Wireguard Ubuntu Deployment

Installation

```
sudo add-apt-repository ppa:wireguard/wireguard   ### Not needed if you're using Ubuntu 20.04 or later
sudo apt install wireguard
```

Enabling IP Forwarding

```
sudo echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
sudo echo "net.ipv4.conf.all.proxy_arp = 1" >> /etc/sysctl.conf
sudo sysctl -p /etc/sysctl.conf
```

This equivalent to commenting the following 2 lines below in /etc/sysctl.conf file and then running `sudo sysctl -p`

```
net.ipv4.ip_forward = 1
net.ipv4.conf.all.proxy_arp = 1
```

Starting Wireguard & Making it a System Service

This is done so Wireguard always starts on system reboot

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

Opening Ports

If you're using UFW for your firewall open up the necessary ports for Wireguard. 51820 is the standard Wireguard port but feel free to use a non-standard port.

```
sudo ufw allow 22/tcp
sudo ufw allow 51820/udp
sudo ufw enable
sudo ufw status verbose
```

Server Configuration

Create a configuration file in `/etc/wireguard/wg0.conf`. An example configuration is below. If you need a private public key pair you can generate one in tunsafe (windows wireguard client).

```
[Interface]
Address = 10.xx.xx.1/24
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
ListenPort = 51820
PrivateKey = <Server's Private Key Here>
SaveConfig = true

[Peer]
PublicKey = <Client's Public Key Here>
AllowedIPs = 10.xx.xx.2/32

[Peer]
PublicKey = <Client's Public Key Heree>
AllowedIPs = 10.xx.xx.3/32
```

Server Config Explanation for [Interface]

Be aware that these `iptables` entries in PostUp & PostDown are for a given interface. Make sure that your **VM's interface** is captured on here you can check with `ip a`. In this above config example if you scroll right, you can see that the **VM's interface** is `eth0`. Additionally, and worth noting, also make sure that your **wireguard interface** also matches the reference on the `iptables` entry. In this above config example, the **wireguard interface** is `wg0`.

For Address = 10.xx.xx.xx/xx create and choose an arbitrary "Private IP address" different from other subnets on this VM's network to avoid IP conflict. Also specify the IP range you're going to use like /24 or /20 etc. You can use a program line tunsafe (Windows) to generate these keys or

you can use line 14+15 here.

SaveConfig = true / false. This setting when set to "true" will automatically save the current live config in standard format into your wg0.conf file whenever wireguard service is turned off. Because it is in standard format any comments you made to the wg0.conf file while be gone. Set this to false if you don't want this to happen. Set this to true if you'd like to add clients while the server is live without turning it off.

Server Config Explanation for [Peer]

For peer just keep incrementing your arbitrary IP address by one & use /32 because it is one IP. Then enter in their public key.

Finally start your wireguard service with...

```
sudo systemctl start wg-quick@wg0 ### to start wireguard server
sudo systemctl status wg-quick@wg0  ### to check wireguard server status
wg show ### alternative way to check wireguard server status
```

Adding Clients to Server

Use Method#1 if you're new. Method #2 and #3 are advanced.

Method #1: Editing After Turning Wireguard Off

```
sudo systemctl stop wg-quick@wg0
# Edit your /etc/wireguard/wg0.conf file and add the peers you need there
sudo systemctl start wg-quick@wg0
```

Method #2: While Wireguard Is Live (wg-quick save wg0)

Also requires `SaveConfig = true` in your config.

```
sudo wg set wg0 peer <Client Public Key> allowed-ips 10.X.X.X/32
sudo wg show
sudo systemctl restart wg-quick@wg0
route 10.X.X.X/32 wg0
```

The difference with using a wg-quick save is that you have to do the 4th command of route add which is easy to fat finger and screw things up.

Method #3: While Wireguard Is Live (Restarting Interface)

This method requires `SaveConfig = true` in your config.

Adding a peer (Changes not saved yet)

```
sudo wg set wg0 peer <Client Public Key> allowed-ips 10.X.X.X/32
```

Check if new peer's public key and ip shows up with

```
sudo wg
```

Finally do a

```
sudo systemctl restart wg-quick@wg0
```

Generating Client Configurations For Users

Example configuration. Please read the gotchas for each OS.

```
[Interface]
PrivateKey = < Client Private Key Here >
Address = 10.X.X.0/24
DNS = 8.8.8.8

[Peer]
PublicKey = < Server Public Key Here >
AllowedIPs = 0.0.0.0/0, ::/0
Endpoint = ServerPublicIPAdress:51820
PersistentKeepalive = 25
```

A couple of gotchas to note.

In **Linux**, the `Address =` line needs to end in /32.

In **Mac OS & Windows** the `Address =` lines needs to end in /24 or the subnet assigned.

Also in **Linux** the `DNS = line` cannot be there it has to be erased.

In **Mac OS** the `DNS = line` needs to be there otherwise client cannot browse Internet.

In **Windows Tunsafe** the `DNS = line` is optional. In **Windows Wireguard** the `DNS = line` is required.

Optional Configurations

Isolating Wireguard Clients From Each Other

This can be achieved with the following IP Tables command below assuming your wireguard interface is "wg0"

```
iptables -I FORWARD -i wg0 -o wg0 -j REJECT
```

Command References

```
sysctl net.ipv4.ip_forward             ### Verifies if IP Forward is working
sudo systemctl enable wg-quick@wg0     ### Makes Wireguard auto-start on boot
sudo systemctl start wg-quick@wg0     #Turn on Wireguard Interface
sudo systemctl stop wg-quick@wg0   #Turn off Wireguard Interface
sudo wg show        #Check if VPN tunnel is running

#command to remove client (peer)
wg set wg0 peer peer_pubkey remove

#Don't know if this command is needed after wg-quick save or removal of client
wg addconf wgnet0 <(wg-quick strip wgnet0)

### Generating Key Pairs ###
umask 077
wg genkey | tee privatekey | wg pubkey > publickey
# Key pairs are saved in same path you typed this command in
### End Generating Key Pairs ###
```

Revision #16
Created 2020-08-06 23:44:39 UTC by Admin
Updated 2021-05-16 02:09:00 UTC by Admin