ACME-29
Assignment 2 Report

Edoardo Gabrielli (1693726), Davide Quaranta (1715742), Alessio Tullio (1809077)

Initial brainstorming

The purpose of this assignment is to configure two **VPNs** for the ACME network, specifically to allow "**road warriors**" to access some internal services, and to establish a secure **site-to-site** tunnel between the two routers, as they should be considered as located in different branches and at the moment their connection is insecure.

There are multiple VPN solutions available, such as OpenVPN, WireGuard and IPsec. For both VPNs we opted for **OpenVPN**, since:

- It is already implemented in **Zentyal**, which is the machine to be used as VPN gateway for the road warriors.
- It is already installed in **OPNsense**, which runs on the two ends of the site-to-site tunnel.
- It is **well established** as one of the standard ways to realize VPNs.
- The configuration options available in both Zentyal and OPNsense are minimal and functional, thus reducing the chances of mis-configuring the network.

In our topology, **road warriors** are thought to be at the same security policy level of the Client hosts, but differently from them, they can not access the External Network.

Road Warriors VPN setup details

Zentyal has several **built-in modules** that can be enabled, in particular: VPN, CA and firewall. As stated in the introduction, the VPN module is based on OpenVPN. All the above modules can be installed via Zentyal's web interface, from the *Software Management→Zentyal Components→Install* section.

The proxy server in the DMZ, acts both as **VPN server** and **Certification Authority**. The foundational requirements are the following:
- A certificate for the VPN server (called `vpn-acme-29`).
- A certificate for each client: Becca, Jim and Huck.

Certificates are necessary to provide privacy, integrity and authenticity of the packets sent through the VPN, as they are used to encrypt the packets sent between the clients and the server. We generated them from the web Interface, in the section *Certification Authority→General*.

Once we have the certificates, we need to set up the VPN server.
From the *VPN→Servers* section we added a new server called `acme-29`.

Main server configuration

Setting	**Value**
Server port	`UDP 1194`
VPN address	`192.168.160.0/24`
Server certificare	`vpn-acme-29`
Redirect gateway	`yes`

We preferred UDP over TCP for performance reasons and maintained the default port for the VPN. The VPN address is the network used to distinguish the VPN clients from the other machines. We also set the optional fields for the nameservers and search domain respectively to `100.100.1.2` and `acme29-dc.lan`.

In our concept the road warriors, while working (namely while connected to the VPN), are considered similar to the users in the `Clients network`: same policy but with a restricted network access (in fact road warriors can not access External Network).

To implement this functionality, we had to:
1. Set up the VPN with a **tap** interface, this means that a “virtual bridge” is created between the server and the clients. This adds some overhead, since all packets generated by the clients go over the VPN (e.g. the ARP packets).
2. Set the **Redirect gateway** `on`, which makes Zentyal the default gateway for the VPN clients.

Advertised networks

In this section we added a list of the advertised networks, namely all the networks that the road warriors can access: `DMZ`, `Internal servers`, `Clients`. We created the `ACME29_VPN` group as a **Network object**.

Client bundles

To let clients connect to the VPN it is required to create a client bundle for each of them. Assuming all road warriors connect to the ACME through a Linux environment, with OpenVPN already installed, the procedure to create bundles is the following; we use `becca` as reference.

From the *VPN→Servers→Download Client Bundle* section:
- **Client type**: `Linux`
- **Client certificate:** `becca`

- **Server Address:** `100.100.6.3`

Finally the generated bundle needs to be transferred to the client. Further details on the configuration can be found in the tests section of this report.

Main Router configuration

In order to make the VPN traffic flow it is necessary to add a gateway and a route for the road warriors network. This can be done in this way:

1. Add a new gateway in *System→Gateways→Single* called `VPN_GATEWAY`, for the `DMZ` interface, with IP `100.100.6.3`.
2. Add a static route in *System→Routes→Configuration* to make the network `192.168.160.0/24` be reachable via `VPN_GATEWAY`.

It is also necessary to adjust the firewall rules to allow VPN traffic, specifically:

Main Firewall Rules

Interface	**Direction**	**Protocol**	**Source**	**Destination**	**Port**
`WAN`	`in`	`IPv4 UDP`	`100.101.0.0/24`	`proxy_server`	`1194`
`OpenVPN`	`in`	`IPv4+6 *`	`VPN_TUNNEL_NET`	`*`	`*`

Where the alias `VPN_TUNNEL_NET` is `10.10.0.0/24,2001:db8:1::/64`.

Internal Firewall Rules

Interface	**Direction**	**Protocol**	**Source**	**Destination**	**Port**
`OpenVPN`	`in`	`IPv4+6 *`	`VPN_TUNNEL_NET`	`*`	`*`

Site-to-site VPN setup details

As motivated in the initial brainstorming section, we opted to use **OpenVPN** to create the site-to-site VPN between the Main and the Internal routers.

To set up this VPN we broadly had to create and configure an OpenVPN **server** on the Main Router and an OpenVPN **client** on the Internal Router, in such a way that their cryptographic and tunnel settings are mirrored. Our setup supports IPv6.

OpenVPN server

Protocol

Setting	**Value**
Server mode	`Peer to peer with shared key`
Protocol	`UDP`
Port	`1194`
Interface	`INTERNAL`

Cryptography

Setting	**Value**
Shared key	`<auto generated key>`
Encryption algorithm	`AES-256-CBC`
Auth digest algorithm	`SHA512`
Hardware Crypto	`No acceleration`

Tunnel

Setting	**Value**
IPv4 Tunnel Network	`10.10.0.0/24`
IPv6 Tunnel Network	`2001:db8:1::/64`

IPv4 Local Network	`100.100.6.0/24`
	`100.100.4.0/24`

IPv6 Local Network	`2001:470:b5b8:1d06::/64`
	`2001:470:b5b8:1d04::/64`

IPv4 Remote Network	`100.100.2.0/24`
	`100.100.1.0/24`

IPv6 Remote Network	`2001:470:b5b8:1df2::/64`
	`2001:470:b5b8:1df1::/64`

Compression	`Adaptive`

We also enabled the **Address Pool** and **Topology** flags for client settings, in order to provide and allocate only one virtual IP to the clients, and to disable the creation of restricted subnetworks per client.

VPN-specific firewall rules

In order to enable VPN traffic flow, we need to allow it via firewall rules, specifically:

Interface	**Direction**	**Protocol**	**Source**	**Port**
`INTERNAL`	`in`	`IPv4+6 UDP`	`*`	`1194`
`OpenVPN`	`in`	`IPv4+6 *`	`VPN_TUNNEL_NET`	`*`

Where the alias `VPN_TUNNEL_NET` is `10.10.0.0/24,2001:db8:1::/64`.

Interfaces, routes, gateways

OPNsense requires to explicitly create a virtual interface and to set static routes for the VPN. To do this we needed to:

1. Create the interface `OPENVPN_SERVER` for the virtual network port `ovpns1`.
2. Create the gateway `OPENVPN_SERVER_GW` for the interface `OPENVPN_SERVER`.
3. Set the routes for the `Client` and `Servers` networks to `OPENVPN_SERVER_GW`.

OpenVPN client

As introduced before, the OpenVPN client configuration on the Internal Router mirrors the one done for the OpenVPN server on the Main Router.

Specifically, the client has the same cryptographic and compression configuration and the rest is configured as follows:

Protocol

Setting	**Value**
Server mode	`Peer to peer with shared key`
Protocol	`UDP`
Remote server	`100.100.254.1:1194`
Interface	`EXTERNAL`

Tunnel

Setting	Value
IPv4 Tunnel Network	`10.10.0.0/24`
IPv6 Tunnel Network	`2001:db8:1::/64`

IPv4 Remote Network	`100.100.6.0/24`
	`100.100.4.0/24`

IPv6 Remote Network	`2001:470:b5b8:1d06::/64`
	`2001:470:b5b8:1d04::/64`

VPN-specific firewall rules

Similarly to the Main Firewall, some rules are needed to enable VPN traffic flow:

Interface	Direction	Protocol	Source	Destination	Port
`OpenVPN`	`in`	`IPv4 TCP`	`VPN_CLIENTS`	`VPN_CLIENTS_ALLOWED_NET`	`22`
`OpenVPN`	`in`	`IPv4 UDP`	`VPN_CLIENTS`	`dc`	`53`
`OpenVPN`	`in`	`IPv4+6 *`	`VPN_TUNNEL_NET`	`*`	`*`

Where the aliases are:

- `DMZ_NET`: `100.100.6.0/24,2001:470:b5b8:1d06::/64`
- `VPN_TUNNEL_NET`: `10.10.0.0/24,2001:db8:1::/64`
- `VPN_CLIENTS`: `192.168.160.0/24`
- `VPN_CLIENTS_ALLOWED_NET`: `100.100.1.0/24,100.100.2.0/24,2001:470:b5b8:1df1::/64,2001:470:b5b8:1df2::/64`

Interfaces, routes, gateways

For the client configuration, OPNsense automatically handles the gateway and routing configuration, so it is only needed to create the interface `OPENVPN_CLIENT` for the virtual network port `ovpnc1`.

Adjustments to the firewall policy

To incorporate the newly created VPN requirements into the existing security policy, it is needed to adjust the firewall rules.

- Since road warriors are similar to hosts in the `Client network`, it is sufficient to move the rules defined in the **Internal Firewall** for the **`EXTERNAL`** interface into the **`OpenVPN`** interface.

- Similarly, in the **Main Firewall** is sufficient to move the rules of the **`INTERNAL`** interface into the **`OpenVPN`** interface.

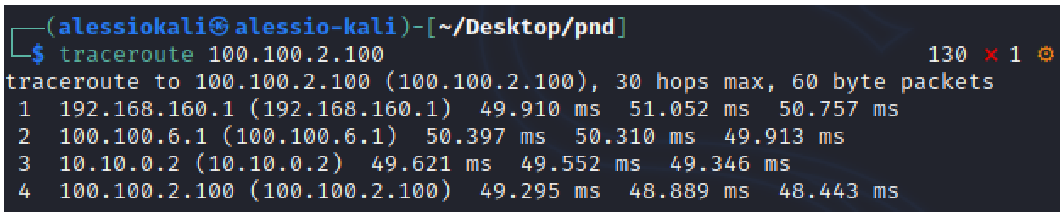
Tests

Road Warriors VPN:

The road warriors VPN can be tested by starting a VM in our host machine, after having connected to our personal VPN for the virtual environment.
Then we can connect from the guest machine to the road warriors VPN, by using one of the generated client bundles, for example Jim's.

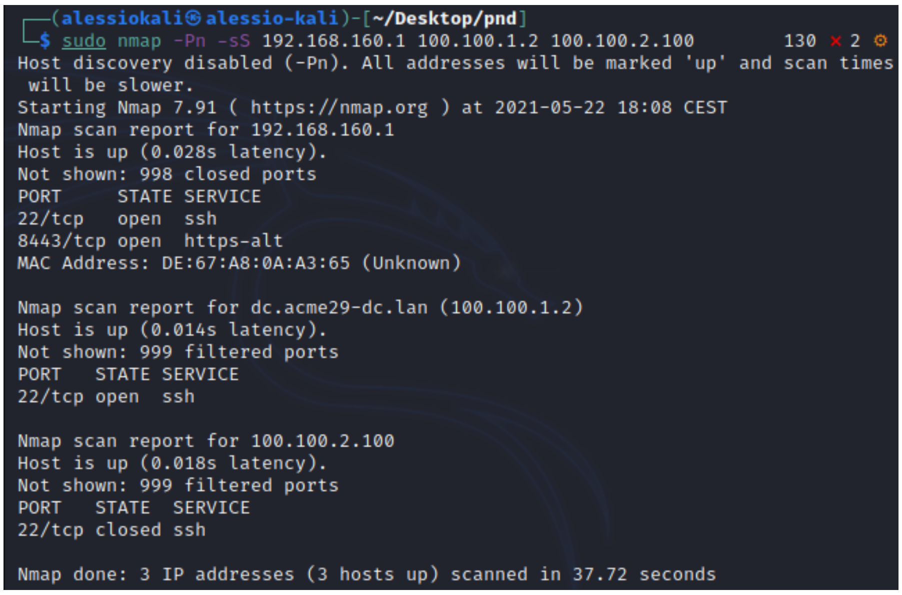
Once connected to the road warriors VPN, checked the connection with **`traceroute`** and scanned the network with **`nmap`**, to make sure that only the allowed services were accessible.

As an example, this is the output of a traceroute between Jim and the Kali machine. Please, **notice** that here we allowed the specific traffic in the firewall rules.



```
┌──(alessiokali㉿alessio-kali)-[~/Desktop/pnd]
└─$ traceroute 100.100.2.100                                         130 ✕ 1 ⚙
traceroute to 100.100.2.100 (100.100.2.100), 30 hops max, 60 byte packets
 1  192.168.160.1 (192.168.160.1)  49.910 ms  51.052 ms  50.757 ms
 2  100.100.6.1 (100.100.6.1)  50.397 ms  50.310 ms  49.913 ms
 3  10.10.0.2 (10.10.0.2)  49.621 ms  49.552 ms  49.346 ms
 4  100.100.2.100 (100.100.2.100)  49.295 ms  48.889 ms  48.443 ms
```

We scanned the whole ACME network with **`nmap`** and the results confirmed the correctness of our firewall rules. We provide here a short example, taken from a moment in which we were debugging the `Proxy Server`, the `Domain Controller` and the `client-ext1` machines.

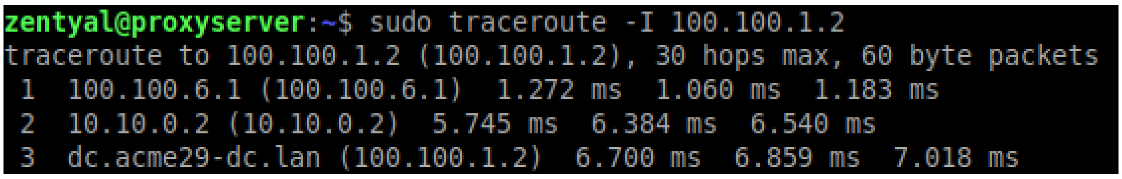


```
┌──(alessiokali㉿alessio-kali)-[~/Desktop/pnd]
└─$ sudo nmap -Pn -sS 192.168.160.1 100.100.1.2 100.100.2.100      130 ✕ 2 ⚙
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times
 will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-22 18:08 CEST
Nmap scan report for 192.168.160.1
Host is up (0.028s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
8443/tcp open  https-alt
MAC Address: DE:67:A8:0A:A3:65 (Unknown)

Nmap scan report for dc.acme29-dc.lan (100.100.1.2)
Host is up (0.014s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap scan report for 100.100.2.100
Host is up (0.018s latency).
Not shown: 999 filtered ports
PORT   STATE  SERVICE
22/tcp closed ssh

Nmap done: 3 IP addresses (3 hosts up) scanned in 37.72 seconds
```

Furthermore, it is possible to capture the traffic from the road warriors through **Wireshark**. The captured traffic is exchanged through the OpenVPN protocol and is encrypted, ensuring us that the VPN is working correctly.

Site-to-site VPN

It is possible to check that the VPN correctly works by performing a `traceroute` from two hosts belonging to the two sites, for example between the `DMZ` and the `Servers network`.



```
zentyal@proxyserver:~$ sudo traceroute -I 100.100.1.2
traceroute to 100.100.1.2 (100.100.1.2), 30 hops max, 60 byte packets
 1  100.100.6.1 (100.100.6.1)  1.272 ms  1.060 ms  1.183 ms
 2  10.10.0.2 (10.10.0.2)  5.745 ms  6.384 ms  6.540 ms
 3  dc.acme29-dc.lan (100.100.1.2)  6.700 ms  6.859 ms  7.018 ms
```

It is also possible to check the connection status from the OPNsense interface.

Final remarks

At this point the two sites are connected via a secure tunnel, and road warriors can access the network through the VPN gateway on the proxy. We needed to add new firewall rules to explicitly allow both VPNs traffic, and to restructure the existing rules for the new virtual interfaces. As mentioned in the previous assignment report, aliases in OPNsense made the configuration process easier to create and debug.

Please, for more details, refer to the configuration files.