Shea Polansky

WireGuard: VPN of the Future

Motivation: Previous VPNs

- IPSec – Security Extensions for Internet Protocol
  - Open standard
  - Built into nearly every OS
  - Sorta-kinda provides VPN functionality out of the box
  - Can also be security layer for Layer 2 Tunneling Protocol (L2TP)
- OpenVPN
  - Open source
  - Basically OpenSSL + virtual network layer
  - Used by ISE

- SSL VPNs
  - Cisco AnyConnect
  - Microsoft SSTP
  - Look like HTTPS connections to firewalls
  - Require proprietary software
- Mesh VPNs
  - ZeroTier
  - Tinc
  - New tech, immature

Motivation: Problems in Previous VPNs

IPSec is F****ing confusing

- About 100 RFCs required to understand the protocol
- Has 2 different operating modes for VPN that work completely differently
- Requires 2 separate pieces of software to run on Linux
- Supports all of the ciphers, so you'll have to pick the right ones
  - Double check you're not accepting NULL ciphers :)

IPSec isn't very reliable for "Road Warrior" setups

- NAT and Firewalls tend to accidentally break it
- "Everything" supports it, but what exactly is supported varies
  - Authentication mechanisms, what exactly shows up as device ID, etc.

Motivation: Problems in Previous VPNs

OpenVPN isn't much better

- Also confusing
- Really, really slow
- Contains full DHCP implementation
- Also has all the ciphers, including NULL

SSL VPNs are generally proprietary

- Also run over TCP – high latency
- Can also have cipher choice issues

Mesh VPNs are immature

Solution: WireGuard®

FAST, MODERN, SECURE VPN TUNNEL

- Brand new, ground-up design VPN protocol
- Kernel Module and Userspace Implementation Available
  - Accepted into the main kernel tree as of 5.6
- Runs on any platform Go compiles to
  - Win/Mac/Linux
    - x86/x64/ARM/MIPS/POWER/SPARC/Whatever
  - iOS/Android
- Based on protocol designed by Moxie Marlinspike (Signal)
- Layer 3 point-to-point tunnels with automatic roaming support

Solution: WireGuard

FAST, MODERN, SECURE VPN TUNNEL

WireGuard design goals

1. Simple to configure – should be as simple as SSH keys to setup
2. Limited scope – WireGuard handles secure tunneling, full stop.
3. Easy to audit – Source code is simple and small.
4. Opinionated – WireGuard has almost no knobs to tune, and none of them are ciphers.
5. State of the Art Crypto – WireGuard uses state of the art crypto with optional quantum resistance

WireGuard is Simple to Configure

OpenVPN Example Configs:

```
server 172.16.0.0 255.255.255.0
dev tun
proto tcp
port PORT
keepalive 10 120
push "route 10.0.0.0 255.255.255.0"

ca cacert.pem
cert SRV.crt
key SRV.key
dh dh1024.pem
```

```
client
dev tun
remote SRV.domain
proto tcp
port PORT

ca cacert.pem
cert CLNT.crt
key CLNT.key
```

From https://www.cs.put.poznan.pl/csobaniec/examples/openvpn/ (first result when googling for example config)

WireGuard is Simple to Configure

OpenVPN Example Configs:

```
server 172.16.0.0 255.255.255.0
dev tun
proto tcp
port PORT
keepalive 10 120
push "route 10.0.0.0 255.255.255.0"

ca cacert.pem
cert SRV.crt
key SRV.key
dh dh1024.pem
```

PKI required

```
client
dev tun
remote SRV.domain
proto tcp
port PORT

ca cacert.pem
cert CLNT.crt
key CLNT.key
```

TCP is very high latency, use UDP where possible

Weak – Logjam Attack

This doesn't cover cipher support, so defaults (more compatible than secure) will be used

From https://www.cs.put.poznan.pl/csobaniec/examples/openvpn/ (first result when googling for example config)

WireGuard is Simple to Configure

Server

```
[Interface]
PrivateKey =
yAnz5TF+lXXJte14tji3zlMNq+hd2rYUIg
JBgB3fBmk=
ListenPort = 51820

[Peer]
PublicKey =
xTIBA5rboUvnH4htodjb6e697QjLERt1NA
B4mZqp8Dg=
AllowedIPs = 10.192.122.3/32,
10.192.124.1/24
```

Client

```
[Interface]
PrivateKey =
gI6EdUSYvn8ugXOt8QQD6Yc+JyiZxIhp3GInSW
RfWGE=
ListenPort = 21841

[Peer]
PublicKey =
HIgo9xNzJMWLKASShiTqIybxZ0U3wGLiUeJ1PK
f8ykw=
Endpoint = 192.95.5.69:51820
AllowedIPs = 0.0.0.0/0
```

WireGuard Has A Limited Scope

WireGuard does…

- …provide encrypted, authenticated, point-to-point tunnels between pre-configured hosts
- …handle roaming, routing, address validation, DoS resistance, and key generation

WireGuard does *not*…

- …handle IP assignment, prefix delegation, or NAT issues
- …support layer 2 tunneling (though you could use L2TP/EoIP)
- …deal with key distribution or public key infrastructure

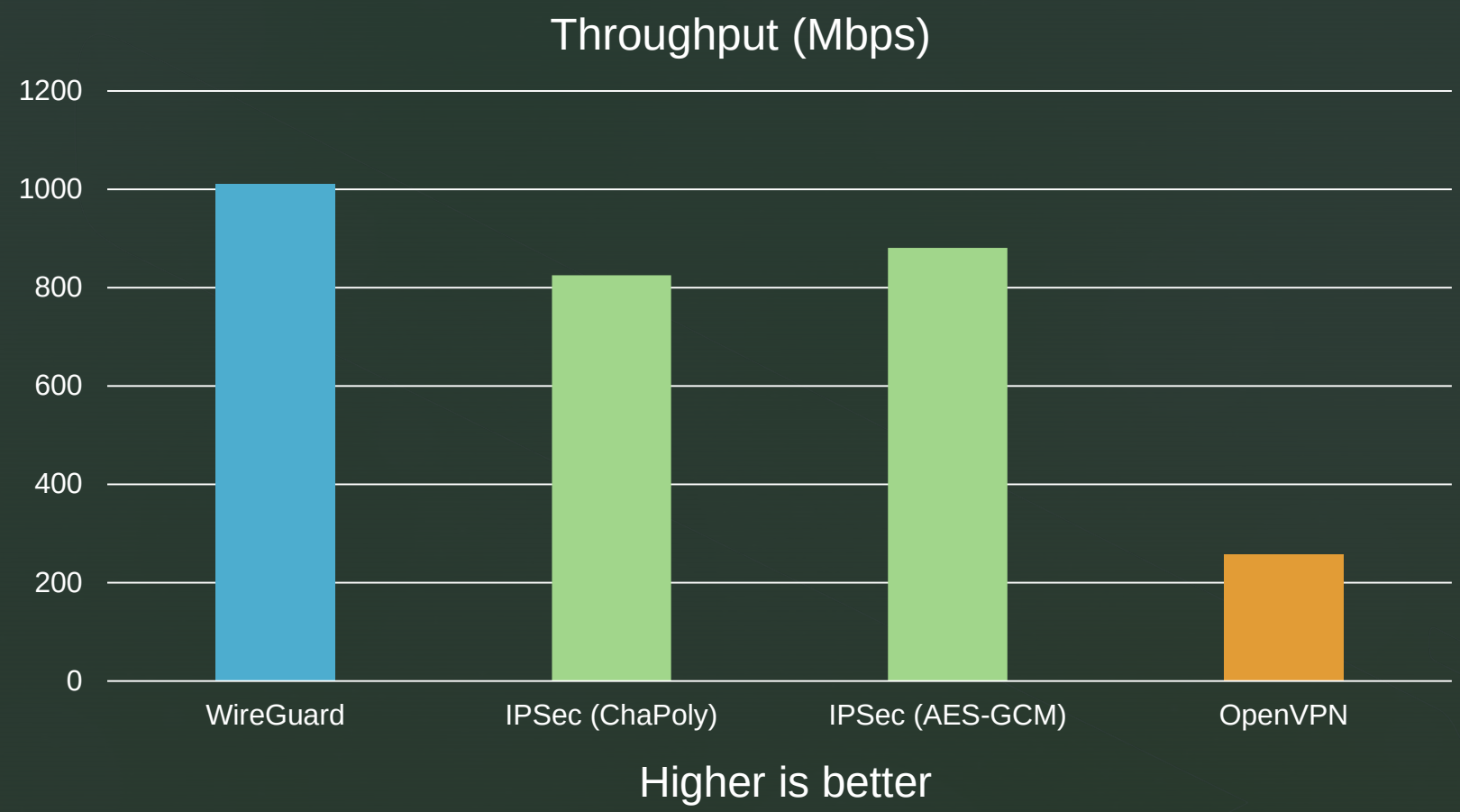
WireGuard is Easy to Audit

- Total code base is 5k LOC
  - OpenVPN is 10k+ *not including OpenSSL*
  - IPSec is 2-3 different code bases, hundreds of thousands of lines
  - Proprietary VPNs: ????
- Protocol is formally verified to provide:
  - Correctness, Strong key agreement & authenticity, Key-compromise impersonation resistance, Unknown key-share attack resistance, Key secrecy, **Forward secrecy**, Session uniqueness, Identity hiding
- Implementation has undergone additional audits

WireGuard is Opinionated & Uses State-of-the-Art Crypto

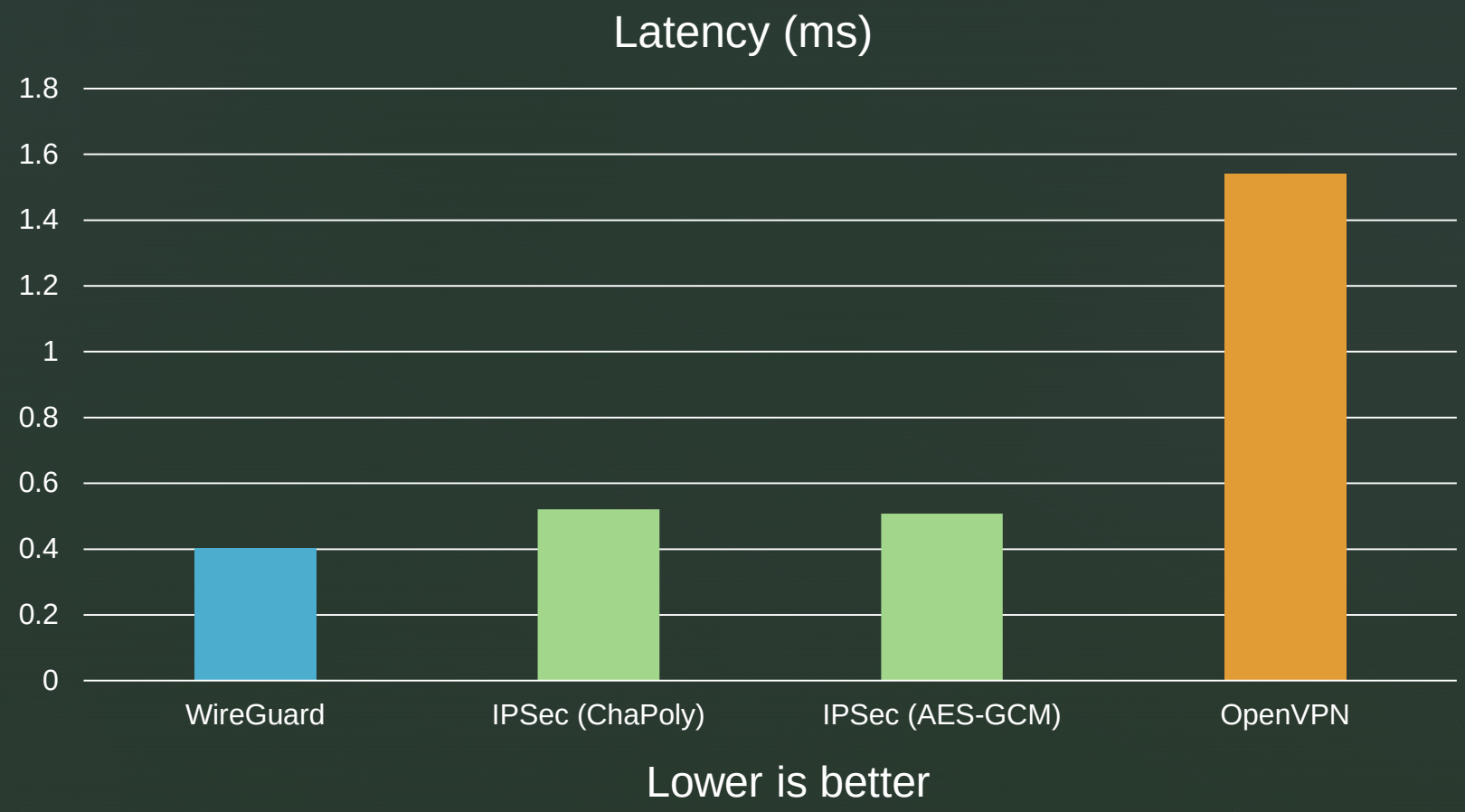
- Virtually no knobs to turn – only one crypto/security related
  - Optional pre-shared key for quantum resistance
    - 99 percent of users will never touch
- State of the art crypto is mandatory
  - ChaCha20/Poly1305 AEAD construction for data
  - ED25519 keys
  - SipHash24/BLAKE2/HKDF for hashing/key derivation
- Hypothetical changes in crypto will be versioned – no compatibility/security weirdness

WireGuard is Very, Very Fast



Source: https://www.wireguard.com/performance/

WireGuard is Very, Very Fast



Source: https://www.wireguard.com/performance/

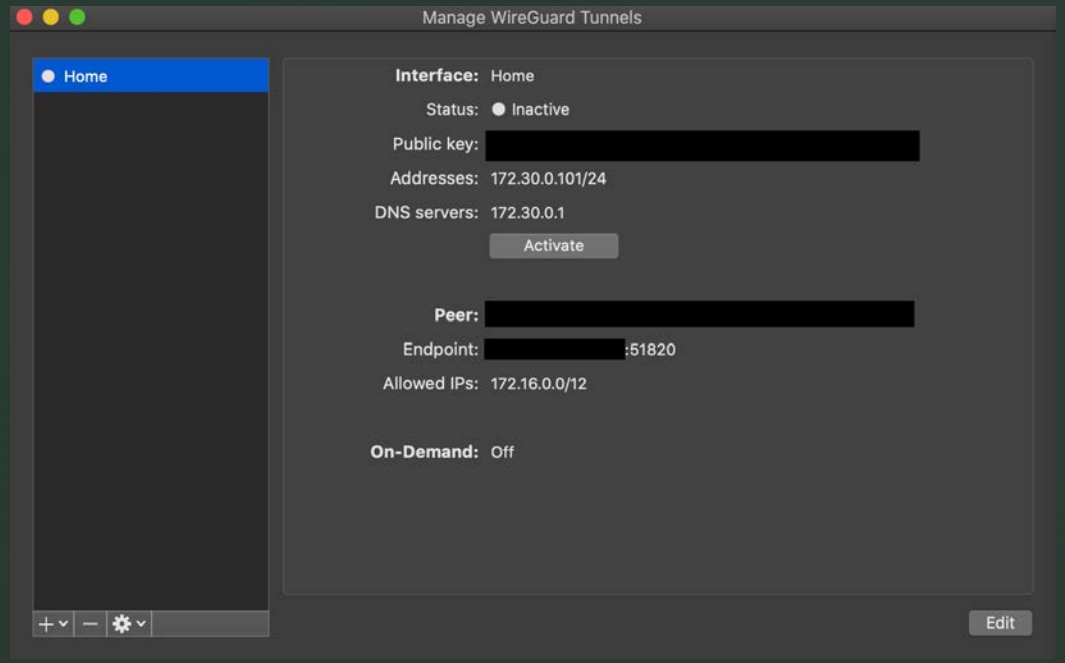
Using WireGuard

Source: https://www.wireguard.com/quickstart/

Using WireGuard

Use the `wg-quick` tool/platform GUIs

- Part of default WG tools package
- Extends config syntax to support QoL features like IP/DNS assignment
- One click/command bring up/teardown
- `wg-quick save <int>` for instant config generation



Using `wg-quick` (Server)

```
[Interface]
PrivateKey = <Private Key>
Address = 10.0.0.1/24, fd86:ea04:1115::1/64
ListenPort = 51820
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING
-o eth0 -j MASQUERADE; ip6tables -A FORWARD -i wg0 -j ACCEPT; ip6tables -t
nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D
POSTROUTING -o eth0 -j MASQUERADE; ip6tables -D FORWARD -i wg0 -j ACCEPT;
ip6tables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
SaveConfig = true

[Peer]
PublicKey = HIgo9xNzJMWLKASShiTqIybxZ0U3wGLiUeJ1PKf8ykw=
Endpoint = 192.95.5.69:51820
AllowedIPs = 0.0.0.0/0
```

Using `wg-quick` (Client)

```
[Interface]
PrivateKey = <Private Key>
Address = 10.0.0.1/24, fd86:ea04:1115::1/64

[Peer]
PublicKey = <Server Public key>
Endpoint = <Server Public IP>:51820
AllowedIPs = 10.0.0.2/24, fd86:ea04:1115::5/64
```

WireGuard Downsides

- IP assignment must be done manually
- Each host must be told about each other host
  - N hosts = N^2 configurations
- No PKI – Key distro can be annoying

There are tools to solve this:

- wg-dynamic – Dynamic mesh VPN configuration daemon
- Subspace Cloud – Self enrollment VPN UI w/ SSO+MFA

…unfortunately, pretty immature

Thank you!
Questions?

Download Links/More Info

- WireGuard homepage: https://www.wireguard.com
- Full tutorial: https://www.linode.com/docs/networking/vpn/set-up-wireguard-vpn-on-ubuntu/
- Installation:
  - MacOS: App Store
    - Also `brew install wireguard-go wireguard-tools`
  - Windows: `choco install -pre wireguard`
  - Debian&co.: `apt install wireguard wireguard-tools`
  - Android: Google Play
  - iOS: App Store