Nastavení WireGuard VPN Road Warrior

emanuelduss.ch/posts/wireguard-vpn-road-warrior-setup

September 29, 2018

Zavedení

WireGuard je relativně nový open-source software pro vytváření VPN tunelů na IP vrstvě pomocí nejmodernější kryptografie. Zúčastnil jsem se samoorganizovaného setkání tvůrce a vývojáře Jasona Donenfelda na 34c3, který vysvětlil, jak WireGuard funguje a jak jej lze používat. Docela mě zaujala jeho jednoduchost a zkusil jsem to. Fungovalo to víceméně po vybalení. Nyní jsem vytvořil pokročilejší nastavení pro přístup k mé domácí síti.

V tomto příspěvku na blogu popíšu, jak jej můžete využít ke vzdálenému přístupu k domácí nebo firemní síti z jakékoli externí sítě jako takzvaný road warrior.

Software WireGuard VPN

Zde je dobrá přednáška od vývojáře WireGuard Jasona Donenfelda, který vysvětluje, co WireGuard umí a jak funguje: https://www.youtube.com/watch?v=eYztYCbV_8U&t=16s

Některé klíčové vlastnosti z této přednášky:

- WireGuard je řešení VPN (alternativa/náhrada např. OpenVPN nebo IPsec).
- Nastavení a konfigurace WireGuard je velmi jednoduché.
- Implementuje tunelovací protokol vrstvy 3 pro IPv4 a IPv6.
- Je napsán v ~ 4k jednotlivých řádcích kódů.
- Zapouzdřené IP pakety jsou uvnitř UDP paketů.
- Využívá nejmodernější kryptografii (podporovány jsou pouze silné algoritmy jako Curve25519, ChaCha20, Poly1305 nebo BLAKE2 a nelze konfigurovat žádné další šifry).
- WireGuard běží v linuxovém jádře (ale existují i implementace v uživatelském prostoru).

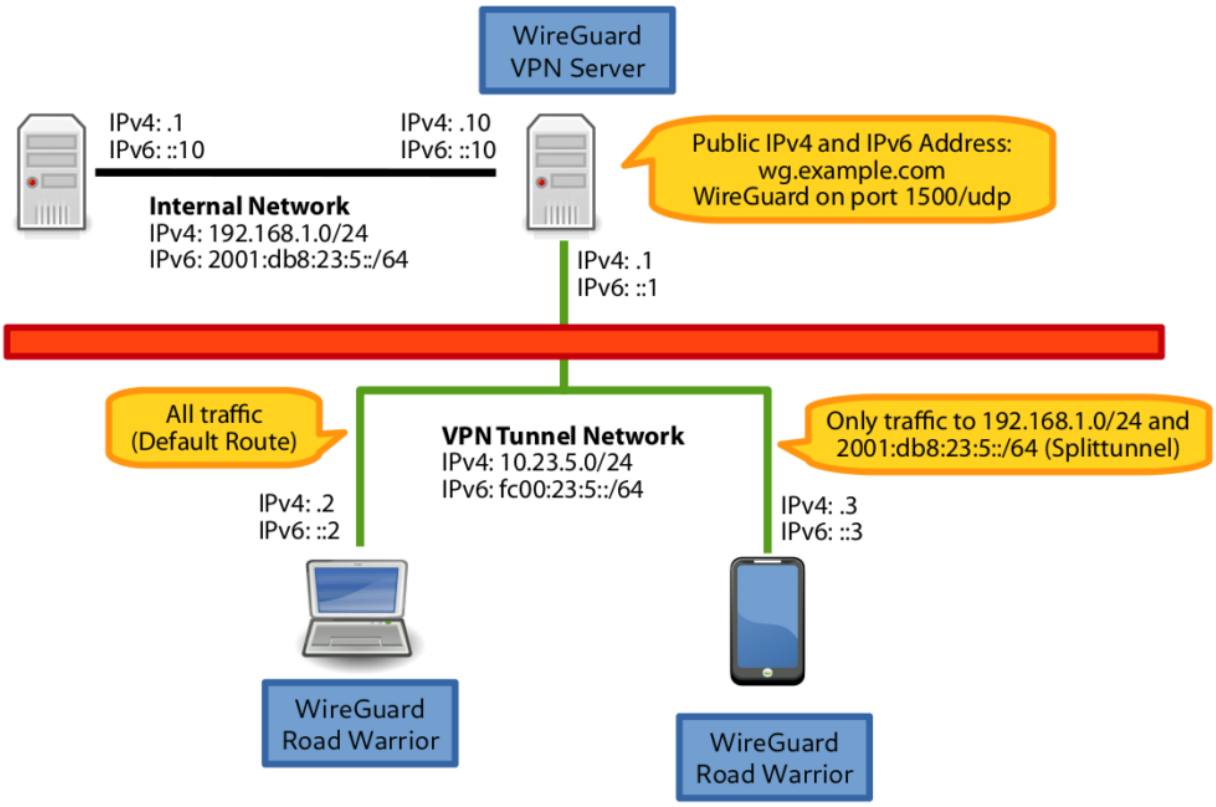
- Lze jej spravovat pomocí běžných síťových nástrojů Linuxu, jako je ip, iptables, …
- Autentizace se provádí pomocí soukromých/veřejných klíčů, podobně jako u SSH klíčů.
- Klienti mohou provádět roaming, jako v mosh (https://mosh.org/) .
- WireGuard nereaguje na neověřené balíčky, takže není možné zjistit, zda na serveru běží WireGuard, pokud odesílatel není autorizován.
- Poskytuje dokonalé dopředné utajení.
- Nezveřejňuje žádnou identitu, protože veřejné klíče nejsou nikdy přenášeny v čistém textu přes internet.
- Výměna klíčů (ECDH) trvá pouze 1 zpáteční cestu.
- WireGuard je rychlý, protože běží v prostoru jádra a protože použité kryptografické algoritmy jsou také velmi rychlé.

Více informací, whitepaper, pokyny k nastavení nebo ukázky lze nalézt na webových stránkách projektu: https://www.wireguard.com/.

V určitém okamžiku bude WireGuard integrován přímo do linuxového jádra. Linus Torvalds řekl „je to umělecké dílo“ a doufá, že bude brzy začleněno do jádra: https://lists.openwall.net/netdev/2018/08/02/124 .

Scénář Road Warrior

Silniční bojovník je osoba, která používá mobilního klienta (např. notebook nebo mobilní telefon) pro připojení ke své firemní nebo domácí síti.



Vlastnosti tohoto nastavení:

- K interní infrastruktuře IPv4 a IPv6 lze přistupovat odkudkoli prostřednictvím IPv4 a IPv6.
- Klienta WireGuard VPN lze nainstalovat a používat na Linuxu a mobilních telefonech, jako je Android.
- Přes VPN lze směrovat buď veškerý provoz (výchozí trasa), nebo pouze provoz požadovaný pro vnitřní síť (split tunneling). To lze nakonfigurovat na klientovi.
- Pokud road warrior nemá připojení IPv6, lze jej zajistit prostřednictvím tunelu VPN.
- VPN server může být také za NAT routerem, protože WireGuard funguje přes UDP.

Poznámka: Pokud roadwarrior naváže VPN spojení s mobilním telefonem a používá mobilní telefon jako WiFi hotspot pro jiné zařízení (např. notebook), provoz z WiFi hotspotu není směrován přes VPN. Nejsem si jistý, proč tomu tak je, ale možná je to omezení operačního systému na mobilním telefonu.

Instalace softwaru WireGuard

Nainstalujte WireGuard podle pokynů k instalaci (https://www.wireguard.com/install/).

Debian

Přidání repozitáře WireGuard a instalace `wireguard`balíčku:

```
echo "deb http://deb.debian.org/debian/ unstable main" | sudo
tee /etc/apt/sources.list.d/unstable-wireguard.list
printf 'Package: *\nPin: release a=unstable\nPin-Priority:
150\n' | sudo tee /etc/apt/preferences.d/limit-unstable
sudo apt update
sudo apt install wireguard
```

Raspberry Pi

Na Raspberry Pi jej musíte zkompilovat ručně podle tohoto návodu k instalaci: https://github.com/adrianmihalko/raspberrypiwireguard . Nemusíte však instalovat hlavičky jádra přes, `rpi-soruce`jak bylo zmíněno. Pro instalaci na Raspberry Pi stačí následující příkazy:

```
sudo apt-get install libmnl-dev build-essential git
git clone https://git.zx2c4.com/WireGuard
cd WireGuard/
cd src/
make
sudo make install
```

Arch Linux

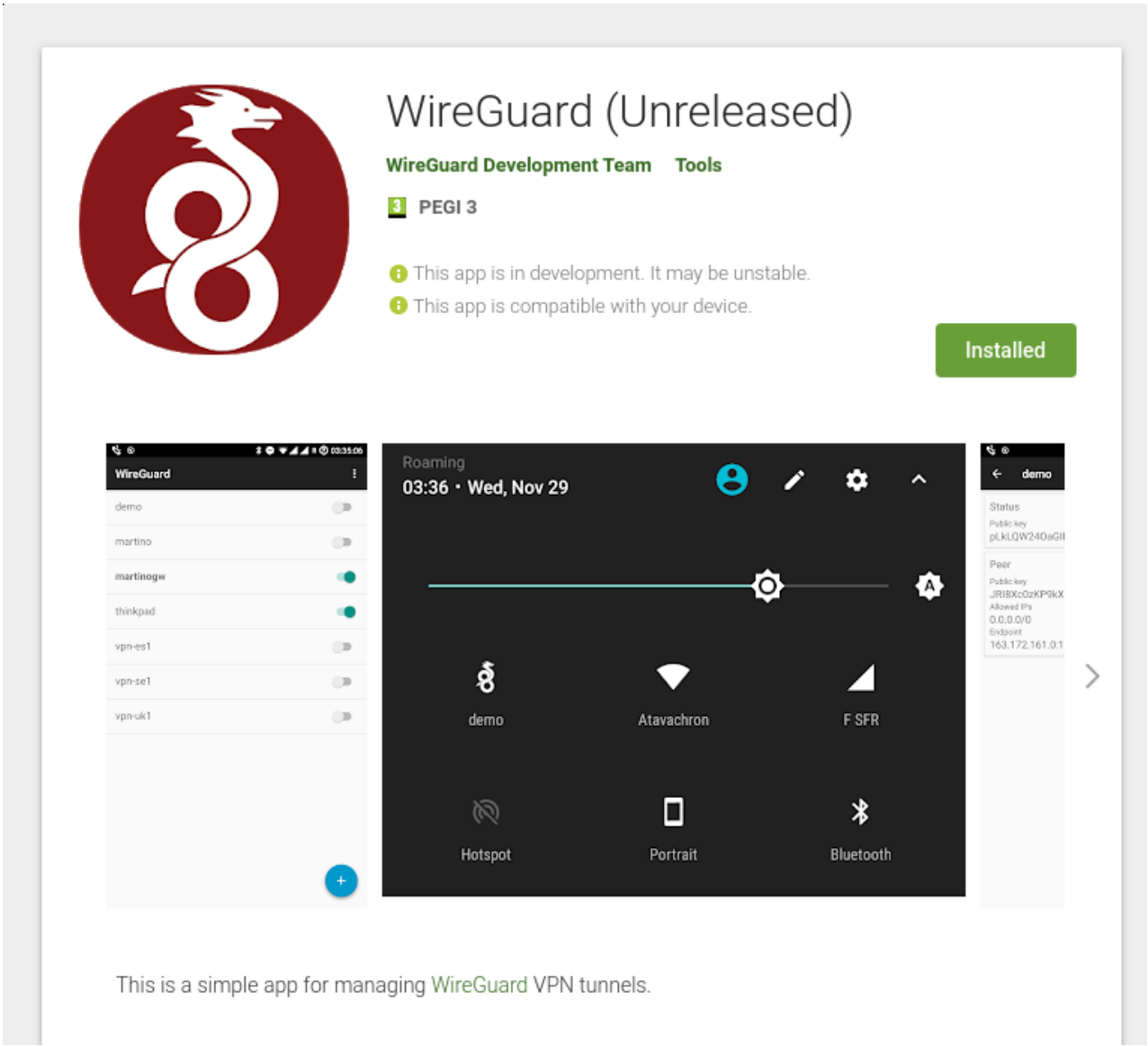
Instalace dvou balíčků wireguard z oficiálních repozitářů a balíčku linux-headers (je to nutné, protože modul Wireguard je nainstalován jako modul DKMS):

```
sudo pacman -S wireguard-dkms wireguard-tools linux-headers
```

Android

Nainstalujte si aplikaci WireGuard z obchodu Play: https://play.google.com/store/apps/details?id=com.wireguard.android&hl=cs . Tato aplikace implementuje

WireGuard v uživatelském prostoru. Pro použití WireGuard tedy telefon nemusí být rootovaný.



Test instalace

Můžete otestovat, zda `wireguard`je načten modul jádra:

```
$ lsmod | grep -i wireguard
wireguard              147456  0
ip6_udp_tunnel          16384  1 wireguard
udp_tunnel              16384  1 wireguard
ipv6                   434176  33
nf_conntrack_ipv6,nf_nat_masquerade_ipv6,nf_defrag_ipv6,wireguard,nf_nat_ipv6
```

Generování klíčů serveru

Aby bylo zajištěno, že všechny soubory mají správná oprávnění (čitelná a zapisovatelná pouze vlastníkem souboru, kterým je v tomto případě uživatel `root`), `umask`musí být nastavena na `077`:

```
umask 077
```

Konfigurace se provádí v `/etc/wireguard`adresáři. Vygenerujte soukromý a veřejný klíč pro server:

```
cd /etc/wireguard
wg genkey | tee server-private.key | wg pubkey > server-
public.key
ls -l server-private.key server-public.key
```

Příklad klíče serveru:

```
# cat server-private.key server-public.key
mNt0Gx2Af/OCkT9FchX3nybsaXUAerglnuMnSud4z1k=
3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
```

Generování klientského klíče

Vygenerujte soukromý a veřejný klíč pro každého klienta. Poznámka: Tyto klíče lze také kompletně vygenerovat na klientovi.

notebook:

```
cd /etc/wireguard
wg genkey | tee notebook-private.key | wg pubkey > notebook-
public.key
ls -l notebook-*
```

Příklad klíče:

```
# cat notebook-private.key notebook-public.key
OPTN5qb4FFulEBFLlrmC1sFTawT6AmdhsAbigpCMemw=
UGyBshzPfAH0U4QAgGJHe07LfUz4RcHA9PhUlUC4cCA=
```

To samé pro mobilní telefon:

```
wg genkey | tee mobile-private.key | wg pubkey > mobile-
public.key
ls -l mobile*
```

Konfigurace serveru

WireGuard

Vytvořte nový konfigurační soubor pro server v `/etc/wireguard/wg0.conf`. Název souboru určuje název síťového rozhraní VPN. V tomto případě bude nové síťové rozhraní pojmenováno `wg0`.

```
# cd /etc/wireguard
# ls -l wg0.conf
-rw------- 1 root root 713 Sep 23 17:33 wg0.conf
# cat wg0.conf
[Interface]
Address = 10.23.5.1/24, fc00:23:5::1/64
ListenPort = 1500
PrivateKey = mNt0Gx2Af/OCkT9FchX3nybsaXUAerglnuMnSud4z1k=
PreUp = iptables -t nat -A POSTROUTING -s 10.23.5.0/24  -o
enxb827eb7dc89a -j MASQUERADE; ip6tables -t nat -A POSTROUTING
-s fc00:23:5::/64 -o enxb827eb7dc89a -j MASQUERADE
PostDown = iptables -t nat -D POSTROUTING -s 10.23.5.0/24  -o
enxb827eb7dc89a -j MASQUERADE; ip6tables -t nat -D POSTROUTING
-s fc00:23:5::/64 -o enxb827eb7dc89a -j MASQUERADE

# Notebook
[Peer]
PublicKey = UGyBshzPfAH0U4QAgGJHe07LfUz4RcHA9PhUlUC4cCA=
AllowedIPs = 10.23.5.2/32, fc00:23:5::2/128

# Mobile
[Peer]
PublicKey = 1xy8XRtUQT/9AwYWlEXsWCezNjfiFjXaBy40UUtAWBo=
AllowedIPs = 10.23.5.3/32, fc00:23:5::3/128
```

Část vysvětlení konfigurace `Interface`:

- `Address`: Server získá přidělenou adresu IPv4 `10.23.5.1/24`a adresu IPv6 `fc00:23:5::1/64`.
- `ListenPort`: Server bude na portu naslouchat paketům UDP`1500.`
- `PrivateKey`: Toto je soukromý klíč serveru vygenerovaný dříve.

- `PreUp`: Pravidlo `iptables`/ `ip6tables`v `nat`řetězci je přidáno před vytvořením rozhraní pro provádění NAT mezi klienty VPN a cílem. Rozhraní (`-o enxb827eb7dc89a`) je třeba upravit podle názvu odchozího rozhraní na VPN serveru.
- `PreDown`: Když je server VPN zastaven, pravidla NAT budou odstraněna.

Část vysvětlení konfigurace `Peer`:

- `PublicKey`: Veřejný klíč klientů (vygenerovaný dříve)
- `AllowedIPs`: Toto je nejsložitější možnost konfigurace, která mě ze začátku trochu mátla. Tato možnost vždy zahrnuje pouze IP adresy nebo sítě, které jsou dostupné na vzdáleném místě. Nejde o IP adresu/síť mimo tunel (takže žádná konfigurace, ze které veřejné IP adresy se klient smí připojit), ale pouze o adresy/sítě, které jsou přenášeny uvnitř tunelu! Ve scénáři road warrior, kde klient neposkytuje serveru celou síť, je síťová maska vždy `/32` na IPv4 nebo `/128`IPv6. Pakety na serveru VPN s touto cílovou IP adresou jsou odesílány tomuto zadanému partnerovi. Tento peer může také odesílat balíčky pouze z této zdrojové IP adresy na server VPN. Je také důležité vědět, že `AllowedIPs` ve stejném konfiguračním souboru nejsou žádní kolegové se stejnými adresami/sítěmi. Pokud by tomu tak bylo, server by nevěděl, kterému peeru má server posílat balíčky odpovídající více peerům se stejnou konfigurovanou sítí.

Přesměrování IP

Přesměrování IP musí být povoleno na IPv4 i IPv6. Vytvoření konfiguračního souboru `/etc/sysctl.d/wireguard.conf`:

```
# ls -l /etc/sysctl.d/wireguard.conf
-rw------- 1 root root 53 Sep 25 22:23
/etc/sysctl.d/wireguard.conf
# cat /etc/sysctl.d/wireguard.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

Načítání konfigurace:

```
# sysctl -p /etc/sysctl.d/wireguard.conf
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

Konfigurace klienta

Směrování veškerého provozu (výchozí trasa)

Konfigurační soubor, který bude směrovat veškerý provoz přes VPN:

```
# cd /etc/wireguard
# ls -l client-notebook_dgw.conf
-rw------- 1 root root 245 Sep 23 20:34 client-
notebook_dgw.conf
# cat client-notebook_dgw.conf
[Interface]
PrivateKey = OPTN5qb4FFulEBFLlrmC1sFTawT6AmdhsAbigpCMemw=
Address = 10.23.5.2/24, fc00:23:5::2/64
DNS = 8.8.8.8

[Peer]
PublicKey = 3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
Endpoint = wg.example.com:1500
AllowedIPs = 0.0.0.0/0, ::/0
```

Část vysvětlení konfigurace `Interface`:

- `Address`: Klient získá přidělenou adresu IPv4 `10.23.5.2/24`a adresu IPv6 `fc00:23:5::2/64`.
- `PrivateKey`: Toto je soukromý klíč klienta.
- `DNS`: Tento server DNS se používá na klientovi. Může to být server DNS ve vzdálené síti.

Část vysvětlení konfigurace `Peer`:

- `PublicKey`: Toto je veřejný klíč serveru VPN.
- `Endpoint`: Toto je název hostitele serveru VPN.

- `AllowedIPs`: Veškerý provoz odpovídající těmto sítím je odesílán přes tunel VPN. V tomto případě `0.0.0.0/0`to `::/0`znamená, že veškerý provoz IPv4 a veškerý provoz IPv6 je směrován přes VPN.

Dělené tunelování

Konfigurační soubor, který bude přes VPN směrovat pouze provoz pro VPN (`10.23.5.0/24`a `fc00:23:5::/64`) a pro vzdálenou síť ( ).`192.168.1.0/24`

```
# cd /etc/wireguard
# ls -l client-mobile_splittunnel.conf
-rw------- 1 root root 355 Sep 23 20:39 client-
mobile_splittunnel.conf
# cat client-mobile_splittunnel.conf
[Interface]
PrivateKey = kKcKPZoC4gULXOmpDi54sAy5tQvnFEn6J8yXC8OxukY=
Address = 10.23.5.3/24, fc00:23:5::3/64
DNS = 8.8.8.8

[Peer]
PublicKey = 3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
Endpoint = wg.example.com:1500
AllowedIPs = 10.23.5.0/24, fc00:23:5::/64, 192.168.1.0/24,
2001:db8:23:5::/64
```

Jediný rozdíl je v `AllowedIPs`direktivě, která vytváří nastavení rozděleného tunelování VPN. Přes VPN je směrován pouze provoz pro poskytované sítě.

Aspekty při používání NAT nebo stavových firewallů

Pokud je server za NAT nebo stavovým firewallem a klient na server po určitou dobu neodesílá žádný provoz, NAT router/firewall odstraní stav hostitele z tabulky připojení. Když nyní server odešle paket klientovi, klient již nebude moci tento paket přijmout, protože NAT router/firewall neví, co s tímto paketem dělat. Chcete-li tento problém vyřešit, `PersistentKeepalive`lze tuto možnost použít k pravidelnému odesílání prázdného ověřeného paketu na server, aby

bylo připojení otevřené. WireGuard navrhuje hodnotu 25 sekund, která by fungovala s širokou škálou firewallů. (Děkuji Rameshovi za komentář .)

`Peer`Pokud je tedy server za NAT nebo stavovým firewallem, měla by být v sekci konfigurace klienta přidána následující možnost :

```
PersistentKeepalive = 25
```

Využití serveru

Start/Stop ručně

Ruční spuštění serveru VPN:

```
# wg-quick up wg0
[#] iptables -t nat -A POSTROUTING -s 10.23.5.0/24 -o
enxb827eb7dc89a -j MASQUERADE; ip6tables -t nat -A POSTROUTING
-s fc00:23:5::/64 -o enxb827eb7dc89a -j MASQUERADE
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip address add 10.23.5.1/24 dev wg0
[#] ip address add fc00:23:5::1/64 dev wg0
[#] ip link set mtu 1420 dev wg0
[#] ip link set wg0 up
[#] ip route add fc00:23:5::/64 dev wg0
```

Opětovné zastavení služby:

```
# wg-quick down wg0
[#] ip link delete dev wg0
[#] iptables -t nat -D POSTROUTING -s 10.23.5.0/24 -o
enxb827eb7dc89a -j MASQUERADE; ip6tables -t nat -D POSTROUTING
-s fc00:23:5::/64 -o enxb827eb7dc89a -j MASQUERADE
```

Start/Stop pomocí systemd

Spuštění služby:

```
# systemctl start wg-quick@wg0
```

Zobrazují se podrobnosti o službě:

```
# systemctl status wg-quick@wg0
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
Loaded: loaded (/lib/systemd/system/wg-quick@.service; enabled;
vendor preset: en
Active: active (exited) since Sun 2018-09-23 20:48:10 CEST; 9s
ago
Docs: man:wg-quick(8)
man:wg(8)
https://www.wireguard.com/
https://www.wireguard.com/quickstart/
https://git.zx2c4.com/WireGuard/about/src/tools/man/wg-quick.8
https://git.zx2c4.com/WireGuard/about/src/tools/man/wg.8
Process: 18609 ExecStart=/usr/bin/wg-quick up wg0 (code=exited,
status=0/SUCCESS)
Main PID: 18609 (code=exited, status=0/SUCCESS)

Sep 23 20:48:09 wgpi systemd[1]: Starting WireGuard via wg-
quick(8) for wg0...
Sep 23 20:48:09 wgpi wg-quick[18609]: [#] iptables -t nat -A
POSTROUTING -s 10.23.5.
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip link add wg0 type
wireguard
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] wg setconf wg0
/dev/fd/63
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip address add
10.23.5.1/24 dev wg0
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip address add
fc00:23:5::1/64 dev wg0
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip link set mtu 1420
dev wg0
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip link set wg0 up
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip route add
fc00:23:5::/64 dev wg0
Sep 23 20:48:10 wgpi systemd[1]: Started WireGuard via wg-
quick(8) for wg0.
```

Opětovné zastavení služby:

```
# systemctl stop wg-quick@wg0
```

Služba je nyní zastavena:

```
# systemctl status wg-quick@wg0
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
Loaded: loaded (/lib/systemd/system/wg-quick@.service; enabled;
vendor preset: en
Active: inactive (dead) since Sun 2018-09-23 20:48:25 CEST; 37s
ago
Docs: man:wg-quick(8)
man:wg(8)
https://www.wireguard.com/
https://www.wireguard.com/quickstart/
https://git.zx2c4.com/WireGuard/about/src/tools/man/wg-quick.8
https://git.zx2c4.com/WireGuard/about/src/tools/man/wg.8
Process: 18667 ExecStop=/usr/bin/wg-quick down wg0
(code=exited, status=0/SUCCESS)
Process: 18609 ExecStart=/usr/bin/wg-quick up wg0 (code=exited,
status=0/SUCCESS)
Main PID: 18609 (code=exited, status=0/SUCCESS)

Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip address add
10.23.5.1/24 dev wg0
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip address add
fc00:23:5::1/64 dev wg0
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip link set mtu 1420
dev wg0
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip link set wg0 up
Sep 23 20:48:10 wgpi wg-quick[18609]: [#] ip route add
fc00:23:5::/64 dev wg0
Sep 23 20:48:10 wgpi systemd[1]: Started WireGuard via wg-
quick(8) for wg0.
Sep 23 20:48:25 wgpi systemd[1]: Stopping WireGuard via wg-
quick(8) for wg0...
Sep 23 20:48:25 wgpi wg-quick[18667]: [#] ip link delete dev
wg0
Sep 23 20:48:25 wgpi wg-quick[18667]: [#] iptables -t nat -D
POSTROUTING -s 10.23.5.
Sep 23 20:48:25 wgpi systemd[1]: Stopped WireGuard via wg-
quick(8) for wg0.
```

Automaticky spustit službu při spuštění systému:

```
# systemctl enable wg-quick@wg0
```

Znovu zakažte službu při spuštění:

```
# systemctl disable wg-quick@wg0
```

Ověřování

`wg0`Při spuštění služby se vytvoří nové síťové rozhraní :

```
# ip a l wg0
26: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue
state UNKNOWN group default qlen 1000
link/none
inet 10.23.5.1/24 scope global wg0
valid_lft forever preferred_lft forever
inet6 fc00:23:5::1/64 scope global
valid_lft forever preferred_lft forever
```

Trasa je odeslána podle `AllowedIPs`směrnice:

```
# ip route
default via 192.168.1.1 dev enxb827eb7dc89a src 192.168.1.10
metric 202
10.23.5.0/24 dev wg0 proto kernel scope link src 10.23.5.1
192.168.1.0/24 dev enxb827eb7dc89a proto kernel scope link src
192.168.1.10 metric 202
```

Zobrazení aktuální konfigurace:

```
# wg show
interface: wg0
public key: 3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
private key: (hidden)
listening port: 1500

peer: UGyBshzPfAH0U4QAgGJHe07LfUz4RcHA9PhUlUC4cCA=
allowed ips: 10.23.5.2/32, fc00:23:5::2/32
peer: 1xy8XRtUQT/9AwYWlEXsWCezNjfiFjXaBy40UUtAWBo=
allowed ips: 10.23.5.3/32, fc00:23:5::3/32
```

Pokud jsou klienti připojeni, zobrazí se další data:

```
# wg show
interface: wg0
public key: 3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
private key: (hidden)
listening port: 1500

peer: 1xy8XRtUQT/9AwYWlEXsWCezNjfiFjXaBy40UUtAWBo=
endpoint: 178.197.42.137:18822
allowed ips: 10.23.5.3/32, fc00:23:5::3/128
latest handshake: 4 seconds ago
transfer: 788 B received, 732 B sent

peer: UGyBshzPfAH0U4QAgGJHe07LfUz4RcHA9PhUlUC4cCA=
endpoint: 178.197.42.137:18821
allowed ips: 10.23.5.2/32, fc00:23:5::2/128
latest handshake: 16 seconds ago
transfer: 6.61 KiB received, 6.07 KiB sent
```

Zobrazení podrobné konfigurace rozhraní:

```
# wg showconf wg0
[Interface]
ListenPort = 1500
PrivateKey = mNt0Gx2Af/OCkT9FchX3nybsaXUAerglnuMnSud4z1k=

[Peer]
PublicKey = UGyBshzPfAH0U4QAgGJHe07LfUz4RcHA9PhUlUC4cCA=

[Peer]
PublicKey = 1xy8XRtUQT/9AwYWlEXsWCezNjfiFjXaBy40UUtAWBo=
AllowedIPs = 10.23.5.0/24, fc00:23:5::/64
```

Použití klienta

Kopírování konfiguračního souboru klienta do `/etc/wireguard`:

```
# cd /etc/wireguard
# ls -l wg0.conf
-rw------- 1 root root 245 Sep 23 21:17 wg0.conf
# cat wg0.conf
[Interface]
PrivateKey = OPTN5qb4FFulEBFLlrmC1sFTawT6AmdhsAbigpCMemw=
Address = 10.23.5.2/24, fc00:23:5::2/64
DNS = 8.8.8.8

[Peer]
PublicKey = 3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
Endpoint = wg.example.com:1500
AllowedIPs = 0.0.0.0/0, ::/0
```

Spuštění služby stejným způsobem jako na serveru:

```
# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip address add 10.23.5.2/24 dev wg0
[#] ip address add fc00:23:5::2/64 dev wg0
[#] ip link set mtu 1420 dev wg0
[#] ip link set wg0 up
[#] resolvconf -a tun.wg0 -m 0 -x
[#] wg set wg0 fwmark 51820
[#] ip -6 route add ::/0 dev wg0 table 51820
[#] ip -6 rule add not fwmark 51820 table 51820
[#] ip -6 rule add table main suppress_prefixlength 0
[#] 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
```

Bylo vytvořeno nové síťové rozhraní:

```
# ip a l wg0
4: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue
state UNKNOWN group default qlen 1
link/none
inet 10.23.5.2/24 scope global wg0
valid_lft forever preferred_lft forever
inet6 fc00:23:5::2/64 scope global
valid_lft forever preferred_lft forever
```

VPN funguje:

```
# ping 10.23.5.1
PING 10.23.5.1 (10.23.5.1) 56(84) bytes of data.
64 bytes from 10.23.5.1: icmp_seq=1 ttl=64 time=284 ms
64 bytes from 10.23.5.1: icmp_seq=2 ttl=64 time=63.3 ms
64 bytes from 10.23.5.1: icmp_seq=3 ttl=64 time=225 ms
64 bytes from 10.23.5.1: icmp_seq=4 ttl=64 time=150 ms
^C
--- 10.23.5.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 63.345/180.886/284.496/82.871 ms
```

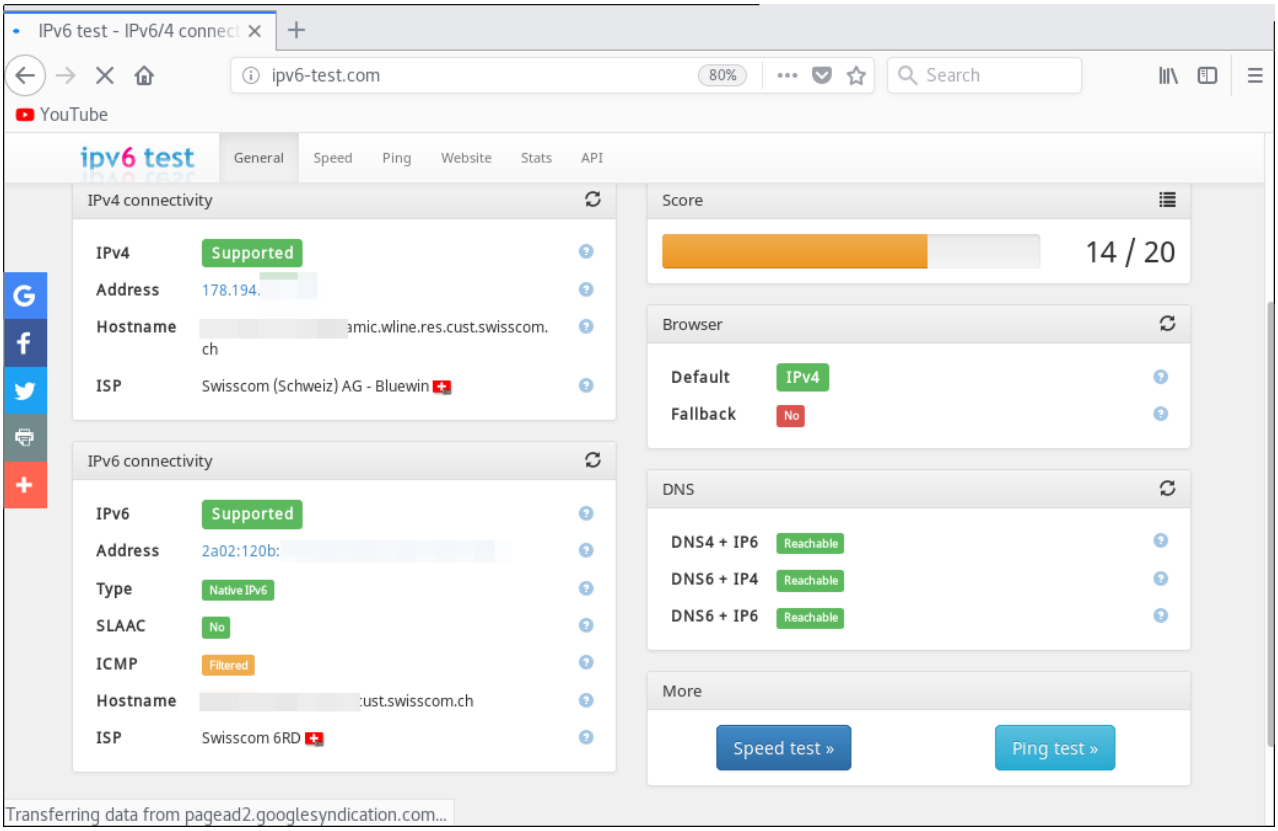
Zobrazení spojení:

```
# wg show
interface: wg0
public key: UGyBshzPfAH0U4QAgGJHe07LfUz4RcHA9PhUlUC4cCA=
private key: (hidden)
listening port: 52682
fwmark: 0xca6c

peer: 3oPm1WH3RXv+8zCEQ7onRAYJWAvuHKo/1OfIiTE5LDc=
endpoint: 178.194.23.5:1500
allowed ips: 0.0.0.0/0, ::/0
latest handshake: 5 seconds ago
transfer: 604 B received, 660 B sent
```

Protože `AllowedIPs`je direktiva nakonfigurována na `0.0.0.0/0`a `::/0`, veškerý provoz je směrován přes VPN:

```
# curl -L motd.ch/ip.php
178.194.23.5
```

IPv4 i IPv6 fungují přes tunel:



Zastavení VPN:

```
# wg-quick down wg0
[#] ip -4 rule delete table 51820
[#] ip -4 rule delete table main suppress_prefixlength 0
[#] ip -6 rule delete table 51820
[#] ip -6 rule delete table main suppress_prefixlength 0
[#] ip link delete dev wg0
[#] resolvconf -d tun.wg0
```

Spuštění přes službu systemd:

```
# systemctl start wg-quick@wg0
```

Zastavení přes systemd:

```
# systemctl stop wg-quick@wg0
```

Postavení:

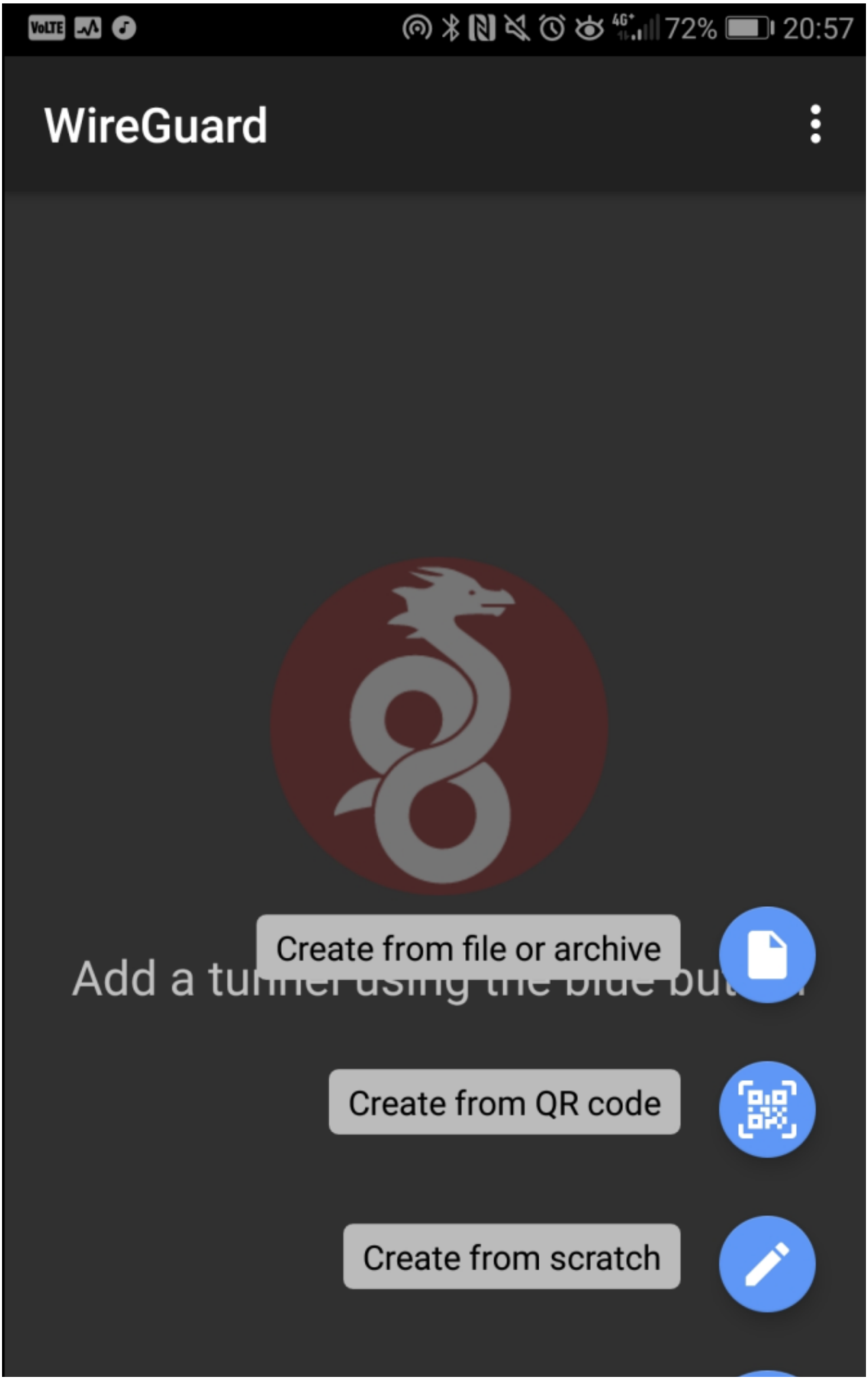
```
# systemctl status wg-quick@wg0
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
Loaded: loaded (/lib/systemd/system/wg-quick@.service;
disabled; vendor prese
Active: inactive (dead)
Docs: man:wg-quick(8)
man:wg(8)
https://www.wireguard.com/
https://www.wireguard.com/quickstart/
https://git.zx2c4.com/WireGuard/about/src/tools/man/wg-quick.8
https://git.zx2c4.com/WireGuard/about/src/tools/man/wg.8

Sep 23 21:54:35 dimmbar wg-quick[1629]: [#] ip -4 rule add
table main suppress_p
Sep 23 21:54:35 dimmbar systemd[1]: Started WireGuard via wg-
quick(8) for wg0.
Sep 23 21:54:46 dimmbar systemd[1]: Stopping WireGuard via wg-
quick(8) for wg0..
Sep 23 21:54:47 dimmbar wg-quick[1732]: [#] ip -4 rule delete
table 51820
Sep 23 21:54:47 dimmbar wg-quick[1732]: [#] ip -4 rule delete
table main suppres
Sep 23 21:54:47 dimmbar wg-quick[1732]: [#] ip -6 rule delete
table 51820
Sep 23 21:54:47 dimmbar wg-quick[1732]: [#] ip -6 rule delete
table main suppres
Sep 23 21:54:47 dimmbar wg-quick[1732]: [#] ip link delete dev
wg0
Sep 23 21:54:47 dimmbar wg-quick[1732]: [#] resolvconf -d
tun.wg0
Sep 23 21:54:47 dimmbar systemd[1]: Stopped WireGuard via wg-
quick(8) for wg0.
```

Použití mobilního klienta

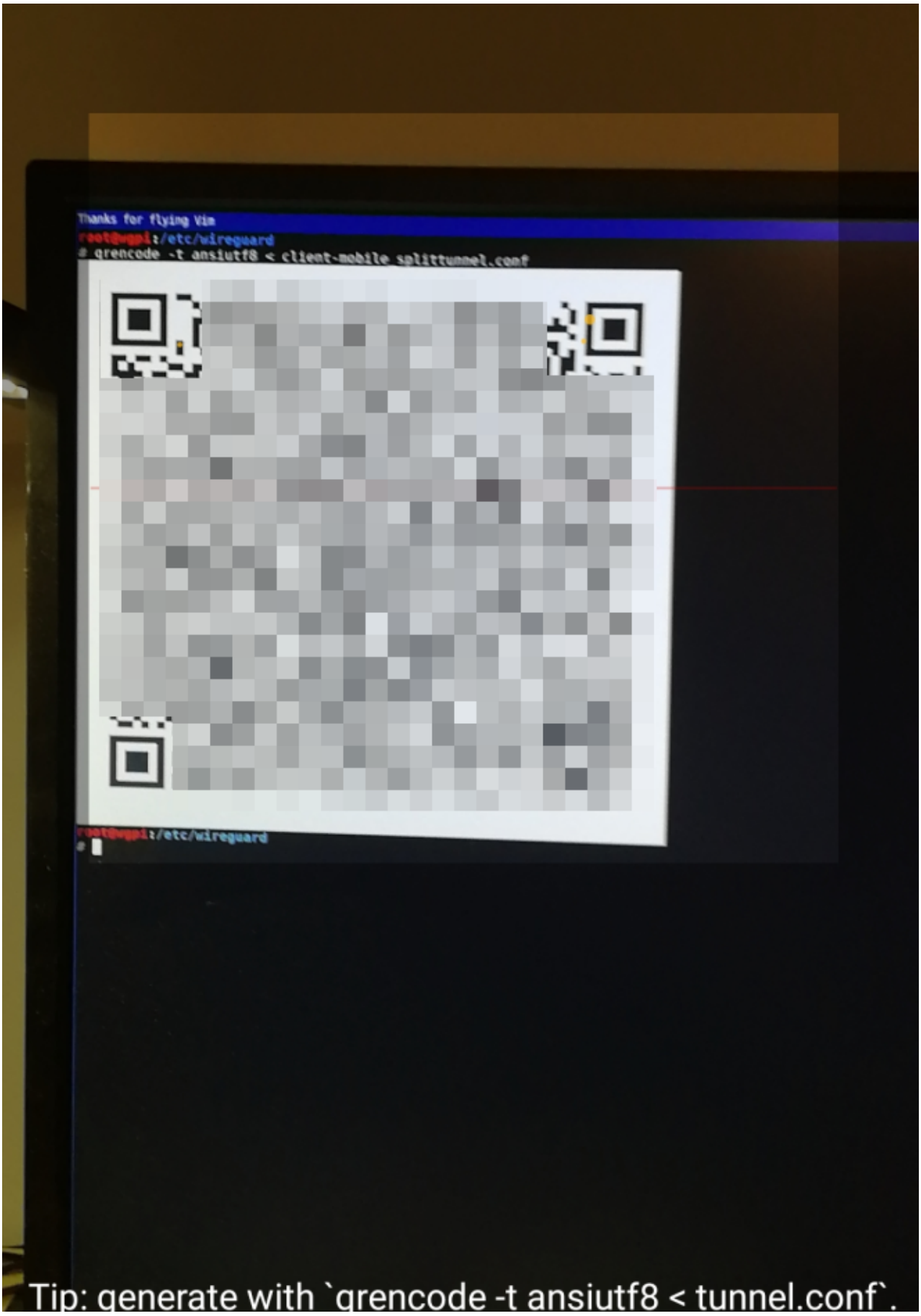
Generování QR kódu pro mobilního klienta:

```
# qrencode -t ansiutf8 < client-mobile_splittunnel.conf
```

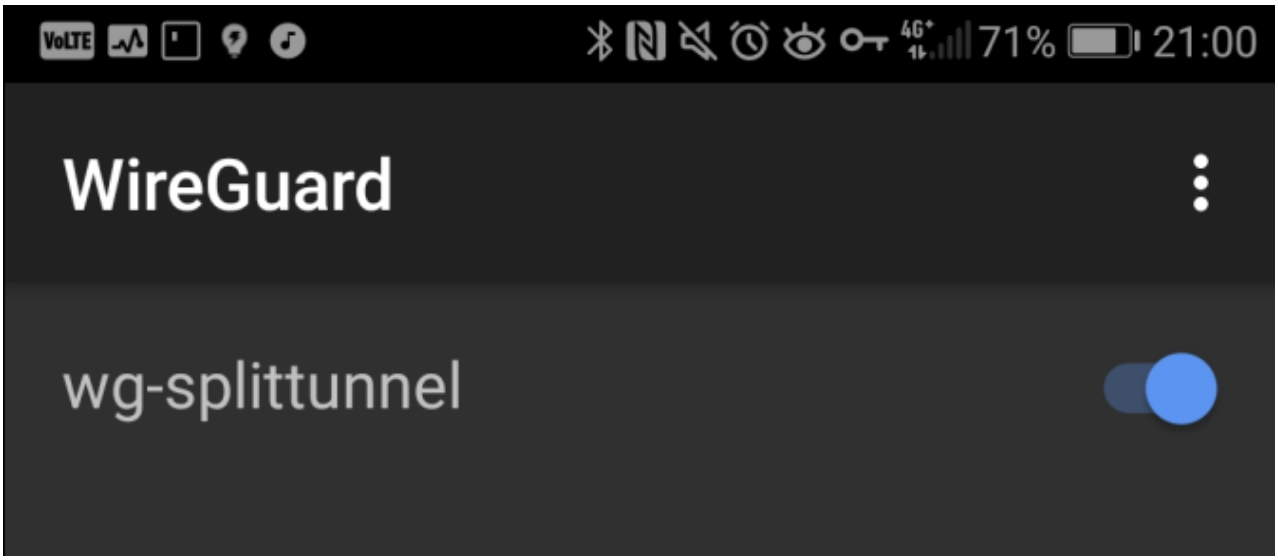
Přidání nového připojení VPN výběrem `Create from QR code`:


72%
20:57
WireGuard
Create from file or archive
Create from QR code
Create from scratch

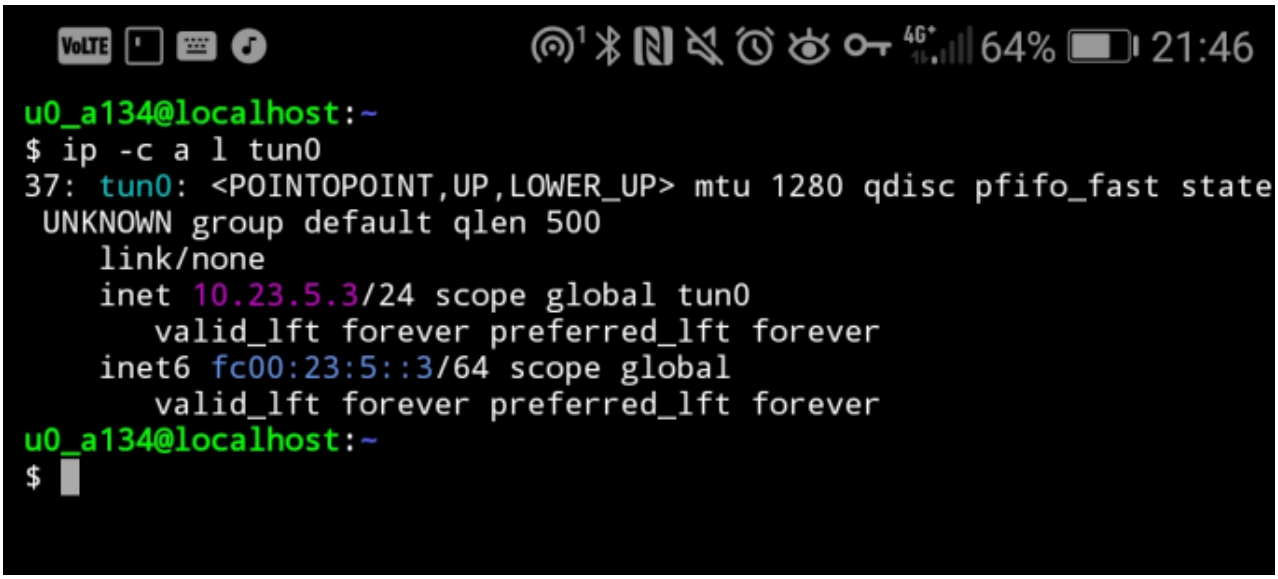
Skenování QR kódu:



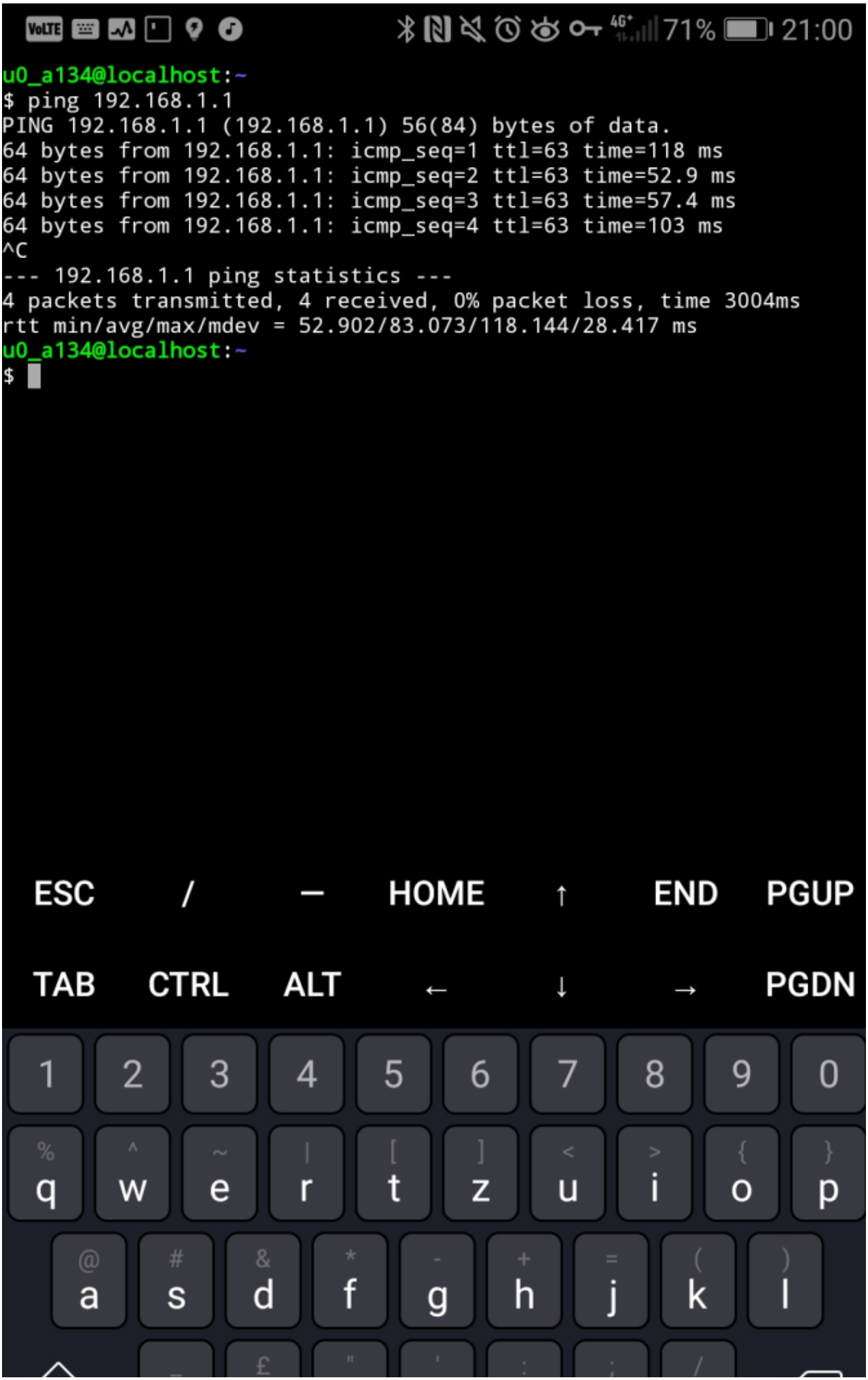
Povolení VPN:

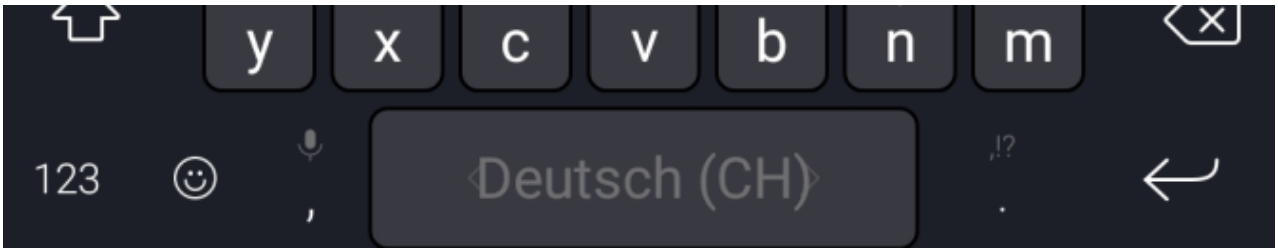


Bylo vytvořeno nové síťové rozhraní s nakonfigurovanými IP adresami:

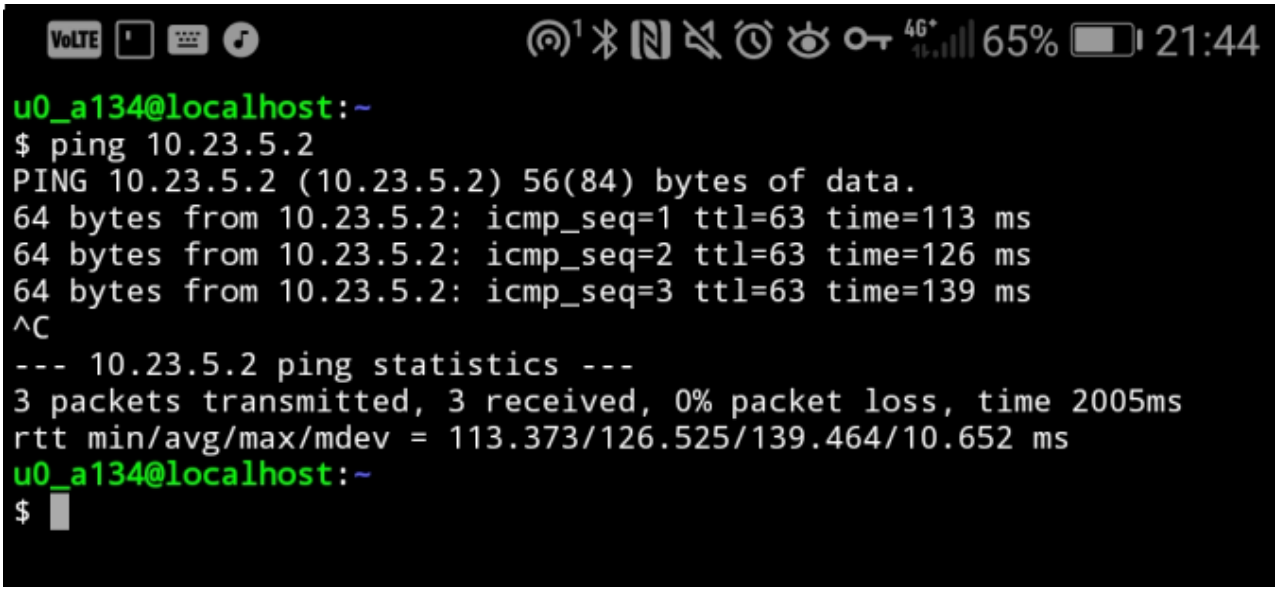


Přístup ke vzdálené síti:

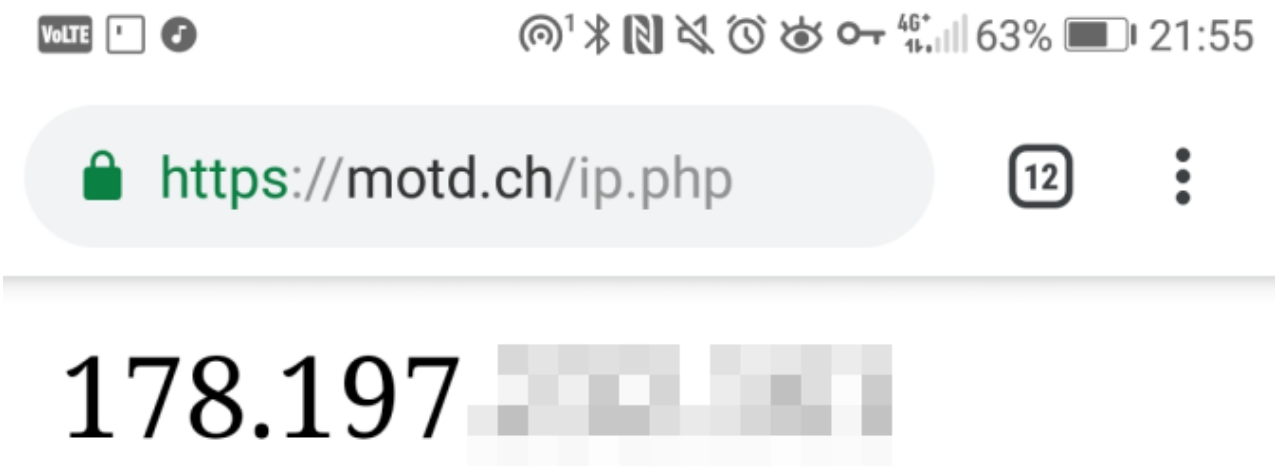

71%
21:00
u0_a134@localhost:~
$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=63 time=118 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=63 time=52.9 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=63 time=57.4 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=63 time=103 ms
^C
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 52.902/83.073/118.144/28.417 ms
u0_a134@localhost:~
$
ESC / — HOME ↑ END PGUP
TAB CTRL ALT ← ↓ → PGDN
1 2 3 4 5 6 7 8 9 0
% q ^ w ~ e | r [t] z < u > i { o } p
@ a # s & d * f - g + h = j (k) l



Je také možné oslovit další klienty VPN (brána firewall tomu nebrání):



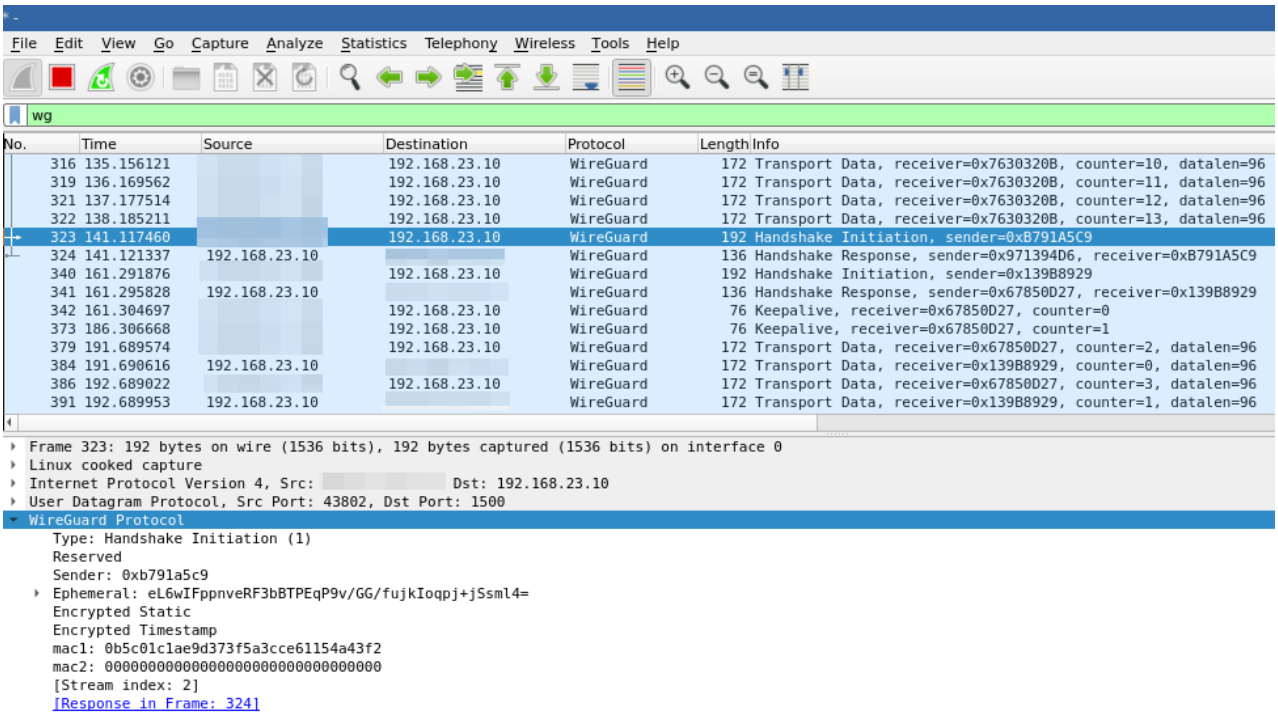
Protože se používá rozdělené tunelování, normální síťový provoz neprochází přes pole VPN:



Ladění (aktualizace z 23. 3. 2019)

Wireshark

Wireshark má disektor pro WireGuard :



Další informace o tom, jak dešifrovat data v rámci Wireshark poskytnutím protokolů klíčů, najdete zde: https://github.com/Lekensteyn/wireguard-dissector .

Používání nftables (aktualizace z 2021-04-18)

Pokud chcete na serveru Wireguard používat nftables místo iptables, můžete to udělat bez problémů. Podle toho musíte nakonfigurovat nftables.

Protože se ke konfiguraci nftables používá několik příkazů, má smysl používat vlastní skripty v konfiguraci serveru Wireguard:

```
PreUp = /etc/wireguard/server-preup
PostDown = /etc/wireguard/server-postdown
```

skript `/etc/wireguard/server-preup`:

```
#!/usr/bin/env bash

VPNIF="wg0"
LANIF="enp2s0"

nft insert rule ip filter FORWARD iifname "$VPNIF" oifname
"$LANIF" accept
nft insert rule ip filter FORWARD iifname "$LANIF" oifname
"$VPNIF" ct state related,established accept

nft insert rule ip6 filter FORWARD iifname "$VPNIF" oifname
"$LANIF" accept
nft insert rule ip6 filter FORWARD iifname "$LANIF" oifname
"$VPNIF" ct state related,established accept

nft add table ip wireguard-nat
nft -- add chain ip wireguard-nat prerouting { type nat hook
prerouting priority -100 \; }
nft add chain ip wireguard-nat postrouting { type nat hook
postrouting priority 100 \; }
nft add rule ip wireguard-nat postrouting oifname "$LANIF"
masquerade

nft add table ip6 wireguard-nat
nft -- add chain ip6 wireguard-nat prerouting { type nat hook
prerouting priority -100 \; }
nft add chain ip6 wireguard-nat postrouting { type nat hook
postrouting priority 100 \; }
nft add rule ip6 wireguard-nat postrouting oifname "$LANIF"
masquerade
```

Tento skript umožňuje přesměrování mezi Wireguard VPN a rozhraním připojeným k LAN a přidává pravidla NAT pro IPv4 a IPv6.

skript `/etc/wireguard/server-postdown`:

```
#!/usr/bin/env bash

VPNIF="wg0"
LANIF="enp2s0"

nft list chain ip filter FORWARD -a | grep "iifname.*$VPNIF" |
awk '{ print $NF }' | while read handle
do
  nft delete rule ip6 filter FORWARD handle "$handle"
done

nft list chain ip6 filter FORWARD -a | grep "iifname.*$VPNIF" |
awk '{ print $NF }' | while read handle
do
  nft delete rule ip6 filter FORWARD handle "$handle"
done

nft delete table wireguard-nat
```

Tento skript odstraní přidaná pravidla. Pravidla NAT musí být odstraněna pomocí ID handleru, protože v tuto chvíli není možné je odstranit pomocí stejné syntaxe, jakou byla přidána (jako v iptables).

Restartujte server Wireguard a jste připraveni jít.

Reference

- Stránka projektu WireGuard: https://www.wireguard.com/
- Skvělé povídání o WireGuard: https://www.youtube.com/watch?v=eYztYCbV_8U&t=2229s
- WireGuard na Raspberry Pi: https://github.com/adrianmihalko/raspberrypiwireguard
- Zdrojový kód (úložiště Git): https://git.zx2c4.com/WireGuard/
- Archiv seznamu adresátů: https://lists.zx2c4.com/pipermail/wireguard/
- IRC: #wireguard na Freenode