Practical Network Defense

Second Assignment - University "La Sapienza"

Group 27: Nicola Bartoloni ******* - Valerio Trenta 1856471

2020-13-06

1 Scope and Initial Considerations

The scope of this assignment comprehends the whole target network: a series of services provided by different machines in several subnetworks in the **ACME** environment are listed, and we focus on how to enable these services to the clients of our network by exposing them on a controlled manner through firewall rules specified in the **OPNSense** service provided by the two main routers of the network.
By catching a quick glimpse of the policy proposed by the assignemnt, we can tell the scope is to configure our network with a series of *white-listed* services that we want to be enabled and reachable from certain machines in the network itself, thus reducing the *attack surface* of the network by also excluding from this white list every other service that is not of interest.
Our analysis should begin with the services that we want to provide in the target network, and then on the firewall rules that make it possible to reach each service on each different machine.
Please notice that, since IPv6 addresses are not part of the scope of this assignment, we will ignore them during the configuration of the firewall rules in section 4.

2 Infrastructure Setup

We identify four main services which require machine-to-machine communication and, thus, are susceptible to the firewall rules that we'll define:

- an **Apache** web server reachable through **TCP** ports **80** (HTTP) and **443** (HTTPS) at **100.100.6.2**;
- a **DNS service** reachable through standard **UDP** port **53** at **100.100.1.2**;
- a **rsyslog at logserver** which we will make reachable through standard **UDP** port **514** at **100.100.1.3**;
- a **proxy server** reachable through port **3128** at **100.100.6.3**, to be configured later on Assignemnt 3.

We thus identify two new subnetworks: the **Internal Servers** subnetwork **100.100.1.0/24** and the **DMZ** subnetwork **100.100.6.0/24**, and for the sake of defining the scope of this assignment and the setup of the whole infrastructure, we should also consider two additional details:

- One of the four services - web server - is already setup and running in the system: it does not need to be configured or launched;
- the **proxy server** is to be configured on the next assignment: for now, knowing from the **zentyal** portal that it is provided on port 3128 is enough.

This leaves us with the only **DNS service** to be configured at the moment, and the **rsyslog** process at **logserver** and at each of its clients to be defined later. The first step has been performed by following the instructions provided with the assignment: by accessing the **zentyal** portal at **100.100.1.2** on port **8843** - credentials have been changed - we added as forwarders the suggested IP addresses and specified the domain name - **acme.group27**. Then, the file located at **/etc/zentyal/dns.conf** on the aforementioned machine was modified including the target subnetworks (Clients and DMZ, that are the ones which will exploit the service).
At this point, pairs of IP addresses and hostnames were specified in the portal, so that the well-known machines of the network can now be associated with the following names:

- **kali**.acme.group27;
- **watchdog**.acme.group27;
- **dc**.acme.group27;
- **web**.acme.group27;
- **proxy**.acme.group27;

- **coffee**.acme.group27;

also, as suggested in the assignment, the external services machines were provided with an external DNS service such as 8.8.8.8 in their DHCPv4 configuration. Please notice also that the *logserver* was not given a name since it is only meant to be accessed by the SPOCK environment or by SSH and is not offering any browser-related service - and the same reasoning could be applied to the first two machines, *kali* and *arpwatch*, which we could exclude from this pairing list.
Last step was to actually modify the DHCPv4 settings in the two main routers to set the **dc** machine as the **DNS server** in the Internal Servers network, after having disabled the **Service Unbound DNS** in both routers.
Keep also in mind that some of the machines with predefined IP addresses (fixed, not assigned by DHCP service) had either to be configured externally, or had to have their */etc/resolv.conf* file re-configured by modifying the IP address of their nameserver (for instance, Proxy and Kali machines).
The **DNS service** configuration is tested in section 5.

2.1 Log server

The **rsyslog** service at **logserver** was configured by modifying the file */etc/rsyslog.conf*, whose lines about **TCP** remote service were uncommented so that the service is listening on **TCP** port **514** for log messages to record from its clients. Then, a **template** was defined in this config file, so that for every log message that is received, the service records it under path */var/log/machine/process.log* - e.g., when process **login** from **webserver** machine sends a log message to the **logserver**, it is recorded under /webserver/login.log under the logging directory.
Then, for each **client** of the service, the very same config file was modified - so for every machine on **DMZ** and **Internal Servers** subnetworks, by appending to the end of the file **.* @@100.100.1.3:514* to specify that each log message from every process of the machine must be sent to the **logserver** through **TCP** protocol on port **514**. Figures **1** and **2** at page 8 show the outcome of this configuration.
The double @ character in the previous string strictly specifies that the protocol to be used is **TCP** and not **UDP**, since it is a best practice to exploit the first and not the latter when performing remote logging given that it is a more reliable and secure protocol.
Please notice that we have also checked that, when further services were implemented in the machines that send remote logs to *logserver* in the next assignments, these services are also logged under the corresponding directory of their machine - like *fail2ban* service.

3 Policy Evaluation

The proposed policy targets the four services listed in the previous paragraph and a series of machines which will either exploit or provide the corresponding services.
The best way of interpreting and understanding this policy is given by its fifth line:

- *"Anything that is not specifically allowed has to be denied"*;

which suggests the *white-listed* approach we have to adopt when defining the firewall rules: the policy is a list of **PASS** rules - meaning, rules that when matched will let the packet pass and continue its journey - while everything that doesn't match the rule has to be rejected, i.e. the packet must be dropped.
If implemented correctly, this policy should allow the Internal services to be exploited and reachable only by the **Clients or DMZ networks**, and it should also allow the internal client hosts in the **Clients** network to either reach the **external web services** through HTTP/HTTPS, or the Internet via the **Proxy service** to be configured in the next assignment on port 3128, being this service in the **DMZ** the only one that is able to actually initiate connections with **WAN** again on HTTP/HTTPS protocols, while the **web server** in **DMZ** should be the only machine which can accept connections initiated by someone else from the Internet and thus outside our target network.
Also, every machine in the internal network (meaning **DMZ**, **Internal servers** and **Clients network**) should be accessible on port 22 (SSH) only from hosts in the **Clients network** and obviously reject any other connection on that port from undesired hosts: this means only the internal clients in **100.100.2.0/24** should be actually able to administrate the machines in our internal network.
Furthermore, external clients may freely choose their DNS service, so they can also request an external DNS service on the Internet, but every other machine in our internal network must exploit our internal DNS Service - and, even if this can be bypassed easily, this means that internal machines are only able to reach by their domain names machines whose names are defined in our internal DNS service, at least at the very beginning.
The two main routers, other than having their default passwords changed, should also only be reachable on their HTTP service by machines in the internal clients network.
Notice that there may be several different working setups to actually implement the policy, and the one we implement and describe in the next paragraph may not be the only one.
Nonetheless, we can generally state that the approach that is suggested, given the infrastructure's architecture, is that of having a *dual-homed host* firewall approach - indeed, we do have two firewalls, and the **Internal** interface of the **Main Firewall** especially should decide whether packets coming from the **WAN** should actually be received by the **Internal Firewall** and viceversa - obviously with a **DMZ** placed in between to implement a **defense-in-depth** strategy.

4 Policy Implementation

While evaluating the policy in the previous paragraph, we have already proposed a high-level description of the rules to be implemented at each interface of the two firewalls. Notice that, for instance, the rules we applied at the **Internal interface** of the **Main Router**, could have also been applied, instead, at the **External interface** of the **Internal Router** or, to have a complete in-depth defence, the same rules could have been applied at both interfaces - which could also be the best solution in order to prevent bad scenarios like the failure of one of the two firewalls.
Also, there is another interesting fact about OPNSense Firewall configuration: it tracks connections by default, so that every packet in a TCP connection which has already been *established* and has been accepted, is automatically accepted (*pass* action) by default. This means that if we set a rule to allow machine A to ping machine B through ICMP protocol and then another one which rejects by default any other connection on that interface, since OPNSense by default applies to the packet the very first rule that matches, machine B's response will be automatically accepted too by the firewall, while if machine B initiates a new connection by pinging machine A, the firewall will not allow it - and we can verify this on the testing paragraph.
While implementing this policy, several **aliases** had to be defined in OPNSense's Firewall service at each of the two routers. These aliases could either be single hosts or single subnetworks, which were in some cases already present by default. This way, machines or subnets that were allowed to send or receive specific packets on an interface were addressed more precisely, and we were able to make important distinctions between hosts in the same subnet but with different policies - i.e., **Web Service** and **Proxy Service** in the **DMZ subnet**, or the two internal services **DNS** and **Log**.
These are the rules that were specified on each of the two routers:
Internal Router:

- **Clients Interface**: only accept incoming packets on this interface with destination ports 80(HTTP), 443(HTTPS), 22(SSH), 53(UDP-DNS) and 3128(Proxy Service). Anything else will be discarder, since clients are not supposed to perform different actions and exploit different protocols than the aforementioned ones. For practical reasons, also packets on port 8443 (for **zentyal** panel service) are allowed - but this might be changed;

- **Servers Interface**: only accept incoming packets on this interface with destination port 53(UDP-DNS) or 514(TCP-SYSLOG, only if coming from **DMZ subnet**) and on port 22(SSH) only if coming from the **Clients subnet**, otherwise discard. This means that the two servers can only be reached by Clients or DMZ subnetworks, and only for the services that they provide (SYSLOG, DNS, SSH);

- **External Interface:** we can either set the same rules of the **Main Router**'s **Internal Interface** to have a more robust setup, or leave this with its default configuration;

Main Router:

- **Internal Interface:** only accept incoming packets on this interface on ports 80(HTTP) and 443(HTTPS) between **Clients subnet** and **External services subnet**, or ports 22(SSH) and 3182(Proxy) between **Clients subnet** and **DMZ subnet**, or ports 53(UDP-DNS) and 514(TCP-SYSLOG) between **DMZ subnet** and **Internal services subnet**, otherwise discard. This means the Client host will only be able to reach the external web services, or the proxy to actually reach the WAN, and won't be able to do it directly by itself. Furthermore, SSH is eanbled for Client hosts to reach the **DMZ subnet** and the services offered by the **Internal servers** are reachable from the machines in the DMZ;

- **DMZ Interface:** we want the proxy server to be able to reach the internet via HTTP/HTTPS protocol, so we specify that this interface has to accept incoming packets specifically from the **Proxy server** and with **any** destination address on ports HTTP/HTTPS, and the same goes (but reversed) for the **web server** which we want to be accessible on ports HTTP/HTTPS from the Internet. However, the **proxy service** itself must be only available for client hosts in the **Clients subnet**, so we should also specify that this interface should accept incoming packets on port 3182 of the Proxy machine with source address in the **Clients subnet**. The aforementioned rules for DNS/SYSLOG services in the Internal Services apply also here on this interface, while every other packet which is not *white-listed* must be rejected;

- **WAN Interface:** this interface should be the first line of prevention against intruders from the Internet, so it only has to accept - as specified by the policy - connections with destination port 80 or 443 on the **web server** machine in DMZ and connections to any other machine in the Internet on port 80 or 443 if they are initiated by the **Proxy** machine in DMZ: every other packet incoming on this interface should be rejected;

- **External services Interface:** since anything that is not specifically allowed must be denied, this interface should only allow external clients to establish a connection with their DNS service and an HTTP/HTTPS connection with the internal clients of our network, while anything else (including HTTP/HTTPS with **WAN** or SSH with internal clients) should not be allowed.

Please notice that for practical reasons, hosts in the Clients subnetwork have also been granted the possibility to access the two routers via their HTTP service. This is not specified by the policy and thus should not be allowed, but since we are still implementing the whole infrastructure, we need access to this service from this subnetwork. At the end of the whole process, this rule should be disabled.

5 Tests

We performed some basic tests for the DNS service and the new policy that were enabled and implemented in this assignment.
To perform thorough tests on the firewall rules we implemented, the WAN interface on the **Main router** was enabled, through the option provided in OPNSense, to accept private connections, so that it could be reached from external hosts (Internet, or our local machines).

5.1 Testing the DNS Service

For the internal DNS service, once defined all the firewall rules, we just tried to solve the domain names through the **host** command on terminal on the machines exploiting the service, verifying that the names were correctly solved to their corresponding IP addresses - e.g., *coffee.acme.group27*, *dc.acme.group27* or *web.acme.group27* were correctly resolved by any machine in **DMZ subnet** and **Clients network** subnet. The same command was then run on machinesd of the **external clients subnet**, verifying that some external domain names were correctly resolved - e.g., *google.com*.
No errors or misconfiguration problems were faced during this test phase.

5.2 Testing the policy

To test the policy implementation and make sure it behaves the way we wanted it to behave as described in the third paragraph, we had to perform some basic tests by exploiting at least one machine in each of our subnetworks, and also an external one on the **WAN interface** simulating an external machine connected from the Internet.
A bunch of basic tests consisted in *pinging* internal or external machines (not in the same LANs obviously), since ICMP is not mentioned in the policy and thus it should be disabled by default - indeed, the ouctomes were always negative: no machine in the target network is able to ping any other machine that is not in the same LAN.
The pictures from **3** to **11** below represent, for each point of the policy that was implemented, a testing command that was executed - either via terminal or via browser - and its corresponding desired output.

Please notice also that in some of these tests we were able to perform an SSH connection and authentication on some of the machines in the DMZ from the **Kali machine**: we indeed verified that machines in DMZ and Internal Services network accept connections on port 22 from the Clients network only, while they reject any other connection on the very same port from different source networks.

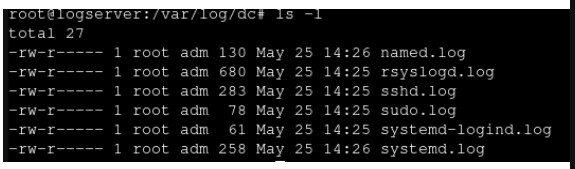

Figure 1: Log files sent by 100.100.1.2 - dc machine.

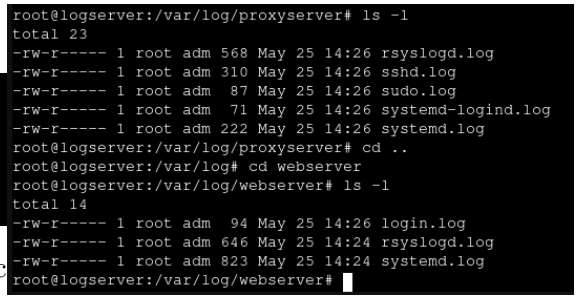

Figure 2: log files sent by machines in DMZ subnet.

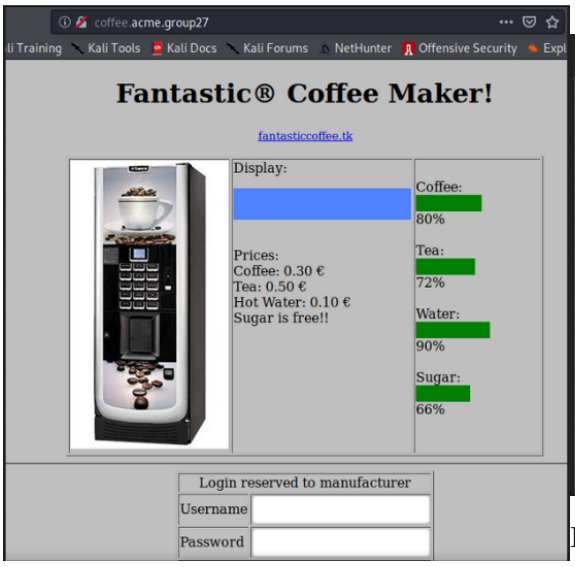

Figure 3: Machine in Clients subnet can connect via HTTP/HTTPS to web service in External Clients subnet.

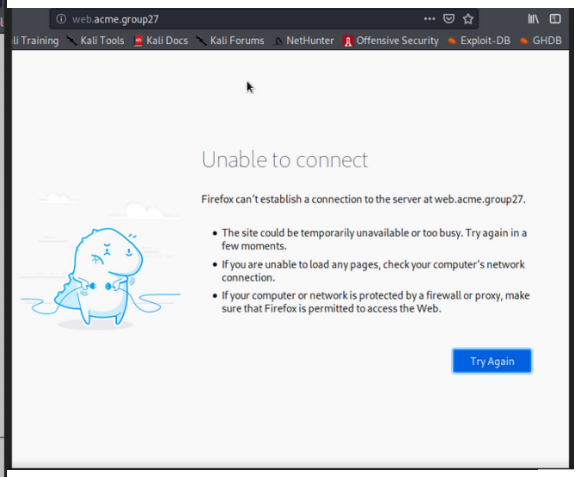

Figure 4: Machine in Clients subnet cannot connect via HTTP/HTTPS to any other web service that is not in External Clients subnet, not even in DMZ.

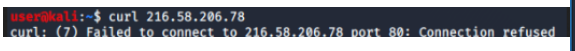

Figure 5: Machine in Clients subnet cannot connect via HTTP/HTTPS outside (WAN).

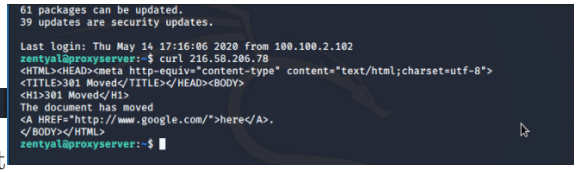

Figure 6: Proxy machine is enabled to connect via HTTP/HTTPS to Google, outside (WAN).

6 Final remarks

The testing phase showed us that the policy, if correctly interpreted, was implemented as specified by the assignment.
Some extra measures could have been taken, and they are very similar to the ones we have seen in the first assignment: we need to perform Linux hardening via sudo and hardening of the SSH protocol for all the machines in the DMZ and Internal Service subnetworks: this was done by following the same procedure that was applied to the **arpwatch machine** in the previous assignment, thus by disabling SSH login via basic authentication on the *root* account of every machine, and by enabling only *public key authentication* for this very same protocol through the *sshd_config* file. In the machines that do not exploit **Zentyal** thus, another user was created and managed via *sudoers* file - on the *logserver* and *webserver* machines. The machines are thus only reachable by the **Kali machine** - the only machine holding the authorized key, protected by a passphrase - through SSH login.
We should also stress the fact that the scope of this assignment was to focus mainly on the **firewall rules** to be defined to achieve our goals and implement the given policy: this means that link−layer attacks were not taken into consideration, and this may be a critical vulnerability in our network. indeed, notice that every machine can be accessed through SSH or other protocols by any other machine on its same subnetwork - i.e., *dc* can access *logserver* through any protocol. This is because the rules we defined to implement our policy are placed on the two firewall−routers, which obviously do not handle packets exchanged between hosts in the same subnetwork: to overcome this issue, probably some extra rules should also be defined locally on each machine via *iptables* - but this was out of our scope for this assignment.
Also, as a last step, when the whole configuration going on in these assignments is over, we should at least enable only one machine in the **Clients** subnetwork - the **Kali** machine, probably - to connect via HTTP service to the two routers, for administration and practical reasons, so as we mentioned in the previous paragraphs, the rule enabling these connections should be actually disabled as a last step.

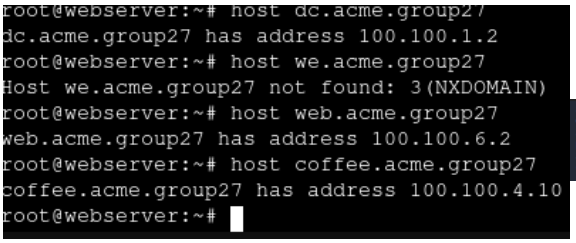

Figure 7: Web server (DMZ) can DNS request on Domain Controller machine.

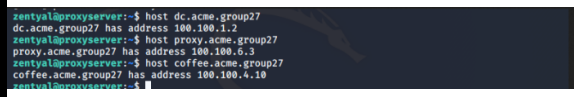

Figure 8: Same for Proxy (DMZ).

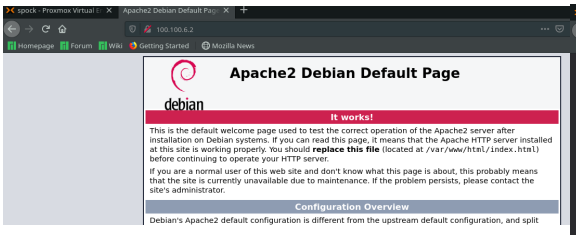

Figure 9: Web server (DMZ) is reachable from the outside (WAN) on HTTP/HTTPS protocol.

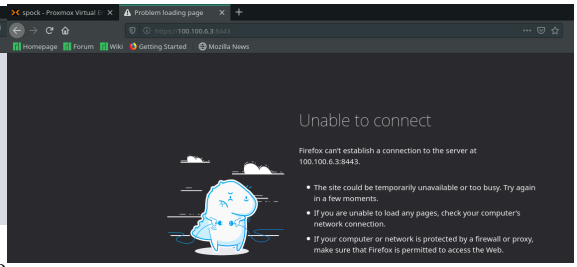

Figure 10: Proxy is not reachable from outside, not even on its Zentyal service port.

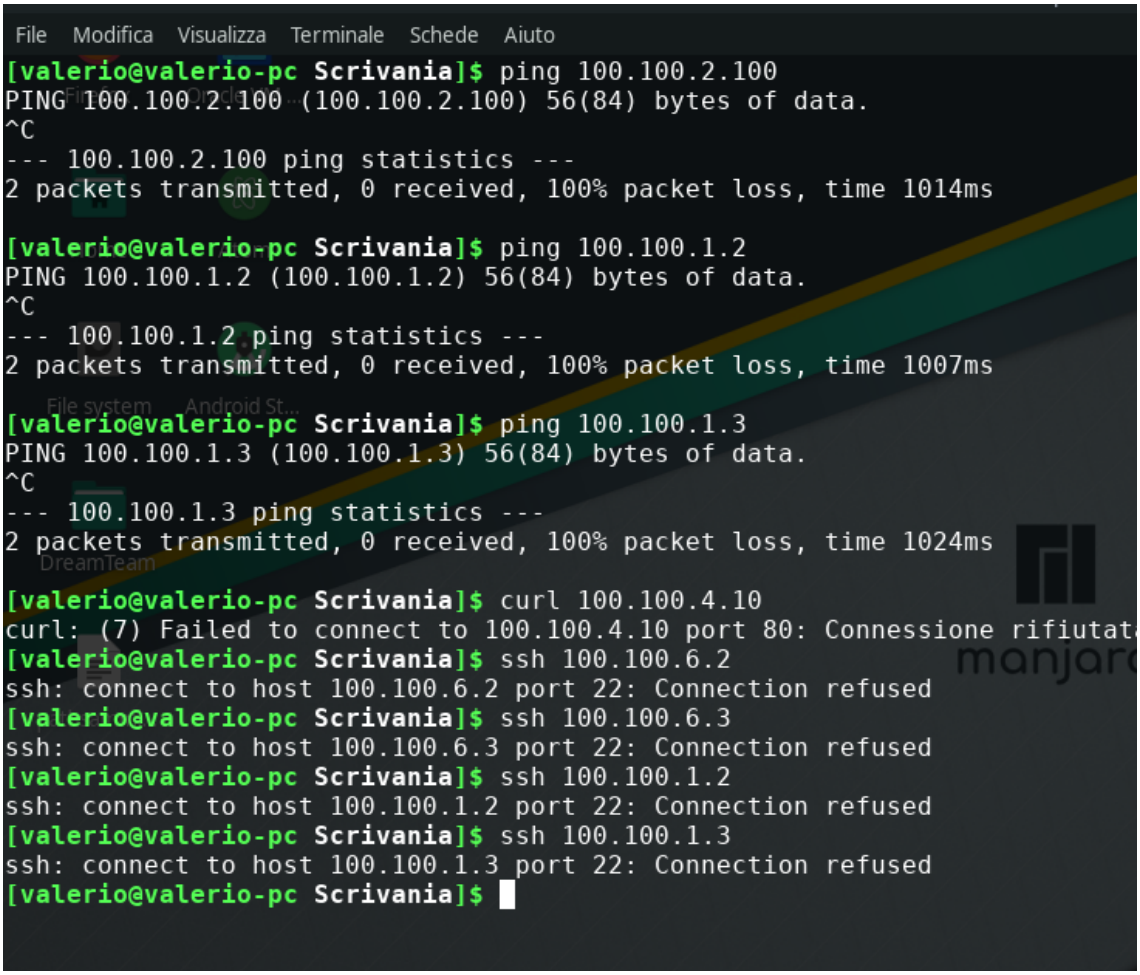

Figure 11: WAN interface rejecting connections to specific hosts in the target network on specific protocols.