Wireguard OpenBSD client

In this post, I will be installing Wireguard on my OpenBSD laptop to be able to connect to my personal services over a secure tunnel.

OpenBSD client setup

Install Wireguard

Wireguard tools are officially included in the OpenBSD repository, but are usually a bit outdated. To install them, type:

```
$ doas pkg_add wireguard-tools
```

> As usual, OpenBSD provides excellent documentation about Wireguard (`man wg`), use it if necessary.

Prepare directories

1. Switch to `root` so you don't have to type `doas` over and over again, also the config directory will only be readable by `root`.

```
$ doas su
```

2. Set `umask` to `077` to allow `rw` access to `root` only.

```
(root)$ umask 077
```

3. Create the config folder and its subdirectories.

```
(root)$ mkdir /etc/wireguard/{keys,psk}
```

Generate keys

1. Move to the `keys` directory and generate client's public and private key. You will put the **public key** to the **server config** later, **private key will never leave the device.**

```
(root)$ cd /etc/wireguard/keys
(root)$ wg genkey | tee wg0_private.key | wg pubkey > wg0_public.key
```

The `wg genkey` command generates a random *private* key in base64 and prints it to standard output (terminal). The output is instead redirected to `tee`, which both prints it to stdout (terminal), but also saves it into a file `wg0_private.key`. The private key printed to stdout is then piped (`|` symbol) to `wg pubkey`, which calculates the public key and prints it in base64 to stdout from a corresponding private key (the one we redirected to it with the pipe), lastly redirect the public key from stdout to a file `wg0_public.key`

You will now have two files in `/etc/wireguard` directory. One containing public, the other private key.

```
wg0_private.key  wg0_public.key
```

2. Create client config file (`wg0.conf`) in `/etc/wireguard`. Make sure you still have `umask` set to `077`.

```
(root)$ touch wg0.conf
```

3. Open the file and make it look like this. Replace IPs with the ones you are planning to use. `[Peer]` section specifies the server's part of config. `AllowedIPs` should point to the interal IP of the server within the Wireguard tunnel. `PublicKey` should contain the servers public key. `PresharedKey` will be generated on the server in a moment. Actually, you can generate this key on the client and then copy it to the server or vice versa, it's up to you. Either way, because it's a *shared secret*, it has to be present in both configuration files – on the server and the client.

```
[Interface]
PrivateKey = GeneratedPrivateKey_from_wg0_private.key
Address = 10.20.20.5/29

[Peer]
PublicKey =
PresharedKey =
AllowedIPs = 10.20.20.1/32
Endpoint = publicIP:port
```

Server setup

We assume that the server is already set up and we are just adding a new client. For a guide how to setup a server, head over here.

1. Go to `/etc/wireguard/psk` and generate the preshared key. Ideally, switch to `077` `umask` again.

```
(root)$ cd /etc/wireguard/psk
(root)$ wg genpsk > openbsd_client.psk
```

2. Add the following `[Peer]` block to the server `wg0.conf`. Replace `PublicKey` with the client public key (`wg0_public.key`). Copy the generated preshared key from `openbsd_client.psk` to `PresharedKey` in **both the server and client config.**

```
[Peer]
PublicKey =
PresharedKey =
AllowedIPs = 10.20.20.5/32
```

Start the Wireguard interface

1. Go back to the OpenBSD client and bring up the Wireguard interface manually:

```
$ doas wg-quick up wg0
```

2. Now check `ifconfig` and confirm connectivity to the server.

Start Wireguard on boot

In order to start Wireguard automatially after boot you need to create a `rc.d` service script that allows you to not only start the service on boot, but also start and stop it on demand.

On OpenBSD the systems daemon configuration database is located in `rc.conf` and `rc.conf.local` file.

> It is recommended to leave `rc.conf` untouched and instead create and edit a new `rc.conf.local` file.

Create script

> **IMPORTANT!** The following scripts were made by **Mark Vainomaa** `<mikroskeem@mikroskeem.eu>`, **not me**. The original script can be found here.

1. Open `/etc/rc.conf.local` file and add a new line containing `wg_quick_flags=wg0` – `wg0` is the name of the interface.

rc.conf.local

```
apmd_flags=-A
sshd_flags=NO
xenodm_flags=
wg_quick_flags=wg0
```

2. Create new file `wg_quick` in `/etc/rc.d`. Paste the following script to it. It has a few comments to make it easier to understand.

wg_quick

```
#!/bin/ksh
#
# Author: Mark Vainomaa <mikroskeem@mikroskeem.eu>

daemon="/usr/local/bin/wg-quick"

rc_reload=NO

. /etc/rc.d/rc.subr

pexp_wg="^/usr/local/bin/bash ${daemon} up ${daemon_flags}"
pexp_route="route -n monitor"

rc_check() {
    # Check the pid of wg-quick script whose parent is init
    pid="$(pgrep -P 1 -f "${pexp_wg}")"
    if [ -z "${pid}" ]; then
        return 1
    fi

    # Check for a stale wg-quick script process handling
    # only `route -n monitor`
    if [ ! -z "$(pgrep -P "${pid}" -f "${pexp_route}")" ]; then
        # TODO: uh-oh, stale wg-quick & route monitor; what should we do?
        return 1
    fi

    return 0
```

```
}

rc_start() {
    ${daemon} up ${daemon_flags}
}

rc_stop() {
    ${daemon} down ${daemon_flags}

    # TODO: remove this when wg-quick gets fixed
    #       this is here to kill 'route -n monitor' which
    #       keeps wg_quick rc script blocked on this function
    pid="$(pgrep -P 1 -f "${pexp_wg}")"
    pid2="$(pgrep -P "${pid}" -f "${pexp_wg}")"
    rpid="$(pgrep -P "${pid2}" -f "${pexp_route}")"

    # Get rid of the stale `route -n monitor' process
    if [ ! -z "${rpid}" ] && kill -0 "${rpid}"; then
        kill -15 "${rpid}"
    fi
}

rc_pre() {
    # Error out if flags are empty
    if [ -z "${daemon_flags}" ]; then
        echo "ERROR: daemon flags cannot be empty and must contain WireGuard tunnel configuration name!"
        return 1
    fi

    # Pass flags through basename so users couldn't do
    # something like `../../x' and other dumb stuff.
    configname="$(basename ${daemon_flags})"

    # Check if WireGuard's configuration file exists
    if [ ! -f "/etc/wireguard/${configname}.conf" ]; then
        echo "ERROR: file \`/etc/wireguard/${configname}.conf' does not exist!"
        return 1
    fi

    # Check if tunnel with given name is already running
```

```
    if [ -f "/var/run/wireguard/${configname}.name" ]; then
        echo "ERROR: tunnel \`${configname}' is already running!"
        return 1
    fi
}

rc_cmd $1
```

3. Adjust permissions to the script – set them to `555` (`rx-rx-rx`)

```
$ doas chmod 555 /etc/rc.d/wg_quick
```

Test script

1. You can now test the script to see if it starts Wireguard.

```
$ doas rcctl start wg_quick
wg_quick(ok)
```

2. Also try to stop the service (one of the problems I ran into when I tried to simplify the script)

```
$ doas rcctl stop wg_quick
wg_quick(ok)
```

> Keep in mind that Wireguard runs as `root`, therefore you need elevated privileges to start or stop the script.

Enable wg_quick

1. To make Wireguard start on boot, enable the service script we have just created.

```
$ doas rcctl enable wg_quick
```

2. Check that wg_quick is actually enabled.

```
$ rcctl ls on
cron
...
wg_quick
xenodm
...
```

That's it, now reboot your device to test if Wireguard starts and you are done.

Revision #18
Created 5 October 2021 00:05:01 by Marek
Updated 10 October 2021 02:44:19 by Marek