версии программ могут отличаться.

## Подготовка и первичная настройка

После покупки сервера и установки системы нам нужно попасть на сервер. Я буду делать это с помощью SSH. Вся конфигурация будет проходить в два этапа: настройка на сервере (включает в себя первичную настройку) и настройка клиентов.
После покупки, скорее всего, тебе дадут доступ по SSH с логином

```
root
```

и паролем. Позже лучше создать обычного пользователя, а рутовые команды выполнять после

```
sudo -i
```

. Нужно это в том числе для защиты от брутфорса — пользователь

```
root
```

общеизвестный, и при попытках брута, вероятней всего, будет использоваться именно он.
Для начала нам понадобится подключить репозиторий EPEL — пакет

```
openvpn
```

лежит именно там.

```
$ yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
noarch.rpm $ yum update -y
```

На RHEL, CentOS, Fedora, OpenSUSE и, возможно, других установлен и включен по умолчанию SELinux. Проверить это можно командой

```
getenforce
```

или

```
sestatus
```

. Чтобы не нырять в дебри его настроек и спастись от возможной головной боли, мы переведем его в режим

```
permissive
```

. В этом режиме он будет оповещать нас о нарушениях политик безопасности, но предпринимать никаких действий не станет. Таким образом, у тебя всегда будет возможность его поизучать. Для этого нужно изменить следующую директиву в файле

```
/etc/selinux/config
```

:

```
SELINUX=permissive
```

Перезагружаемся и ставим необходимые пакеты:

```
$ yum install -y iptables-services openvpn unzip
```

- `iptables-services` — файлы `.service` для управления утилитой `iptables`;
- `openvpn` — сам сервер OpenVPN;
- зачем нужен `unzip`, попробуй догадаться сам.

## Базовая защита

Поскольку китайские боты и скрипт-киддиз не дремлют и, скорее всего, уже сейчас пробуют подобрать пароль к твоему серверу, перенесем

```
sshd
```

на другой порт и запретим логин от рута. Перед тем как это сделать, нужно убедиться, что в системе существует другой пользователь с доступом по SSH или добавить нового и установить для него пароль.

```
$ useradd -G wheel -m eakj $ passwd eakj
```

где

```
eakj
```

— имя пользователя. В идеале нужно использовать ключи SSH, но в этом случае обойдемся паролем. Не забудь проверить, раскомментирована ли строчка

```
%wheel ALL=(ALL) ALL
```

в файле

```
/etc/sudoers
```

. Теперь изменим следующие директивы в файле

```
/etc/ssh/sshd_config
```

:

```
Port 12222 PermitRootLogin no PasswordAuthentication yes
```

Перечитаем конфиги (

```
systemctl reload sshd
```

), убедимся, что sshd поднялся без проблем (

```
systemctl status sshd
```

), и попробуем открыть дополнительную сессию SSH, не закрывая текущей.

```
[root@ip-172-31-26-46 ~]# systemctl reload sshd
[root@ip-172-31-26-46 ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2017-12-04 21:17:03 UTC; 6min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 1000 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)
 Main PID: 903 (sshd)
   CGroup: /system.slice/sshd.service
           └─903 /usr/sbin/sshd -D
```

Статус sshd

# Работа на сервере

## Easy-rsa

Утилита easy-rsa была создана, чтобы облегчить процесс создания Certificate Authority (CA) и управления ими, а также серверными и клиентскими сертификатами. В идеале для CA нужно выделить специальную машину, но для экономии времени можно использовать все ту же. Поддержку ECDSA добавили в версии 3.0, а в репозиториях у нас 2.2.2, поэтому скачаем последнюю версию с GitHub. Это бинарник, поэтому ничего компилировать уже не придется.

```
$ cd /opt/ && curl -O -L https://github.com/OpenVPN/easy-rsa/archive/master.z
ip $ unzip master.zip && rm -f master.zip $ cd easy-rsa-master/easyrsa3/ && c
p vars.example vars
```

Далее в файле

```
vars
```

нужно раскомментировать и настроить некоторые параметры.

**/opt/easy-rsa-master/easyrsa3/vars**

```
set_var EASYRSA_DN "cn_only" set_var EASYRSA_ALGO ec set_var EASYRSA_CURVE se
cp521r1 set_var EASYRSA_CA_EXPIRE 3650 set_var EASYRSA_CERT_EXPIRE 3650 set_v
ar EASYRSA_CRL_DAYS 3650
```

Здесь указано, что использовать нужно только Common Name (CN) для Distinguished Name (DN) и криптографию на эллиптических кривых (ec). Также задано название кривой (secp521r1) и время действия сертификатов.

# INFO

Некоторые версии OpenSSL отличаются нестабильной работой, так что рекомендую выбирать проверенные эллиптические кривые:

`prime256v1`

,

`secp384r1`

,

`secp521r1`

. В RHEL и родственных дистрибутивах доступны только они. Список можно посмотреть при помощи

```
openssl ecparam -list_curves
```

.
По умолчанию,

`easyrsa`

будет искать

`vars`

в той же директории, где и сам исполняемый файл. Но для надежности мы объявим переменную окружения:

```
$ export EASYRSA_VARS_FILE=/opt/easy-rsa-master/easyrsa3/vars
```

Создаем свой CA, а также генерируем сертификаты и ключи сервера и клиентов:

```
$ ./easyrsa init-pki $ ./easyrsa --batch build-ca nopass $ ./easyrsa build-se
rver-full openvpn-server nopass $ ./easyrsa build-client-full openvpn-client
nopass
```

где

`openvpn-server`

и

`openvpn-client`

— это наш CN для сервера и клиента.

```
[root@ip-172-31-26-46 easyrsa3]# ./easyrsa init-pki

Note: using Easy-RSA configuration from: /opt/easy-rsa-master/easyrsa3/vars

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /opt/easy-rsa-master/easyrsa3/pki

[root@ip-172-31-26-46 easyrsa3]# ./easyrsa --batch build-ca nopass
Generating a 521 bit EC private key
writing new private key to '/opt/easy-rsa-master/easyrsa3/pki/private/ca.key.PM8xTr1SPE'
-----
[root@ip-172-31-26-46 easyrsa3]# ./easyrsa build-server-full openvpn-server nopass

Note: using Easy-RSA configuration from: /opt/easy-rsa-master/easyrsa3/vars
Generating a 521 bit EC private key
writing new private key to '/opt/easy-rsa-master/easyrsa3/pki/private/openvpn-server.key.r1HO9uRzjM'
-----
Using configuration from ./openssl-easyrsa.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'openvpn-server'
Certificate is to be certified until Dec  2 22:02:28 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[root@ip-172-31-26-46 easyrsa3]# ./easyrsa build-client-full openvpn-client nopass

Note: using Easy-RSA configuration from: /opt/easy-rsa-master/easyrsa3/vars
Generating a 521 bit EC private key
writing new private key to '/opt/easy-rsa-master/easyrsa3/pki/private/openvpn-client.key.0jWHFViAth'
-----
Using configuration from ./openssl-easyrsa.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'openvpn-client'
Certificate is to be certified until Dec  2 22:02:32 2027 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```

easy-rsa

Скопируем сертификат и ключ сервера в

`/etc/openvpn/server`

, а сертификат и ключ клиента — в

`/tmp`

.

```
$ cp -p pki/ca.crt pki/private/openvpn-server.key pki/issued/openvpn-server.c
rt /etc/openvpn/server/ $ cp -p pki/ca.crt pki/private/openvpn-client.key pk
i/issued/openvpn-client.crt /tmp/
```

## OpenVPN-сервер

Приступим к созданию конфигов и настройки сервера OpenVPN.

```
$ cd /etc/openvpn/server/ $ openvpn --genkey --secret ta.key
```

Файл

```
ta.key
```

нужен для директивы

```
tls-auth
```

, которая предоставляет дополнительный уровень защиты для нашего OpenVPN. Этот ключ должен быть и у клиента, поэтому скопируем его в

```
/tmp
```

:

```
$ cp -p ta.key /tmp/
```

Пример конфига:

**/etc/openvpn/server/openvpn-server.conf**

```
### Bind на loopback-адрес и стандартный порт, ### так как коннект из интерне
та все равно получает stunnel local 127.0.0.1 port 1194 ### Stunnel туннелиру
ет только TCP proto tcp ### Режим туннеля dev tun ### Файлы сертификатов и кл
ючей ca ca.crt cert openvpn-server.crt key openvpn-server.key ### Включаем ис
пользование Elliptic Curve Diffie — Hellman (ECDH) dh none ### На сервере ста
вим 0 tls-auth ta.key 0 ### Явно указываем, что можно использовать для канала
управления (control channel) tls-cipher TLS-ECDHE-ECDSA-WITH-AES-256-GCM-SHA3
84:TLS-ECDHE-ECDSA-WITH-CHACHA20-POLY1305-SHA256 ### Шифр для канала данных
(data channel) cipher AES-256-GCM server 10.8.8.0 255.255.255.0 ### Указываем
клиентам перенаправлять весь трафик в туннель, ### где 52.214.41.150 — IP сер
вера push "redirect-gateway def1" push "route 52.214.41.150 255.255.255.255 n
et_gateway" ### Указываем использовать эти DNS push "dhcp-option DNS 208.67.2
22.222" push "dhcp-option DNS 208.67.220.220" ### Включаем возможность исполь
зования одного клиентского ### сертификата на многих устройствах одновременно
### для большего контроля — можно выключить (закомментировать) duplicate-cn k
eepalive 10 120 user nobody group nobody persist-key persist-tun ### Никаких
логов. Есть смысл включить для отладки ### в случае сбоя status /dev/null log
/dev/null verb 0
```

Также этот конфиг не позволяет клиентам общаться между собой в сети. Если хочется больше контроля, то можно запретить использование одного сертификата на многих устройствах одновременно. Тогда придется генерировать клиентский сертификат для каждого устройства отдельно. Для этого нужно закомментировать

```
duplicate-cn
```

.

Чтобы отключить шифрование, устанавливаем

```
cipher none
```

, остальное — без изменений. В этом режиме все еще будет проходить аутентификация, но канал данных шифроваться не будет.

Пробуем стартовать:

```
$ systemctl start openvpn-server@openvpn-server
```

Смотрим статус:

```
$ systemctl status openvpn-server@openvpn-server
```

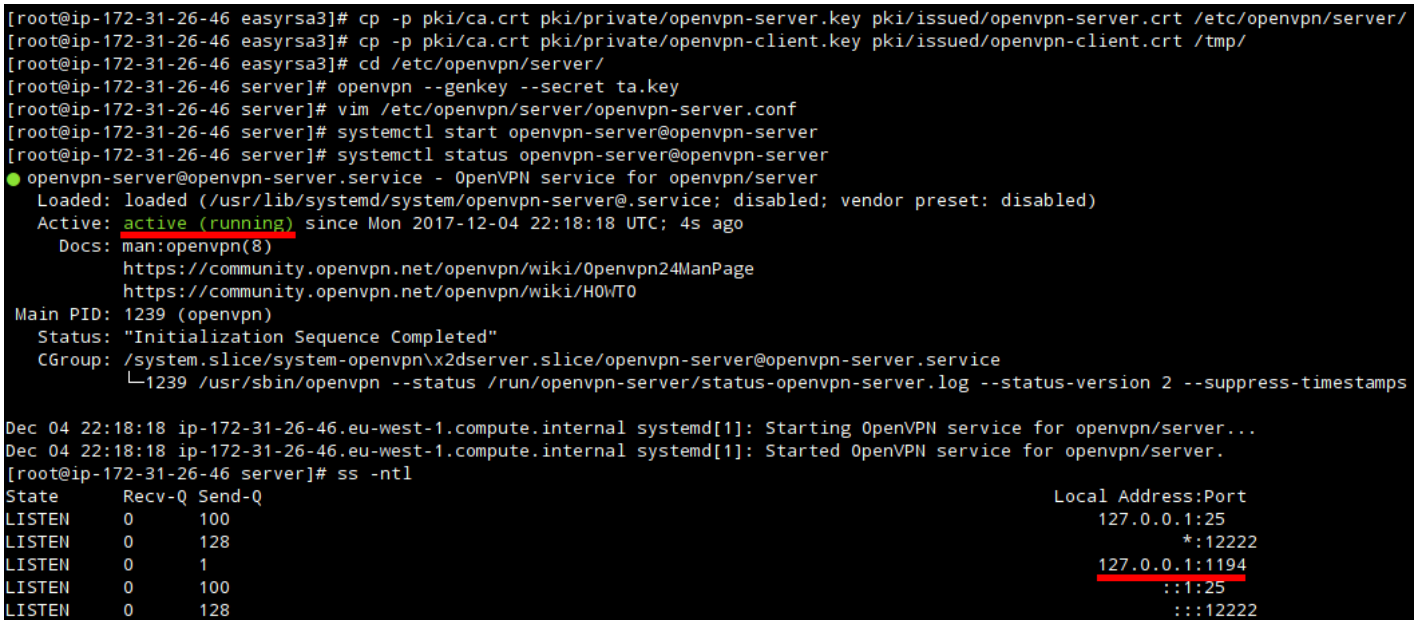
То, что идет после **@**, — это название файла с конфигом. Например, если он у тебя называется просто

```
server.conf
```

, тогда будет

```
systemctl start openvpn-server@server
```

.




Статус OpenVPN

Если все хорошо, добавляем автоматическую загрузку:

```
$ systemctl enable openvpn-server@openvpn-server
```

## Сервер stunnel

В репозиториях у нас версия stunnel 4.56, которая не поддерживает верификацию клиентов со стороны сервера, поэтому установим более свежую:

```
$ cd /opt && curl -O -L https://rpmfind.net/linux/fedora/linux/updates/25/x86_64/Packages/s/stunnel-5.41-1.fc25.x86_64.rpm $ rpm -ivh stunnel-5.41-1.fc25.x86_64.rpm ### Проверим $ rpm -qi stunnel
```

Теперь добавим нового пользователя

```
stunnel
```

и создадим ему домашнюю директорию в

```
/var/stunnel
```

:

```
$ useradd -d /var/stunnel -m -s /bin/false stunnel
```

Проверим, что все успешно:

```
$ ls -ld /var/stunnel
```

Это делается для того, чтобы не запускать

```
stunnel
```

от

```
root
```

, и дает возможность использовать chroot.

```
[root@ip-172-31-26-46 server]# cd /opt && curl -O -L https://rpmfind.net/linux/fedora/linux/updates/25/x86_64/Packages/s/stunnel-5.41-1.fc25.x86_64.rpm
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  150k  100  150k    0     0   387k      0 --:--:-- --:--:-- --:--:--  388k
[root@ip-172-31-26-46 opt]# rpm -ivh stunnel-5.41-1.fc25.x86_64.rpm
warning: stunnel-5.41-1.fc25.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fdb19c98: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:stunnel-5.41-1.fc25              ################################# [100%]
[root@ip-172-31-26-46 opt]# rpm -qi stunnel
Name        : stunnel
Version     : 5.41
Release     : 1.fc25
Architecture: x86_64
Install Date: Mon 04 Dec 2017 10:26:11 PM UTC
Group       : Applications/Internet
Size        : 294922
License     : GPLv2
Signature   : RSA/SHA256, Mon 17 Apr 2017 06:33:03 PM UTC, Key ID 4089d8f2fdb19c98
Source RPM  : stunnel-5.41-1.fc25.src.rpm
Build Date  : Mon 17 Apr 2017 06:20:40 PM UTC
Build Host  : buildvm-23.phx2.fedoraproject.org
Relocations : (not relocatable)
Packager    : Fedora Project
Vendor      : Fedora Project
URL         : http://www.stunnel.org/
Summary     : A TLS-encrypting socket wrapper
Description :
Stunnel is a socket wrapper which can provide TLS/SSL
(Transport Layer Security/Secure Sockets Layer) support
to ordinary applications. For example, it can be used in
conjunction with imapd to create a TLS secure IMAP server.
[root@ip-172-31-26-46 opt]# useradd -d /var/stunnel -m -s /bin/false stunnel
[root@ip-172-31-26-46 opt]# ls -ld /var/stunnel
drwx------. 2 stunnel stunnel 62 Dec  4 22:26 /var/stunnel
```

Статус stunnel

Примерно так должен выглядеть

```
/etc/stunnel/stunnel.conf
```

:

```
### Помни: exec исполняется из каталога chroot! chroot = /var/stunnel setuid
= stunnel setgid = stunnel pid = /stunnel.pid debug = 0 ## performance tunnin
g socket = l:TCP_NODELAY=1 socket = r:TCP_NODELAY=1 ### curve used for ECDHE
curve = secp521r1 sslVersion = all options = NO_SSLv2 options = NO_SSLv3 [ope
nvpn] accept = 443 connect = 127.0.0.1:1194 renegotiation = no ### RSA cipher
s = ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-RSA-AES256-
SHA cert = /etc/stunnel/stunnel-server.crt key = /etc/stunnel/stunnel-server.
key CAfile = /etc/stunnel/clients.crt verifyPeer = yes
```

Основные директивы в этом файле:

- `accept = [address:]<port>`

  — указывает, на каком адресе и порте будет слушать наш

  `stunnel`

  ;
- `connect = [address:]<port>`

  — указывает, на какой адрес и порт будет перенаправляться трафик;
- `cert = <full path to certificate>`

  — абсолютный путь к сертификату;
- `key = <full path to key>`

  — абсолютный путь к ключу от этого сертификата. Эту директиву можно опустить, если ключ встроен в файл с сертификатом.

Имя сервиса

```
[openvpn]
```

заключается в квадратные скобки и может быть произвольным.

Создадим ключи и сертификаты:

```
$ cd /etc/stunnel $ openssl req -newkey rsa:2048 -nodes -keyout stunnel-serve
r.key \ -x509 -days 3650 -subj "/CN=stunnel-server" \ -out stunnel-server.crt
```

```
[root@ip-172-31-26-46 stunnel]# openssl req -newkey rsa:2048 -nodes -keyout stunnel-server.key \
> -x509 -days 3650 -subj "/CN=stunnel-server" \
> -out stunnel-server.crt
Generating a 2048 bit RSA private key
.....................................................+++
.............................................+++
writing new private key to 'stunnel-server.key'
-----
```

Так как все сертификаты клиентов должны быть записаны в

`clients.crt`

на сервере, сгенерируем их:

```
$ openssl req -newkey rsa:2048 -nodes -keyout eakj-desktop.key \ -x509 -days 3650 -subj "/CN=eakj-desktop" \ -out eakj-desktop.crt $ openssl req -newkey rsa:2048 -nodes -keyout eakj-mobile.key \ -x509 -days 3650 -subj "/CN=eakj-mobile" \ -out eakj-mobile.crt $ openssl pkcs12 -export -in eakj-mobile.crt \ -inkey eakj-mobile.key -out eakj-mobile.p12
```

При генерации

`eakj-mobile.p12`

нужно будет ввести пароль, не забудь его.

```
[root@ip-172-31-26-46 stunnel]# openssl req -newkey rsa:2048 -nodes -keyout eakj-desktop.key \
> -x509 -days 3650 -subj "/CN=eakj-desktop" \
> -out eakj-desktop.crt
Generating a 2048 bit RSA private key
.............+++
.....................................................................................................+++
writing new private key to 'eakj-desktop.key'
-----
[root@ip-172-31-26-46 stunnel]# openssl req -newkey rsa:2048 -nodes -keyout eakj-mobile.key \
> -x509 -days 3650 -subj "/CN=eakj-mobile" \
> -out eakj-mobile.crt
Generating a 2048 bit RSA private key
.....................................................................+++
.....................+++
writing new private key to 'eakj-mobile.key'
-----
[root@ip-172-31-26-46 stunnel]# openssl pkcs12 -export -in eakj-mobile.crt \
> -inkey eakj-mobile.key -out eakj-mobile.p12
Enter Export Password:
Verifying - Enter Export Password:
[root@ip-172-31-26-46 stunnel]# ls -l
total 32
-rw-r--r--. 1 root root 1103 Dec  4 22:52 eakj-desktop.crt
-rw-r--r--. 1 root root 1704 Dec  4 22:52 eakj-desktop.key
-rw-r--r--. 1 root root 1099 Dec  4 22:52 eakj-mobile.crt
-rw-r--r--. 1 root root 1708 Dec  4 22:52 eakj-mobile.key
-rw-r--r--. 1 root root 2373 Dec  4 22:52 eakj-mobile.p12
-rw-r--r--. 1 root root  547 Dec  4 22:29 stunnel.conf
-rw-r--r--. 1 root root 1107 Dec  4 22:51 stunnel-server.crt
-rw-r--r--. 1 root root 1708 Dec  4 22:51 stunnel-server.key
```

Запишем все клиентские сертификаты в

`clients.crt`

. Вот что должно примерно получиться:

**/etc/stunnel/clients.crt**

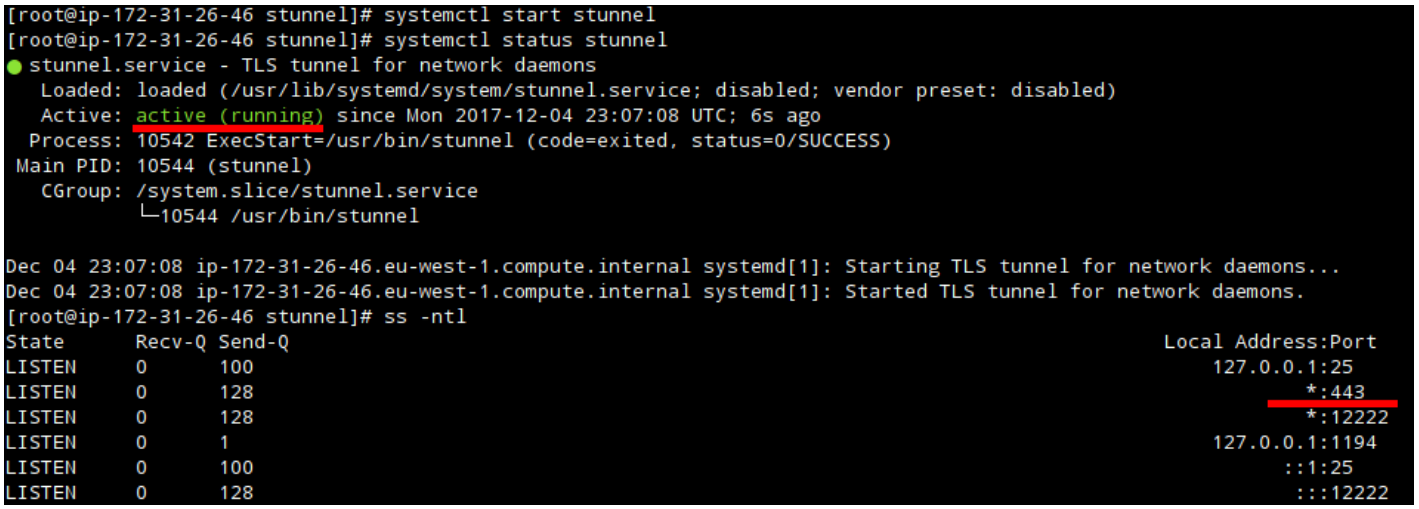
```
### eakj-desktop -----BEGIN CERTIFICATE----- ... -----END CERTIFICATE----- ##
# eakj-mobile -----BEGIN CERTIFICATE----- ... -----END CERTIFICATE-----
```

Стартуем:

```
$ systemctl start stunnel
```

И проверяем:

```
$ systemctl status stunnel`
```




Статус stunnel

Если все хорошо, то добавляем на автостарт:

```
$ systemctl enable stunnel
```

Как и в случае с OpenVPN, скопируем клиентские файлы, а также серверный сертификат в

```
/tmp
```

:

```
$ cp -p eakj-* stunnel-server.crt /tmp/
```

## Настройка файрвола и маршрутизации

## WARNING

Утилита

```
iptables
```

— это мощный инструмент, но он не терпит ошибок в конфигурации. Стоит быть предельно аккуратным и перепроверять правила, чтобы случайно не закрыть себе доступ. Открыть его после этого бывает трудно, и все зависит от провайдера VPS. Чтобы снизить риск перекрытия доступа самому себе, рекомендую не прописывать iptables в автозагрузку и для начала хорошенько проверить содержимое файла

```
/etc/sysconfig/iptables
```

. Если что-то пойдет не так, то ты сможешь перезагрузить машину и продолжить эксперименты.
Ну а если уверен, что все настроил правильно, то пиши:

```
$ systemctl enable iptables
```

В некоторых случаях

```
firewalld
```

может быть установлен и включен по умолчанию, например на минимальной установке CentOS 7.4. Так что для начала лучше проверь:

```
$ systemctl status firewalld
```

```
[root@localhost ~]# cat /etc/centos-release
CentOS Linux release 7.4.1708 (Core)
[root@localhost ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2017-12-05 15:00:19 GMT; 55s ago
     Docs: man:firewalld(1)
 Main PID: 1344 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─1344 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
```

Firewalld в CentOS 7.4.1708
Если он включен, то нужно его остановить и убрать из автозагрузки.

```
$ systemctl stop firewalld $ systemctl disable firewalld $ systemctl status f
irewalld
```

```
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@localhost ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

Firewalld отключен

Убедимся, что список правил у нас пуст (

`iptables -nvL`

), и приступим.

```
[root@ip-172-31-26-46 ~]# iptables -nvL
Chain INPUT (policy ACCEPT 2686 packets, 1037K bytes)
 pkts bytes target     prot opt in     out     source               destination

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

Chain OUTPUT (policy ACCEPT 1933 packets, 320K bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Список правил iptables

Нам нужно создать набор базовых правил для файрвола, который говорит:

- позволять пинг;
- принимать любой трафик на интерфейс

  `lo`

  ;
- пропускать пакеты на порты 443 и 12222 (еще не забыл, что мы перенесли наш SSH?);
- а также принимать **ответы** на наши исходящие **запросы**. Больше информации о

  `conntrack`

  можно найти в интернете.

```
$ iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT $ ip
tables -A INPUT -i lo -j ACCEPT $ iptables -A INPUT -p icmp --icmp-type 8 -j
ACCEPT $ iptables -A INPUT -p tcp --dport 443 -j ACCEPT $ iptables -A INPUT -
p tcp --dport 12222 -j ACCEPT
```

Пока эти правила ничем не опасны, так как у них стоит действие

`ACCEPT`

, то есть принимать и пропускать. Смотрим, все ли добавилось (снова

```
iptables -nvL
```

).

```
[root@ip-172-31-26-46 stunnel]# iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
[root@ip-172-31-26-46 stunnel]# iptables -A INPUT -i lo -j ACCEPT
[root@ip-172-31-26-46 stunnel]# iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
[root@ip-172-31-26-46 stunnel]# iptables -A INPUT -p tcp --dport 443 -j ACCEPT
[root@ip-172-31-26-46 stunnel]# iptables -A INPUT -p tcp --dport 12222 -j ACCEPT
[root@ip-172-31-26-46 stunnel]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   19  1432 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmptype 8
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:12222

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

Chain OUTPUT (policy ACCEPT 3 packets, 352 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Основной набор правил iptables

Далее нам понадобится несколько правил переадресации.

```
$ iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT $
iptables -A FORWARD -i tun+ -s 10.8.8.0/24 -j ACCEPT
```

Здесь

```
10.8.8.0/24
```

— адрес и маска подсети, которую мы указывали в

```
openvpn-server.conf
```

.

Добавим правила, которые позволят клиентам отсылать и принимать пакеты из интернета. Тут возможны два варианта: вариант, когда у тебя есть статический IP, и вариант, когда адрес динамический. Первый лучше тем, что не тратятся ресурсы на определение IP-адреса. Преимущество второго варианта — в том, что не надо иметь статический IP. Если ты не уверен или не знаешь, какой у тебя, используй второй вариант. Для начала узнаем свой адрес командой

```
ip a
```

. У меня сервер за NAT, поэтому и IP тут локальной сети.

```
[root@ip-172-31-26-46 stunnel]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP qlen 1000
    link/ether 06:9b:08:6c:1a:d6 brd ff:ff:ff:ff:ff:ff
    inet 172.31.26.46/20 brd 172.31.31.255 scope global dynamic eth0
       valid_lft 2914sec preferred_lft 2914sec
    inet6 fe80::49b:8ff:fe6c:1ad6/64 scope link
       valid_lft forever preferred_lft forever
```

Вот варианты правил. Помни, что тебе нужно **либо 1, либо 2**, вместе их не добавляй.

1. `iptables -t nat -A POSTROUTING -s 10.8.8.0/24 -o eth0 -j SNAT --to-source 172.31.26.46`

   , где

   `172.31.26.46`

   — это IP, присвоенный интерфейсу, который смотрит в инет, а

   `eth0`

   — это сам интерфейс.

2. `iptables -t nat -A POSTROUTING -s 10.8.8.0/24 -o eth0 -j MASQUERADE`

   .

```
[root@ip-172-31-26-46 ~]# iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 57 packets, 3504 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 6 packets, 360 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 53 packets, 4776 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 53 packets, 4776 bytes)
 pkts bytes target     prot opt in     out     source               destination
   51  3144 SNAT       all  --  *      eth0    10.8.8.0/24          0.0.0.0/0           to:172.31.26.46
```

Настройки форвардинга

Осталось указать политику

`DROP`

по умолчанию для цепочек

`INPUT`

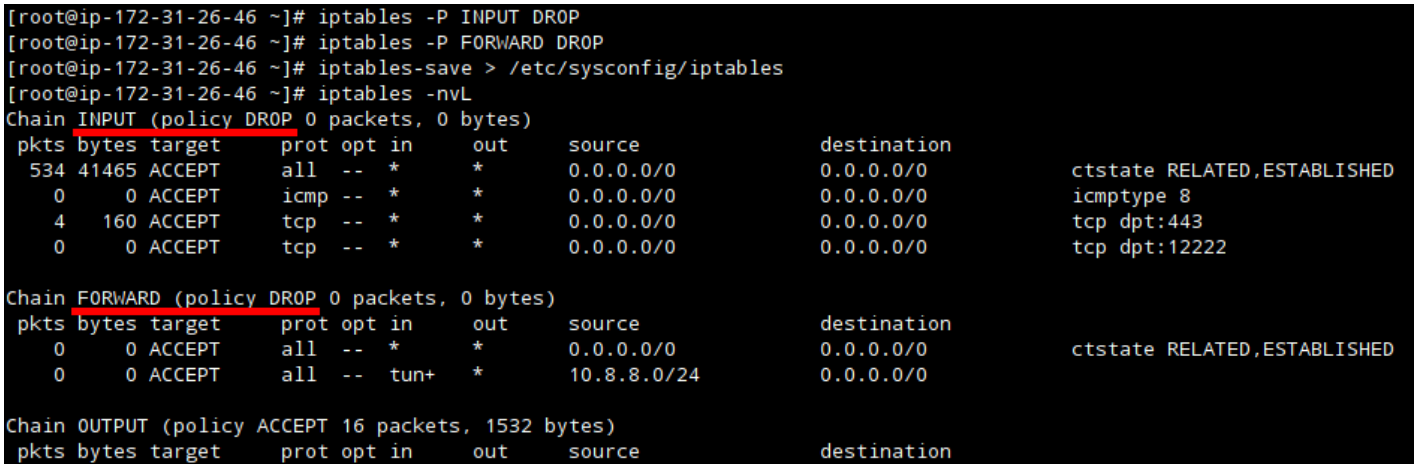
и

```
FORWARD
```

, после чего сохраним наши правила.

```
$ iptables -P INPUT DROP $ iptables -P FORWARD DROP $ iptables-save > /etc/sy
sconfig/iptables
```

Политика по умолчанию срабатывает после того, как отработали все правила в цепочке. Поэтому перед тем как ее применить, стоит удостовериться, что все нужные порты открыты.




Финальные настройки

Если на этом этапе ты еще не потерял доступ к своему серверу, тогда добавим правила в автозагрузку и продолжим.

```
$ systemctl enable iptables
```

Далее нам нужно включить форвардинг пакетов, так как по умолчанию он выключен. Проверить это можно командой

```
$ sysctl net.ipv4.ip_forward
```

Если вывод команды показывает

```
net.ipv4.ip_forward = 1
```

, то ничего делать не нужно, форвардинг уже включен. Если же

```
net.ipv4.ip_forward = 0
```

, то в файле

```
/etc/sysctl.conf
```

нужно изменить уже существующую или добавить новую строчку

```
net.ipv4.ip_forward = 1
```

. Это позволит нашим изменениям переживать перезагрузку. Далее выполним команду

```
sysctl -p
```

, чтобы изменения применились немедленно.

## Настройка клиентов

Настройка сервера закончена. Сейчас у нас в

```
/tmp
```

должны быть все необходимые файлы для клиентов.

```
[root@ip-172-31-26-46 ~]# ls -l /tmp/
total 40
-rw-------. 1 root root  806 Dec  4 22:02 ca.crt
-rw-r--r--. 1 root root 1103 Dec  4 22:52 eakj-desktop.crt
-rw-r--r--. 1 root root 1704 Dec  4 22:52 eakj-desktop.key
-rw-r--r--. 1 root root 1099 Dec  4 22:52 eakj-mobile.crt
-rw-r--r--. 1 root root 1708 Dec  4 22:52 eakj-mobile.key
-rw-r--r--. 1 root root 2373 Dec  4 22:52 eakj-mobile.p12
-rw-------. 1 root root 3113 Dec  4 22:02 openvpn-client.crt
-rw-------. 1 root root  384 Dec  4 22:02 openvpn-client.key
-rw-r--r--. 1 root root 1107 Dec  4 22:51 stunnel-server.crt
drwx------. 3 root root   17 Dec  4 21:17 systemd-private-fcee1a6762
drwx------. 3 root root   17 Dec  4 22:18 systemd-private-fcee1a6762
drwx------. 3 root root   17 Dec  4 23:07 systemd-private-fcee1a6762
-rw-------. 1 root root  636 Dec  4 22:13 ta.key
```

Содержимое папки /tmp

Как видно, некоторые файлы доступны для чтения только руту, что не позволит их скачать при помощи

```
scp
```

(ведь логин от рута у нас запрещен). Поэтому присвоим им другого владельца командой

```
$ chown eakj: /tmp/{ta.key,ca.crt,openvpn-client.crt,openvpn-client.key}
```

где eakj — это имя пользователя, которого мы создали в начале для доступа по SSH. Не забудь удалить эти файлы из

```
/tmp
```

на сервере. После того как настроишь все свои клиенты, они там только для удобства скачивания.

Для экономии времени я возьму один и тот же сертификат и ключ для подключения к OpenVPN в Linux, Windows и Android. Но на Android будет другой сертификат и ключ для подключения к stunnel, так как там придется использовать формат PKCS#12.

## Клиент stunnel

### Linux

С правами разобрались, перейдем к настройке клиента. Нужно скачать и установить

`stunnel`

, обычно он есть в репозиториях и с установкой нет проблем. Также исходники можно найти на официальном сайте.
В Fedora надо набрать

```
dnf install -y stunnel
```

, в Arch Linux:

```
pacman -S stunnel
```

, в Ubuntu:

```
apt install stunnel4
```

.

# INFO

В Ubuntu 16.04.3 LTS пакет

`stunnel4`

имеет версию 5.30, которая не поддерживает верификацию (verifyPeer), поэтому придется или найти свежий пакет, или закомментировать

`verifyPeer`

и

`CAfile`

. Также нужно изменить

`ENABLED=0`

на

`ENABLED=1`

в файле

`/etc/default/stunnel4`

. Возможны и другие мелкие отличия, для их обнаружения понадобится включить логирование.
Теперь скачаем клиентские файлы командой

`scp`

. Обрати внимание, что для записи в

`/etc/stunnel`

нужны права рута, поэтому и

`scp`

нужно запустить от суперпользователя.

```
$ scp -P 12222 eakj@52.214.41.150:"/tmp/{eakj-*,stunnel-server.crt}" /etc/stu
nnel/
```

Теперь нужно создать клиентский

`/etc/stunnel/stunnel.conf`

. Вот пример.

```
[openvpn] client = yes accept = 127.0.0.1:1194 connect = 52.214.41.150:443 ##
# Проверить сервер verifyPeer = yes ### Для этого нужен его сертификат CAfile
= /etc/stunnel/stunnel-server.crt ### Сертификат и ключ нужен для проверки ##
# клиента (verifyPeer) на сервере cert = /etc/stunnel/eakj-desktop.crt key =
/etc/stunnel/eakj-desktop.key
```

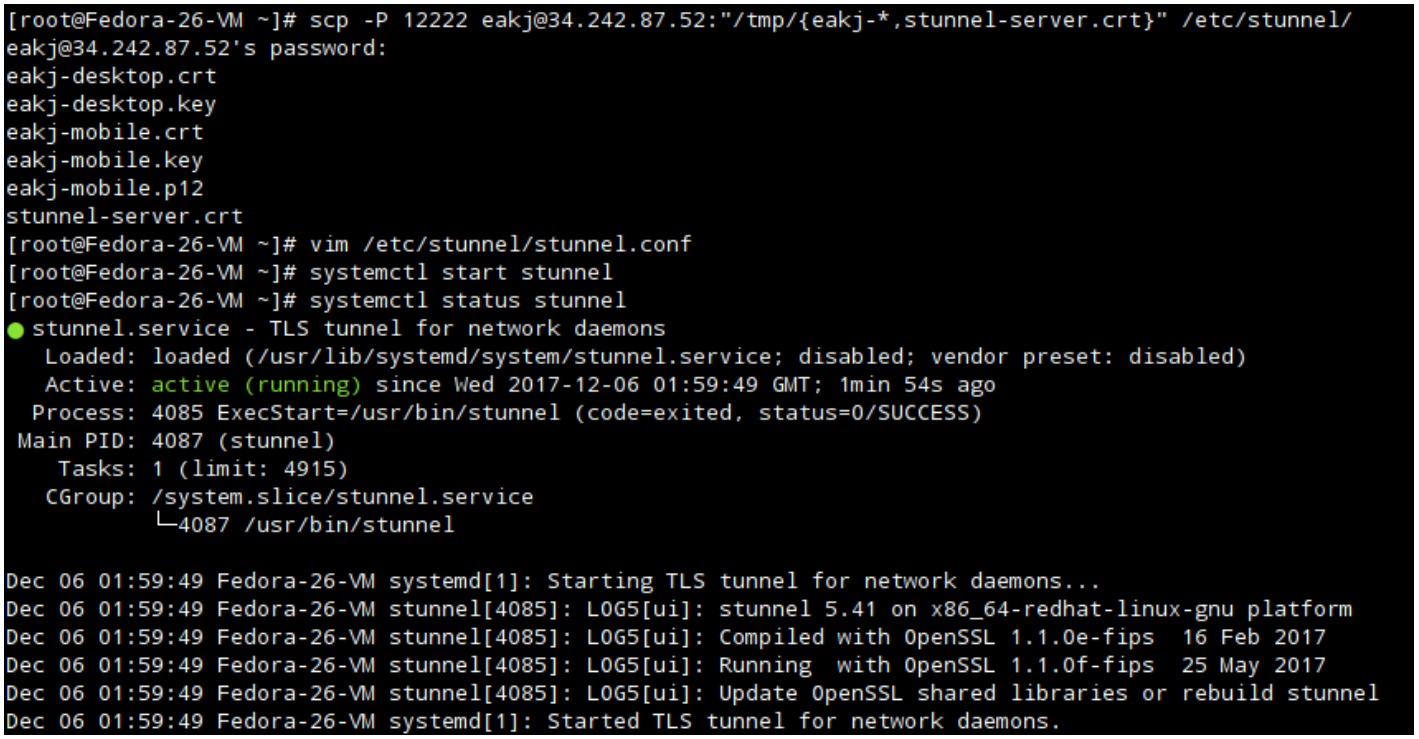
Запускаем (

`systemctl start stunnel`

) и проверяем (

`systemctl status stunnel`

).



Клиент stunnel в Fedora

**Windows**

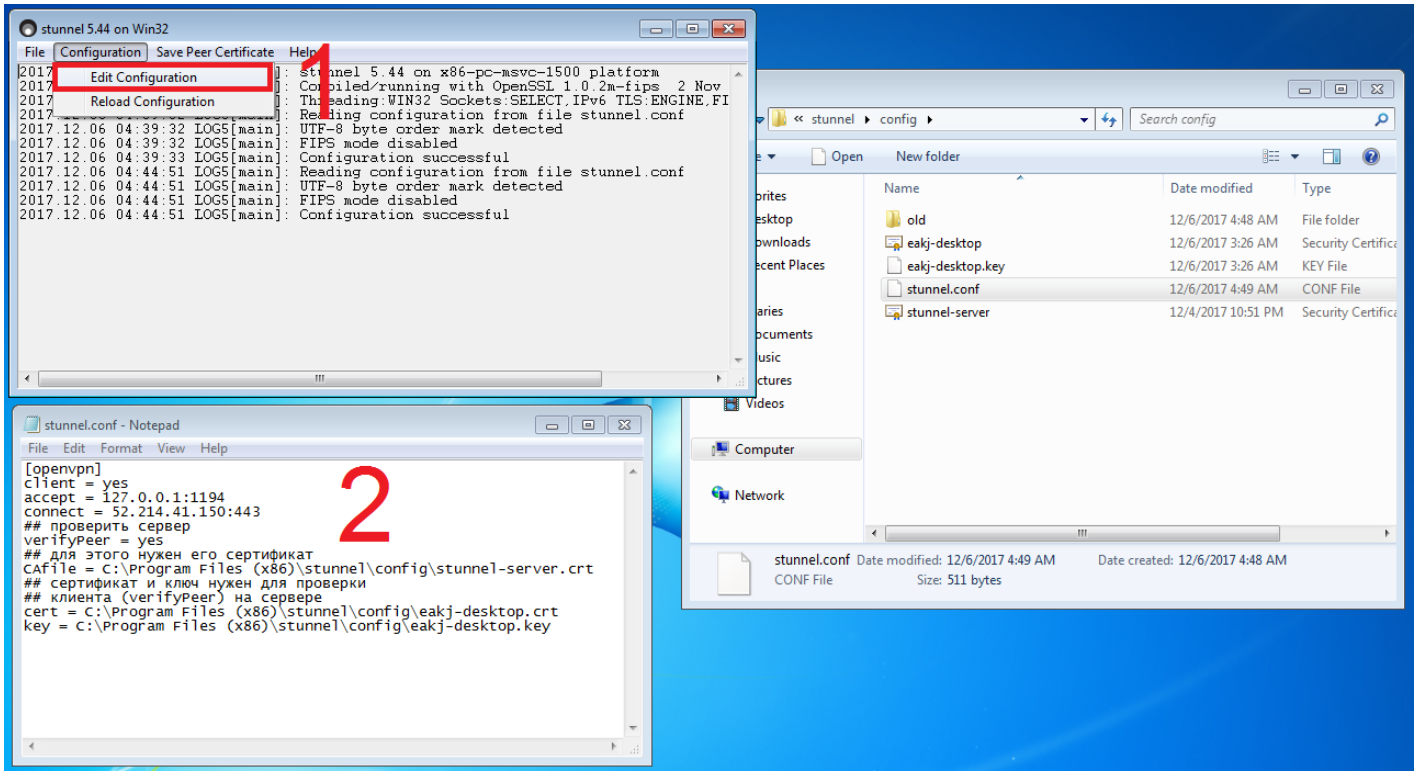
Для начала нужно скачать и установить

```
stunnel
```

. Найти его можно на официальном сайте.
Затем нужно скачать клиентские сертификаты и ключи с нашего сервера и поместить их в

```
C:\Program Files (x86)\stunnel\config
```

, как показано на скрине. Я делал это при помощи WinSCP, ты можешь использовать любой удобный тебе клиент SSH. После установки и запуска увидишь главное окно программы.




Настройки stunnel в Windows

Нужно отредактировать стандартный конфиг, а также переместить или удалить сгенерированные при установке ключи и сертификаты. Я переместил их в папку

```
old
```

. Для Windows конфиг почти идентичен линуксовому, только с другими путями.

```
[openvpn] client = yes accept = 127.0.0.1:1194 connect = 52.214.41.150:443 ## # Проверить сервер verifyPeer = yes ### Для этого нужен его сертификат CAfile = C:\Program Files (x86)\stunnel\config\stunnel-server.crt ### Сертификат и ключ нужны для проверки ### клиента (verifyPeer) на сервере cert = C:\Program Files (x86)\stunnel\config\eakj-desktop.crt key = C:\Program Files (x86)\stunnel\config\eakj-desktop.key
```

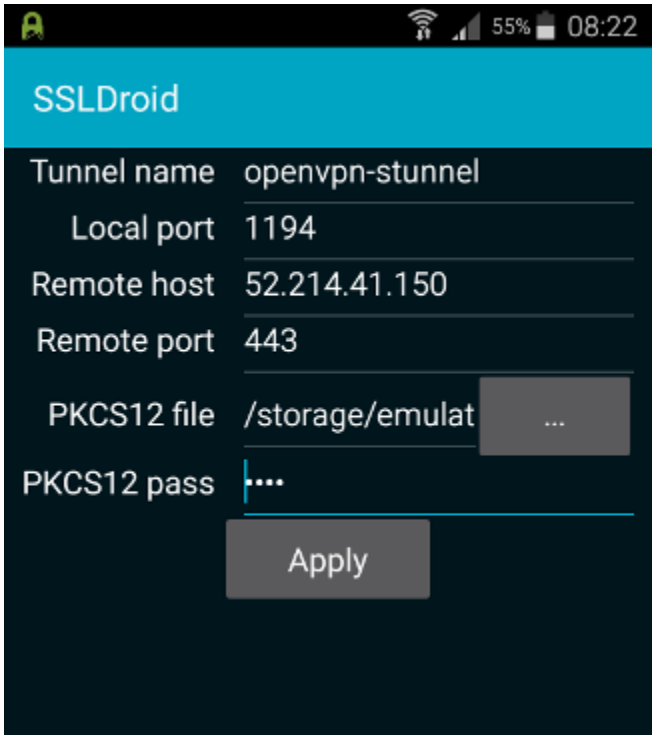
После того как изменил и сохранил конфиг, в окне stunnel жмешь Configuration → Reload Configuration.

**Android**

В качестве клиента stunnel на мобильных устройствах будет использоваться приложение SSLDroid. Перенесем на телефон файл

```
eakj-mobile.p12
```

и настроим приложение.

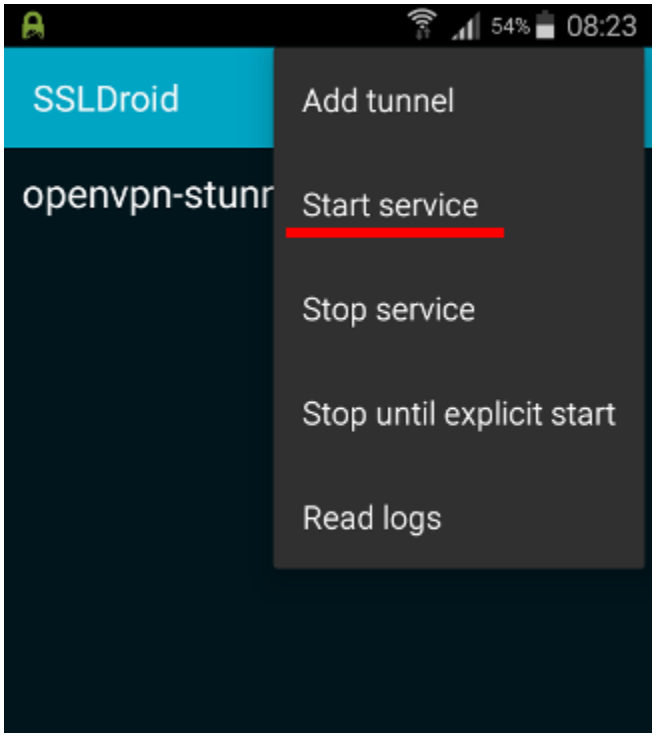


SSLDroid config

Укажем путь к файлу, введем пароль, который запомнили при генерации

```
eakj-mobile.p12
```

, сохраним настройки и запустим сервис.



SSLDroid service start

## Клиент OpenVPN

**Linux**

Скачаем и установим

`openvpn`

из репозитория. Клиент и сервер идут вместе. В Fedora пили:

```
dnf install -y openvpn
```

, в Arch Linux:

```
pacman -S openvpn
```

, в Ubuntu:

```
apt install openvpn
```

.

Скачаем файлы клиента с сервера при помощи

`scp`

. Для записи в

`/etc/openvpn/client/`

также нужны права рута.

## Ubuntu

В Ubuntu клиентские файлы хранятся в

`/etc/openvpn/`

и сервисы называются

`openvpn@<имя конфиг файла>`

. Например:

```
$ systemctl start openvpn@openvpn-client
```

В репозиториях Ubuntu 16.04.3 LTS пакет

`openvpn`

имеет версию 2.3.10, а поддержка AES-256-GCM появилась в 2.4. Придется или найти свежий пакет, или использовать шифрование AES-256-CBC (не забудь изменить и на сервере). Также могут возникнуть проблемы с отсутствием группы

`nobody`

(фиксится командой

```
groupadd nobody
```

). Если что-то еще пойдет не так, включай логи и чини.

```
$ scp -P 12222 eakj@52.214.41.150:"/tmp/{openvpn-client*,ca.crt,ta.key}" /et
c/openvpn/client/
```

Примерный файл конфигурации

```
/etc/openvpn/client/openvpn-client.conf
```

:

```
client dev tun proto tcp remote 127.0.0.1 1194 resolv-retry infinite nobind u
ser nobody group nobody persist-key persist-tun ca ca.crt cert openvpn-clien
t.crt key openvpn-client.key ### на клиенте 1 tls-auth ta.key 1 remote-cert-t
ls server cipher AES-256-GCM verb 3
```

Для отключения шифрования делаем то же, что и на сервере:

```
cipher none
```

.

При таком конфигурационном файле клиенту всегда нужно носить с собой пять файлов:

- файл конфигурации (openvpn-client.conf);
- сертификат CA (ca.crt);
- клиентский сертификат (openvpn-client.key);
- клиентский ключ (openvpn-client.key);
- ключ для tls-auth (ta.key).

Это может быть не всегда удобно, поэтому есть возможность записать их содержимое в конфигурационный файл. Вместо

```
ca ca.crt
```

,

```
cert client.crt
```

,

```
key client.key
```

и

```
tls-auth ta.key 1
```

используется

```
<ca></ca>
```

,

```
<cert></cert>
```

,

```
<key></key>
```

,

```
key-direction 1
```

и

```
<tls-auth></tls-auth>
```

. Выглядит это примерно так:

```
<ca> содержимое ca.crt </ca> <cert> содержимое openvpn-client.crt </cert> <ke
y> содержимое openvpn-client.key </key> ### Указываем что tls-auth на стороне
клиента key-direction 1 <tls-auth> содержимое ta.key </tls-auth>
```

Детальнее пример такого конфига рассмотрим при настройке клиента Android.
Стартуем:

```
$ systemctl start openvpn-client@openvpn-client
```

и проверяем:

```
$ systemctl status openvpn-client@openvpn-client
```

**Windows**

Скачать клиент OpenVPN для Windows можно с официального сайта. Конфиги хранятся в

```
C:\Program Files\OpenVPN\config
```

. После установки нужно скачать клиентские сертификаты и ключи с нашего сервера и поместить их в

```
C:\Program Files\OpenVPN\config
```

, как показано на скрине. Будем использовать те же сертификаты и ключи, что и для Linux (привет

```
duplicate-cn
```

).

Чтобы создать клиентский конфиг, открывай «Блокнот» от админа, копируй пример и удаляй

```
user nobody
```

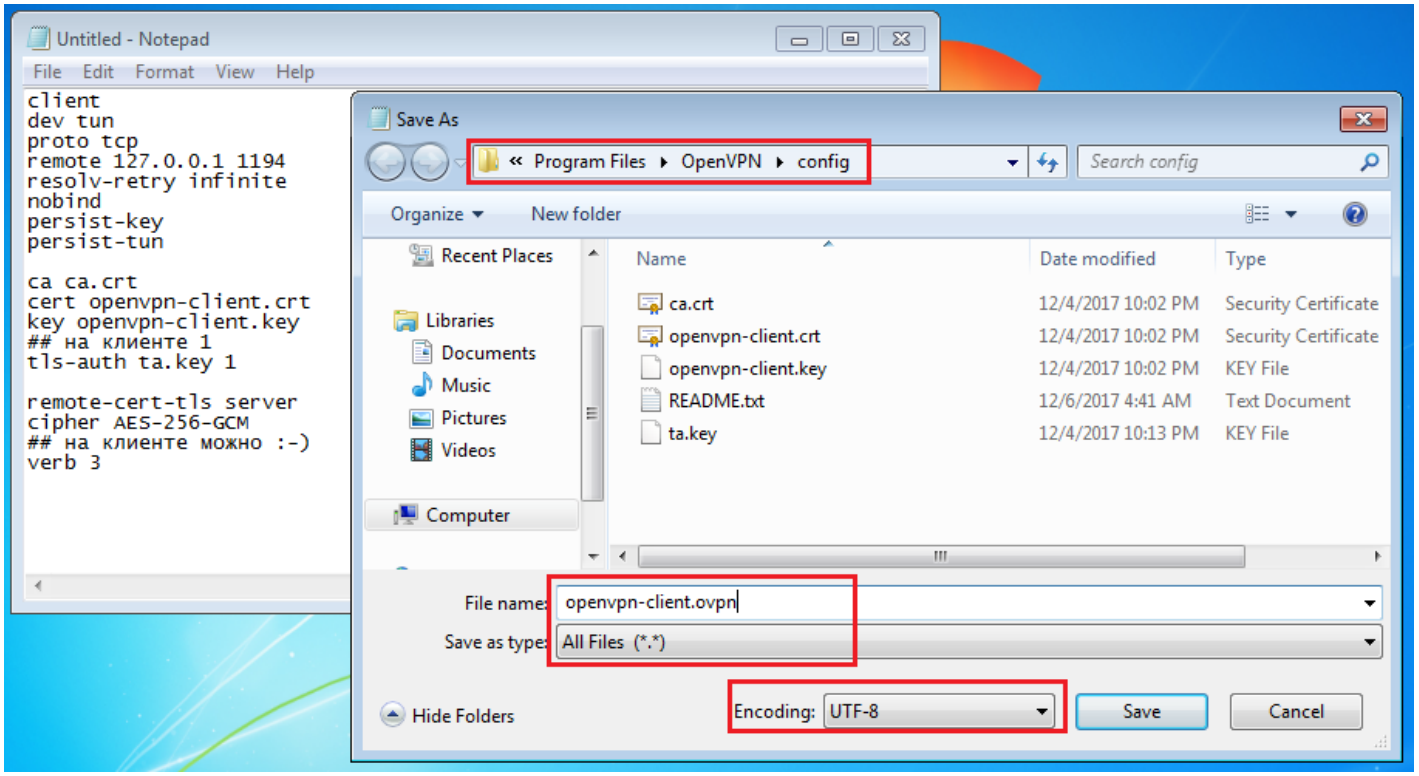
и

```
group nobody
```

. Должно получиться как на скрине. Заметь, используется расширение

```
.ovpn
```

.

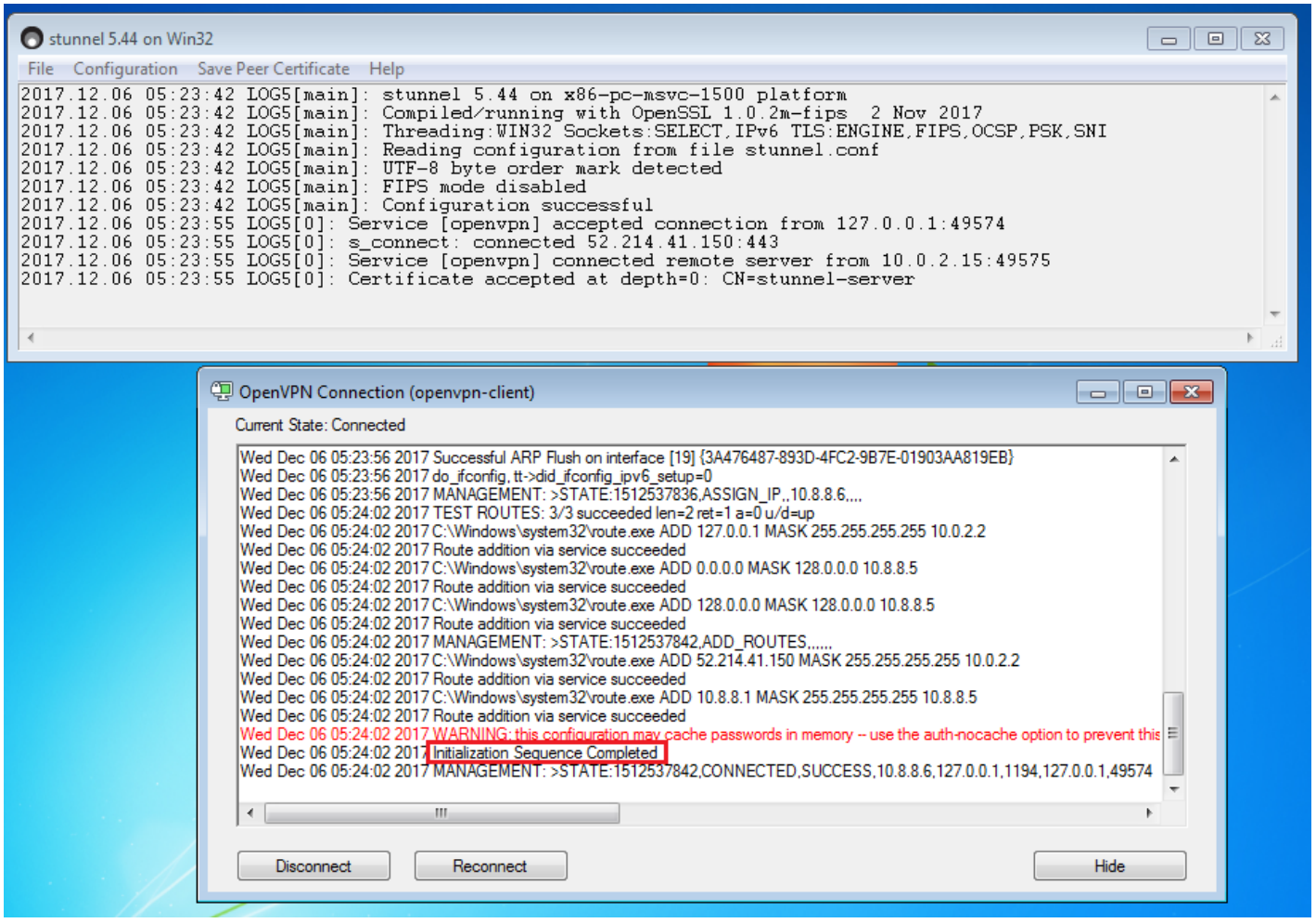



Конфиг OpenVPN для Windows

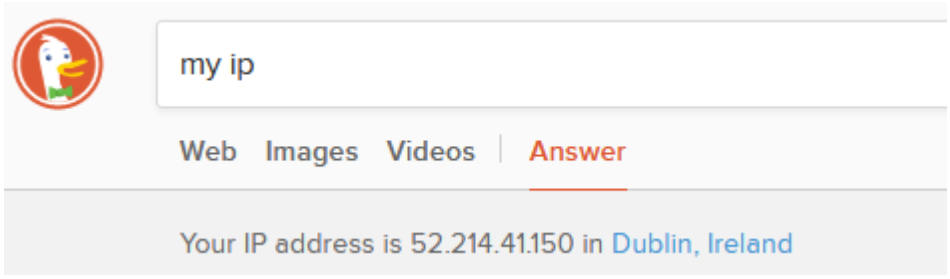
Стартуешь и видишь заветную надпись

```
Initialization Sequence Completed
```

— значит, все работает.




OpenVPN и stunnel в Windows




Убедимся

**Android**

Для Android есть приложение OpenVPN for Android, будем использовать именно его. Конфиг с записанными в него сертификатами и ключами выглядит примерно так:

```
client dev tun proto tcp remote 127.0.0.1 1194 resolv-retry infinite nobind u
ser nobody group nobody persist-key persist-tun remote-cert-tls server cipher
AES-256-GCM ### На клиенте можно :-) verb 3 ### Содержимое ca.crt <ca> -----B
EGIN CERTIFICATE----- ... -----END CERTIFICATE----- </ca> ### Содержимое open
vpn-client.crt <cert> -----BEGIN CERTIFICATE----- ... -----END CERTIFICATE---
-- </cert> ### Содержимое openvpn-client.key <key> -----BEGIN PRIVATE KEY----
- ... -----END PRIVATE KEY----- </key> ### Указываем, что tls-auth на стороне
клиента key-direction 1 ### Содержимое ta.key <tls-auth> ## ## 2048 bit OpenV
PN static key ## -----BEGIN OpenVPN Static key V1----- ... -----END OpenVPN S
tatic key V1----- </tls-auth>
```

Как ты мог заметить, файл

```
openvpn-client.crt
```

в начале содержит примерно следующую информацию.

```
Certificate: Data: Version: 3 (0x2) Serial Number: 9b:0a:56:f3:d4:70:97:66:d
9:92:81:54:26:fc:9c:53 Signature Algorithm: ecdsa-with-SHA256 Issuer: CN=Chan
geMe Validity Not Before: Dec 4 22:02:32 2017 GMT Not After : Dec 2 22:02:32
2027 GMT Subject: CN=openvpn-client
```

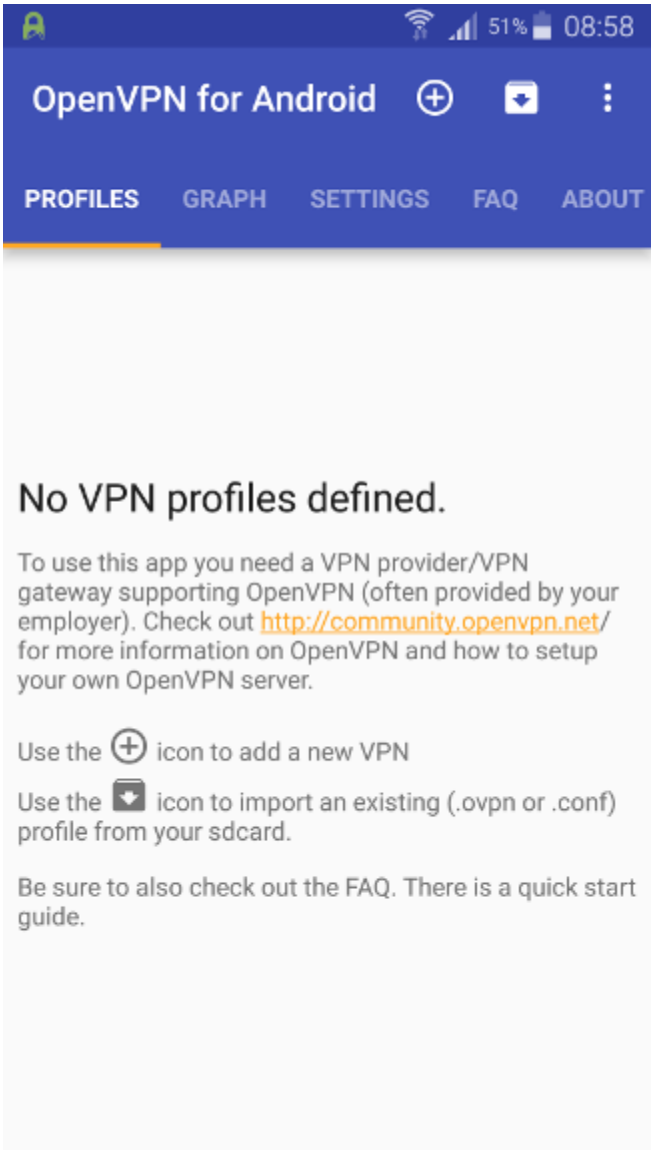
Все это можно опустить и добавить в файл конфига только сам сертификат (от пометок

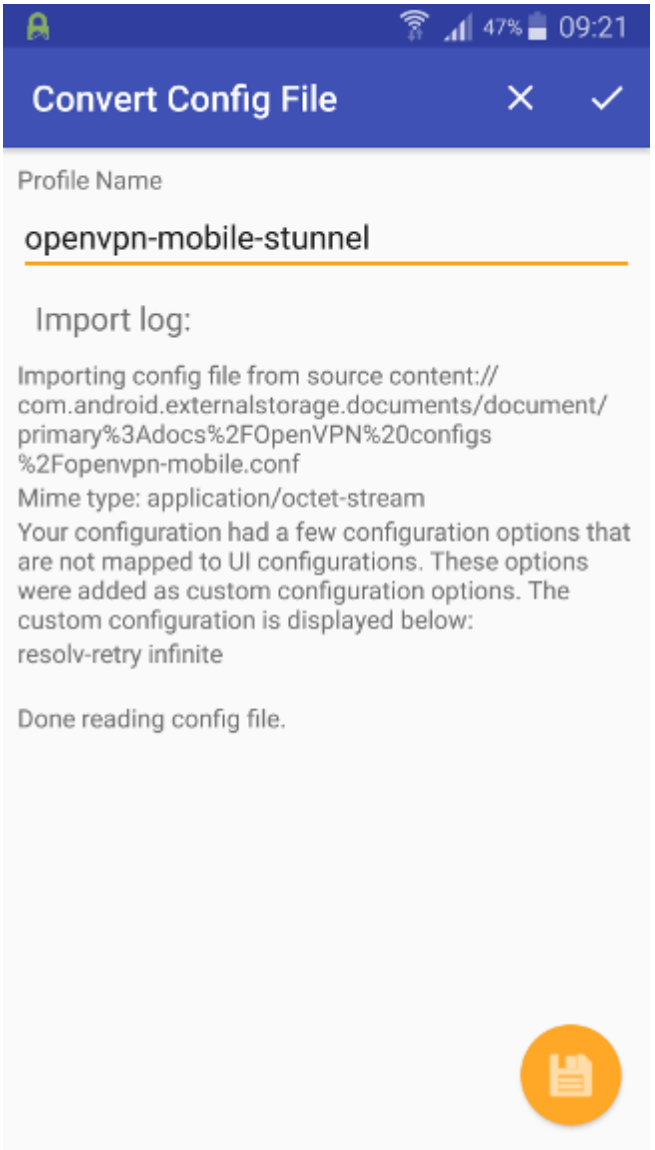
```
BEGIN
```

до

```
END
```

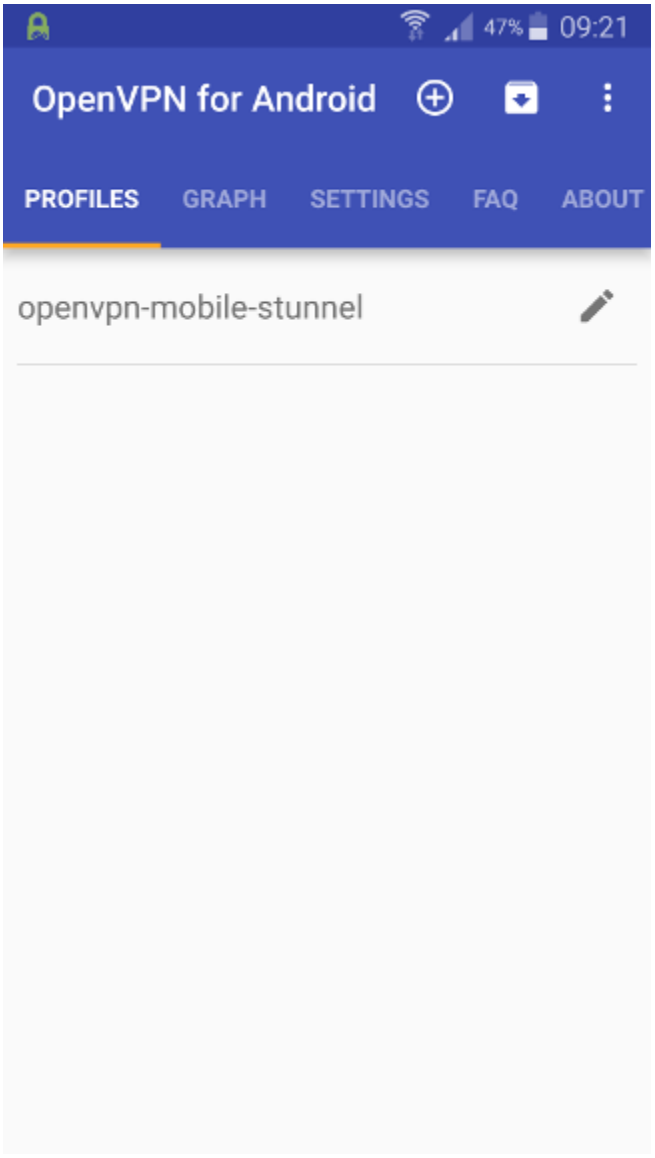
), как сделано в примере.
Переместим готовый конфиг на устройство и начнем настройку приложения. В главном меню жмешь плюсик, потом «Импорт», выбираешь нужный конфиг и сохраняешь.



OpenVPN на Android: главное меню и импорт профиля

Нажимаешь на импортированный профиль и смотришь за процессом подключения.





Подключается. Ура!

Наконец, видим три заветных слова: Initialization Sequence Completed. Все работает! В некоторых случаях может понадобиться подкорректировать значение MSS, о чем это приложение нам любезно сообщит. Удачной отладки!