**Wireguard VPN**

**Оглавление**

[**Справка и первоначальные настройки** 2](#_Toc30885429)

[**Принцип работы** 4](#_Toc30885430)

[**Настройка** 5](#_Toc30885431)

[1. Установка необходимых пакетов 5](#_Toc30885432)

[2. Генерация ключей 5](#_Toc30885433)

[3.1 Настройка конфигурации сервера 5](#_Toc30885434)

[3.2 Настройка конфигурации клиента 6](#_Toc30885435)

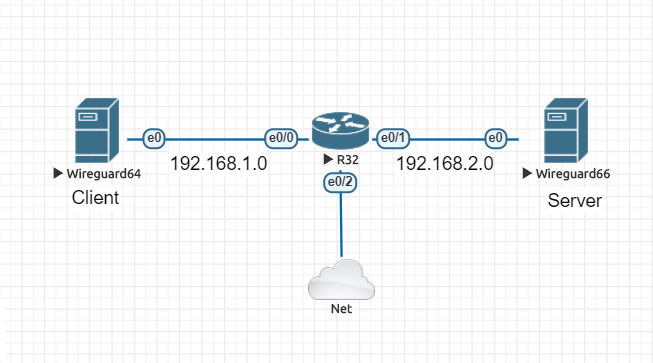
[4. Включение интерфейсов wg0 6](#_Toc30885436)

[5. Перенаправление IP-адресов на сервере 8](#_Toc30885437)

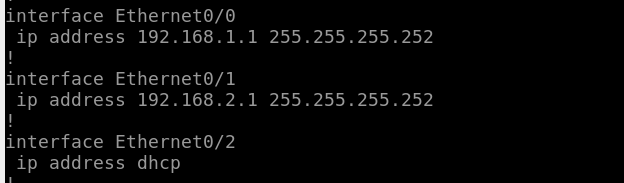
[6. Настройка iptables, необходимая для управления трафиком 8](#_Toc30885438)

# **Справка и первоначальные настройки**

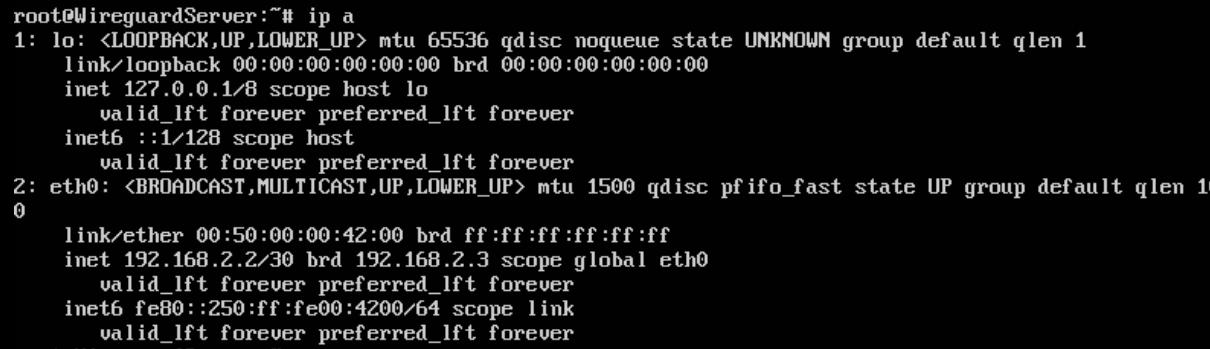
Для настройки VPN-туннеля используем простейшую схему, состоящую из двух устройств с ОС Ubuntu 16.04, которые связаны с роутером и имеют выход в интернет.

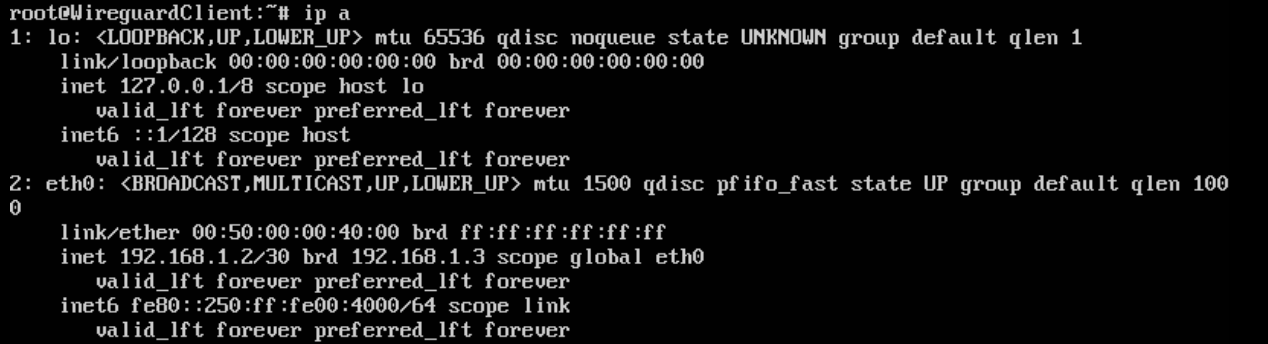


На роутере настроены только IP-адреса на соответствующих интерфейсах:

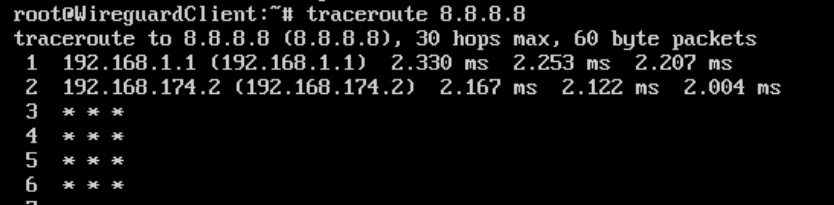


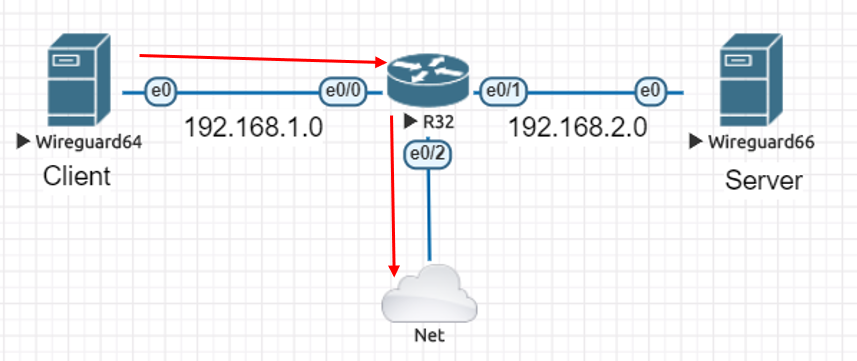
Также на интерфейсах eth0 и на сервере, и на клиенте настроены IP-адреса:





С помощью команды traceroute мы можем проверить, как идет трафик до настройки Wireguard:





Проблема текущих VPN решений в том, что их тяжело правильно настроить, дорого обслуживать, а так же в них полно legacy кода сомнительного качества.

Заявленные преимущества WireGuard над другими VPN решениями:

* Простой в использовании.
* Использует современную криптографию: Noise protocol framework, Curve25519, ChaCha20, Poly1305, BLAKE2, SipHash24, HKDF и т.д.
* Компактный читаемый код, проще исследовать на уязвимости.
* Высокая производительность.
* Не теряет соединение, когда устройство выключается
* Четкая и проработанная спецификация.

# **Принцип работы**

Принципы работы можно описать примерно так:

* Создается WireGuard интерфейс, ему назначается приватный ключ и IP адрес. Загружаются настройки других пиров: их публичные ключи, IP адреса и т.д.
* Все IP пакеты, приходящие на WireGuard интерфейс инкапсулируются в UDP и безопасно добавляются другим пирам.
* Клиенты задают публичный IP адрес сервера в настройках. Сервер автоматически узнает внешние адреса клиентов, когда от них приходят корректно аутентифицированные данные.
* Сервер может менять публичный IP адрес не прерывая работы. При этом он отправит оповещение подключенным клиентам и они обновят свою конфигурацию на лету.
* Используется концепт маршрутизации Cryptokey Routing. WireGuard принимает и отправляет пакеты на основании публичного ключа пира. Когда сервер расшифровывает корректно аутентифицированный пакет, проверяется его src поле. Если оно соответствует с конфигурацией allowed-ips аутентифицированного пира, то пакет принимается интерфейсом WireGuard. При отправке исходящего пакета происходит соответственная процедура: берется dst поле пакета и на основании его выбирается соответсвующий пир, пакет подписывается своим ключом, шифруется ключом пира и отправляется на remote endpoint.

# **Настройка**

Все действия совершаются в режиме root. Каждая новая команда начинается с символа #, его вводить не нужно.

## 1. Установка необходимых пакетов

На обоих устройствах производим обновление пакета и производим установку следующих пакетов:

# apt-get update

# add-apt-repository ppa:wireguard/wireguard

# apt-get install wireguard-dkms wireguard-tools

## 2. Генерация ключей

На каждом из устройств генерируем приватный и публичный ключи. Приватный выводится сразу же в файл wg0.conf, а публичный сохраняется в pubkey.

# (umask 077 && printf "[Interface]\nPrivateKey = " | sudo tee /etc/wireguard/wg0.conf > /dev/null)

# wg genkey | sudo tee -a /etc/wireguard/wg0.conf | wg pubkey | sudo tee /etc/wireguard/publickey

## 3.1 Настройка конфигурации сервера

С помощью команды

# nano /etc/wireguard/wg0.conf

открываем созданный раннее файл с уже сохраненным туда приватным ключом соединения. Добавляем туда следующие настройки:

[Interface]

PrivateKey = <Уже вписанный приватный ключ>

Address = 10.200.200.1/24

SaveConfig = true

ListenPort = 51820

[Peer]

PublicKey = <Публичный ключ клиента>

AllowedIPs = 10.200.200.2/32

Выделенные красным параметры необходимо заменить на свои.

*Address* – адрес будущего VPN-сервера. Не совпадает с адресом, настроенным на интерфейсе eth0.

*SaveConfig* имеет значение true, чтобы сервис wg-quick мог автоматически сохранять свою активную конфигурацию в этом файле при завершении работы.

*ListenPort* - UDP порт, который будет прослушиваться. Его можно заменить на любой другой.

*AllowedIPs* говорит о том, что к данному туннелю может подключиться только хост 10.200.200.2, что дополнительно повышает безопасность соединения.

Публичный ключ клиента необходимо вписать вручную. Он был сгенерирован нами ранее на втором устройстве. Просмотреть его можно с помощью:

# cat /etc/wireguard/publickey

## 3.2 Настройка конфигурации клиента

Аналогично пункту 3.1 открываем файл (был сгенерирован на шаге 2):

# nano /etc/wireguard/wg0.conf

Приватный ключ в этом файле вновь уже имеется. Остается вписать недостающие настройки:

[Interface]

PrivateKey = <Сгенерированный приватный ключ>

Address = 10.200.200.2/32

DNS = 10.200.200.1

[Peer]

PublicKey = <Публичный ключ сервера>

Endpoint = <IP-адрес, который настроен на eth0>:51820

AllowedIPs = 0.0.0.0/0

PersistentKeepalive = 21

*AllowedIPs* = 0.0.0.0/0 будет разрешать направлять весь трафик на клиенте через VPN-туннель.

*DNS* = 10.200.200.1 установит IP-адрес DNS-преобразователя на наш VPN-сервер.

## 4. Включение интерфейсов wg0

На сервере:

# chown -v root:root /etc/wireguard/wg0.conf

# chmod -v 600 /etc/wireguard/wg0.conf

# wg-quick up wg0

# systemctl enable wg-quick@wg0.service

На клиенте действуем аналогично:

# wg-quick up wg0

# systemctl enable wg-quick@wg0.service

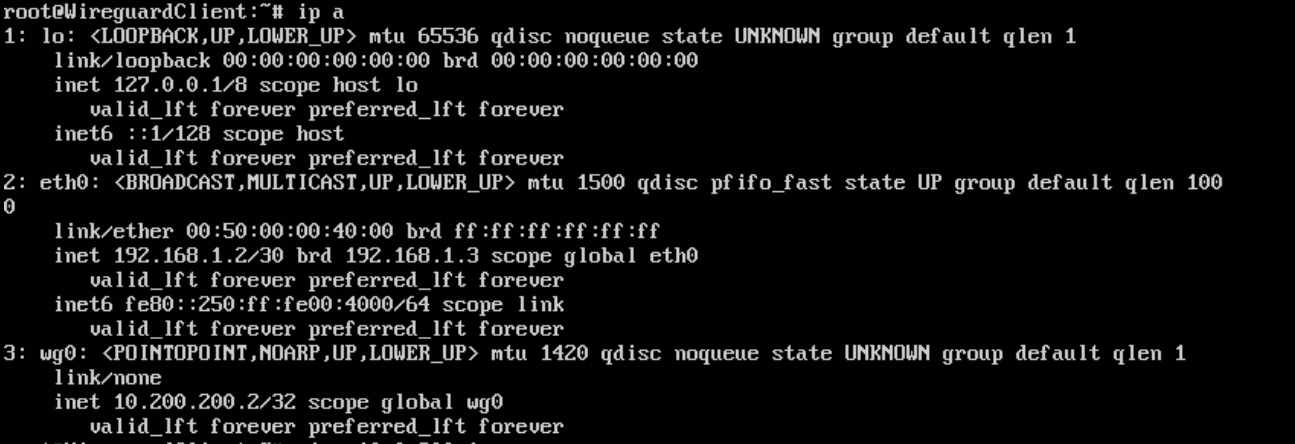
Для разрыва VPN-соединения можно использовать следующую команду:

# wg-quick down wg0

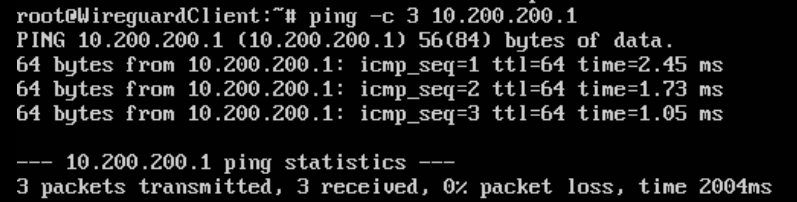
При помощи команды

# ip a

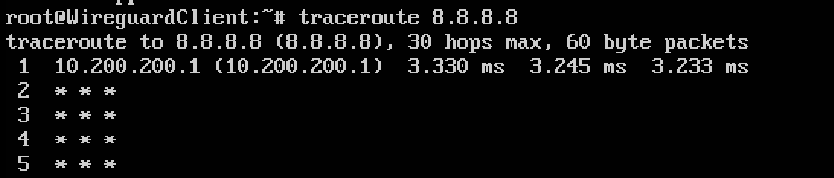
на обоих устройствах можно увидеть появившийся интерфейс wg0, имеющий установленный ранее нами в файле /etc/wireguard/wg0.conf IP-адрес.



На данный момент мы уже можем пропинговать по внутреннему IP-адресу сервер с клиента, все пакеты будут доставлены успешно, если всё сделано верно.



Однако, если попробовать сделать traceroute, трафик доходит до сервера, но далее никуда не идет.



Эта проблема решается с помощью настройки брандмауэра и перенаправления IP-адресов на сервере.

## 5. Перенаправление IP-адресов на сервере

Отредактируйте следующий файл:

# nano /etc/sysctl.conf

В нем необходимо раскомментировать следующую строку:

net.ipv4.ip\_forward=1

Теперь выполните следующие действия, чтобы избежать необходимость перезагрузки сервера:

# sysctl -p

# echo 1 > /proc/sys/net/ipv4/ip\_forward

## 6. Настройка iptables, необходимая для управления трафиком

Отслеживание VPN-подключения:

# iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

# iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

Разрешение входящего трафика VPN на прослушивающем порту:

# iptables -A INPUT -p udp -m udp --dport 51820 -m conntrack --ctstate NEW -j ACCEPT

Разрешаем TCP и UDP трафик:

# iptables -A INPUT -s 10.200.200.0/24 -p tcp -m tcp --dport 53 -m conntrack --ctstate NEW -j ACCEPT

# iptables -A INPUT -s 10.200.200.0/24 -p udp -m udp --dport 53 -m conntrack --ctstate NEW -j ACCEPT

Разрешить пересылку пакетов, которые остаются в VPN-туннеле:

# iptables -A FORWARD -i wg0 -o wg0 -m conntrack --ctstate NEW -j ACCEPT

Настройка NAT:

# iptables -t nat -A POSTROUTING -s 10.200.200.0/24 -o eth0 -j MASQUERADE

Для того, чтобы во время перезагрузки правила остались неизменными, воспользуемся утилитой netfilter-persistent:

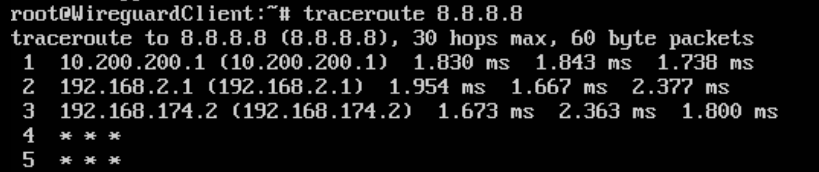
# apt-get install iptables-persistent

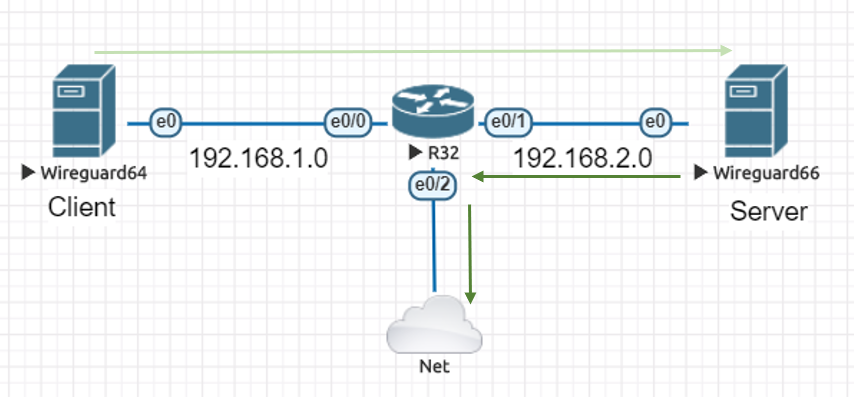
# systemctl enable netfilter-persistent

# netfilter-persistent save

# netfilter-persistent reload

Теперь, вновь выполнив traceroute, мы видим, что наш маршрут изменился:





Абсолютно аналогично работает Wireguard и на более сложной схеме, при этом с той же самой настройкой. Данные сначала передаются по туннелю на сервер, а уже потом уходят в сеть.

