WireGuard

cryptokey routing table with modern cryptography

tsaimh (2024, 2026)
? (?-2023)

Source: https://archive.fosdem.org/2017/interviews/jason-a-donenfeld/

Introduction

- WireGuard is a communication protocol and free and open-source software that implements encrypted VPNs, and was designed with the goals of ease of use, high speed performance, and low attack surface.
- It aims to be smaller and better performing than IPsec and OpenVPN.
- The WireGuard protocol passes traffic over UDP.
- Initially released by Jason A. Donenfeld for the Linux kernel in 2015, it is now cross-platform (Windows, macOS, BSD, iOS, Android) and widely deployable.

Jason A. Donenfeld
President of
EdgeSecurity

Introduction (cont.)

- Simple and fast VPN solution
  - ✓ Low overhead
  - ✓ Deep integration with Linux kernel
  - ✓ Over UDP
- Peer to Peer
- Secure
- Built-in Roaming
  - ✓ Connections keep alive even if the underlay network change

Source: https://openvpn.net/blog/the-history-of-openvpn/

Installation

- https://www.wireguard.com/install/
- Linux (kernel >=3.10)
  - ✓ sudo apt install wireguard
- FreeBSD (13.0 or later)
  - ✓ $ pkg install wireguard

Tools

- Tools provided by WireGuard
  - ✓ wg
    set and retrieve configuration of WireGuard interface
  - ✓ wg-quick
    set up a WireGuard interface simply
- System tools
  - ✓ ip / ifconfig
    setup wg interfaces
  - ✓ systemd
    auto start after boot

Setup by hand (Linux)

- Add interface
  - ✓ $ ip link add dev wg0 type wireguard
- Setup IP address
  - ✓ $ ip address add dev wg0 192.168.2.1/24
  - ✓ $ ip address add dev wg0 192.168.2.1 peer 192.168.2.2
- Setup wg configurations
  - ✓ $ wg setconf wg0 myconfig.conf
  - ✓ $ wg set wg0 listen-port 51820 private-key /path/to/private-key peer ABCDEF... allowed-ips 192.168.88.0/24 endpoint 209.202.254.14:8172
- Setup interface
  - ✓ $ ip link set up dev wg0

Setup by configuration

- Configuration file
  - ✓ /etc/wireguard/wg0.conf
- Enable and start interface
  - ✓ $ systemctl enable wg-quick@wg0
  - ✓ $ wg-quick up wg0

Example Configurations – Client

```
[Interface]
Address = 10.113.0.4/16
PrivateKey = [CLIENT PRIVATE KEY]

[Peer]
PublicKey = [SERVER PUBLICKEY]
AllowedIPs = 10.113.0.0/16, 10.123.45.0/24, 1234:4567:89ab::/48
Endpoint = [SERVER ENDPOINT]:51820
PersistentKeepalive = 25
```

Example Configurations – Server

```
[Interface]
Address = 10.113.0.254/16
ListenPort = 51820
PrivateKey = [SERVER PRIVATE KEY]

# note - substitute eth0 in the following lines to match the Internet-facing interface
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j
MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j
MASQUERADE

[Peer]
PublicKey = [FOO's PUBLIC KEY]
PresharedKey = [PRE-SHARED KEY]
AllowedIPs = 10.113.0.1/32, 10.113.1.0/24

[Peer]
PublicKey = [BAR's PUBLIC KEY]
AllowedIPs = 10.113.0.2/32, 10.113.2.0/24
```

Configuration – Interface

- Address (optional)
  IP address and netmask of the interface
- ListenPort
  Wg service listen port
- PrivateKey
  Private key of the interface
- PreUp / PreDown / PostUp / PostDown
  Run shell scripts before / after interface up / down
  (E.g. Setup firewall rules)

Configuration – Peer

- PublicKey
  Public key of the peer
- AllowedIPs
  IP addresses that are allowed to pass through this peer
- Endpoint (Optional)
  Location of the peer
  Wg will also use the previous connections to detect this configuration
- PersistentKeepalive (Optional)
  By default, Wg send packs only if there are data to be send
  Send packs to peer periodically to bypass NAT or Firewall
- PresharedKey (Optional)
  Pre-shared key for additional symmetric encryption

Generate Key Pair

- Key pair
  - ✓ $ wg genkey > privatekey
  - ✓ $ wg pubkey < privatekey > publickey
- Pre-shared key
  - ✓ $ wg genpsk > preshared

Cryptokey Routing

- WireGuard will add routing rules to system routing table according to the configurations.
- Once packets go inside WireGuard, it is routed according to Cryptokey Routing.
  - ✓ When sending packets, the list of allowed IPs behaves as a sort of routing table
  - ✓ When receiving packets, the list of allowed IPs behaves as a sort of access control list

Built-in Roaming

- When the client connects to server, server record the IP of client, and communicate with client by this IP.
- When client (or even server) change its IP, it sends data to the peer and the peer will update the IP.
- Both client and server send encrypted data to the most recent IP endpoint for which they authentically decrypted data. Thus, there is full IP roaming on both ends.

Example – Build a Bridge VPN Server

- Follow the setup guide and build a Wg peer as a VPN server
- Enable ip forwarding
  - ✓ sysctl net.ipv4.ip_forward=1
- Setup NAT so clients can connect to internet through the VPN server
  - ✓ Add these lines to wg0.conf
    - ➢ PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
    - ➢ PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

Connect from mobile

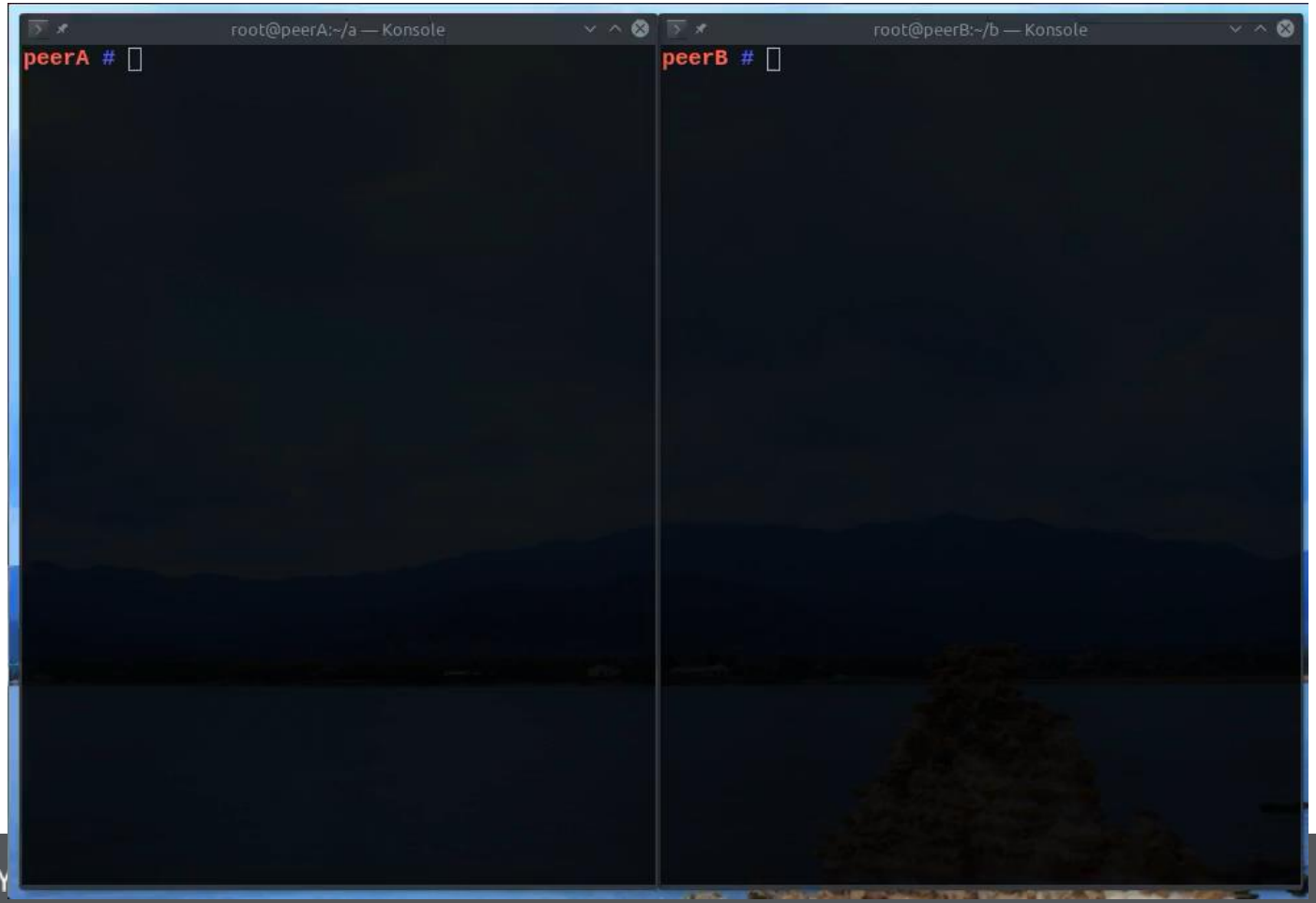
- For mobile app, user can use QR-Code to import configuration file, instead of copy-paste private key from other ways
  - ✓ $ qrencode -t ansiutf8 < wgconfig.conf

User authentication

- Every peer has its own private key for identity authentication
- Integration with other authentication system (like LDAP) may need other software support
  - ✓ For now, WireGuard only provide simple tunnel connections between peers.

Side by Side Configuration

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



Reference

- https://www.wireguard.com/
- https://www.wireguard.com/quickstart/
- https://www.wireguard.com/papers/wireguard.pdf
- https://www.zenarmor.com/docs/network-security-tutorials/how-to-install-wireguard-on-freebsd
- https://wiki.archlinux.org/index.php/WireGuard