Туннели. VPN

VPN. OpenVPN. 6to4

[VPN — Virtual Private Network](#_vqnwkm1ttzvn)

[OpenVPN](#_5kcwmzdqwrad)

Практическое [задание](#_3tfrjxxltv85)

[Дополнительные материалы](#_52pbnarpzji0)

[Используемая литература](#_uvp6qax5r1ok)

# 

# VPN — Virtual Private Network

Рассмотрим, как связать две сети предприятия, находящиеся в разных городах. В рамках клиент-серверного взаимодействия шлюз одной сети (VPN-клиент) будет подключаться к шлюзу другой (VPN-сервер). Транспортировка будет осуществляться через внешнюю сеть. Для этого необходимо выполнить три задачи.

* Обеспечить аутентификацию, чтобы удостовериться, что VPN-клиент — действительно тот, за кого себя выдает.
* Обеспечить шифрование трафика, чтобы злоумышленники в Интернете не могли его перехватить.
* Обеспечить туннелирование — следование IP-пакетов, упакованных в конечном итоге в другие пакеты. Таким образом, все хопы (прыжки), которые делаются при нетуннельном соединении, в данном случае будут не видны. Между шлюзами по туннелю будет всего один прыжок, что позволит настроить маршрут через туннель и объединить две физически не связанные сети в единую безопасную сеть.

Сети, построенные таким образом, называют виртуальными частными сетями (VPN — Virtual Private Networks).

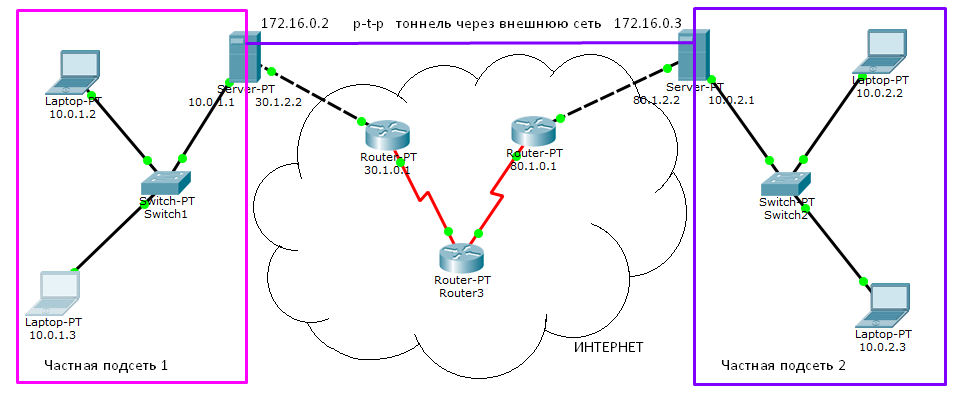
Существуют разные туннелирующие протоколы. Например, PPPoE является туннелирующим, но работает на канальном уровне. С его помощью можно соединить клиент и сервер (как правило, шлюз в Интернете), если машины присутствуют в одной сети. При типе сетевого соединения **bridge** можно настроить это и в виртуальной машине. Провайдеры используют PPPoE из-за возможности аутентифицировать и логировать действия пользователя.

PPTP формально является защищенным туннелирующим протоколом. Но на практике он не криптостойкий. Используется там, где безопасность как раз не требуется. В отличие от PPPoE протокол PPTP работает поверх сетевого уровня и не налагает требование находиться в одной физической сети. Поэтому он используется провайдерами для предоставления «белого» IP-адреса клиенту. Обычно инфраструктура провайдера состоит из «серых» IP-адресов, отсюда потребность в туннелировании. В остальном PPTP предоставляет аналогичные PPPoE возможности: аутентифицировать и логировать действия пользователя. Также PPPoE используется в DSL-модемах.

Оба вышеуказанных протокола поддерживаются в Windows, но возможно настроить и в Linux.

Одной из полноценных реализаций защищенного VPN является OpenVPN. Он позволяет надежно передавать данные, подключать отдельных удаленных пользователей к офисным сетям, связывать географически удаленные сети предприятия в одну защищенную. Для доступа к OpenVPN в Windows потребуется установка дополнительного программного обеспечения.

Схема объединения двух офисных сетей в одну частную.



Подобное объединение может быть выполнено с помощью OpenVPN или PPTP, но невозможно с помощью PPPoE. С точки зрения логической инфраструктуры (в этом несложно убедиться с помощью команды **traceroute**) для абонентов частных подсетей 1 и 2 соединение выглядит так, словно шлюзы напрямую соединены кабелем. На самом деле кабель виртуальный (соединение «точка–точка» и есть туннель). Один прыжок туннелированного пакета с точки зрения маршрута внутри тоннеля физически реализуется несколькими прыжками несущего его пакета. PPTP и OpenVPN предоставляют аналогичные возможности, но не устойчивы к взлому с точки зрения безопасности PPTP.

## OpenVPN

Представим две машины, играющие роль шлюза. На них может быть установлен DHCP-сервер, кеширующий DNS-сервер, настроен сетевой фильтр. Остальные машины локальной сети выходят в Интернет через него.

Задача — связать две сети так, чтобы можно было работать в безопасной среде, объединяющей обе подсети предприятия.

Напрямую прописать маршрут в сеть 10.0.2.0/255.255.255.0 через 80.1.2.2 мы не можем, так как.

* Небезопасно передавать внутренний трафик через внешнюю сеть.
* Адреса вида 10.0.2.0 и 10.0.1.0 не маршрутизируются в сети Интернет и будут отброшены сетевыми маршрутизаторами.
* Нельзя назначить шлюзом адрес машины, недоступной на канальном уровне.

Понадобится туннелирование, построенное по клиент-серверному взаимодействию. VPN-сервер слушает указанный порт (например, UDP-порт 6666), а клиент подключается к нему. В случае разрыва соединения переподключаться будет именно клиент. Но когда соединение установлено, обе сети будут равноправными. По крайней мере, можно настроить, чтобы связанные подсети маршрутизировались в обе стороны. Это не всегда нужно, например, может быть излишне, когда задача — подключить к частной сети компьютер удаленного сотрудника.

Предположим, компьютер с адресами 10.0.1.1 (в локальную сеть) и 30.1.2.2 (во внешнюю сеть) будет клиентом. Машина с адресами 10.0.2.1 (во внутреннюю сеть) и 80.1.2.2 (во внешнюю сеть) — сервером.

Подключаемся по ssh на компьютер 80.1.2.2 (VPN-сервер). Устанавливаем **openvpn**.

|  |
| --- |
| # sudo apt install openvpn |

Создаем сертификаты. Для этого воспользуемся программой **easy-rsa** для генерации сертификатов. Ранее она входила в состав **openssh**, а сейчас это отдельный проект, потому скачаем ее дополнительно.

Создаем директорию для ключей и переходим в нее.

|  |
| --- |
| # mkdir /etc/openvpn/keys  # cd /etc/openvpn/keys |

Скачиваем и распаковываем утилиту **easy-rsa**.

|  |
| --- |
| # wget https://github.com/OpenVPN/easy-rsa/archive/master.zip  # unzip master.zip |

Создаем структуру публичных PKI-ключей.

|  |
| --- |
| # cd /etc/openvpn/keys/easy-rsa-master/easyrsa3  # mv vars.example vars  # ./easyrsa init-pki |

Будет создана папка **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki**. Создаем удостоверяющий центр **CA**.

|  |
| --- |
| # ./easyrsa build-ca |

Вводим и запоминаем пароль для центра сертификации, он нам понадобится. Получим два ключа. Секретный (приватный) будет храниться на сервере и не должен его покидать: **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki/private/ca.key**. Открытый ключ вместе с пользовательскими сертификатами будем передавать клиентам: **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki/ca.crt**.

Создаем запрос сертификата для сервера без пароля с помощью опции **nopass**:

|  |
| --- |
| # ./easyrsa gen-req server nopass |

Подписываем запрос на получение сертификата у нашего CA.

|  |
| --- |
| # ./easyrsa sign-req server server |

Вводим пароль от CA, который указывали раньше, и отвечаем на вопрос «Yes». Результат – подписанный удостоверяющим центром сертификат для сервера: **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki/issued/server.crt**

Также понадобится файл с параметрами для работы алгоритма Диффи-Хеллмана (протокол Диффи-Хеллмана позволяет получить общий секретный симметричный ключ при работе через незащищенный от прослушивания канал). Создаем его (на это потребуется время).

|  |
| --- |
| # ./easyrsa gen-dh |

Получаем файл dh-сертификата: **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki/dh.pem.** Копируем в папку **/etc/openvpn** все ключи, необходимые для работы OpenVPN-сервера.

|  |
| --- |
| # cp pki/ca.crt /etc/openvpn/ca.crt  # cp pki/dh.pem /etc/openvpn/dh.pem  # cp pki/issued/server.crt /etc/openvpn/server.crt  # cp pki/private/server.key /etc/openvpn/server.key |

Создадим ключи для клиента.

|  |
| --- |
| # ./easyrsa gen-req client nopass  # ./easyrsa sign-req client client |

Будет запрошен **Common name** клиента. Рекомендуется указать имя хоста клиента — понадобится в настройках. В результате получаем.

* **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki/issued/client.crt**
* **/etc/openvpn/keys/easy-rsa-master/easyrsa3/pki/private/client.key**

Создаем папку, например.

|  |
| --- |
| # mkdir /etc/openvpn/clientfiles |

Копируем в папку оба клиентских файла и **ca.crt**.

|  |
| --- |
| # cp pki/ca.crt /etc/openvpn/clientfiles  # cp pki/issued/client.crt /etc/openvpn/clientfiles  # cp pki/ private/client.key /etc/openvpn/clientfiles |

Запаковываем в архив.

|  |
| --- |
| # cd /etc/openvpn/clientfiles  # tar -pczf clientfiles.tgz \* |

И отправляем на клиентскую машину с помощью **scp** (на наших машинах установлен **ssh**).

|  |
| --- |
| # scp clientfiles.tgz user@30.1.2.2:/home/user |

Настроим конфигурационный файл для сервера.

|  |
| --- |
| # mcedit /etc/openvpn/server.conf |

Практически создаем с нуля.

|  |
| --- |
| port 6666 # 6666 — если предпочитаете использовать нестандартные порты для работы  # либо используйте стандартный 1194  proto udp # протокол может быть и tcp, если есть необходимость в этом (слишком много потерь)  dev tun # tun работает на сетевом уровне, tap — на канальном. Нам достаточно tun  # файлы сертификатов и ключей  ca /etc/openvpn/ca.crt  cert /etc/openvpn/server.crt  key /etc/openvpn/server.key  dh /etc/openvpn/dh.pem  server 172.16.0.0 255.255.255.0 # подсеть для туннеля может быть любой  route 10.0.2.0 255.255.255.0 # указываем подсеть, к которой будем обращаться через vpn,  # это клиентская подсеть  # Следующие три опции можно пропустить, если создаем удаленное рабочее место  # Без них только клиент будет иметь доступ к серверным ресурсам  # Они понадобятся для соединения офисов  ifconfig-pool-persist ipp.txt # файл с записями соответствий clinet – ip  client-to-client # позволяет клиентам OpenVPN подключаться друг к другу  client-config-dir /etc/openvpn/ccd # директория с индивидуальными настройками клиентов  keepalive 10 20 #Время ожидания, через которое производится реконнект, поэтому лучше ставить 10-20 секунд, особенно для серверов-клиентов  comp-lzo #по дефолту no, сжатие траффика  persist-key  persist-tun  status /var/log/openvpn/openvpn-status.log  log /var/log/openvpn/openvpn.log  verb 3 |

Создаем необходимые директории.

|  |
| --- |
| # mkdir /etc/openvpn/ccd && mkdir /var/log/openvpn |

Создаем файл конфигурации клиента в директории, указанной в параметре **client-config-dir**, учитывая сказанное выше. Эти настройки нужны, чтобы связать две подсети (чтобы клиентская сеть тоже маршрутизировалась со стороны сервера).

|  |
| --- |
| # mcedit /etc/openvpn/ccd/client |

Обратите внимание, что **client** — имя сервера (**common name**). Желательно, чтобы оно совпадало с доменным именем, прописанным в том числе при генерации клиентского сертификата. Если же в качестве **common name** указан IP-адрес, придется использовать его.

|  |
| --- |
| iroute 10.0.2.0 255.255.255.0 |

Запускаем сервер.

|  |
| --- |
| # systemctl start openvpn@server.service  # systemctl enable openvpn@server.service |

Проверяем.

|  |
| --- |
| # netstat -ulnp | grep 6666 |

Если все правильно сделано, увидим прослушиваемый порт.

Так как рассматриваем задачу для соединения локальных сетей, необходимо разрешить форвардинг пакетов и добавить в **IPTABLES** правила, разрешающие пересылку пакетов между подсетями.

|  |
| --- |
| # Включить IP-форвардинг в ядре  echo "1" > /proc/sys/net/ipv4/ip\_forward #будет работать до ребута только, поэтому надо поправить  # /etc/sysctl.conf  # Разрешить входящие соединения на порт OpenVPN  iptables -A INPUT -p UDP --dport 6666 -j ACCEPT  # Разрешить форвардинг между подсетью OpenVPN и локальной  # Правила следует поставить в начале цепочки, если уже определены другие правила  iptables -A FORWARD -s 10.0.1.0/24 -d 10.0.2.0/24 -j ACCEPT  iptables -A FORWARD -d 10.0.2.0/24 -s 10.0.1.0/24 -j ACCEPT |

Не забываем изменить или добавить опцию **IP-forwarding** в **/etc/sysctl.conf**.

|  |
| --- |
| net.ipv4.ip\_forward = 1 |

Перейдем к клиенту. Подключаемся по **ssh** на компьютер 30.1.2.2 (VPN-клиент). Устанавливаем **openvpn**.

|  |
| --- |
| # sudo apt install openvpn |

Распаковываем принятые ранее файлы.

|  |
| --- |
| # cp /home/user/clientfiles.tgz /etc/openvpn  # cd /etc/openvpn  # tar -xzf clientfiles.tgz |

Настраиваем файл конфигурации **client.conf**.

|  |
| --- |
| dev tun  proto udp  remote 80.1.2.2 6666 # Здесь IP-адрес и порт сервера, видимый на данный момент  client  resolv-retry infinite  ca /etc/openvpn/ca.crt  cert /etc/openvpn/client.crt  key /etc/openvpn/client.key  route 10.0.1.0 255.255.255.0 # Подсеть со стороны сервера  persist-key  persist-tun  comp-lzo  verb 3  status /var/log/openvpn/openvpn-status.log 1  status-version 3  log-append /var/log/openvpn/openvpn-client.log |

Создаем каталог для логов.

|  |
| --- |
| # mkdir /var/log/openvpn |

Запускаем.

|  |
| --- |
| # systemctl start openvpn@client.service  # systemctl enable openvpn@client.service |

Также можно проверить.

|  |
| --- |
| # ifconfig  # route  # ping 10.0.1.1 |

Чтобы абоненты тоже могли подключаться к другой половине сети.

|  |
| --- |
| # Включить IP-форвардинг в ядре (если стоит 0)  echo "1" > /proc/sys/net/ipv4/ip\_forward  # Разрешить форвардинг между подсетью OpenVPN и локальной  # Правила следует поставить в начале цепочки, если определены другие правила  iptables -A FORWARD -s 10.0.1.0/24 -d 10.0.2.0/24 -j ACCEPT  iptables -A FORWARD -d 10.0.2.0/24 -s 10.0.1.0/24 -j ACCEPT |

Не забываем изменить опцию IP-forwarding (если установлена в 0) или добавить (если отсутствует) в **/etc/sysctl.conf**.

|  |
| --- |
| net.ipv4.ip\_forward = 1 |

Если хотим, чтобы клиентская сеть пользовалась OpenVPN-сервером для шлюза доступа в Интернет, понадобится на сервере настроить **NAT** через **iptables**.

|  |
| --- |
| iptables -t nat -A POSTROUTING -s 10.2.0.0/24 -o ens33 -j MASQUERADE |

Уточните наименование сетевого интерфейса на вашей машине (вместо **ens33** может быть **eth0** и так далее).

Если VDS-хостер дает **/48** или **/64** сеть IPv6-адресов, можем подключить в нее все домашние компьютеры или виртуальные машины, используя туннель. Но многие хостеры экономят и выдают по одному адресу. Это неправильно, но можно попробовать воспользоваться не единственным IPv6-адресом, а механизмом **6to4**.

Небольшой скрипт позволит настроить туннель (не VPN, а туннель 6to4) **ip6to4.sh**.

|  |
| --- |
| #!/bin/bash  ip tunnel add tun6to4 mode sit remote any local 185.195.5.2 ttl 64  ip link set dev tun6to4 up  ip -6 addr add 2002:B9C3:0502::FACE/128 dev tun6to4  ip -6 route add 2000::/3 via ::192.88.99.1 dev tun6to4 metric 1 |

6to4-адрес получаем из IPv4-адреса, записав его в шестнадцатеричной системе в двух хекстетах после **2002:,** и получаем сеть **/48**, которой можем распоряжаться по своему усмотрению.

Осталось отредактировать конфигурацию **openvpn** на сервере.

|  |
| --- |
| server-ipv6 2001:B9C3:0502:FACE::1/64  topology subnet   # IPv6 routes push "route-ipv6 2001:D9C3:0502:FACE::/64" push "route-ipv6 ::/0" |

В **dhcp-options** можно указать IPv6 DNS-серверов (например, от Google). Чтобы все работало, нужно в **iptables** разрешить **anycast** адрес 192.88.99.1.

|  |
| --- |
| -A INPUT -s 192.88.99.0/24 -j ACCEPT  -A OUTPUT -d 192.88.99.0/24 -j ACCEPT |

И настроить **ip6tables**.

|  |
| --- |
| # Generated by ip6tables-save v1.6.0 on Wed Jul 19 12:10:11 2017 \*filter :INPUT DROP [0:0] :FORWARD DROP [0:0] :OUTPUT ACCEPT [0:0] -A INPUT -i lo -j ACCEPT -A INPUT -p tcp -m conntrack --ctstate NEW -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -j DROP -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 128 -j ACCEPT -A FORWARD -p tcp -m conntrack --ctstate NEW -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -j DROP -A FORWARD -p ipv6-icmp -m icmp6 --icmpv6-type 128 -j ACCEPT -A FORWARD -i tun0 -j ACCEPT -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT COMMIT # Completed on Wed Jul 19 12:10:11 2017 |

**iptables** полностью.

|  |
| --- |
| # Generated by iptables-save v1.6.0 on Mon Jul 17 16:52:01 2017  \*nat  :PREROUTING ACCEPT [210:12952]  :INPUT ACCEPT [87:5224]  :OUTPUT ACCEPT [7:546]  :POSTROUTING ACCEPT [0:0]  -A POSTROUTING -o ens3 -j MASQUERADE  COMMIT  # Completed on Mon Jul 17 16:52:01 2017  # Generated by iptables-save v1.6.0 on Mon Jul 17 16:52:01 2017  \*filter  :INPUT DROP [110:7021]  :FORWARD DROP [12:872]  :OUTPUT DROP [0:0]  -A INPUT -i lo -j ACCEPT  -A INPUT -s 192.88.99.0/24 -j ACCEPT  -A OUTPUT -d 192.88.99.0/24 -j ACCEPT  -A INPUT -p icmp -j ACCEPT  -A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT  -A INPUT -i ens3 -p tcp -m multiport --dports 22,25,80,443 -j ACCEPT  -A INPUT -i ens3 -p udp -m udp --dport 6666 -j ACCEPT  -A FORWARD -s 10.0.1.0/24 -d 10.0.2.0/24 -j ACCEPT  -A FORWARD -s 10.0.1.0/24 -d 10.0.2.0/24 -j ACCEPT  -A FORWARD -s 172.16.0.0/24 -j ACCEPT  -A FORWARD -d 172.16.0.0/24 -j ACCEPT  -A OUTPUT -o lo -j ACCEPT  -A OUTPUT -p icmp -j ACCEPT  -A OUTPUT -p tcp -m tcp --sport 32768:60999 -j ACCEPT  -A OUTPUT -p udp -m udp --sport 32768:60999 -j ACCEPT  -A OUTPUT -o ens3 -p tcp -m multiport --dports 22,25,80,443 -j ACCEPT  -A OUTPUT -o ens3 -p udp -m udp --sport 6666 -j ACCEPT  COMMIT  # Completed on Mon Jul 17 16:52:01 2017 |

Не забываем на сервере включить **ip forwarding** для **ipv6**.

# Практическое задание

1. Настроить OpenVPN, связать несколько виртуальных машин с его помощью.
2. \* Сделать одну из машин, настроенных в задании выше, шлюзом доступа в Интернет. Настроить NAT.
3. \* Настроить OpenVPN на VDS и клиент в VirtualBox/VmWare. Предоставить в VMWare ipv6 с помощью 6to4.

# Дополнительные материалы

1. Быстрый старт OpenVPN: <http://notessysadmin.com/quickstart-openvpn-server>
2. Настройка OpenVPN: <http://linux-bash.ru/mseti/51-openvpn.html>
3. <http://ru.wikipedia.org/wiki/6to4>
4. <http://version6.ru>
5. <https://habr.com/post/321486/>
6. Установка **pptpd**: <https://jtprog.ru/simple-vpn/>

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы.

1. [https://ru.wikipedia.org/w/index.php?title=Сетевая\_модель\_OSI&oldid=83830272](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&oldid=83830272)
2. [https://ru.wikipedia.org/wiki/Динамический\_порт](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%82)
3. <http://www.resoo.org/docs/iptables_ipchains/iptables-help.pdf>
4. <https://www.cyberciti.biz/faq/setting-up-an-network-interfaces-file/>
5. <https://help.ubuntu.com/community/IptablesHowTo>
6. [http://help.ubuntu.ru/wiki/настройка\_сети\_вручную](http://help.ubuntu.ru/wiki/%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0_%D1%81%D0%B5%D1%82%D0%B8_%D0%B2%D1%80%D1%83%D1%87%D0%BD%D1%83%D1%8E)
7. <https://srez.org/2015/09/razdacha-ipv6-po-openvpn-tunnelyu.html>