Наверх

# Строим маршрутизатор в SOCKS на ноутбуке с Debian 10

**2020-03-08 в 22:02, admin**, рубрики: Debian, DNS, dnscrypt, linux, ntp, systemd, systemd-networkd, tun2socks, Беспроводные технологии, Настройка Linux, Серверное администрирование, Сетевые технологии, системное администрирование

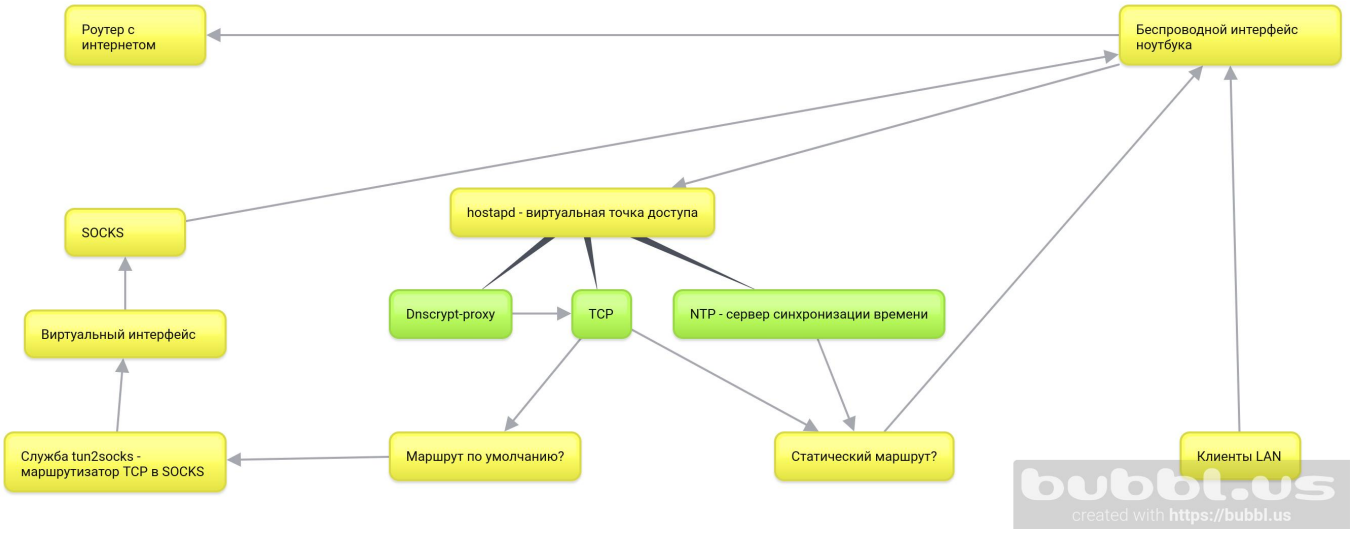
Целый год (или два) я откладывал публикацию данной статьи по главной причине — мной уже были опубликованы две статьи, в которых я описал процесс создания полноценного маршрутизатора в SOCKS из самого обычного ноутбка с Debian.

Однако, с тех пор стабильная версия Debian обновилась до Buster, мне в личку обратилось достаточное количество людей с просьбой помочь в настройке, а значит, мои предыдущие статьи не являются исчерпывающими. Что ж, я и сам догадывался, что методы, изложеннные в них, не до конца раскрывают все тонкости настройки Linux для маршрутизации в SOCKS. К тому же они написаны для Debian Stretch, а после обновления до Buster, в системе инициализации systemd, я заметил небольшие изменения во взаимодействии служб. Да и в самих статьях я не использовал systemd-networkd, хотя она лучше всего подходит для сложных сетевых конфигураций.

Кроме вышеуказаных изменений, в мою конфигурацию добавились такие службы как **hostapd** — служба для виртуализации точки доступа, **ntp** для синхронизации времени клиентов локальной сети, **dnscrypt-proxy** для шифрования соединений по протоколу DNS и отключения рекламы на клиентах

локальной сети, а также, как я упомянал ранее, **systemd-networkd** для конфигруации сетевых интерфейсов.

Вот простейшая блок-схема, внутреннего устройства такого маршрутизатора.




Итак, я напомню, какие цели преследует цикл данных статей:

1. Маршрутизировать в SOCKS все соединения ОС, а также соединения всех устройств, состоящих в одной сети с ноутбуком.
2. Ноутбук в моем случае должен оставаться полностью мобильным. То есть, давать возможность использовать окружение рабочего стола и не быть привязанным к физическому месуположению.
3. Последний пункт подразумевает подключение и маршрутизацию только через встроенный беспроводной интерфейс.
4. Ну, и конечно, создание исчерпывающего руководства, а также разбор соответсвующих технологий в меру моих скромных познаний.

Что будет рассмотрено в данной статье:

1. **git** — скачаем репоизитории проектов **tun2socks**, необходимого для маршрутизации трафика TCP в SOCKS, и **create_ap** — скрипта для автоматизации настройки виртуальной точки доступа с помощью **hostapd**.
2. **tun2socks** — построим и установим службу systemd в систему.
3. **systemd-networkd** — настроим беспроводной и виртуальные интерфейсы, таблицы статической маршрутизации и перенаправление пакетов.
4. **create_ap** — установим службу systemd в систему, настроим и запустим виртуальную точку доступа.

Необязательные шаги:

- **ntp** — установим и настроим сервер для сихронизации времени на клиентах виртуальной точки доступа.
- **dnscrypt-proxy** — зашифруем запросы DNS, маршрутизируем их в SOCKS и отключим рекламные домены для локальной сети.

# Зачем всё это?

Это один из способов организации защиты TCP-соединений локальной сети. Главное преимущество — все соединения идут в SOCKS, если для них не построен статический маршрут через оригинальный шлюз. Это значит, что не нужно прописывать настройки SOCKS-сервера ни отдельным программам, ни

клиентам в локальной сети — они все идут в SOCKS по-умолчанию, так как он является шлюзом по-умолчанию, пока мы не укажем обратного.

По сути мы добавляем второй шифрующий роутер в качестве ноутбука перед оригинальным роутером и используем интернет соединение оригинального роутера для уже зашифрованных SOCKS-запросов ноутбука, который, в свою очередь, маршрутизирует и шифрует запросы клиентов локальной сети.

С точки зрения провайдера мы постоянно подключены к одному серверу с зашифрованным трафиком.

Соответственно, все устройства подключаются к виртуальной точке доступа ноутбука.

# Установите в систему tun2socks

Пока на вашей машине есть интернет, скачайте все необходимые инструменты.

```
apt update
```

```
apt install git make cmake
```

Скачайте пакет badvpn

```
git clone https://github.com/ambrop72/badvpn
```

В вашей системе появится папка

`badvpn`

. Создайте отдельную папку для сборки

```
mkdir badvpn-build
```

Перейдите в нее

```
cd badvpn-build
```

Соберите

`tun2socks`

```
cmake ../badvpn -DBUILD_NOTHING_BY_DEFAULT=1 -DBUILD_TUN2SOCKS=1
```

Установите в систему

```
make install
```

- Параметр

  `-DBUILD_NOTHING_BY_DEFAULT=1`

  отключает сборку всех компонентов репозитория badvpn.
- `-DBUILD_TUN2SOCKS=1`

  включает в сборку компонент **tun2socks**.

- `make install`

  — установит бинарник tun2socks в вашу систему по адресу

  `/usr/local/bin/badvpn-tun2socks`

  .

## Установите службу tun2socks в systemd

Создайте файл

`/etc/systemd/system/tun2socks.service`

со следующим содержимым:

```
[Unit]
Description=SOCKS TCP Relay

[Service]
ExecStart=/usr/local/bin/badvpn-tun2socks --tundev tun2socks --netif-ipaddr 172.16.1.1 --netif-netma

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

- `--tundev`

  — принимает имя виртуального интерфейса, который мы инициализируем с помощью systemd-networkd.

- `--netif-ipaddr`

  — сетевой адрес «маршрутизатора» tun2socks, к которому подключается виртуальный интерфейс. Лучше сделать отдельной зарезервированной подсетью.

- `--socks-server-addr`

  — принимает сокет (

  `адрес:порт`

  сервера SOCKS).

Если ваш SOCKS сервер требует аутентификации, вы можете указать параметры

`--username`

и

`--password`

.

Далее зарегистрируйте службу

```
systemctl daemon-reload
```

И включите

```
systemctl enable tun2socks
```

Прежде чем запускать службу, обеспечим её виртуальным сетевым интерфейсом.

# Переходим на systemd-networkd

Включаем

`systemd-networkd`

:

```
systemctl enable systemd-networkd
```

Отключаем текущие сетевые службы.

```
systemctl disable networking NetworkManager NetworkManager-wait-online
```

- **NetworkManager-wait-online** — это служба, которая ждет наличие работающего сетевого подключения прежде, чем systemd продолжит запуск других служб, зависящих от наличия сети. Мы отключаем её, так как перейдем на аналог systemd-networkd.

Давайте сразу включим его:

```
systemctl enable systemd-networkd-wait-online
```

## Настройте беспроводной сетевой интерфейс

Создайте файл конфигурации systemd-networkd для беспроводного сетевого интерфейса

`/etc/systemd/network/25-wlp6s0.network`

.

```
[Match]
Name=wlp6s0

[Network]
Address=192.168.1.2/24
IPForward=yes
```

- **Name** — это имя вашего беспроводного интерфейса. Идентифицируйте его командой

  `ip a`

  .
- **IPForward** — директива, которая включает перенаправление пакетов на сетевом интерфейсе.
- **Address** отвечает за присвоение IP-адреса беспроводному интерфейсу. Мы указываем его статически потому, что при эквивалентной директиве

  `DHCP=yes`

  , systemd-networkd создаёт в системе шлюз по-умолчанию. Тогда весь трафик пойдет через оригинальный шлюз, а не через будущий виртуальный интерфейс в отличной подсети. Вы можете проверить текущий шлюз по-умолчанию командой

  `ip r`

### Создайте статический маршрут для удалённого сервера SOCKS

Если ваш SOCKS сервер не локальный, а удалённый, то вам необходимо создать для него статический

маршрут. Для этого добавьте секцию

`Route`

в конец созданного вами файла конфигурации беспроводного интерфейса со следующим содержанием:

```
[Route]
Gateway=192.168.1.1
Destination=0.0.0.0
```

- **`Gateway`**
  — это шлюз по-умолчанию или адрес вашей оригинальной точки доступа.
- **`Destination`**
  — адрес сервера SOCKS.

## Настройте wpa_supplicant для systemd-networkd

systemd-networkd использует wpa_supplicant для подключения к защищенной точке доступа. При попытке «поднять» беспроводной интерфейс systemd-networkd запускает службу

`wpa_supplicant@имя`

, где *имя* — это имя беспроводного интерфейса. Если вы не использовали systemd-networkd до этого момента, то, наверняка, эта служба в вашей системе отсутствует.

Поэтому создайте ее командой:

```
systemctl enable wpa_supplicant@wlp6s0
```

Я использовал

`wlp6s0`

в качестве имени своего беспроводного интерфейса. У вас это имя может отличаться. Вы можете узнать его командой

`ip l`

.

Теперь созданная служба

`wpa_supplicant@wlp6s0`

будет запускаться при «поднятии» беспроводного интерфейса, однако она, в свою очередь, будет искать настройки SSID и пароля точки доступа в файле

`/etc/wpa_supplicant/wpa_supplicant-wlp6s0`

. Поэтому необходимо создать его с помощью утилиты

`wpa_passphrase`

.

Для этого выполните команду:

```
wpa_passphrase SSID password>/etc/wpa_supplicant/wpa_supplicant-wlp6s0.conf
```

где **SSID** — это имя вашей точки доступа, password — пароль, а **wlp6s0** — имя вашего беспроводного интерфейса.

### Инициализируйте виртуальный интерфейс для tun2socks

Создайте файл для инициализации нового вирутального интерфейса в системе

`/etc/systemd/network/25-tun2socks.netdev`

```
[NetDev]
Name=tun2socks
Kind=tun
```

- **Name** — это имя, которое systemd-networkd назначит будущему виртуальному интерфейсу при его инициализации.
- **Kind** — это тип виртуального интерфейса. Исходя из названия службы tun2socks, вы можете догататься, что она использует интерфейс типа

  `tun`

  .
- **netdev** — это расширение файлов, которые

  `systemd-networkd`

  использует для инициализации виртуальных сетевых интерфейсов. Адрес и другие сетевые настройки для этих интерфейсов указываются в **.network**-файлах.

Создайте такой файл

`/etc/systemd/network/25-tun2socks.network`

со следующим содержимым:

```
[Match]
Name=tun2socks

[Network]
Address=172.16.1.2/24
Gateway=172.16.1.1
```

- `Name`

  — имя виртуального интерфейса, которое вы указали в **netdev**-файле.
- `Address`

  — IP адрес, который будет назначен вирутальному интерфейсу. Должен быть в одной сети с адресом, который вы указали в службе tun2socks
- `Gateway`

  — IP адрес «маршрутизатора» **tun2socks**, который вы указали при создании службы systemd.

Таким образом, интерфейс **tun2socks** имеет адрес

`172.16.1.2`

, а служба **tun2socks** —

`172.16.1.1`

, то есть является шлюзом для всех соединений с вирутального интерфейса.

# Настройте виртуальную точку доступа

Установите зависимости:

```
apt install util-linux procps hostapd iw haveged
```

Скачайте репозиторий **create_ap** на свою машину:

```
git clone https://github.com/oblique/create_ap
```

Перейдите в папку репозитория на вашей машине:

```
cd create_ap
```

Установите в систему:

```
make install
```

В вашей системе появится конфиг
`/etc/create_ap.conf`
. Вот основные опции для правки:

- **GATEWAY=10.0.0.1**
  — лучше сделать отдельной зарезервированной подсетью.
- **NO_DNS=1**
  — выключите, так как этим параметром будет управлять виртуальный интерфейс systemd-networkd.
- **NO_DNSMASQ=1**
  — выключите по той же причине.
- **WIFI_IFACE=wlp6s0**
  — беспроводной интерфейс ноутбука.
- **INTERNET_IFACE=tun2socks**
  — виртуальный интерфейс, созданный для tun2socks.
- **SSID=hostapd**
  — имя виртуальной точки доступа.
- **PASSPHRASE=12345678**
  — пароль.

Не забудьте включить службу:

```
systemctl enable create_ap
```

## Включите DHCP сервер в systemd-networkd

Служба
`create_ap`
инициализирует в системе виртуальный интерфейс **ap0**. По идее, на этом интерфейсе «висит» dnsmasq, но зачем устанавливать лишние службы, если systemd-networkd содержит встроенный DHCP-сервер?

Чтобы включить его, определим сетевые настройки для виртуальной точки. Для этого создайте файл

`/etc/systemd/network/25-ap0.network`

со следующим содержимым:

```
[Match]
Name=ap0

[Network]
Address=10.0.0.1/24
DHCPServer=yes

[DHCPServer]
EmitDNS=yes
DNS=10.0.0.1
EmitNTP=yes
NTP=10.0.0.1
```

После того, как служба create_ap инициализирует виртуальный интерфейс

`ap0`

, systemd-networkd автоматически присвоит ему IP-адрес и включит DHCP сервер.

Строки

**`EmitDNS=yes`**

и

**`DNS=10.0.0.1`**

передают настройки DNS сервера устройствам, подключенным к точке доступа.

Если вы не планируете использовать локальный DNS сервер — в моём случае это dnscrypt-proxy — можете установить

**`DNS=10.0.0.1`**

в

**`DNS=192.168.1.1`**

, где *192.168.1.1* — адрес вашего оригинального шлюза. Тогда запросы DNS вашего хоста и локальной сети пойдут в незашифрованном виде через серверы провайдера.

**`EmitNTP=yes`**

и

**`NTP=192.168.1.1`**

передают настройки NTP.

То же самое касается строки

**`NTP=10.0.0.1`**

.

# Установите и настройте NTP сервер

Установите в систему:

```
apt install ntp
```

Правьте конфиг

`/etc/ntp.conf`

. Закомментируйте адреса стандартных пулов:

```
#pool 0.debian.pool.ntp.org iburst
#pool 1.debian.pool.ntp.org iburst
#pool 2.debian.pool.ntp.org iburst
#pool 3.debian.pool.ntp.org iburst
```

Добавьте адреса публичных серверов, например, Google Public NTP:

```
server time1.google.com ibrust
server time2.google.com ibrust
server time3.google.com ibrust
server time4.google.com ibrust
```

Предоставьте доступ к серверу клиентам из вашей сети:

```
restrict 10.0.0.0 mask 255.255.255.0
```

Включите трансляцию в вашу сеть:

```
broadcast 10.0.0.255
```

Наконец, добавьте адреса этих серверов в таблицу статической маршрутизации. Для этого откройте файл конфигурации беспроводного интерфейса

`/etc/systemd/network/25-wlp6s0.network`

и добавьте в конец секции

`Route`

.

```
[Route]
Gateway=192.168.1.1
Destination=216.239.35.0

[Route]
Gateway=192.168.1.1
Destination=216.239.35.4

[Route]
Gateway=192.168.1.1
Destination=216.239.35.8

[Route]
Gateway=192.168.1.1
Destination=216.239.35.12
```

Вы можете узнать адреса ваших серверов NTP, используя утилиту

`host`

следующим образом:

```
host time1.google.com
```

# Установите dnscrypt-proxy, уберите рекламу и скройте DNS трафик от провайдера

```
apt install dnscrypt-proxy
```

Чтобы обслуживать DNS запросы хоста и локальной сети, правьте сокет `/lib/systemd/system/dnscrypt-proxy.socket`. Измените следующие строки:

```
ListenStream=0.0.0.0:53
ListenDatagram=0.0.0.0:53
```

Перезапустите `systemd`:

```
systemctl daemon-reload
```

Правьте конфиг `/etc/dnscrypt-proxy/dnscrypt-proxy.toml`:

```
server_names = ['adguard-dns']
```

Чтобы направить соединения dnscrypt-proxy через tun2socks, добавьте ниже:

```
force_tcp = true
```

Правьте конфиг `/etc/resolv.conf`, который сообщает DNS сервер хосту.

```
nameserver 127.0.0.1
nameserver 192.168.1.1
```

Первая строчка включает использование dnscrypt-proxy, вторая — использует оригинальный шлюз, в случае, когда сервер dnscrypt-proxy недоступен.

**Готово!**

Перезагрузитесь или остановите действующие сетевые службы:

```
systemctl stop networking NetworkManager NetworkManager-wait-online
```

И перезапустите все необходимые:

```
systemctl restart systemd-networkd tun2socks create_ap dnscrypt-proxy ntp
```

Псоле перезагрузки или перезапуска у вас появится вторая точка доступа, которая маршрутизирует хост и устройства локальной сети в SOCKS.

Примерно так выглядит вывод

```
ip a
```

обычного ноутбука:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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: tun2socks: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group def
    link/none
    inet 172.16.1.2/24 brd 172.16.1.255 scope global tun2socks
       valid_lft forever preferred_lft forever
    inet6 fe80::122b:260:6590:1b0e/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
3: enp4s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default ql
    link/ether e8:11:32:0e:01:50 brd ff:ff:ff:ff:ff:ff
4: wlp6s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 4c:ed:de:cb:cf:85 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 brd 192.168.1.255 scope global wlp6s0
       valid_lft forever preferred_lft forever
    inet6 fe80::4eed:deff:fecb:cf85/64 scope link
       valid_lft forever preferred_lft forever
5: ap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 4c:ed:de:cb:cf:86 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global ap0
       valid_lft forever preferred_lft forever
    inet6 fe80::4eed:deff:fecb:cf86/64 scope link
       valid_lft forever preferred_lft forever
```

# В итоге

1. Провайдер видит только зашифрованное соединение к вашему серверу SOCKS, а значит, ничего не видит.
2. И всё же он видит ваши NTP запросы, чтобы предотвратить это, удалите статические маршруты для NTP серверов. Однако, не факт, что ваш сервер SOCKS разрешает протокол NTP.

## Костыль, замеченный на Debain 10

Если попытаться перезапустить сетевую службу из консоли, то она упадет с ошибкой. Связанно это с тем, что её часть в виде виртуального интерфейса привязана к службе tun2socks, а значит используется. Чтобы перезапустить сетевую службу, сначала нужно остановить службу tun2socks. Но, я думаю, если вы дочитали до конца, для вас это точно не пробема!

*Автор: Анатолий Никифоров*



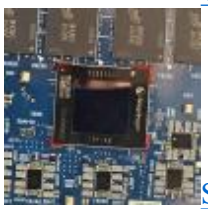
*Источник*

## Рекомендованный контент


Представлен похожий на фантастический истребитель маршрутизатор Thor X4 AX6000


Быстро и дёшево: новейший маршрутизатор Xiaomi с поддержкой WiFi 7 стоит $35

Сборка Debian пакетов для расширений PHP


Snapdragon X Elite в ноутбуке работает одинаково хорошо как от сети, так и от батареи. Как минимум в Asus Vivobook S 15


Система охлаждения Snapdragon X Elite в ноутбуке Asus вызывает вопросы относительно энергопотребления. Появились фото

Самый мощный игровой мобильный процессор AMD в самом мощном игровом ноутбуке MSI. Представлен Titan 18 PRO Ryzen Edition

* Ваше имя*

Ваш e-mail (не отображается в списке сообщений)

* - обязательные к заполнению поля

Отправить!

---

**Новости**

- Смартфоны Google Pixel принято ругать за их платформы, но тесты показали, что Tensor G4 одна из самых энергоэффективных на рынке
- Каждый из этих пузырей почти в 100 раз больше Солнца. Астрономы показали далёкую звезду с невиданной ранее детализацией ещё и на видео
- Делаем простой рисовальщик в PySide6
- Взлет и падение Силиконовой Долины как центра инноваций 21 века, 1995-2024 года
- Redox OS: операционная система на Rust с микроядром и поддержкой Unix. Что это за ОС?
- Решаем загадку Джиндоша на SQL в пять строчек
- Как организовать партнёрскую сеть и привлечь 60 партнёров за один год + чек-лист
- Правда ли в России умер маркетинг, и что ожидать дальше? Реалии 2024 года

**Актуальные темы**

android apple c++ e-commerce Facebook Google iOS java javascript linux mail.ru group microsoft PHP python Windows 8 Вконтакте Госвеб Нам пишут Онлайн-медиа Программирование Роскомнадзор Россия СМИ США Текучка аналитика безопасность блокировка законотворчество запуск инвестиции интернет-реклама кадры кейсы медиа мобильные приложения НОВОСТИ онлайн видео советы соцсети статистика суд телеком эффективность бизнеса яндекс

**Архив**

- Сентябрь 2024 (1249)
- Август 2024 (2071)
- Июль 2024 (583)
- Июнь 2024 (595)
- Май 2024 (537)
- Апрель 2024 (452)
- Март 2024 (445)
- Февраль 2024 (351)
- Январь 2024 (757)
- Декабрь 2023 (548)




https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js