Inhaltsverzeichnis

Testaufbau 3
Installieren und kompilieren auf alten Linux Kernel 3
installation auf dem Raspi 3
Installation auf pfSense 4
testen vom Raspi aus 5

Wireguard

Nachdem ich heute endlich mal das Whitepaper von Wireguard gelesen habe, musste ich Wireguard gleich mal ausprobieren. Es ist supereinfach damit ein funktionierendes VPN hinzubekommen. Und weil es ein Kernelmodul ist, enfällt das hin- und herschieben der Pakete wie bei OpenVPN. Das macht die Verbindung massiv schneller.

Testaufbau

Client soll ein Raspi sein, im WLAN. Als Server nehmen wir pfSense[1] damit wir unterschiedliche Kernel-Versionen haben.

Ich habe als Netzwek 10.0.8.0/24 genommen, das Raspi hat die ip 10.0.8.9 und die pfSense hat 10.0.8.8 im Beispiel.

Installieren und kompilieren auf alten Linux Kernel

Da ich noch einige alte Kernel habe, hier dann die Anleitung, wie es auf einem alten Linux installiert werden kann. Dazu muss der Kernel selber übersetzt werden. Dazu brauchen wir eine Build Umgebung:

```
 sudo apt-get install libelf-dev linux-headers-$(uname -r) build-essential
pkg-config
```

Wireguard Kernelmodule herunterladen und übersetzen

```
git clone https://git.zx2c4.com/wireguard-linux-compat
make -C wireguard-linux-compat/src -j$(nproc)
```

der Abschnitt "Installieren und kompilieren auf alten Linux Kernel" ist noch nicht vollständig und wird später ergänzt

.:.

installation auf dem Raspi

Das Kernelmodul ist schon integriert, kann mit `modprobe wireguard && echo "Wireguard KernelModule geladen"`[2] getestet werden. Die Userspace Tools fehlen, diese werden mit

```
sudo apt install wireguard wireguard-tools
```

installiert. Danach sollte das Programm `wg` bereits im Pfad sein. Ohne Argumente ausgeführt, zeigt es

die konfigurierten Interfaces an (haben wir noch nicht)

Einen neuen Public- und Private- Key generieren geht mit

```
wg genkey | tee /etc/wireguard/private.key | wg pubkey >
/etc/wireguard/public.key
```

Konfigurieren des Interfaces mit

```
wg set wg0 listen-port 1234 private-key /etc/wireguard/private.key peer \
<Public-Key von pfSense> allowed-ips 0.0.0.0/0 endpoint <IP von
pfSense>:1234
```

Damit hören wir auf UDP Port 1234 für ankommende Verbindungen, auf pfSense nehmen wir mal den selben Port. <IP von pfSense> ist dabei die IP im lokalen Netz, also die IP unter der wir pfSense jetzt schon erreichen können.

Installation auf pfSense

Auf pfSense unter `System` → `Package Manager` → `Available Packages` nach `WireGuard` suchen und auf `+ Install` klicken. (Die Warnung, das die Software noch nicht stabil sei, ignorieren wir mal) Danach unter VPN → `WireGuard` → `Tunnel` ein neuer Tunnel anlegen mit `+ Add Tunnel`

```
Enable          [ x ]
Description     [ wireguard-network ]
Listen Port     [ 1234 ]
Interface Keys  [ ........................... ] [
................................ ]   [ Generate ]
                 Private key for this tunnel     Public key for this tunnel
(copy)      New Keys
```

Auf `Generate` new Keys klicken um neue Schlüssel anzulegen, `Enabled` anwählen, ein Name geben und den Port setzen. Danach unten auf `Save Tunnel` klicken und die Änderungen anwenden. Erst jetzt bekommen wir ein Netzwerkinterface `wg0`, dieses unter Interfaces hinzufügen, eine IP für die Firewall vergeben. Achtung: Die Firewall filtert auf "WireGuard" und nicht auf dem Interface "wg0"[3)]

Also dann unter `Firewall` → `Rules` → `Wireguard` die Firewall zum testen öffnen. Ich lasse z.B. ICMP Traffic zu, damit Ping funktioniert.

Nun noch unter VPN → `Wireguard` → `Tunnels` → `tun_wg0` unser Raspi als Peer hinzufügen, mit einem Klick auf `+ Add Peer` unter `Actions`.

Dann brauchen wir den PublicKey vom Raspi `cat /etc/wireguard/public.key`, der Pre-shared Key freilassen und unter `Allowed IPs` eintragen mit welchen IPs das Raspi daherkommen darf. (ich nehme mal die IP welche ich dem Raspi im Wireguard-Netz gegeben habe und sonst nix.[4)]

`Save Peer` und die Änderungen anwenden, und dann geht zum testen.

testen vom Raspi aus

Erst mal die Netzwerkkarte wie gewohnt konfigurieren.

```
ip address add dev wg0 10.0.8.9/24
ip link set up dev wg0
```

Bei meinem ersten Test hat's nicht funktioniert → Firewallregeln auf der pfSense! Müssen am richtigen Ort eingetragen werden.

```
root@raspberrypi:~# ping 10.0.8.8
PING 10.0.8.8 (10.10.8.8) 56(84) bytes of data.
From 10.0.8.9 icmp_seq=1 Destination Host Unreachable
ping: sendmsg: Der notwendige Schlüssel ist nicht verfügbar
```

Aber was wir schön sehen ist die Fehlermeldung. Neben der ICMP "Destination unreachable" bekommen wir vom KernelTreiber zusätzlich die -ENOKEY Fehlermeldung das kein Schlüssel für pfSense verfügbar ist.

Da war mein Fehler, dass ich für pfSense eine falsche `allowed-ips` eingetragen hatte:

```
wg set wg0 listen-port 1234 private-key /etc/wireguard/private.key peer \
<Public-Key von pfSense> allowed-ips 0.0.0.0 endpoint <IP von pfSense>:1234
```

Findest du den Fehler?

Danach funktioniert's von beiden Seiten her.

Soll nun vom Raspi über den Wireguard Tunnel in ein Netz hinter der pfSense Firewall geroutet werden, so muss dieses Netz auf dem Raspi auch in die Routing Tabelle eingetragen werden.

- IPs und Netzwerk einfacher abbilden
- Unterschied vom Tunnel Netzwerk und externem Netzwerk einfacher darstellen
- Beispiel für Routing Tabelle im Raspi hinzufügen

1)
obwohl das KernelModule für Free BSD im pfSense Repo "veraltet" ist, sollte die Verbindung doch möglich sein
2)
wenn's funktioniert, dann kommt keine Fehlermeldung und es steht "Wireguard KernelModule geladen"
3)
Die Firewall filtert nicht auf dem Interface, (genau wie bei OpenVPN) ich finde das verwirrend
4)
es können aber auch ganze Netze angegeben werden, welche durch das Raspi weitergeletet (geroutet) werden.

From:
https://aha-it.ch/wiki/ - **AHa-IT**

Permanent link:
https://aha-it.ch/wiki/lx/net/wireguard?rev=1668057577

Last update: **10.11.2022 05:19**