Part 5 – Adjusting Firewall (iptables)

Adjusting Firewall rules

Everything regarding the Wireguard VPN should be configured:

- Wireguard tunnel between a server and 2 peers (clients)
- Services running on the Wireguard interface only (Nginx)
- Correct DNS resolving for both the server and Android client

Now it's time to fix the Firewall, so what are our goals here?

- DROP everything except:
  - HTTP/S and DNS on Wireguard interface
  - Ping on all interfaces
  - SSH on Wireguard interface only (later, since SSH is our only access to the VPS and breaking it would cause immense headache)

I am using very powerful `iptables` with `iptables-restore` utility to write my rules to a config file instead to the command line. The config file is located in `/etc/iptables/rules.v4` (may be different in your case).

Open the file and edit it. Everything will be explained in the following code block behind #. For easier reading, I have divided it into parts:

Setup the 3 main chains – `INPUT`, `OUTPUT` and `FORWARD`. It is recommended to always set `INPUT` as implicit `DENY` and only whitelist approved services and ports.

```
*filter
# CONFIGURE INPUT, OUTPUT AND FORWARD CHAINS
# Drop forwarded traffic, we don't need that since we are not acting as a router
:FORWARD DROP [0:0]

# Accept all outgoing connections
:OUTPUT ACCEPT [0:0]
```

```
# Block all incoming traffic (default DENY)
:INPUT DROP [0:0]
```

When everything is blocked by default, these rules have to exist to allow communication on localhost and any established connections (when you initiace connection from the server)

```
# BASIC RULES FOR THINGS TO WORK WITH DEFAULT DENY ON INPUT CHAIN
# Do not block localhost traffic to itself
-A INPUT -i lo -j ACCEPT

# Allow established and related incoming connections
-A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Rules to allow devices within the Wireguard tunnel to access web services (ports `80` and `443`) + `53` for DNS.

```
# WIREGUARD INTERFACE CONFIG
# Allow web traffic for my search engine (only on the Wireguard interface)
-A INPUT -p tcp -m tcp --dport 80 -i wg0 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -i wg0 -j ACCEPT

# Allow DNS queries on the Wireguard interface
-A INPUT -p udp -m udp --dport 53 -i wg0 -j ACCEPT
```

Rules for the public IP of the server. The only thing that needs to be open is the Wireguard port. You can also configure the server to accept `ICMP` ping requests for easier troubleshooting. Without this rule, the server won't be "pingable" even if it's up.

```
# PUBLIC IP RULES
# Allow ICMP pings on all interfaces (for easier troubleshooting)
-A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Allow Wireguard on the public IP
-A INPUT -p udp -m udp --dport 51895 -i eth0 -j ACCEPT
```

`SSH` is the most critical service and therefore will still be available on all interface and I will move it to the Wireguard interface after testing.

```
# SSH
# Allow SSH on the port that it's running on
-A INPUT -p tcp -m tcp --dport 7985 -j ACCEPT
```

```
# Commit rules
COMMIT
```

The entire config looks like this:

```
*filter
# CONFIGURE INPUT, OUTPUT AND FORWARD CHAINS
# Drop forwarded traffic, we don't need that since we are not acting as a router
:FORWARD DROP [0:0]

# Accept all outgoing connections
:OUTPUT ACCEPT [0:0]

# Block all incoming traffic (default DENY)
:INPUT DROP [0:0]

# BASIC RULES FOR THINGS TO WORK WITH DEFAULT DENY ON INPUT CHAIN
# Do not block localhost traffic to itself
-A INPUT -i lo -j ACCEPT

# Allow established and related incoming connections
-A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# WIREGUARD INTERFACE CONFIG
# Allow web traffic for my websites (only on the Wireguard interface)
-A INPUT -p tcp -m tcp --dport 80 -i wg0 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -i wg0 -j ACCEPT

# Allow DNS queries on the Wireguard interface
-A INPUT -p udp -m udp --dport 53 -i wg0 -j ACCEPT

# PUBLIC IP RULES
# Allow ICMP pings on all interfaces (for easier troubleshooting)
-A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Allow Wireguard on the public IP
-A INPUT -p udp -m udp --dport 51895 -j ACCEPT

# SSH
# Allow SSH on the port that it's running on
```

```
-A INPUT -p tcp -m tcp --dport 7985 -j ACCEPT

# Commit rules
COMMIT
```

Apply the configuration with `iptables-restore`. First make sure all rules make sense to you.

```
$ sudo iptables-restore /etc/iptables/rules.v4
```

List and check the rules (`-n` to not resolve DNS and `-v` for verbose output)

```
$ sudo iptables -L -nv

Chain INPUT (policy DROP 32 packets, 1941 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
   47  2876 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  wg0    *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  wg0    *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
    0     0 ACCEPT     udp  --  wg0    *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
    2   168 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmptype 8
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:51895
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:7985

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 34 packets, 5056 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Revision #3
Created 22 September 2021 01:52:51 by Marek
Updated 22 September 2021 01:59:16 by Marek