ACME-29
Assignment 1 Report

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

Initial brainstorming

After a brief introduction to the network topology and the tools offered by the virtual environment (Proxmox), we reasoned on the best ways to configure the network, according to the requirements.

We opted for a **step by step** approach: starting from configuring the Main Firewall, then the neighbor subnetworks (`External` and `DMZ`), and finally the Internal Firewall with its networks.

What to do

We followed this plan:

1. Preliminary security issues (credentials change, updates).
1. IPv6 addressing.
2. DNS configuration.
3. Firewall rules.

How to do it

We decided to configure the **IPv6** addressing in this way:

- The Main Router should obtain a `/56` prefix from the ISP via **DHCPv6-PD**.
- The Main Router should redistribute the prefix on its interfaces.
- The Internal Router should ask for a **prefix delegation** from the Main, and redistribute it as well.
- Each host should receive the prefix and assign itself the Interface ID (**SLAAC**).

Regarding the **DNS**:

- DNS (and extra info) should be distributed via **DHCPv4** and **Router Advertisements**, from both the Main and the Internal Routers.
- The **Domain Controller** should act as the internal nameserver.

Infrastructure setup for IPv6 addressing

In this section we report the general configuration for the two routers, and document in detail the specific configurations that had been done in the hosts.

Generally, our goal for the IPv6 addressing was to **mirror the IPv4 structure**, so if the web server has `100.100.6.2`, the Interface ID should be `::2`.

Main Firewall-Router

We configured all the interfaces using the OPNsense's web interface. For all of them we enabled the options to prevent the interface removal, and removed the block for bogon and private addresses, due to the type of address used in the network.

In order to make the Main router ask the ISP for a **prefix delegation** of a `/56` subnet, we configured the **`WAN` interface** as follows:

- **IPv6 configuration type**: DHCPv6
- **Configuration mode**: basic
- **Prefix delegation size**: `56`

The router will create `/64` addresses to assign to the following interfaces, combining the delegated prefix with the Prefix ID specified in each interface.

Interface	Configuration type	Tracked interface	Prefix ID
`DMZ`	Track	`WAN`	`0x6`
`EXTERNAL_CLIENTS`	Track	`WAN`	`0x4`
`INTERNAL`	Track	`WAN`	`0x54`

Internal Firewall-Router

Similarly to the Main Router, the **`EXTERNAL` interface** is configured as follows:

- **IPv6 configuration type**: DHCPv6
- **Configuration mode**: basic
- **Prefix delegation size**: `62`

After receiving a prefix of a `/62` subnet, the router will create `/64` addresses to assign to the following interface, combining the delegated prefix with the Prefix ID specified in each interface.

Interface	Configuration type	Tracked interface	Prefix ID
`CLIENTS`	Track	`EXTERNAL`	`0x1`
`SERVERS`	Track	`EXTERNAL`	`0x2`

Machine-specific settings

This section will describe the main configurations applied to the machines.

To set the IPv6 addresses for machines that need a static address (e.g. servers), we decided to use the **IP Tokenization** approach, which consists in instructing the machine to set a **fixed Interface ID** to the prefix obtained via Router Advertisements.

For example, to implement it in the webserver, we added a file `/etc/network/if-up/ip-token` which contains:

```
#!/bin/bash
ip token set ::2 dev eth0
```

We also modified the **sysctl** configurations (in `/etc/sysctl.d/`) to make sure that the machines were appropriately configured for IPv6.

Client ext 1

Since this machine is not a server and doesn't need a fixed interface ID, we preferred to generate a **stable privacy address**, by setting `net.ipv6.conf.all.addr_gen_mode=3` in the sysctl configurations.

Arpwatch

Since the Arpwatch machine will probably be used to monitor the network for MAC-IPv4 pairings, we disabled the IPv6 addressing by placing `net.ipv6.conf.all.disable_ipv6=1` in the sysctl configurations.

IPv6 addresses recap

Network	Machine	Interface ID
DMZ	Web Server	::2
DMZ	Proxy Server	::3
INTERNAL SERVERS	Domain Controller	::2
INTERNAL SERVERS	Log Server	::3
EXTERNAL	Client ext 1	privacy_ip
CLIENTS	Kali	privacy_ip

DNS configuration

To **fully support IPv6** in the network we decided to configure the DNS directly via **dnsmasq**, instead of relying on Zentyal, since it doesn't support IPv6.

Dnsmasq has been configured as follows (`/etc/dnsmasq.d/main.conf`):

```
# interface to listen on
interface=eth0

# addresses to listen on
listen-address=::1,2001:470:b5b8:1df1::2,127.0.0.1,100.100.1.2

# nameservers
server=8.8.8.8
server=8.8.4.4
server=2001:4860:4860::8888
server=2001:4860:4860::8844

# domain configuration, to automatically expand hostnames to
fully-qualified domain names (host -> host.acme29-dc.lan)
expand-hosts
domain=acme29-dc.lan
addn-hosts=/etc/hosts.acme29-dc.lan

# do not resolve addresses with /etc/resolv.conf
no-resolv
```

Note that:

- The `/etc/hosts.acme29-dc.lan` file contains a list of both IPv4 and IPv6 for each host within the ACME network.
- The routers send DNS info via **DHCPv4** and **Router Advertisements**.

Evaluation of the security policy

We refined and synthesized the security policy to implement, in order to **minimize the rules** and **generalize** them. The policies needed to be evaluated as a whole to fully understand the big picture; for example:

- *"All the services provided by the hosts in the `Internal server network` have to be accessible only by the `Client network` and the `DMZ` hosts."*
- *"All the hosts (but the `Client` network hosts) have to use the syslog service on the Log server (syslog)"*

These two rules ultimately mean that only the `DMZ` hosts can access the syslog.

The **security policy** that has been implemented is:

1. *All the hosts have to use as DNS resolver the internal DNS.*

   The Internal DNS (Domain Controller) uses Google's Public DNS (8.8.8.8 and 8.8.4.4). The `DMZ` and the `Client network` have to use the internal DNS, while the `External network` doesn't have to as stated in the rule #4.

2. *Only the webserver service provided in the DMZ has to be accessible from the Internet.*

3. *The proxy service provided in the `DMZ` has to be accessible only from the hosts of the ACME network. However, the proxy needs internet access.*

The proxy needs access to the internet, but can not be queried by hosts not belonging to the ACME network.

4. *All the services provided by hosts in the `Internal server network` have to be accessible only by `Client network` and `DMZ` hosts.*

   Intersecting this rule with the #6, the result is to allow the `DMZ` hosts to communicate with the whole `Internal servers network` (DC and Log Server), while the `Client network` can only communicate with the DC.

5. *Anything that is not specifically allowed has to be denied.*

   OPNsense's default policy is already to deny.

6. *All the hosts (but the Client network hosts) have to use the syslog service on the Log server (syslog).*

   Only the `DMZ` hosts have to access the syslog service provided by Log server since, as stated in #4, only the `Clients network` and `DMZ` hosts have access to the `Internal servers network`.

7. *All the hosts of the network have to be managed via ssh only from hosts within the `Clients network`.*

8. *All the `Clients network` hosts have to only access external web services (http/https).*

   Here we decided that it makes sense to let the `Clients network` hosts communicate with external web services through the Proxy server, so the rule actually allows them to contact the proxy.

Policy implementation in OPNsense

The exported configuration of the two firewalls is included in the archive.

Taking advantage of the **statefulness** of the firewall and its support for **aliases**, we only had to set up the rules for the input direction, and associated an arbitrary name (alias) to multiple IPs or networks.

To make the firewall rules management easier, we created aliases for each server and subnetwork, including one for the whole ACME-29 network, for both IPv4 and IPv6 addresses.

OPNsense's **default policy** is **`DENY`** for each interface, to drop all the traffic that is not specifically allowed.

Test of the configuration

To test whether the configuration was correctly done, we proceeded in this way, for both IPv4 and IPv6:

1. Ensure that the prefix delegation works, by checking if the OPNsense interface shows a prefix for each interface.

2. Ensure that the IP Tokenization mechanism works by:
    - Shutting the host's interfaces down and up.
    - Restarting the `networking` service.
    - Restarting the host.

3. Ensure that Router Advertisements are correctly set up (e.g. right options, DNS) by:
    - Inspecting the packets with Wireshark.
    - Checking the content of `/etc/resolv.conf` in the hosts.

4. Ensure that the DNS works by checking if hosts can perform DNS queries with the Domain Controller, e.g. with `dig github.com @100.100.1.2` or `host github.com 100.100.1.2`.

Firewall policies

To test the firewall policies we defined and used the following **framework**:

DNS (#1)

Test from each host in each network if:

- The internal DNS is correctly set up as a resolver.
- The host can query the internal DNS.
- The host can't query with a different DNS than the internal.
- The host can't query with the upstream DNS that the internal one uses.

Web server reachability (#2)

Test from the `WAN` if:

- The web server can be accessed on port 80 from the WAN.
- No other host can be accessed from the WAN.

Proxy reachability (#3, #4)

- Test if every ACME-29 host can access the proxy.
- Test if the proxy can initiate connection to outside.

Internal servers reachability (#5)

Test if:

- Hosts in the `DMZ` can access the internal services.
- Hosts in the `Clients network` can access the internal services.
- Hosts in the `External services network` can't access the internal services.

Syslog (#6)

Test if:

- Hosts in the `DMZ` can access the syslog.
- No other host can use the syslog.

SSH (#7)

Test if:

- Hosts in the `Clients network` can use SSH on every ACME-29 hosts that has an ssh daemon running.
- No other host can use SSH on hosts in different networks.

Clients web browsing via proxy (#8)

Test if hosts in the `Clients network` can access the proxy on its default port.

Final remarks

In this section we report some issues that we faced and how we solved them.

OPNsense

It should be taken into account that the OPNsense's web interface doesn't report what causes the failure of some services. For example, if a service is improperly configured, OPNsense will just fail in starting the service, without telling the error on screen.

Although the interface is fairly usable, reading OPNsense's documentation is required to understand its behavior and to properly configure some services.

Zentyal

There are two servers running Zentyal: the Proxy and the DC. It should be taken into account that Zentyal doesn't support IPv6 and that there are some issues that may arise in the upgrade process.

In our case, we decided to **upgrade** the Zentyal distribution on the DC, which ultimately resulted in a broken - then fixed - state, due to a **bug** in Zentyal's post-update script[1].

Specifically, the bug was about the webadmin SSL certificate regeneration:

```
if [ `redis-cli get ca/conf/Certificates/keys/crt1 | jq .enable` -eq 0 ];
```

The script doesn't handle the case in which the value is empty but not zero (our case), and the certificate regeneration never happens. **To solve it** is sufficient to manually execute the commands wrapped in the `if` and to restart Zentyal's webadmin module with the `zs` CLI.

After this experience we decided not to upgrade Zentyal on the Proxy, in order to avoid possible unexpected or unstable states.

[1]https://github.com/zentyal/zentyal/blob/80eca4ab66374e2c7f662f9bd09dd1314dcacd8f/main/core/src/scripts/release-upgrade#L244