WireGuard

jnlin

WireGuard Introduction

- ❑ 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

TUN/TAP

TAP	TUN
Layer 2	Layer 3
More overhead(L2)	Less Overhead(L3)
Transfer any protocol	Only IPv4 , IPv6
Support L2+ services	Support L3+ services

Installation

- https://www.wireguard.com/install/
- Linux kernel >= 3.10
- CentOS
  - $ sudo curl -Lo /etc/yum.repos.d/wireguard.repo https://copr.fedorainfracloud.org/coprs/jdoss/wireguard/repo/epel-7/jdoss-wireguard-epel-7.repo
  - $ sudo yum install epel-release
  - $ sudo yum install wireguard-dkms wireguard-tools
- FreeBSD
  - # pkg install wireguard

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
  - # 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
- Start interface
  - # ip link set up dev wg0

Setup by configuration

- ❑ Configuration file
  - /etc/wireguard/wg0.conf
- ❑ 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]
# client foo
PublicKey = [FOO's PUBLIC KEY]
PresharedKey = [PRE-SHARED KEY]
AllowedIPs = 10.113.0.1/32, 10.113.1.0/24

[Peer]
# client bar
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

Reference

- https://www.wireguard.com/
- https://www.wireguard.com/quickstart/
- https://wiki.archlinux.org/index.php/WireGuard