Wireguard

Virtual Private Network (VPN)

Miguel Frade & Francisco Santos

POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Introduction

Wireguard

- written by Jason A. Donenfeld (`https://www.wireguard.com/`)
- a VPN that aims to be faster, simpler, and leaner
  - the number of code lines is only about 1% of either OpenVPN or IPsec
  - praised by the Linux kernel creator, Linus Torvalds, as a “work of art”

Wireguard

- written by Jason A. Donenfeld (`https://www.wireguard.com/`)
- a VPN that aims to be faster, simpler, and leaner
  - the number of code lines is only about 1% of either OpenVPN or IPsec
  - praised by the Linux kernel creator, Linus Torvalds, as a “work of art”
- it is cross-platform (Windows, macOS, BSD, iOS, Android)
- only works over UDP
- has full support for IPv6

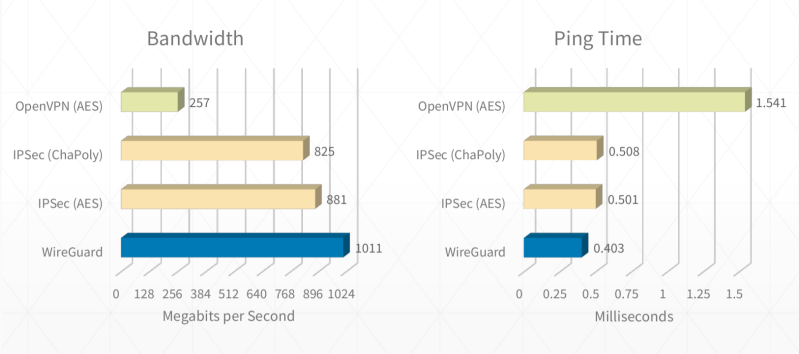
Wireguard

- written by Jason A. Donenfeld (`https://www.wireguard.com/`)
- a VPN that aims to be faster, simpler, and leaner
  - the number of code lines is only about 1% of either OpenVPN or IPsec
  - praised by the Linux kernel creator, Linus Torvalds, as a “work of art”
- it is cross-platform (Windows, macOS, BSD, iOS, Android)
- only works over UDP
- has full support for IPv6
- supports multiple topologies:
  - point-to-point
  - star (server/client)
  - mesh

Wireguard utilizes state-of-the-art cryptography

- has been reviewed by cryptographers
- ChaCha20 for symmetric encryption, authenticated with Poly1305, using RFC7539's AEAD construction
- Curve25519 for ECDH (key exchange)
- BLAKE2s for hashing and keyed hashing (HMAC), described in RFC7693
- HMAC-based Key Derivation Function (HKDF), described in RFC5869

Wireguard performance comparision



Source: Donenfeld, J. A. (2017, November). WireGuard: Next Generation Kernel Network Tunnel. In NDSS.

Setup test scenario

Recommended setup to test Wireguard with virtual machines:

1. **server** – setup a virtual machine with Ubuntu 20.04 server edition
   - with 2 network interfaces, one "NAT" and the other as "Host Only";
   - configure the "Host Only" interface with a static IP address by editing `/etc/netplan/00-installer-config.yaml` and add:

```
network:
  ethernets:
    enp0s8:
      dhcp4: no
      # the IP address must be belong to the same network configured on your
      # host network manager
      addresses: [192.168.56.90/24]
```

   and apply the changes: `sudo netplan apply`

Recommended setup to test Wireguard with virtual machines:

1. **server** – setup a virtual machine with Ubuntu 20.04 server edition
   - with 2 network interfaces, one "NAT" and the other as "Host Only";
   - configure the "Host Only" interface with a static IP address by editing `/etc/netplan/00-installer-config.yaml` and add:

```
network:
  ethernets:
    enp0s8:
      dhcp4: no
      # the IP address must be belong to the same network configured on your
      # host network manager
      addresses: [192.168.56.90/24]
```

   and apply the changes: `sudo netplan apply`

2. **client** – setup a virtual machine with (k)ubuntu 20.04 desktop edition
   - with 2 network interfaces, one "NAT" and the other as "Host Only";
   - later the "NAT" interface must be shutdown:
     - with only the "Host Only" interface this setup will prevent the virtual to have access to the Internet, but should be able to access it through the server after the VPN is properly setup;
     - shutdown "NAT" interface: `sudo ifdown enp3s0  # replace enp3s0 with the name of your NAT interface`
     - activate "NAT" interface: `sudo ifup enp3s0`

SETUP TEST SCENARIO

Alternative setup to test Wireguard without virtual machines:

1. 2 classroom computers running Ubuntu 20.04
   - one working as a server and the other working as a client
2. configure Wireguard on both server and client computers accordingly to these slides;

Note

These slides were written taking into account the virtual machine scenario. If you are using the classroom computers, you must adapt some of the commands.

Installation

INSTALLATION

Windows	https://download.wireguard.com/windows-client/wireguard-installer.exe
macOS	https://itunes.apple.com/us/app/wireguard/id1451685025?ls=1&mt=12
iOS	https://itunes.apple.com/us/app/wireguard/id1441195209?ls=1&mt=8
Android	https://play.google.com/store/apps/details?id=com.wireguard.android
Ubuntu	

Terminal

```
user@linux:~$ sudo apt install wireguard
```

Configuration

On the server – Enable packet forwarding

- read the current state of IP forwarding

```
Terminal
user@server:~$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

- enable forwarding
  - temporarily (lost after a reboot)

```
Terminal
user@server:~$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

  - persistently (remains after a reboot), edit `/etc/sysctl.conf`

```
# change value to "1":
net.ipv4.ip_forward = 1

# if IPv6 is required change also:
net.ipv6.conf.all.forwarding = 1
```

Generate a key pair for each device

- on the server

```
Terminal
user@server:~$ sudo -i
[sudo] password for user:
root@server:~# cd /etc/wireguard
root@server:~/etc/wireguard# wg genkey | tee server-privatekey | wg pubkey > server-publickey
```

- on the client

```
Terminal
user@client:~$ sudo -i
[sudo] password for user:
root@client:~# cd /etc/wireguard
root@client:~/etc/wireguard# wg genkey | tee client-privatekey | wg pubkey > client-publickey
```

On the server – Create the configuration file

- create the file `/etc/wireguard/wg0.conf`
- and add the following for IPv4
  - replace `enp0s3` by the name of the interface with Internet connection

```
[Interface]
Address = 10.8.0.1/24
SaveConfig = true
PostUp = iptables -A FORWARD -i %i -j ACCEPT
PostUp = iptables -A FORWARD -o %i -j ACCEPT
PostUp = iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT
PostDown = iptables -D FORWARD -o %i -j ACCEPT
PostDown = iptables -t nat -D POSTROUTING -o enp0s3 -j MASQUERADE
ListenPort = 51820
PrivateKey = <server private key>
```

On the server – Example of the file `/etc/wireguard/wg0.conf` with support for both IPv4 and IPv6

- replace `eth0` by the name of the interface with Internet connection

```
[Interface]
Address = 10.8.0.1/24
Address = fd86:ea04:1115::1/64
SaveConfig = true
PostUp = iptables -A FORWARD -i %i -j ACCEPT
PostUp = iptables -A FORWARD -o %i -j ACCEPT
PostUp = iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostUp = ip6tables -A FORWARD -i %i -j ACCEPT
PostUp = ip6tables -A FORWARD -o %i -j ACCEPT
PostUp = ip6tables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT
PostDown = iptables -D FORWARD -o %i -j ACCEPT
PostDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
PostDown = ip6tables -D FORWARD -i %i -j ACCEPT
PostDown = ip6tables -D FORWARD -o %i -j ACCEPT
PostDown = ip6tables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
ListenPort = 51820
PrivateKey = <server private key>
```

On the server – Enable wireguard

- enable `wg0` interface

```
Terminal
user@server:~$ sudo wg-quick up wg0
[ ] ip link add wg0 type wireguard
[ ] wg setconf wg0 /dev/fd/63
[ ] ip -4 address add 10.8.0.1/24 dev wg0
[ ] ip link set mtu 1420 up dev wg0
[ ] iptables -A FORWARD -i wg0 -j ACCEPT
[ ] iptables -A FORWARD -o wg0 -j ACCEPT
[ ] iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

- enable the wireguard service to start automatically after each boot

```
Terminal
user@server:~$ sudo systemctl enable wg-quick@wg0
Created symlink /etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service →
↪  /lib/systemd/system/wg-quick@.service.
```

On the server – Check if wireguard is running

```
Terminal
user@server:~$ sudo wg
interface: wg0
  public key: <server public key>
  private key: (hidden)
  listening port: 51820
user@server:~$ ip address show wg0
4: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default
↪  qlen 1000
    link/none
    inet 10.8.0.1/24 scope global wg0
       valid_lft forever preferred_lft forever
```

On the client – Create the configuration file

- create the file `/etc/wireguard/wg0.conf`
- and add the following for IPv4 and IPv6 support
  - replace `192.168.56.90` with the IP address of the host only interface of the server

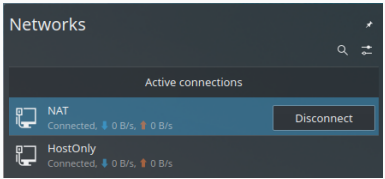
```
[Interface]
PrivateKey = <client private key>
Address = 10.8.0.2/32,fd86:ea04:1115::2/128

[Peer]
PublicKey = <server public key>
Endpoint = 192.168.56.90:51820
AllowedIPs = 0.0.0.0/0,::/0     # to allow all traffic through the VPN
PersistentKeepalive = 25        # to prevent the tunnel from dying
```

On the server – Add the client

```
Terminal
user@server:~$ sudo wg set wg0 peer <client public key>
```

On the client – Turn off the NAT interface through the Network-Manager GUI



Testing

On the client – Enable `wg0` wireguard interface

Terminal

```
user@client:~$ sudo wg-quick up wg0
[ ] ip link add wg0 type wireguard
[ ] wg setconf wg0 /dev/fd/63
[ ] ip -4 address add 10.8.0.2/32 dev wg0
[ ] ip link set mtu 1420 up dev wg0
[ ] wg set wg0 fwmark 51820
[ ] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[ ] ip -4 rule add not fwmark 51820 table 51820
[ ] ip -4 rule add table main suppress_prefixlength 0
[ ] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[ ] iptables-restore -n
```

On the client – Check wireguard configuration (1)

Terminal

```
user@client:~$ sudo wg
interface: wg0
  public key: <client public key>
  private key: (hidden)
  listening port: 49037
  fwmark: 0xca6c

peer: <server public key>
  endpoint: 192.168.56.90:51820
  allowed ips: 0.0.0.0/0, ::/0
  latest handshake: 8 seconds ago
  transfer: 15.95 KiB received, 13.81 KiB sent
  persistent keepalive: every 25 seconds
```

On the client – Check wireguard configuration (2)

Terminal

```
user@client:~$ ip address show wg0
ip address show wg0
8: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default
↪  qlen 1000
    link/none
    inet 10.8.0.2/32 scope global wg0
       valid_lft forever preferred_lft forever
    inet6 fd86:ea04:1115::2/128 scope global
       valid_lft forever preferred_lft forever
```

On the client – Test connection

- to the server

```
Terminal
user@client:~$ ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.96 ms
```

- to the Internet, through the VPN

```
Terminal
user@client:~$ wget https://www.publico.pt
--2020-11-29 16:04:32--  https://www.publico.pt/
Resolving www.publico.pt (www.publico.pt)... 172.67.37.74, 104.22.79.206, 104.22.78.206,
Connecting to www.publico.pt (www.publico.pt)|172.67.37.74|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘index.html.1’

index.html        [               <=>                     ] 852,59K  3,71MB/s    in 0,2s

2020-11-29 16:04:33 (3,71 MB/s) - ‘index.html’ saved [873049]
```

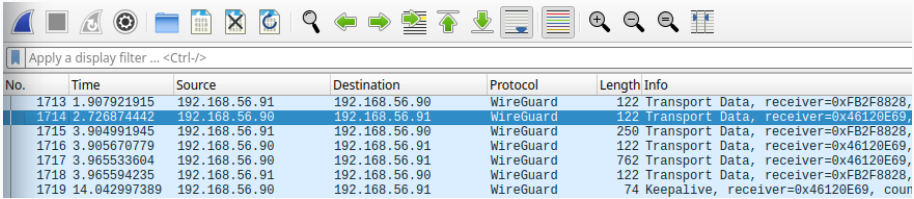
On the client – Capture packets

- install wireshark

```
Terminal
user@client:~$ sudo apt install wireshark
```

- capture the network traffic on the `enp0s8` interface

```
Terminal
user@client:~$ sudo wireshark -i enp0s8 -k
```

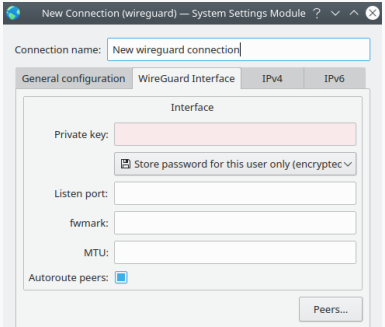


No.	Time	Source	Destination	Protocol	Length	Info
1713	1.907921915	192.168.56.91	192.168.56.90	WireGuard	122	Transport Data, receiver=0xFB2F8828,
1714	2.726874442	192.168.56.90	192.168.56.91	WireGuard	122	Transport Data, receiver=0x46120E69,
1715	3.904991945	192.168.56.91	192.168.56.90	WireGuard	250	Transport Data, receiver=0xFB2F8828,
1716	3.905670779	192.168.56.90	192.168.56.91	WireGuard	122	Transport Data, receiver=0x46120E69,
1717	3.965533604	192.168.56.90	192.168.56.91	WireGuard	762	Transport Data, receiver=0x46120E69,
1718	3.965594235	192.168.56.91	192.168.56.90	WireGuard	122	Transport Data, receiver=0xFB2F8828,
1719	14.042997389	192.168.56.90	192.168.56.91	WireGuard	74	Keepalive, receiver=0x46120E69, coun

Exercise

EXERCISES

1. setup a Wireguard VPN between 2 computers
2. test the connection
   - with the `ping` command
   - with the browser
   - capture the packets from the host only network interface
3. on the client use the Network-Manager GUI
   - to configure the VPN
   - the IPv4 must set to “manual”



For more information:

- Wireguard Quick Start (official webpage)
- WireGuard VPN Road Warrior Setup Tutorial

```
Terminal
user@linux:~$ man wg
user@linux:~$ man wg-quick
```