Homework 2 Report - Group number 27

Name	Student ID	email
Simone di Biasio	1823213	dibiasio.1823213@studenti.uniroma1.it
Leonardo Persiani	1795525	persiani.1795525@studenti.uniroma1.it

0. Brainstorming

In order to implement the requested VPNs we decided to use `wireguard` for two main reasons:

1. `OPNsense` provides a `wireguard` plugin with a GUI that allows to easiliy setup a tunnel
2. Since it offers minimal customisation (e.g. in choosing cryptographic algorithms), it's fast and simple to configure compared to openvpn or IPSec.

The idea is to use the `OPNsense` plugin to create the site-to-site tunnel and the command line interface to setup the roadwarrior tunnel; then, add the required firewall rules to allow VPN traffic.

1. Road Warriors VPN

We first installed the `wireguard` package on the `proxy` machine and on each client (in this case our main machines) via `apt`, and then generated the configuration files needed as explained below.

We assigned the clients addresses according to the given IPv4 subnet and using IPv6 network `2001:470:b5b8:1b32::/64` as follows:

- Proxy Endpoint: `100.100.253.1, 2001:470:b5b8:1b32::1`
- Becca: `100.100.253.2, 2001:470:b5b8:1b32::2`
- Huck: `100.100.253.3, 2001:470:b5b8:1b32::3`
- Jim: `100.100.253.4, 2001:470:b5b8:1b32::4`

For each involved host we had to generate public and private keys using wireguard builtin commands `wg genkey` and `wg pubkey`: the clients will receive the server's public key and the server will have all the clients public keys.

All these things, along with server's private key and clients' public keys, are specified in the config file of the server's wireguard interface.

1.1 Proxy Server endpoint

In order to configure the proxy server as a vpn endpoint, we enabled forwarding:

- `sysctl -w net.ipv4.ip_forward=1`
- `sysctl -w net.ipv6.conf.all.forwarding=1`

Additionally, we had to use NAT for traffic coming from the tunnel and directed towards internal networks: this can be easily accomplished using `iptables`.

The server's config file will hold details about its interface (private key, address, port and command to run before/after bringing up/down the interface) and about its clients (public key and their address).

```
[Interface]
Address = 100.100.253.1/24, 2001:470:b5b8:1b32::1/64
PrivateKey = mJ429hxBv3HNqPkcxlp+4t+HPn9YkWsu9biUKMadAWY=
ListenPort = 51820 # default port
PostUp = iptables -t nat -A POSTROUTING -s 100.100.253.0/24 ! -d
100.100.253.0/24 -o eth0 -j MASQUERADE && ip6tables -t nat -A POSTROUTING -
s 2001:470:b5b8:1b32::/64 ! -d 2001:470:b5b8:1b32::/64 -o eth0 -j
MASQUERADE
PostDown = iptables -t nat -D POSTROUTING -s 100.100.253.0/24 ! -d
100.100.253.0/24 -o eth0 -j MASQUERADE && ip6tables -t nat -D POSTROUTING -
s 2001:470:b5b8:1b32::/64 ! -d 2001:470:b5b8:1b32::/64 -o eth0 -j
MASQUERADE

# BEGIN_PEER becca
[Peer]
PublicKey = AMNxf90hnA/fp0UcPrvdjixr4W0jxeNrR7Hlzgr4uxk=
PresharedKey = WykcgLVnt/6EQi+h70tiEypdQ7sdK43S37Tm7lavkII=
AllowedIPs = 100.100.253.2/32, 2001:470:b5b8:1b32::2/128
# END_PEER becca

# BEGIN_PEER huck
[Peer]
PublicKey = CcO9tvfIuLj3nWdMDurybek1t4TNVJQ8ci2ahyNW5ng=
PresharedKey = WHuYnigYf8jq/M9t65EOHX7+VsjTAf10H9+go1rfGgM=
AllowedIPs = 100.100.253.3/32, 2001:470:b5b8:1b32::3/128
# END_PEER huck

# BEGIN_PEER jim
[Peer]
PublicKey = 0j4OrDuyanFTMkHxdhF320MahmYLvltfpjNXmOMonwU=
PresharedKey = bR2U+XqigG/jEAdjn6tXAr5ILqe2rP1kOG2LyBLAWl8=
AllowedIPs = 100.100.253.4/32, 2001:470:b5b8:1b32::4/128
# END_PEER jim
```

1.1.1 IPv6

To make the tunnel work with IPv6 as well, we had to assign a static ip address to the proxy server via `interfaces` file specifying also the gateway and the dns server (the proxy ignores the router advertisements since `net.ipv6.conf.all.forwarding = 1`).

```
iface eth0 inet6 static
    address 2001:470:b5b8:1b06:8410:9bff:fe35:525c
    netmask 64
    gateway fe80::44fd:65ff:fe1b:8895
    dns-nameservers 2001:470:b5b8:1b11:4c34:16ff:fe3d:beb3
```

1.2 Clients (/etc/wireguard/CLIENT.conf)

In the client's config file we had to specify the interface address, the server's endpoint, the DNS and the networks reachable through the tunnel, which are the tunnel itself, the Internal Servers Network and the Clients Network.

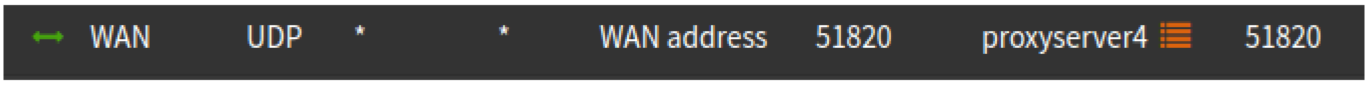
We can see here Becca's config file:

```
# Becca
[Interface]
Address = 100.100.253.2/24, 2001:470:b5b8:1b32::2/64
DNS = 100.100.1.2, 2001:470:b5b8:1b11:4c34:16ff:fe3d:beb3
PrivateKey = 6ANRR7xR9mLa/8ZUyKslXMHGe3c2GNgGLzayGCZiEkE=

[Peer]
PublicKey = Ha3oK4D4E1IW7DiPrUQqSUYtnFgueFn7Nd1opGZeY1s=
PresharedKey = WykcgLVnt/6EQi+h70tiEypdQ7sdK43S37Tm7lavkII=
AllowedIPs = 100.100.253.0/24, 100.100.1.0/24, 100.100.2.0/24,
2001:470:b5b8:1b32::0/64, 2001:470:b5b8:1b11::0/64,
2001:470:b5b8:1b12::0/64
Endpoint = 100.100.0.2:51820
PersistentKeepalive = 25
```

The configuration files are placed inside `/etc/wireguard` directory of the corresponding host.

1.3 Firewall rule

Before bringing up the tunnel we had to enable port forwarding on port UDP 51820 towards the `proxyserver`:

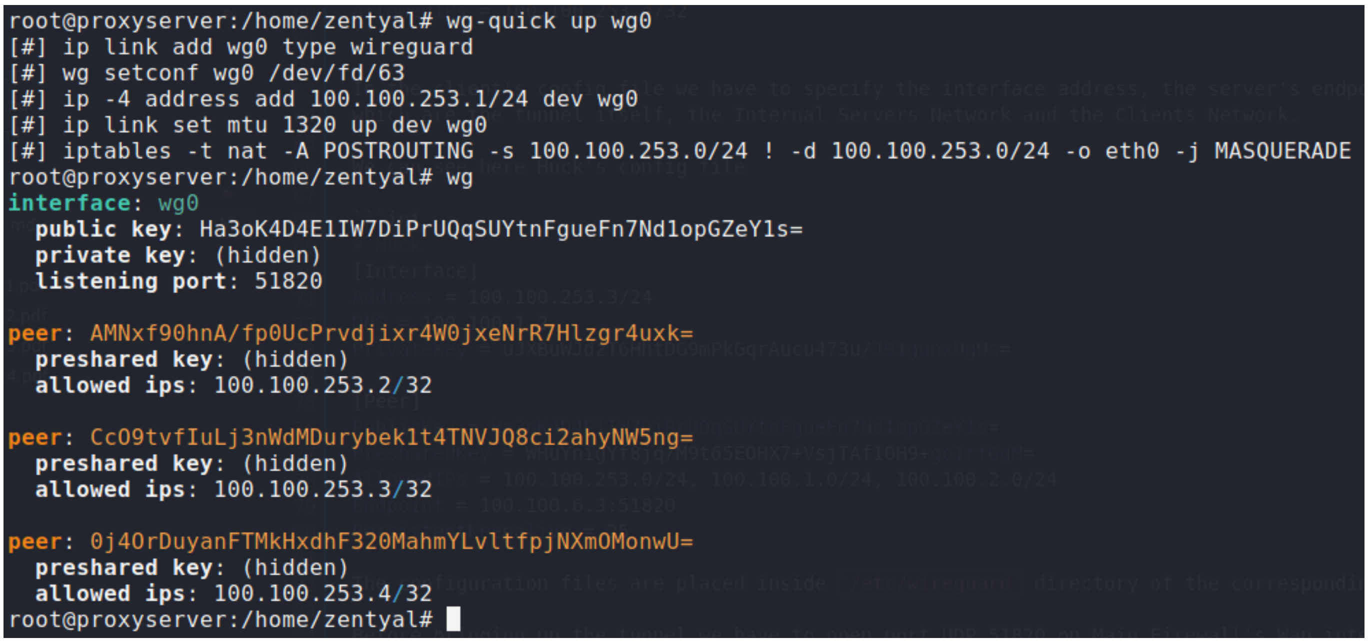


```
WAN    UDP    *    *    WAN address    51820    proxyserver4    51820
```

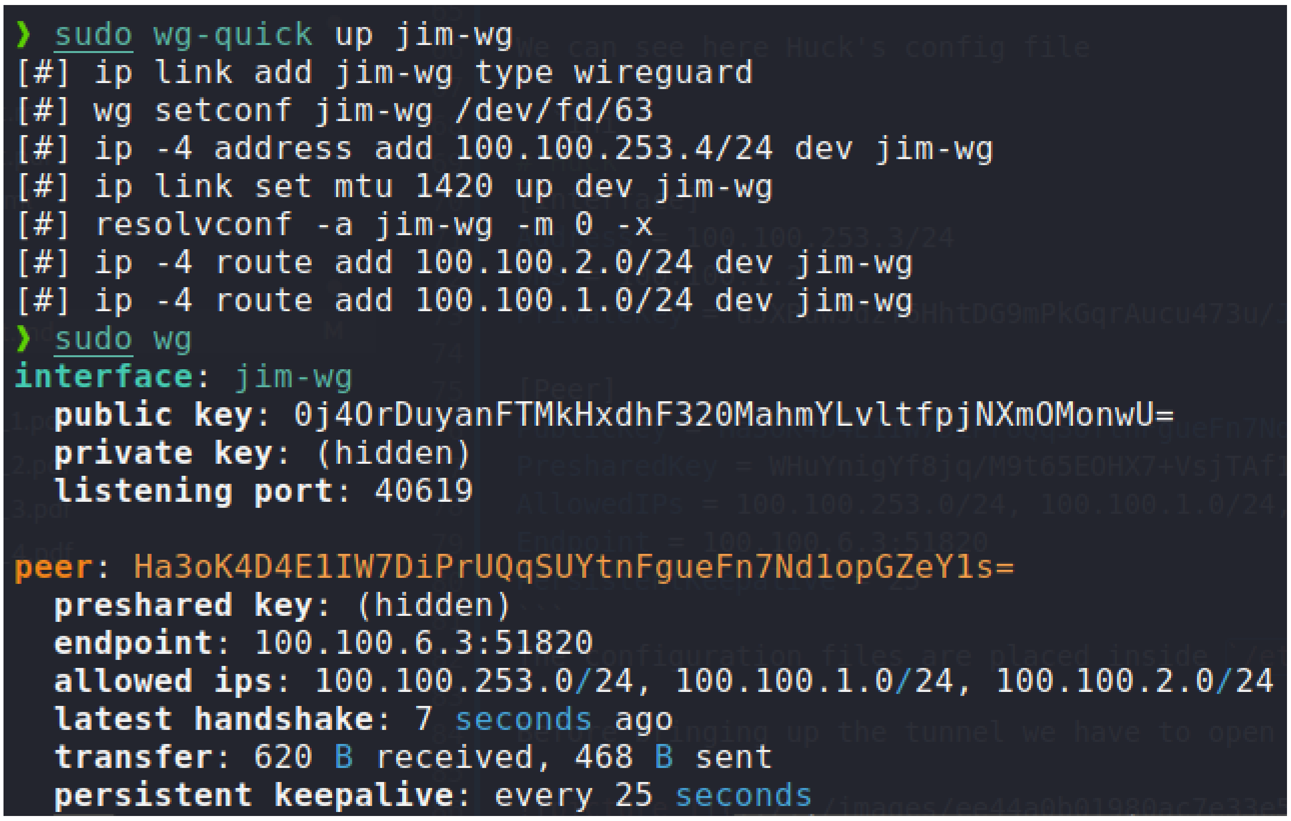
The Main Firewall will act as the apparent VPN endpoint, but it will actually forward VPN traffic to the real VPN endpoint.

1.4 Activation of the tunnel

Finally, we ran `wg-quick up wg0` on the server and `wg-quick up CLIENT-wg` on the clients to create the tunnel:



```
root@proxyserver:/home/zentyal# wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 100.100.253.1/24 dev wg0
[#] ip link set mtu 1320 up dev wg0
[#] iptables -t nat -A POSTROUTING -s 100.100.253.0/24 ! -d 100.100.253.0/24 -o eth0 -j MASQUERADE
root@proxyserver:/home/zentyal# wg
interface: wg0
  public key: Ha3oK4D4E1IW7DiPrUQqSUYtnFgueFn7Nd1opGZeY1s=
  private key: (hidden)
  listening port: 51820

peer: AMNxf90hnA/fp0UcPrvdjixr4W0jxeNrR7Hlzgr4uxk=
  preshared key: (hidden)
  allowed ips: 100.100.253.2/32

peer: Cc09tvfIuLj3nWdMDurybek1t4TNVJQ8ci2ahyNW5ng=
  preshared key: (hidden)
  allowed ips: 100.100.253.3/32

peer: 0j4OrDuyanFTMkHxdhF320MahmYLvltfpjNXmOMonwU=
  preshared key: (hidden)
  allowed ips: 100.100.253.4/32
root@proxyserver:/home/zentyal# 
```



```
❯ sudo wg-quick up jim-wg
[#] ip link add jim-wg type wireguard
[#] wg setconf jim-wg /dev/fd/63
[#] ip -4 address add 100.100.253.4/24 dev jim-wg
[#] ip link set mtu 1420 up dev jim-wg
[#] resolvconf -a jim-wg -m 0 -x
[#] ip -4 route add 100.100.2.0/24 dev jim-wg
[#] ip -4 route add 100.100.1.0/24 dev jim-wg
❯ sudo wg
interface: jim-wg
  public key: 0j4OrDuyanFTMkHxdhF320MahmYLvltfpjNXmOMonwU=
  private key: (hidden)
  listening port: 40619

peer: Ha3oK4D4E1IW7DiPrUQqSUYtnFgueFn7NdlopGZeY1s=
  preshared key: (hidden)
  endpoint: 100.100.6.3:51820
  allowed ips: 100.100.253.0/24, 100.100.1.0/24, 100.100.2.0/24
  latest handshake: 7 seconds ago
  transfer: 620 B received, 468 B sent
  persistent keepalive: every 25 seconds
```

2. Site to Site VPN

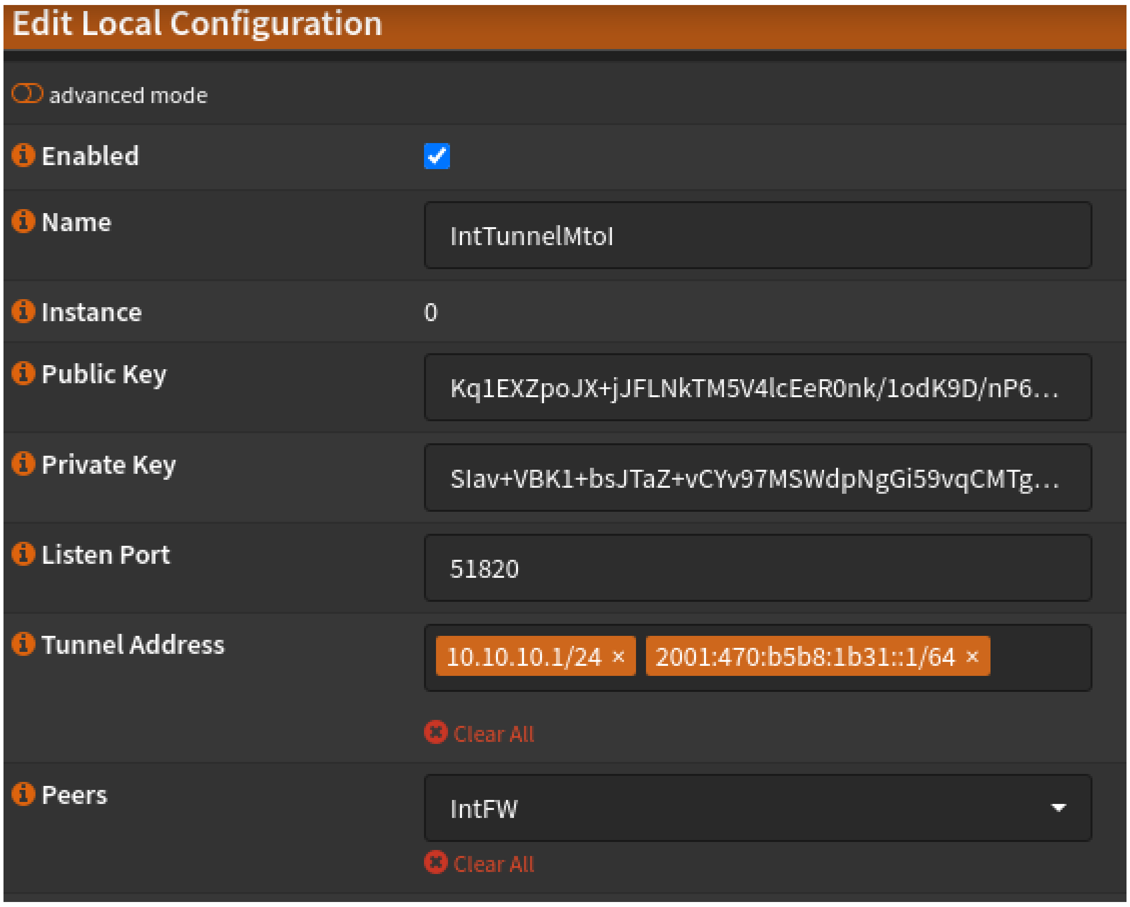
For the site to site VPN we took advantage of `OPNsense`'s package which presents a useful Graphical Interface.

The procedure was pretty much the same on both routers, so we'll just go through the main firewall's setup. For more details, here's the opnsense tutorial we followed.

In this case no tunnel network was provided, so we chose `10.10.10.0/24` for IPv4 and `2001:470:b5b8:1b31::/64` for IPv6 and assigned:

- `10.10.10.1`, `2001:470:b5b8:1b31::1` to Main Firewall's tunnel interface
- `10.10.10.2`, `2001:470:b5b8:1b31::2` to Internal Firewall's tunnel interface

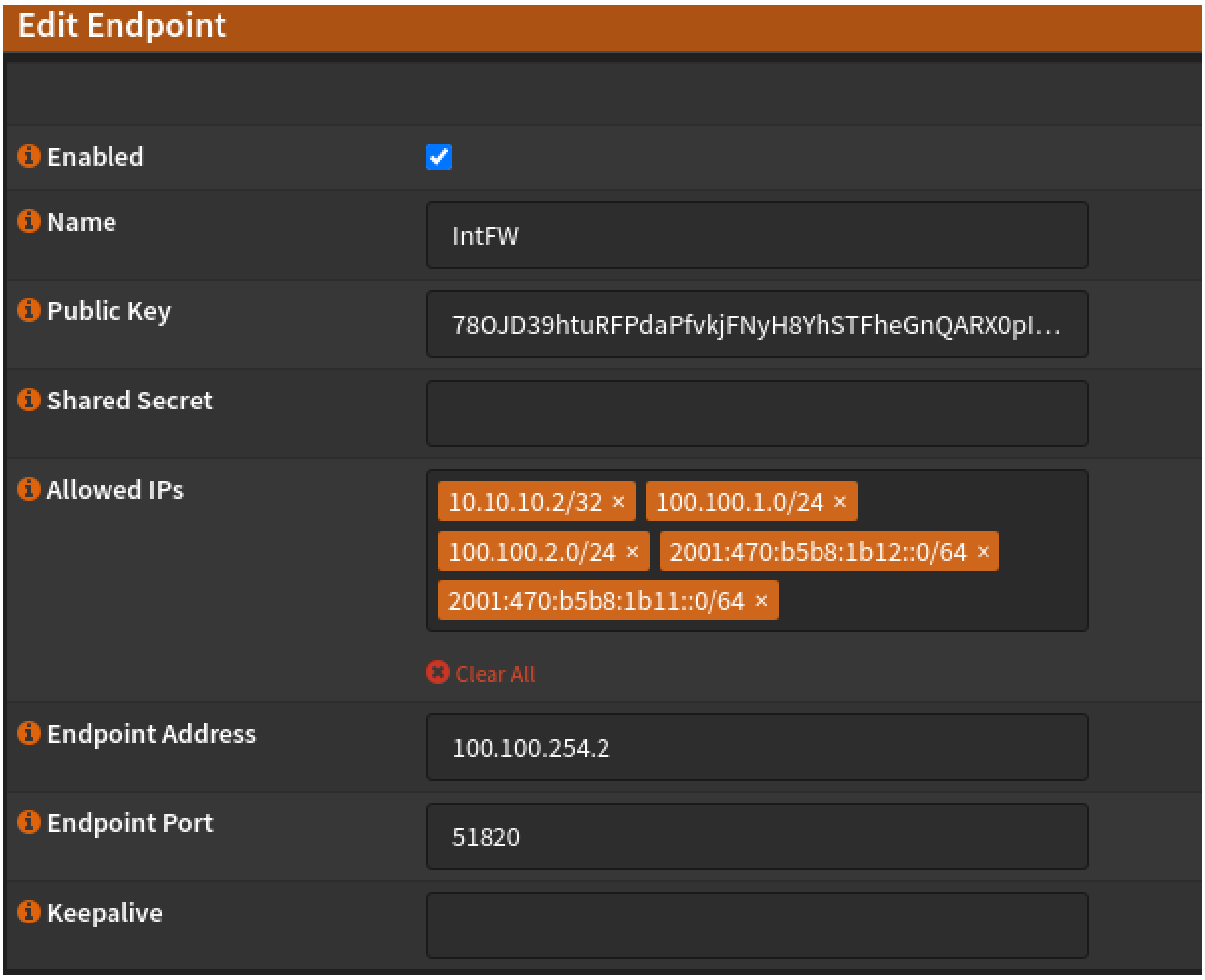
After installing the wireguard package, we added a new Local configuration specifying the tunnel address and the peer, while public and private keys were automatically generated:


Edit Local Configuration

advanced mode

Enabled	✓
Name	IntTunnelMtoI
Instance	0
Public Key	Kq1EXZpoJX+jJFLNkTM5V4lcEeR0nk/1odK9D/nP6…
Private Key	Slav+VBK1+bsJTaZ+vCYv97MSWdpNgGi59vqCMTg…
Listen Port	51820
Tunnel Address	10.10.10.1/24 × 2001:470:b5b8:1b31::1/64 ×
Peers	IntFW

Then we created a new endpoint to connect to, including its public ip address (Internal Firewall's external interface) and the subnets we can reach through the tunnel, which are Internal Servers Network and Clients Network

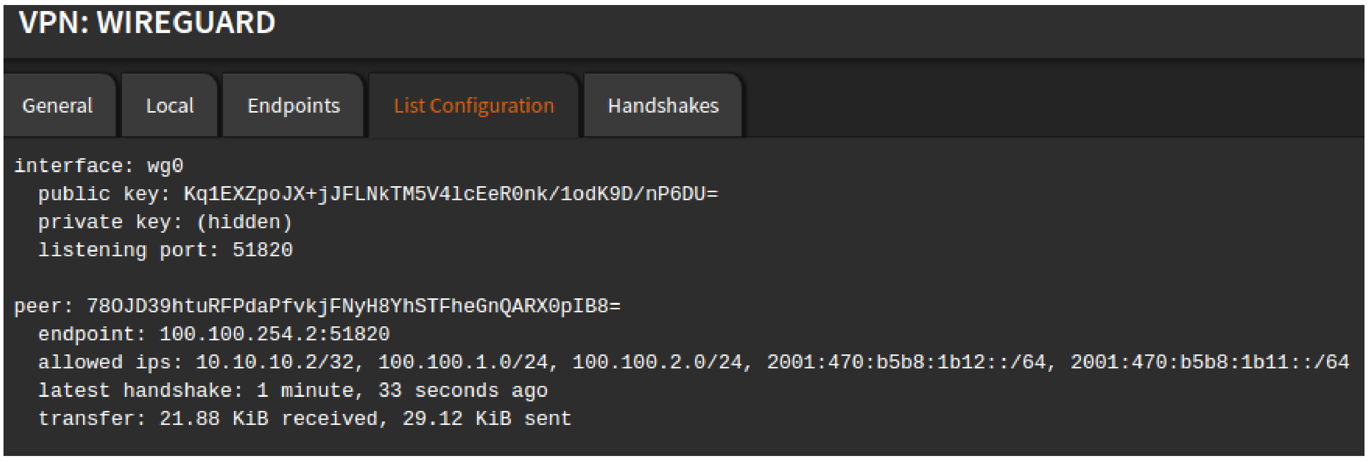


Edit Endpoint

Enabled	✓
Name	IntFW
Public Key	78OJD39htuRFPdaPfvkjFNyH8YhSTFheGnQARX0pI...
Shared Secret	
Allowed IPs	10.10.10.2/32 × 100.100.1.0/24 × 100.100.2.0/24 × 2001:470:b5b8:1b12::0/64 × 2001:470:b5b8:1b11::0/64 ×
Endpoint Address	100.100.254.2
Endpoint Port	51820
Keepalive	

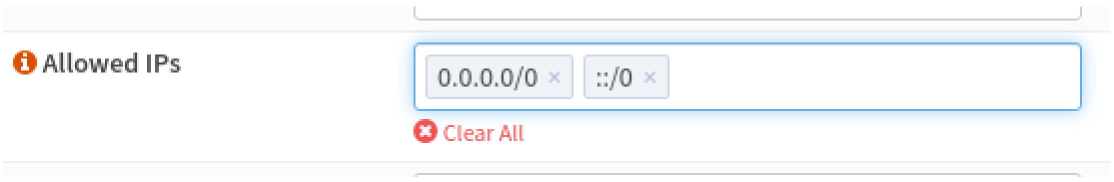
Routes introduced by the VPN were conflicting with preexisting ones and the interface wouldn't enable itself: since the traffic had to pass through the tunnel, we deleted the old routes and kept the new ones.

We finally enabled the wireguard plugin and the tunnel was created:



```
interface: wg0
  public key: Kq1EXZpoJX+jJFLNkTM5V4lcEeR0nk/1odK9D/nP6DU=
  private key: (hidden)
  listening port: 51820

peer: 78OJD39htuRFPdaPfvkjFNyH8YhSTFheGnQARX0pIB8=
  endpoint: 100.100.254.2:51820
  allowed ips: 10.10.10.2/32, 100.100.1.0/24, 100.100.2.0/24, 2001:470:b5b8:1b12::/64, 2001:470:b5b8:1b11::/64
  latest handshake: 1 minute, 33 seconds ago
  transfer: 21.88 KiB received, 29.12 KiB sent
```

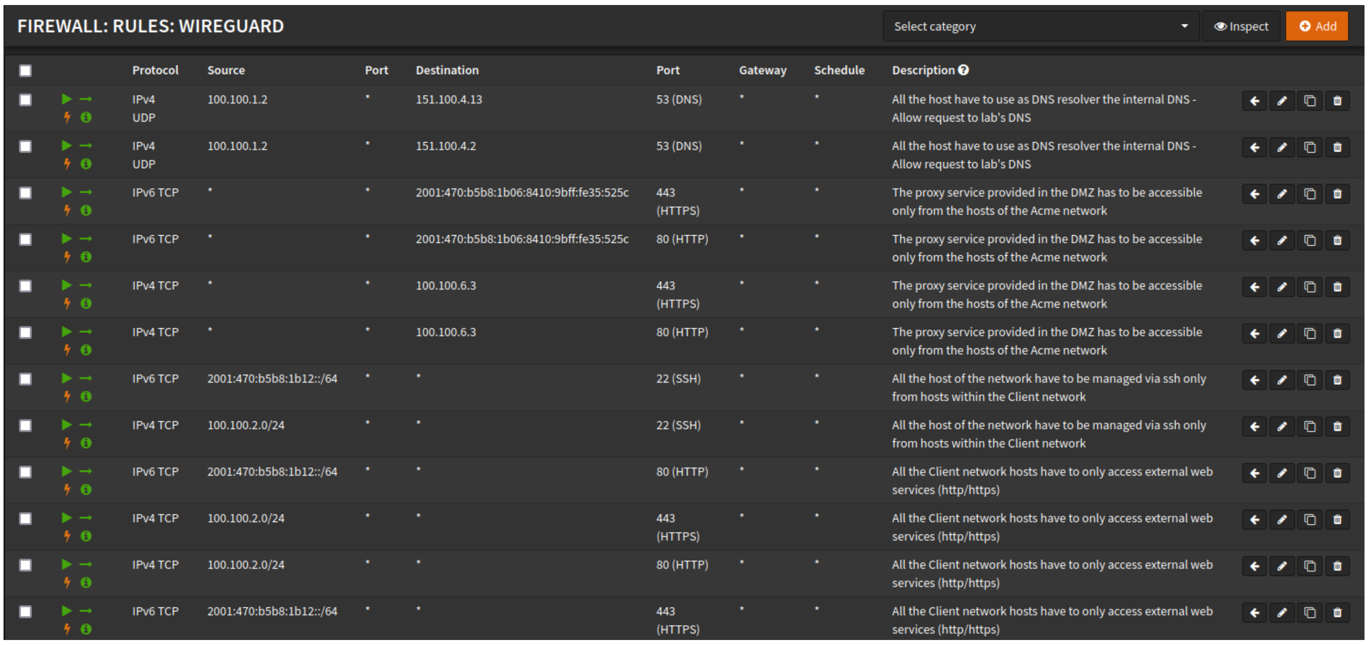
The only difference in the Internal Firewall's setup is that all the traffic is now routed through the tunnel, and this is specified in the `Allowed IPs` field:



Allowed IPs	0.0.0.0/0 × ::/0 ×

3. Firewall adjustments

We have moved the internal and external interfaces rules to the wireguard interfaces, since now all the traffic goes through the tunnel:



FIREWALL: RULES: WIREGUARD

Select category | Inspect | Add

Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
IPv4 UDP	100.100.1.2	*	151.100.4.13	53 (DNS)	*	*	All the host have to use as DNS resolver the internal DNS - Allow request to lab's DNS
IPv4 UDP	100.100.1.2	*	151.100.4.2	53 (DNS)	*	*	All the host have to use as DNS resolver the internal DNS - Allow request to lab's DNS
IPv6 TCP	*	*	2001:470:b5b8:1b06:8410:9bff:fe35:525c	443 (HTTPS)	*	*	The proxy service provided in the DMZ has to be accessible only from the hosts of the Acme network
IPv6 TCP	*	*	2001:470:b5b8:1b06:8410:9bff:fe35:525c	80 (HTTP)	*	*	The proxy service provided in the DMZ has to be accessible only from the hosts of the Acme network
IPv4 TCP	*	*	100.100.6.3	443 (HTTPS)	*	*	The proxy service provided in the DMZ has to be accessible only from the hosts of the Acme network
IPv4 TCP	*	*	100.100.6.3	80 (HTTP)	*	*	The proxy service provided in the DMZ has to be accessible only from the hosts of the Acme network
IPv6 TCP	2001:470:b5b8:1b12::/64	*	*	22 (SSH)	*	*	All the host of the network have to be managed via ssh only from hosts within the Client network
IPv4 TCP	100.100.2.0/24	*	*	22 (SSH)	*	*	All the host of the network have to be managed via ssh only from hosts within the Client network
IPv6 TCP	2001:470:b5b8:1b12::/64	*	*	80 (HTTP)	*	*	All the Client network hosts have to only access external web services (http/https)
IPv4 TCP	100.100.2.0/24	*	*	443 (HTTPS)	*	*	All the Client network hosts have to only access external web services (http/https)
IPv4 TCP	100.100.2.0/24	*	*	80 (HTTP)	*	*	All the Client network hosts have to only access external web services (http/https)
IPv6 TCP	2001:470:b5b8:1b12::/64	*	*	443 (HTTPS)	*	*	All the Client network hosts have to only access external web services (http/https)

Since the only active services in the ACME internal networks are DNS and syslog and we're SNAT-ing requests coming from Roadwarrior VPN, we didn't modify any rule in the internal firewall (for example to allow traffic towards Internal Clients network). If more services will we added, rules can be adjusted accordingly.

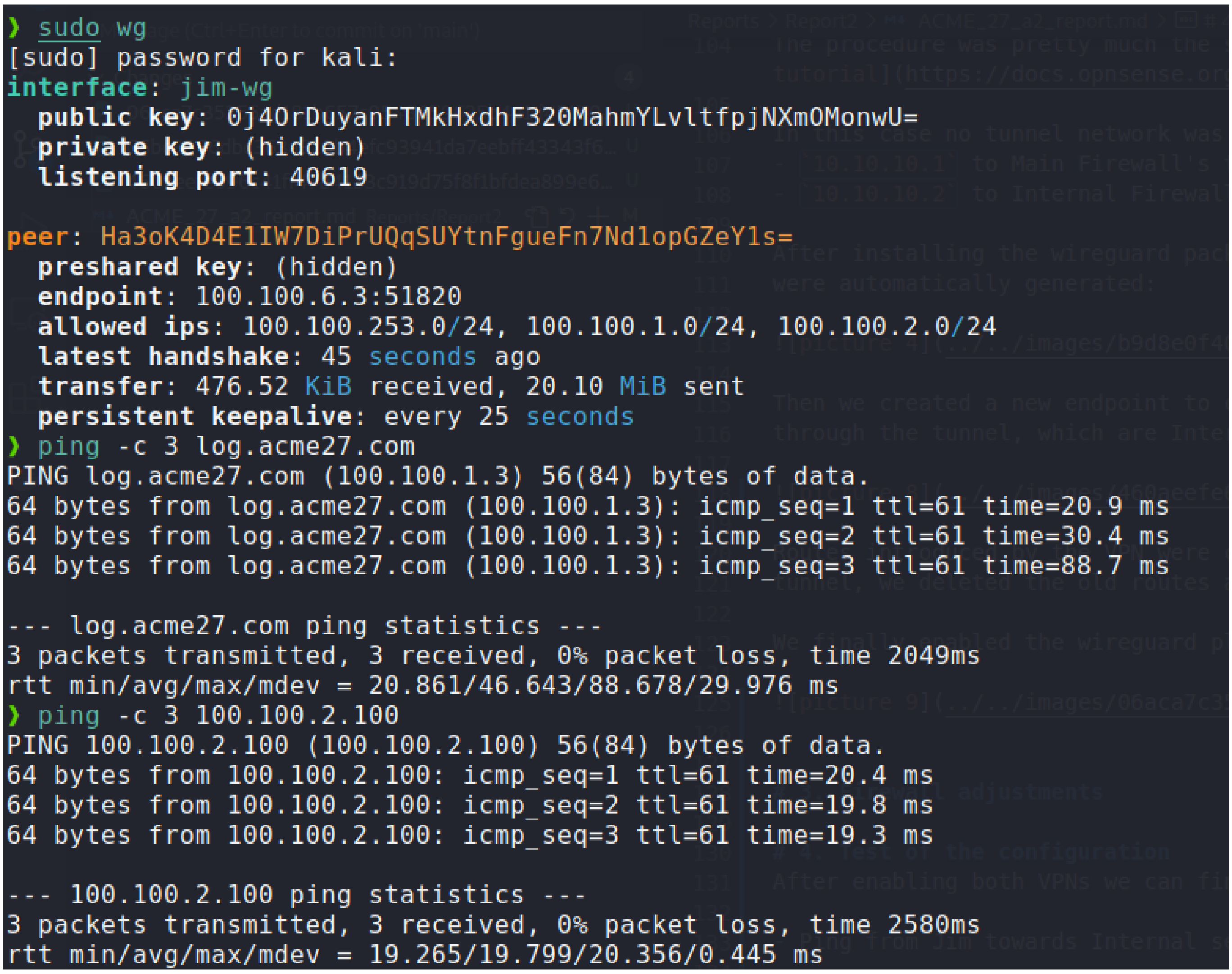
4. Test of the configuration

To test Roadwarrior VPN we must:

- enable proxy interface: `wg-quick up wg0` (on proxyserver)
- enable client interface: `wg-quick up /path/to/client.conf` (on our machines)

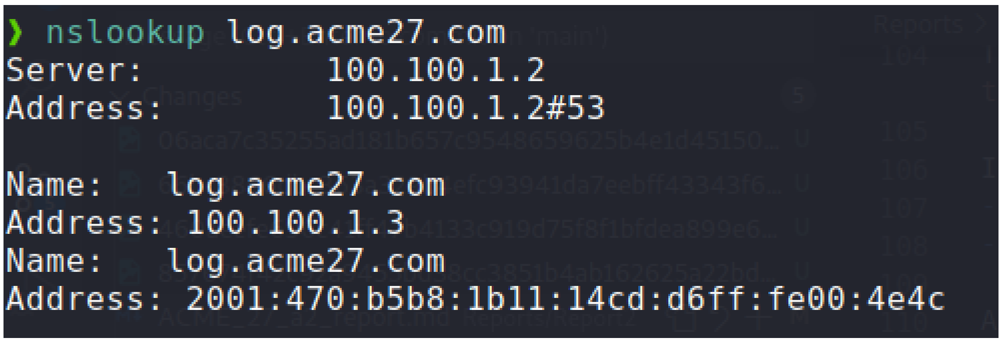
The Site to Site VPN is already enabled. We can now test our setup:

- Ping from a roadwarrior client (Jim) towards Internal servers network and Clients network



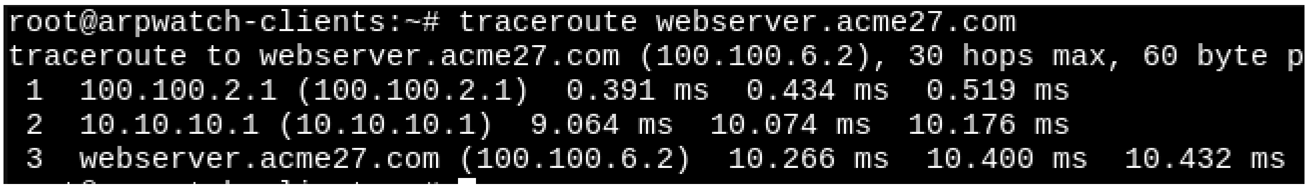
```
❯ sudo wg
[sudo] password for kali:
interface: jim-wg
  public key: 0j4OrDuyanFTMkHxdhF320MahmYLvltfpjNXmOMonwU=
  private key: (hidden)
  listening port: 40619

peer: Ha3oK4D4E1IW7DiPrUQqSUYtnFgueFn7Nd1opGZeY1s=
  preshared key: (hidden)
  endpoint: 100.100.6.3:51820
  allowed ips: 100.100.253.0/24, 100.100.1.0/24, 100.100.2.0/24
  latest handshake: 45 seconds ago
  transfer: 476.52 KiB received, 20.10 MiB sent
  persistent keepalive: every 25 seconds
❯ ping -c 3 log.acme27.com
PING log.acme27.com (100.100.1.3) 56(84) bytes of data.
64 bytes from log.acme27.com (100.100.1.3): icmp_seq=1 ttl=61 time=20.9 ms
64 bytes from log.acme27.com (100.100.1.3): icmp_seq=2 ttl=61 time=30.4 ms
64 bytes from log.acme27.com (100.100.1.3): icmp_seq=3 ttl=61 time=88.7 ms

--- log.acme27.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 20.861/46.643/88.678/29.976 ms
❯ ping -c 3 100.100.2.100
PING 100.100.2.100 (100.100.2.100) 56(84) bytes of data.
64 bytes from 100.100.2.100: icmp_seq=1 ttl=61 time=20.4 ms
64 bytes from 100.100.2.100: icmp_seq=2 ttl=61 time=19.8 ms
64 bytes from 100.100.2.100: icmp_seq=3 ttl=61 time=19.3 ms

--- 100.100.2.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2580ms
rtt min/avg/max/mdev = 19.265/19.799/20.356/0.445 ms
```

- Name resolution by a roadwarrior client (Jim)



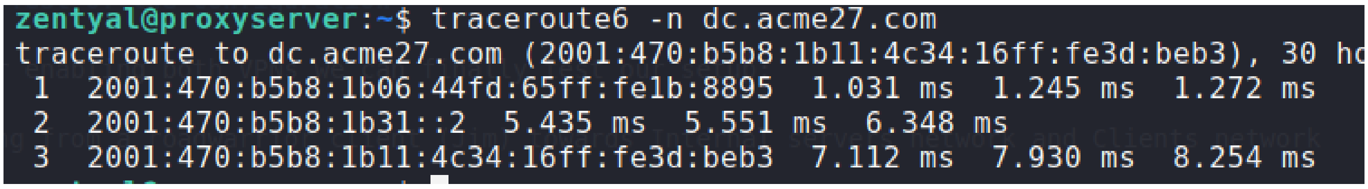
```
❯ nslookup log.acme27.com
Server:         100.100.1.2
Address:        100.100.1.2#53

Name:   log.acme27.com
Address: 100.100.1.3
Name:   log.acme27.com
Address: 2001:470:b5b8:1b11:14cd:d6ff:fe00:4e4c
```

- `traceroute` from `arpwatch` towards `webserver` to test that the traffic passes through the tunnel



```
root@arpwatch-clients:~# traceroute webserver.acme27.com
traceroute to webserver.acme27.com (100.100.6.2), 30 hops max, 60 byte p
 1  100.100.2.1 (100.100.2.1)  0.391 ms  0.434 ms  0.519 ms
 2  10.10.10.1 (10.10.10.1)  9.064 ms  10.074 ms  10.176 ms
 3  webserver.acme27.com (100.100.6.2)  10.266 ms  10.400 ms  10.432 ms
```

- `traceroute6` from `proxy` to `dc` to test IPv6 Site to Site tunnel



```
zentyal@proxyserver:~$ traceroute6 -n dc.acme27.com
traceroute to dc.acme27.com (2001:470:b5b8:1b11:4c34:16ff:fe3d:beb3), 30 hc
 1  2001:470:b5b8:1b06:44fd:65ff:fe1b:8895  1.031 ms  1.245 ms  1.272 ms
 2  2001:470:b5b8:1b31::2  5.435 ms  5.551 ms  6.348 ms
 3  2001:470:b5b8:1b11:4c34:16ff:fe3d:beb3  7.112 ms  7.930 ms  8.254 ms
```

5. Final Remarks

The Assignment seemed pretty simple at first, but, as always, we struggled using `OPNsense` integrated tools, while the Roadwarrior setup was easier.

It turns out that simply disabling routing rules is not enough if wireguard has conflicting rules: we had to permanently delete them. And even if the Site to Site plugin was up and running, without adding a specific rule allowing traffic on port 51820 the new wireguard interface would not show up in the firewalls rule section, even though we were allowing all kind of traffic for debugging.

These little things made us waste a significant amount of time, but overall we definetly learned some new things.