SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF CYBERSECURITY

Assignment 2

ACME-02

PRACTICAL NETWORK DEFENSE

Students:
Federica Bianchi, 1891952
Giovanna Camporeale, 2086227
Francesco Rizzo, 1955615
Alessio Vigilante, 1839055

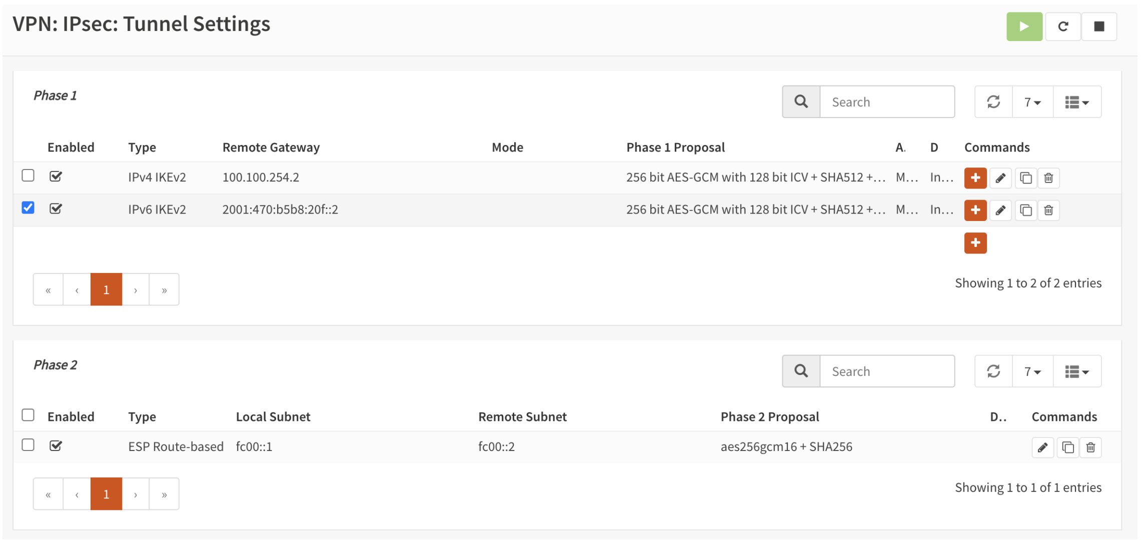
Professor:
Angelo Spognardi

Academic Year 2022/2023

Contents

1 Brainstorming

Before starting with this assignment, we decided to implement **IPv6**. We followed the guide and for every host we assigned the IPv6 address. After that, we readjusted the rules and configuration of the previous assignment. For example, we needed to adapt IPv6 also for the IPsec tunnel, so we created another route-based tunnel as we did for IPv4 (first report).



VPN: IPsec: Tunnel Settings

Phase 1

	Enabled	Type	Remote Gateway	Mode	Phase 1 Proposal	A.	D	Commands
☐	✓	IPv4 IKEv2	100.100.254.2		256 bit AES-GCM with 128 bit ICV + SHA512 +...	M...	In...	
☑	✓	IPv6 IKEv2	2001:470:b5b8:20f::2		256 bit AES-GCM with 128 bit ICV + SHA512 +...	M...	In...	

Showing 1 to 2 of 2 entries

Phase 2

☐	Enabled	Type	Local Subnet	Remote Subnet	Phase 2 Proposal	D..	Commands
☐	✓	ESP Route-based	fc00::1	fc00::2	aes256gcm16 + SHA256		

Showing 1 to 1 of 1 entries

Figure 1: Tunnel IPsec for IPv6

After that, we also decided to implement in the two firewalls the possibility to log in only using HTTPS. We simply did this accessing OPNsense on the section System:Settings:Administration and selected the protocol HTTPS.

Then, we started analyzing the topology of the network and we did a recap of the previous assignment and on the firewall rules that we had.

Then we looked at all the policies to implement, in order to start visualizing them.

2 Evaluation of the security policy

1. **All hosts must use the internal DNS Server as a DNS resolver**.

   First of all, we gathered more information about the different implementation of a dns resolver, like dnsmasq and unbound DNS.
   After that, we saw the video explanation of how to configure the dns on the server and the different hosts.

   As the policy requires, all the host must use the internal DNS server as a DNS resolver, so we decided directly to remove the previous DNS and to only use the new one. For this reason, we didn't add any additional firewall rule.

Finally, we discussed about the host *Client ext 1*, which has a dynamical address managed by DHCPv4. So, it needed to be configured in a more specific way in OPNsense. The same thing happens with the VPN (as the addresses are assigned dynamically).

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

   First of all we've defined the scope of the rule to add: only the WAN interface of the Main firewall.

   Then we stated that, as a default rule, the firewalls block all the traffic so we just needed to allow the traffic directed to the web server. Since it is a web server, we decided that it was better, and more coherent, to allow only the HTTP and HTTPS traffic, and block all the other.

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 (see below, section Services of the ACME co.)**

   At the beginning we thought, in order to make the proxy service in the DMZ accessible from the hosts of the acme network, to allow the traffic HTTP/HTTPS towards the proxy server from every acme subnetwork. Then, we realized that the hosts could also connect to the proxy using different ports, so we noticed that this policy was already satisfied based on the firewall's rules that we had.

   In addition with the configuration that we had, the proxy already had access to internet.

4. **Beside the DNS resolver, the other services in the Internal server network have to be accessible only by hosts of Client and DMZ networks.**

   For this rule, at the beginning, we thought to add blocking rules in output on the SERVERS interface of the Internal firewall, but then we decided to follow a different logic by adding only rules in input to the interfaces of the firewall.

   So we blocked the traffic towards the services of the Internal server network on the interface IPsec of the Internal firewall.

5. **All the hosts (but the Client network hosts) have to use the syslog and the log collector services on the Log server (syslog) and on Graylog server.**

   For this policy we first collected some information about the management of the log in a network.

   Then we discussed about a possible contradiction with the policy of point 4. In fact, in point 4 is said to allow the Client network to access the services of the Internal server network and to deny the access to all the External services network. In point 5, instead, is said that the Client network can't use the log services on the Internal server network while the External services network has to.

For the External services network, we decided to apply the principle of the more restrictive policy, and so we completely blocked the access to the log services on the Log server and on Graylog server.
For the Clients network, we decided to permits the access to the services of the Internal server network, as specified in point 4, but to not allow the log service, as said in point 5.

6. **The Greenbone server has to access all the hosts of the network.**

   By evaluating this policy, we stated that with the configuration of the firewalls that we had, all the network's hosts could already access the Greenbone.

   We decided to enforce only some rules to block the access to the Greenbone from the Internet, since it is not an host of the network.

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

   For this policy, we decided to block the use of ssh to all the hosts (towards all the others) except for the ones in the Client network.

   So the scope of the rule are all the interfaces of both the internal and main firewall, except for the CLIENTS interface of the internal.

8. **All the Client network hosts have only access to external web services (HTTP/HTTPS).**

   Considering all the policies before, we interpreted this one as follows: whenever the Client network hosts want to access some external services (on the Internet) they can only if are web services, so using the protocols HTTP/HTTPS.

9. **Any packet received by the Main Firewall on port 65432 should be redirected to port 80 of the fantasticcoffee host.**

   First of all, we defined the scope of the policy which is, of course, only the Main firewall.

   Then we decided to redirect the traffic using a NAT rule.

10. **The firewalls should protect against IP address spoofing.**

    The idea to protect against spoofing is to specify for each interface of the firewalls linked to a subnetwork of ACME, the source IP address of that specific subnetowrk. In this way, is not possible to create a packet with a source that does not belong to the subnet address, because the firewall will discard it.

11. **All the internal hosts should use the public IP address of the Main Firewall to exit towards the Internet.**

    By evaluating this rule, we noticed that it was already implemented based on the rules we had on the firewall.

    Of course, we needed to test this, as described in Section 4.

12. **The rate of ICMP echo request packets should be limited to 10 Kbit/s.**

    In order to satisfy this policy, we decided to implement a rule in both the Main and Internal firewall.

13. **Anything that is not explicitly allowed has to be denied.**

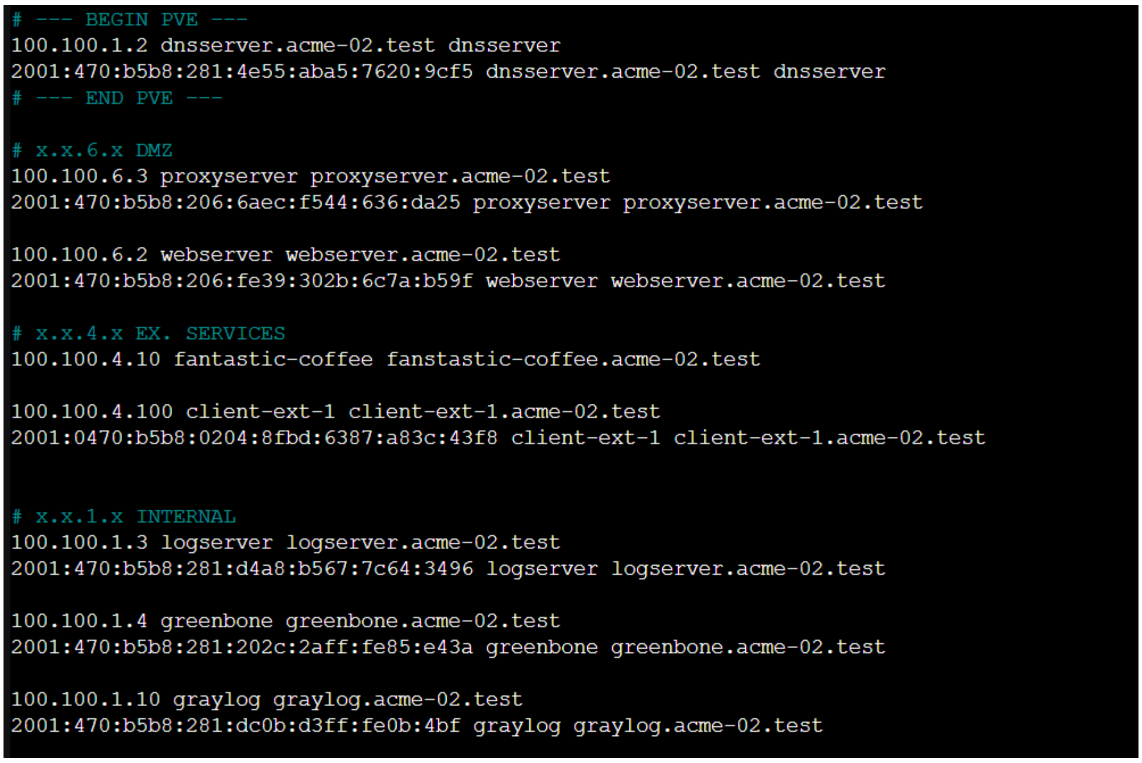
    Since we considered the default block rules of the firewall and allowed only specific traffic to pass, we stated that this policy was already satisfied.

3 Policy implementation in opnsense

1. **All hosts must use the internal DNS Server as a DNS resolver**.

   In order to configure the DNS service, we installed dnsmasq on the *DNS server* (100.100.1.2) and we configured it to manage the acme-02.test domain using the /etc/hosts, file following the guide provided.

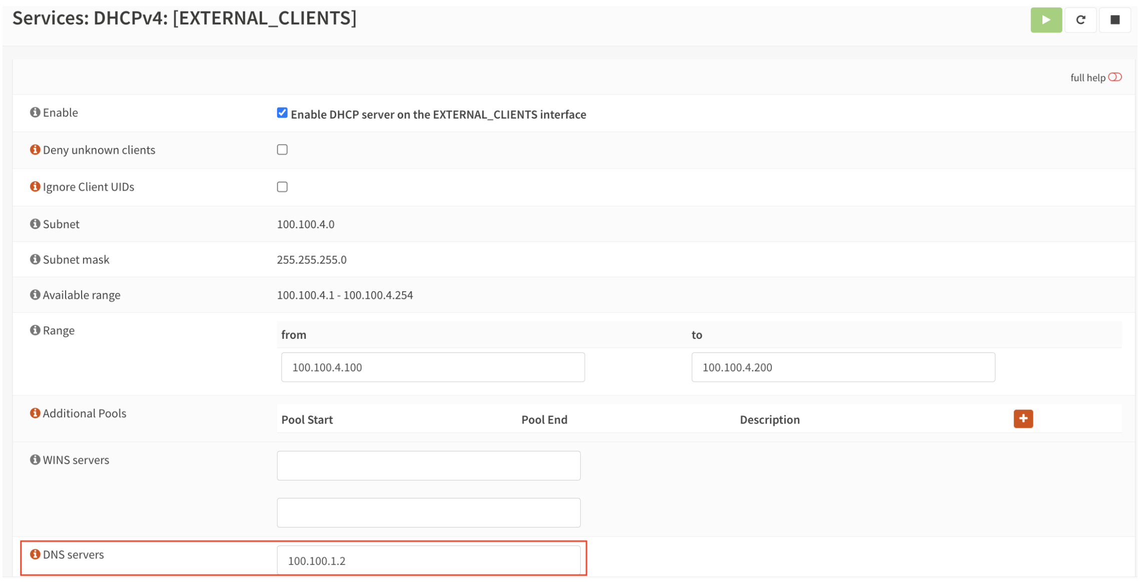
   In the /etc/hosts file, we also added the alias of all the hosts of the network.



```
# --- BEGIN PVE ---
100.100.1.2 dnsserver.acme-02.test dnsserver
2001:470:b5b8:281:4e55:aba5:7620:9cf5 dnsserver.acme-02.test dnsserver
# --- END PVE ---

# x.x.6.x DMZ
100.100.6.3 proxyserver proxyserver.acme-02.test
2001:470:b5b8:206:6aec:f544:636:da25 proxyserver proxyserver.acme-02.test

100.100.6.2 webserver webserver.acme-02.test
2001:470:b5b8:206:fe39:302b:6c7a:b59f webserver webserver.acme-02.test

# x.x.4.x EX. SERVICES
100.100.4.10 fantastic-coffee fanstastic-coffee.acme-02.test

100.100.4.100 client-ext-1 client-ext-1.acme-02.test
2001:0470:b5b8:0204:8fbd:6387:a83c:43f8 client-ext-1 client-ext-1.acme-02.test


# x.x.1.x INTERNAL
100.100.1.3 logserver logserver.acme-02.test
2001:470:b5b8:281:d4a8:b567:7c64:3496 logserver logserver.acme-02.test

100.100.1.4 greenbone greenbone.acme-02.test
2001:470:b5b8:281:202c:2aff:fe85:e43a greenbone greenbone.acme-02.test

100.100.1.10 graylog graylog.acme-02.test
2001:470:b5b8:281:dc0b:d3ff:fe0b:4bf graylog graylog.acme-02.test
```

Figure 2: Part of /etc/hosts file in the webserver

   After that, for each host we configured the DNS server in the /etc/resolv.conf file. Then, we needed to configure the DNS also for the dynamic addresses, namely the VPN and the client-ext-1 host and we did this in OPNsense. For the client-ext-1, in the main firewall, we set the dns:



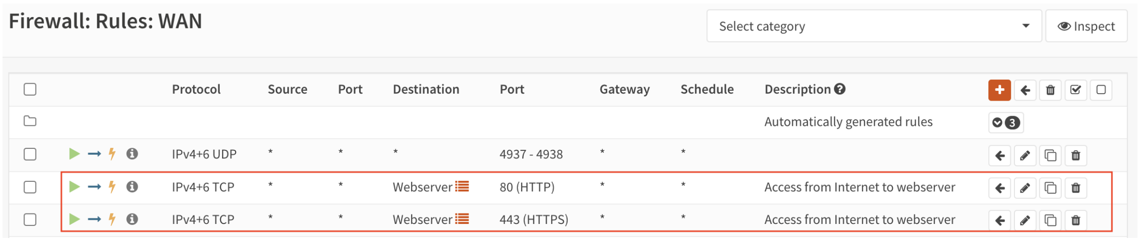
Services: DHCPv4: [EXTERNAL_CLIENTS]

Enable	☑ Enable DHCP server on the EXTERNAL_CLIENTS interface
Deny unknown clients	☐
Ignore Client UIDs	☐
Subnet	100.100.4.0
Subnet mask	255.255.255.0
Available range	100.100.4.1 - 100.100.4.254
Range	from 100.100.4.100 to 100.100.4.200
Additional Pools	Pool Start / Pool End / Description
WINS servers	
DNS servers	100.100.1.2

Figure 3: DNS on client-ext-1

For the VPN, we did something similar, in the section *VPN:OPENvpn:servers*. Finally, we also configured the default DNS in the Main firewall and the Internal firewall, directly in OPNsense, without doing this from the command line.

2. **Only the webserver service provided in the DMZ has to be accessible from the Internet.** In the interface WAN of the Main firewall, we modified rule that we added for the previous assignment and we allowed only the traffic towards the web serve, on the port 80 (HTTP) and 443 (HTTPS).



Firewall: Rules: WAN

Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
							Automatically generated rules
IPv4+6 UDP	*	*	*	4937 - 4938	*	*	
IPv4+6 TCP	*	*	Webserver	80 (HTTP)	*	*	Access from Internet to webserver
IPv4+6 TCP	*	*	Webserver	443 (HTTPS)	*	*	Access from Internet to webserver

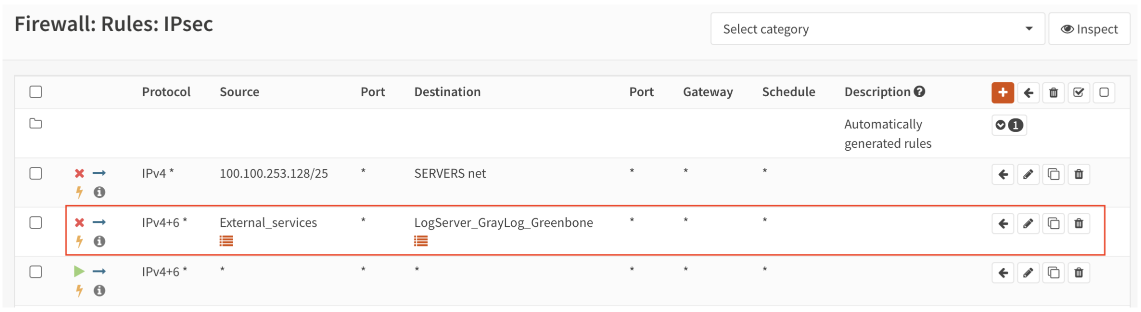
Figure 4: Rules on the WAN interface of the Main firewall

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 (see below, section Services of the ACME co.)**

   As said in the Section 2, we didn't implement additional firewall's rules.

4. **Beside the DNS resolver, the other services in the Internal server network have to be accessible only by hosts of Client and DMZ networks.**

We implemented this policy in the IPsec interface of the Internal firewall. The idea is to block all the traffic coming from the External services newtork (100.100.4.0/24) and directed toward the Log server, Greenbone and Graylog.

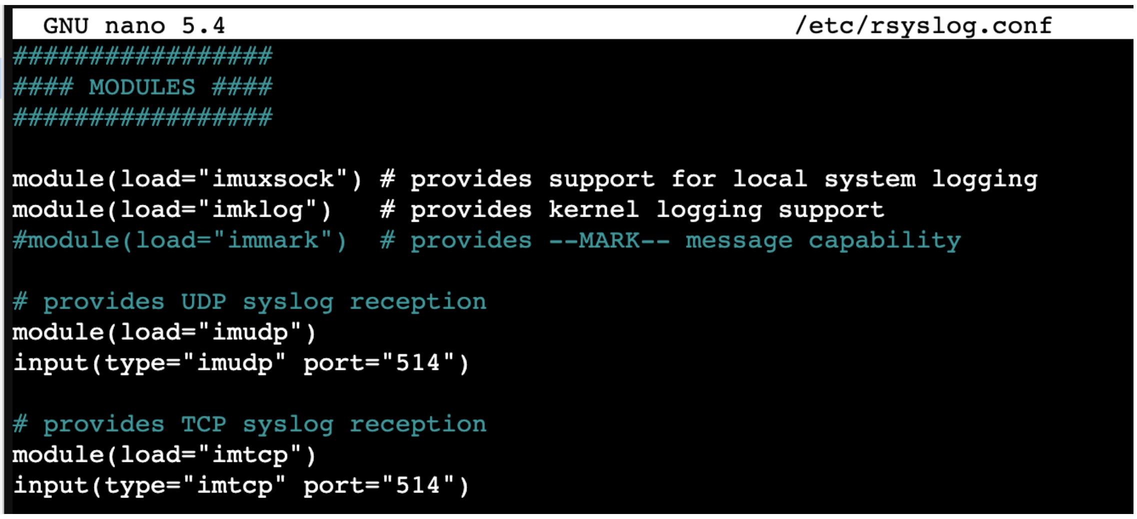

Firewall: Rules: IPsec

	Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
								Automatically generated rules
✖ →	IPv4 *	100.100.253.128/25	*	SERVERS net	*	*	*	
✖ →	IPv4+6 *	External_services	*	LogServer_GrayLog_Greenbone	*	*	*	
▶ →	IPv4+6 *	*	*	*	*	*	*	

Figure 5: Rules on the IPsec interface of the Internal firewall

5. **All the hosts (but the Client network hosts) have to use the syslog and the log collector services on the Log server (syslog) and on Graylog server.**

   First of all, we needed to configure the Log server and the Graylog as rsyslog servers. We did this by modifying the /etc/rsyslog.conf file:

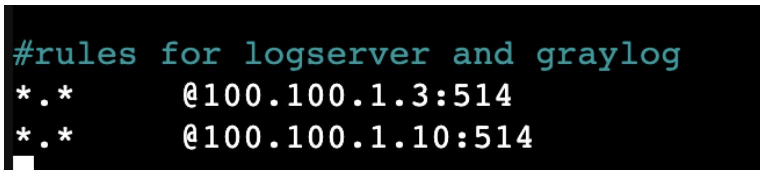


```
  GNU nano 5.4                                 /etc/rsyslog.conf
#################
#### MODULES ####
#################

module(load="imuxsock") # provides support for local system logging
module(load="imklog")   # provides kernel logging support
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

Figure 6: /etc/rsyslog.conf file of Graylog and Logserver.

We uncommented the lines related to the UDP and TCP syslog reception.

Then, for each host (except to the ones in the Client and External services networks) we added to the same file the following lines to configure the log forwarding to both the Log server and the Graylog:



```
#rules for logserver and graylog
*.*       @100.100.1.3:514
*.*       @100.100.1.10:514
```

Figure 7: /etc/rsyslog.conf file of the host that need to send the logs.

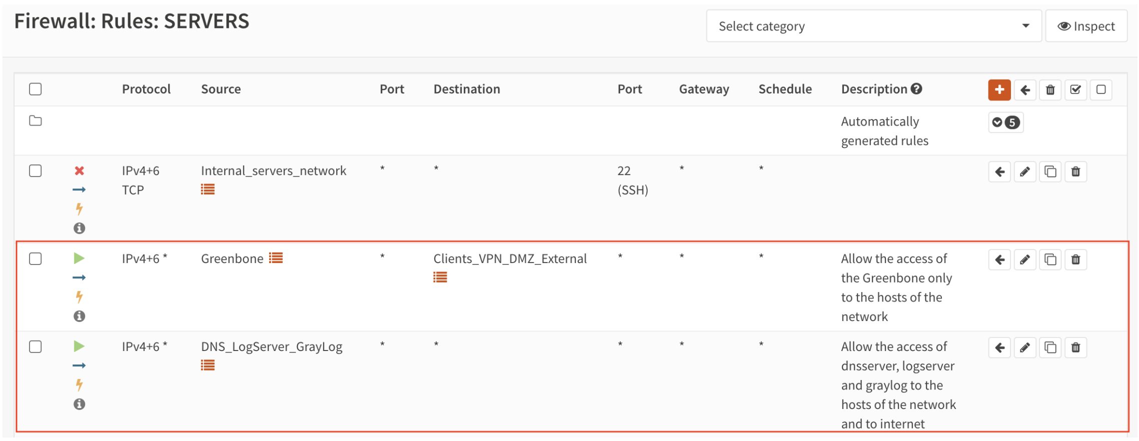
We also configured the log for the Main and the Internal firewall. In this case, we did it in OPNsense, in the section System:Settings:Logging/targets. We added as target machine logserver.acme-02.test listening on port 514.

We didn't add any additional rule on the firewalls, since they already allowed the log service.

6. **The Greenbone server has to access all the hosts of the network.**

   As we said in the Section 2, we only implemented some rules to block the access to the Greenbone from the Internet, since it is not an host of the network. In order to do this, we inserted two rules in the SERVERS interface of the Internal Firewall.

   One allows the Greenbone to access only to the DMZ, VPN, Clients and External services networks, excluding the WAN, and so the access to Internet. The other rule permits, instead, to all the other hosts of the Internal servers network to access all the destinations (so Internet included).



Firewall: Rules: SERVERS

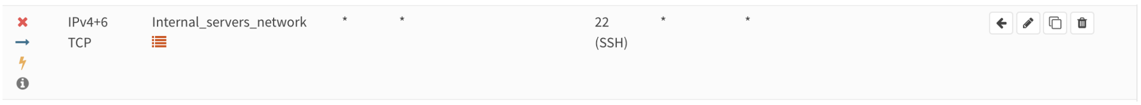
	Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
								Automatically generated rules
✗	IPv4+6 TCP	Internal_servers_network	*	*	22 (SSH)	*	*	
▶	IPv4+6 *	Greenbone	*	Clients_VPN_DMZ_External	*	*	*	Allow the access of the Greenbone only to the hosts of the network
▶	IPv4+6 *	DNS_LogServer_GrayLog	*	*	*	*	*	Allow the access of dnsserver, logserver and graylog to the hosts of the network and to internet

Figure 8: Rules in the SERVERS interface of the Internal firewall.

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

   In order to implement this policy, we added a block rule for each interface of the Main and Internal firewall directly connected to all the hosts subnetworks except for the Client one.

In particular we blocked the traffic with source one of that subnetworks when the destination port is 22, so the one used for ssh.



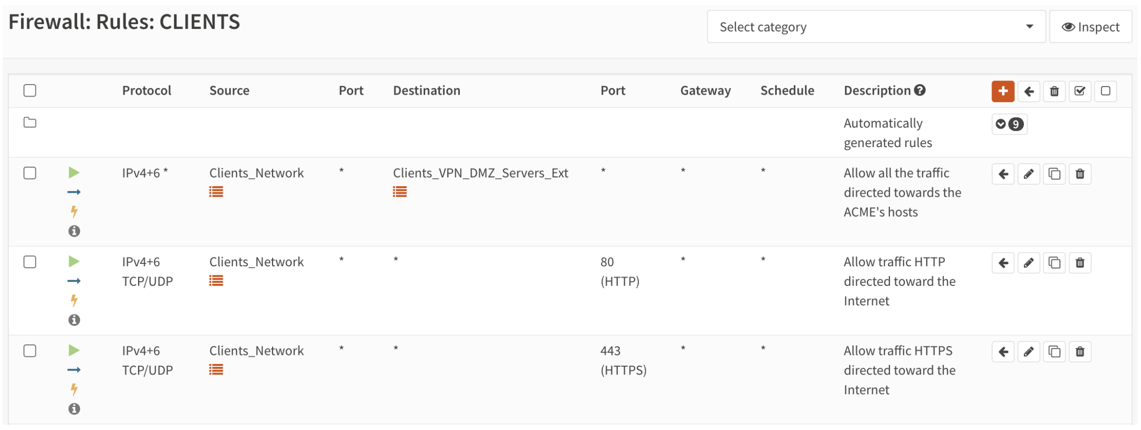
Protocol	Source	Port	Destination	Port	Gateway	Schedule
IPv4+6 TCP	Internal_servers_network	*	*	22 (SSH)	*	*

Figure 9: Examples of rule implemented in the Internal servers network.

8. **All the Client network hosts have only access to external web services (HTTP/HTTPS).**

   We implemented this policy on the CLIENTS interface of the Internal firewall. In particular, we allowed all the traffic (so for any port) only if it is directed toward the hosts of the ACME network.

   Otherwise, if the traffic is directed to the Internet, then it is allowed only using protocols HTTP and HTTPS.



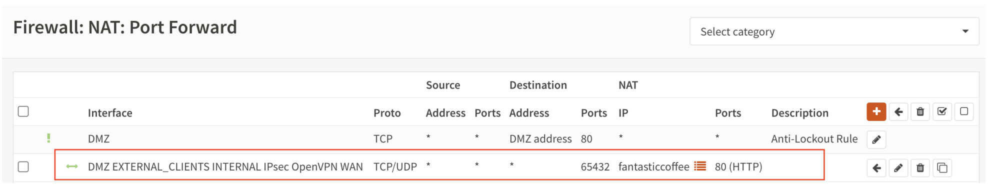
Firewall: Rules: CLIENTS

Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
							Automatically generated rules
IPv4+6 *	Clients_Network	*	Clients_VPN_DMZ_Servers_Ext	*	*	*	Allow all the traffic directed towards the ACME's hosts
IPv4+6 TCP/UDP	Clients_Network	*	*	80 (HTTP)	*	*	Allow traffic HTTP directed toward the Internet
IPv4+6 TCP/UDP	Clients_Network	*	*	443 (HTTPS)	*	*	Allow traffic HTTPS directed toward the Internet

Figure 10: Rules in the CLIENTS interface of the Internal firewall.

9. **Any packet received by the Main Firewall on port 65432 should be redirected to port 80 of the fantasticcoffee host.**

   We implemented this policy as a NAT port forwarding in the Main firewall. Every incoming packet directed to port 65432 of the Main firewall, is redirected to port 80 of fantasticcoffee host.



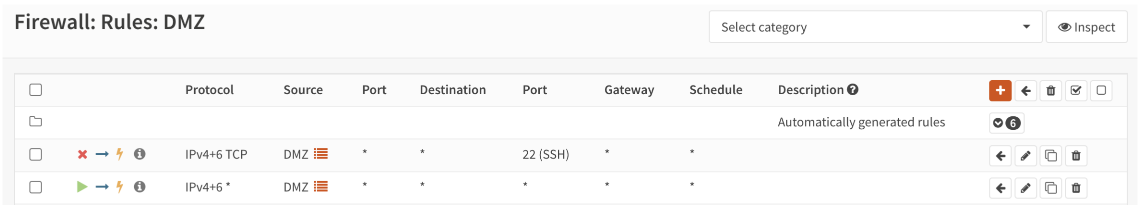
Firewall: NAT: Port Forward

		Source		Destination		NAT		
Interface	Proto	Address	Ports	Address	Ports	IP	Ports	Description
DMZ	TCP	*	*	DMZ address	80	*	*	Anti-Lockout Rule
DMZ EXTERNAL_CLIENTS INTERNAL IPsec OpenVPN WAN	TCP/UDP	*	*	*	65432	fantasticcoffee	80 (HTTP)	

Figure 11: NAT port forwarding

10. **The firewalls should protect against IP address spoofing.**

    For each interface of the two firewalls (connected to an ACME's subnetwork), we inserted rules that accept only packets with that IP source. For example:



Firewall: Rules: DMZ

		Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
									Automatically generated rules
	✗	IPv4+6 TCP	DMZ	*	*	22 (SSH)	*	*	
	▶	IPv4+6 *	DMZ	*	*	*	*	*	

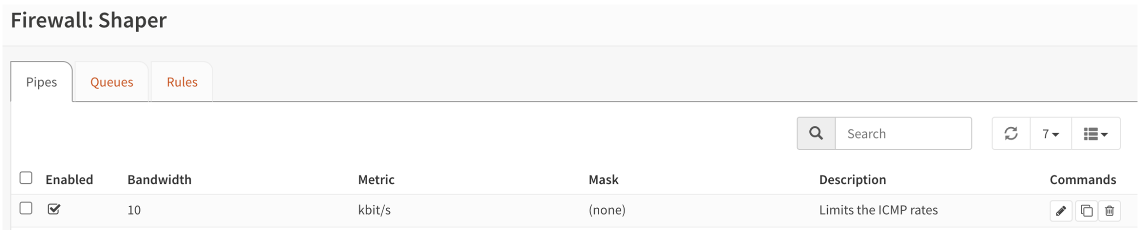
Figure 12: IP address spoofing

11. **All the internal hosts should use the public IP address of the Main Firewall to exit towards the Internet.**

    We didn't need to implement any additional rule for this policy.

12. **The rate of ICMP echo request packets should be limited to 10 Kbit/s.**

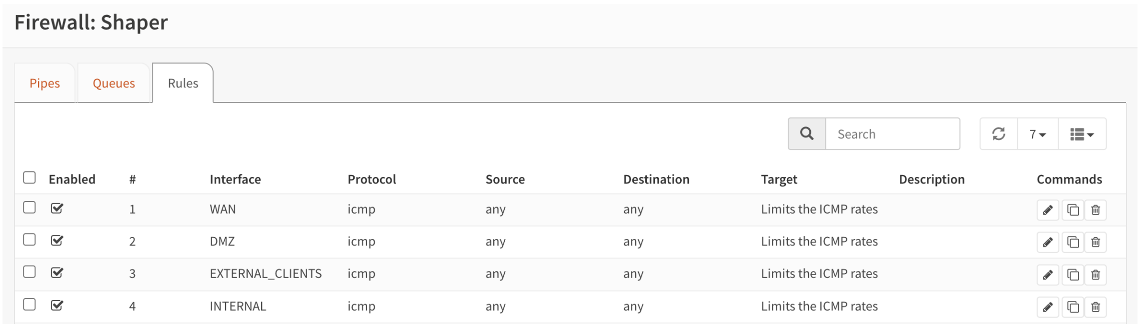
    In order to implement this policy, we first created a pipe to limit the traffic of ICMP packets to 10 Kbits/s, in both the Main and Internal firewall. After that, we created a rule using that pipe for each interface of the firewalls.



Firewall: Shaper

Enabled	Bandwidth	Metric	Mask	Description	Commands
✓	10	kbit/s	(none)	Limits the ICMP rates	

Figure 13: Example of pipe in the Main firewall



Firewall: Shaper

Enabled	#	Interface	Protocol	Source	Destination	Target	Description	Commands
✓	1	WAN	icmp	any	any	Limits the ICMP rates		
✓	2	DMZ	icmp	any	any	Limits the ICMP rates		
✓	3	EXTERNAL_CLIENTS	icmp	any	any	Limits the ICMP rates		
✓	4	INTERNAL	icmp	any	any	Limits the ICMP rates		

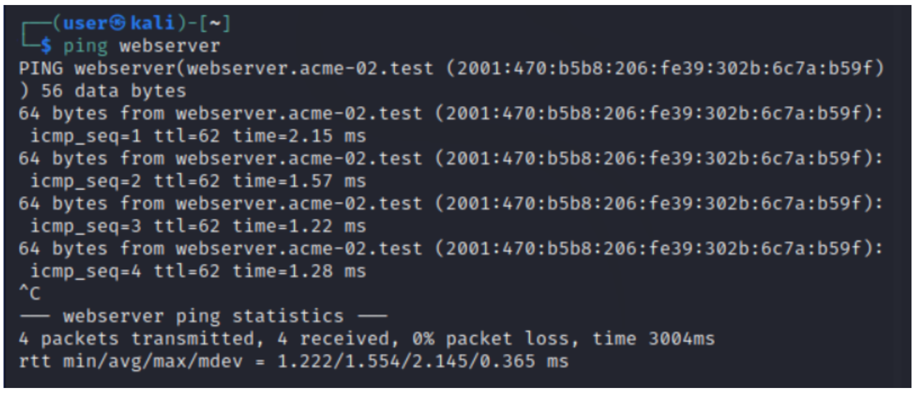
Figure 14: Example of shaper in the Main firewall

13. **Anything that is not explicitly allowed has to be denied.**

    Since we considered the default blocking rules of the firewalls and allowed only the specific policies we wanted, this policy is already satisfied.

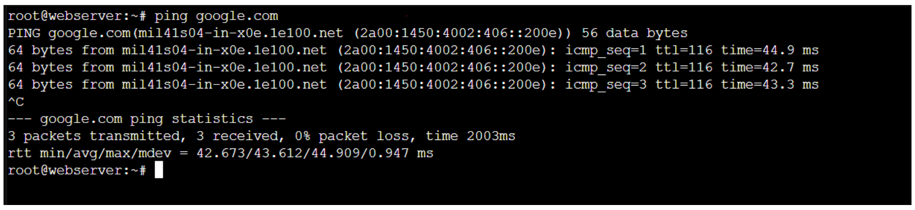
4 Test of the configuration

1. Point 1 was tested firstly trying to ping acme hosts using their name, (see below) and then trying to ping external domains, e.g. ping google.com.



```
┌──(user㉿kali)-[~]
└─$ ping webserver
PING webserver(webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f)
) 56 data bytes
64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f):
 icmp_seq=1 ttl=62 time=2.15 ms
64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f):
 icmp_seq=2 ttl=62 time=1.57 ms
64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f):
 icmp_seq=3 ttl=62 time=1.22 ms
64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f):
 icmp_seq=4 ttl=62 time=1.28 ms
^C
—— webserver ping statistics ——
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.222/1.554/2.145/0.365 ms
```

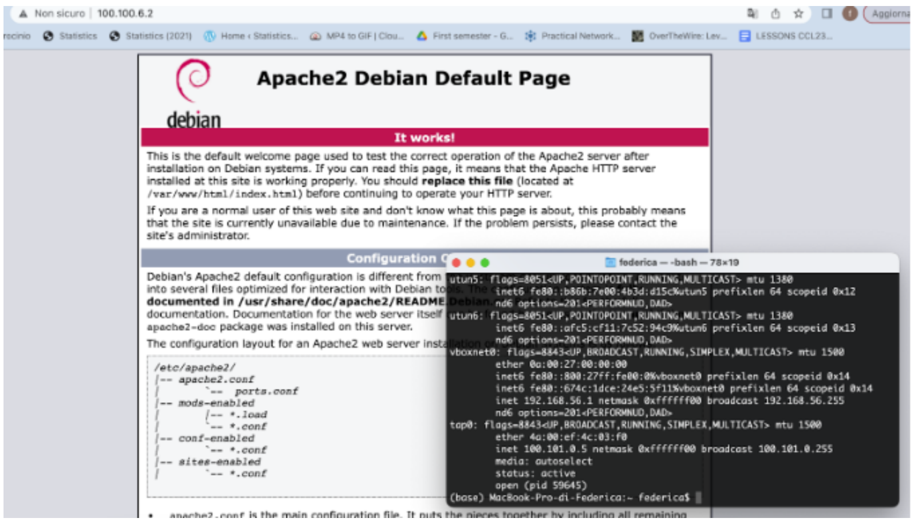
Figure 15: Ping to Webserver by Kali.



```
root@webserver:~# ping google.com
PING google.com(mil41s04-in-x0e.1e100.net (2a00:1450:4002:406::200e)) 56 data bytes
64 bytes from mil41s04-in-x0e.1e100.net (2a00:1450:4002:406::200e): icmp_seq=1 ttl=116 time=44.9 ms
64 bytes from mil41s04-in-x0e.1e100.net (2a00:1450:4002:406::200e): icmp_seq=2 ttl=116 time=42.7 ms
64 bytes from mil41s04-in-x0e.1e100.net (2a00:1450:4002:406::200e): icmp_seq=3 ttl=116 time=43.3 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 42.673/43.612/44.909/0.947 ms
root@webserver:~# 
```

Figure 16: Ping to Google by webserver.

2. Point 2 was tested by accessing from the Internet the web page with address 100.100.6.2 of the webserver.



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration O

Debian's Apache2 default configuration is different from
into several files optimized for interaction with Debian to
documented in /usr/share/doc/apache2/README.Debian.
documentation. Documentation for the web server itself
apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server insta

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

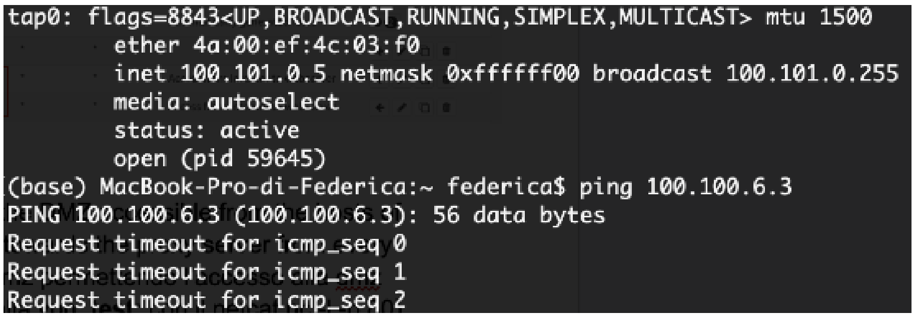
- apache2.conf is the main configuration file. It puts the pieces together by including all remaining

federica — -bash — 78×19

```
utun5: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
	inet6 fe80::b86b:7e00:4b3d:d15c%utun5 prefixlen 64 scopeid 0x12
	nd6 options=201<PERFORMNUD,DAD>
utun6: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
	inet6 fe80::afc5:cf11:7c52:94c9%utun6 prefixlen 64 scopeid 0x13
	nd6 options=201<PERFORMNUD,DAD>
vboxnet0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
	ether 0a:00:27:00:00:00
	inet6 fe80::800:27ff:fe00:0%vboxnet0 prefixlen 64 scopeid 0x14
	inet6 fe80::674c:1dce:24e5:5f11%vboxnet0 prefixlen 64 scopeid 0x14
	inet 192.168.56.1 netmask 0xffffff00 broadcast 192.168.56.255
	nd6 options=201<PERFORMNUD,DAD>
tap0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
	ether 4a:00:ef:4c:03:f0
	inet 100.101.0.5 netmask 0xffffff00 broadcast 100.101.0.255
	media: autoselect
	status: active
	open (pid 59645)
(base) MacBook-Pro-di-Federica:~ federica$ 
```

Figure 17: Web page 100.100.6.2

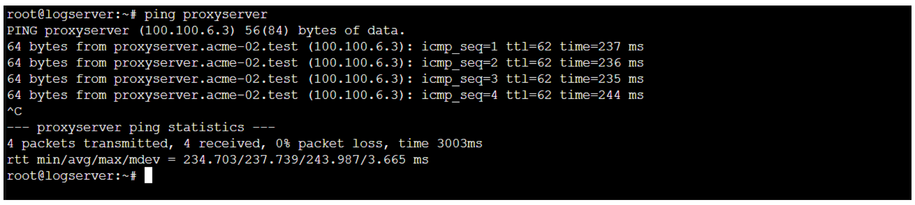
Instead, we see that we have no access to the proxyserver, infact we can't ping neither access the web page.



```
tap0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 4a:00:ef:4c:03:f0
        inet 100.101.0.5 netmask 0xffffff00 broadcast 100.101.0.255
        media: autoselect
        status: active
        open (pid 59645)
(base) MacBook-Pro-di-Federica:~ federica$ ping 100.100.6.3
PING 100.100.6.3 (100.100.6.3): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
```

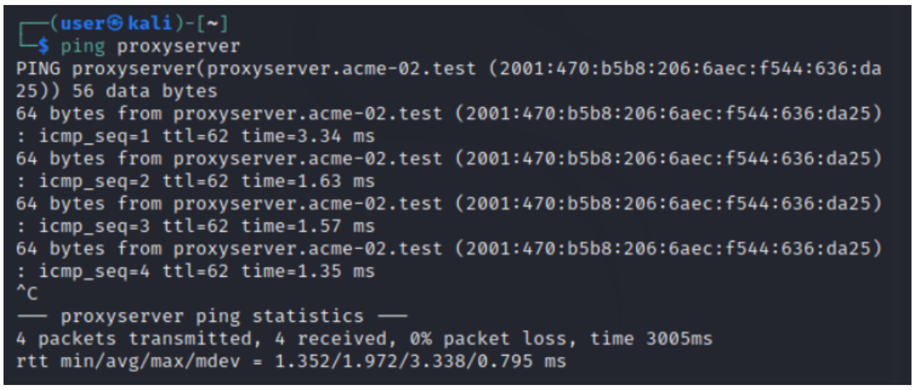
Figure 18: Ping to the proxyserver from an Internet host.

3. Point 3 was tested by verifying, through pings, that different subnets could reach the proxyserver.



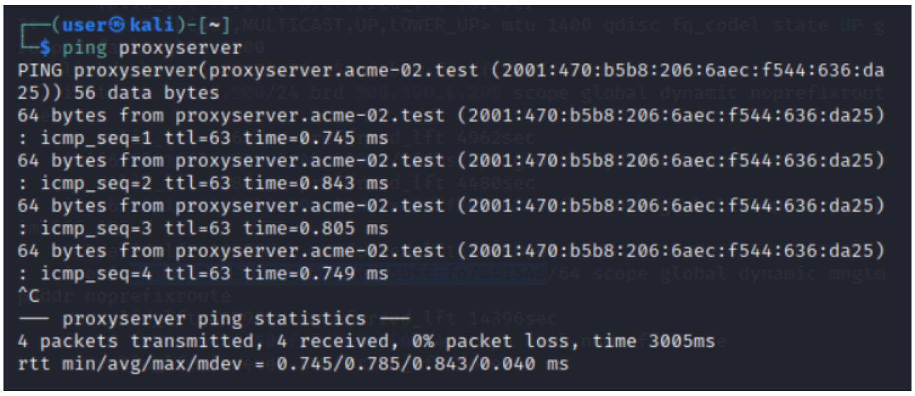
```
root@logserver:~# ping proxyserver
PING proxyserver (100.100.6.3) 56(84) bytes of data.
64 bytes from proxyserver.acme-02.test (100.100.6.3): icmp_seq=1 ttl=62 time=237 ms
64 bytes from proxyserver.acme-02.test (100.100.6.3): icmp_seq=2 ttl=62 time=236 ms
64 bytes from proxyserver.acme-02.test (100.100.6.3): icmp_seq=3 ttl=62 time=235 ms
64 bytes from proxyserver.acme-02.test (100.100.6.3): icmp_seq=4 ttl=62 time=244 ms
^C
--- proxyserver ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 234.703/237.739/243.987/3.665 ms
root@logserver:~#
```

Figure 19: Ping to the proxyserver by the logserver.



```
┌──(user㉿kali)-[~]
└─$ ping proxyserver
PING proxyserver(proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da
25)) 56 data bytes
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=1 ttl=62 time=3.34 ms
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=2 ttl=62 time=1.63 ms
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=3 ttl=62 time=1.57 ms
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=4 ttl=62 time=1.35 ms
^C
── proxyserver ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.352/1.972/3.338/0.795 ms
```

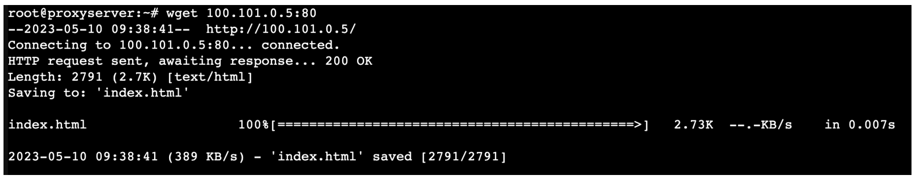
Figure 20: Ping to the proxyserver by Kali.



```
┌──(user㉿kali)-[~]
└─$ ping proxyserver
PING proxyserver(proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da
25)) 56 data bytes
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=1 ttl=63 time=0.745 ms
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=2 ttl=63 time=0.843 ms
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=3 ttl=63 time=0.805 ms
64 bytes from proxyserver.acme-02.test (2001:470:b5b8:206:6aec:f544:636:da25)
: icmp_seq=4 ttl=63 time=0.749 ms
^C
── proxyserver ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.745/0.785/0.843/0.040 ms
```

Figure 21: Ping to the proxyserver from Client-ext-1.

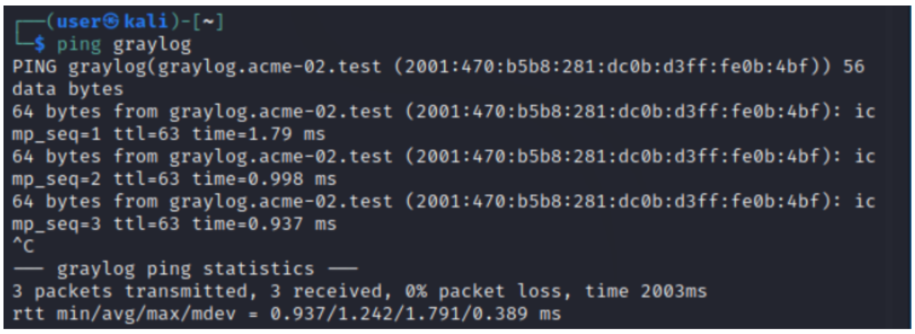
Now, let's verify that the proxyserver has internet access.



```
root@proxyserver:~# wget 100.101.0.5:80
--2023-05-10 09:38:41--  http://100.101.0.5/
Connecting to 100.101.0.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2791 (2.7K) [text/html]
Saving to: 'index.html'

index.html                  100%[===========================================>]   2.73K  --.-KB/s    in 0.007s

2023-05-10 09:38:41 (389 KB/s) - 'index.html' saved [2791/2791]
```

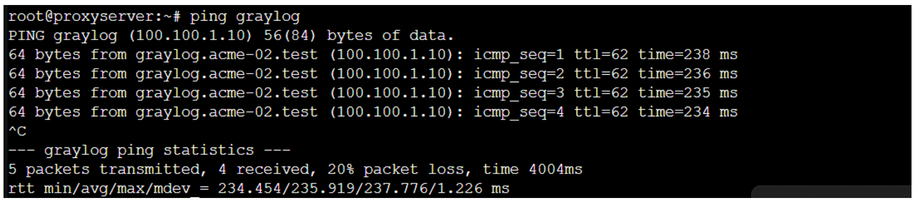
Figure 22: Connection to Internet from proxyserver.

4. The access of the Client Networks and DMZ to the Internal Server hosts (in this case the Graylog) was verified with pings.



```
┌──(user㉿kali)-[~]
└─$ ping graylog
PING graylog(graylog.acme-02.test (2001:470:b5b8:281:dc0b:d3ff:fe0b:4bf)) 56
data bytes
64 bytes from graylog.acme-02.test (2001:470:b5b8:281:dc0b:d3ff:fe0b:4bf): ic
mp_seq=1 ttl=63 time=1.79 ms
64 bytes from graylog.acme-02.test (2001:470:b5b8:281:dc0b:d3ff:fe0b:4bf): ic
mp_seq=2 ttl=63 time=0.998 ms
64 bytes from graylog.acme-02.test (2001:470:b5b8:281:dc0b:d3ff:fe0b:4bf): ic
mp_seq=3 ttl=63 time=0.937 ms
^C
── graylog ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.937/1.242/1.791/0.389 ms
```

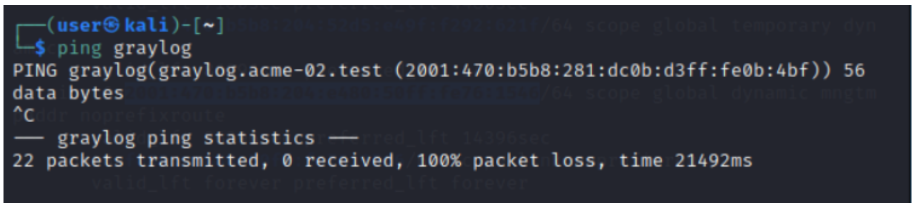
Figure 23: Ping the graylog by Kali (Client network).



```
root@proxyserver:~# ping graylog
PING graylog (100.100.1.10) 56(84) bytes of data.
64 bytes from graylog.acme-02.test (100.100.1.10): icmp_seq=1 ttl=62 time=238 ms
64 bytes from graylog.acme-02.test (100.100.1.10): icmp_seq=2 ttl=62 time=236 ms
64 bytes from graylog.acme-02.test (100.100.1.10): icmp_seq=3 ttl=62 time=235 ms
64 bytes from graylog.acme-02.test (100.100.1.10): icmp_seq=4 ttl=62 time=234 ms
^C
--- graylog ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4004ms
rtt min/avg/max/mdev = 234.454/235.919/237.776/1.226 ms
```

Figure 24: Ping the graylog by proxyserver (DMZ).

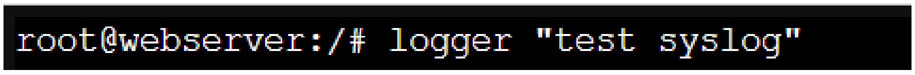
In addition, we verified that hosts in the external services network (in this case the Client-ext-1) **can't** access the Internal Server hosts.



```
┌──(user㉿kali)-[~]
└─$ ping graylog
PING graylog(graylog.acme-02.test (2001:470:b5b8:281:dc0b:d3ff:fe0b:4bf)) 56
data bytes
^C
── graylog ping statistics ──
22 packets transmitted, 0 received, 100% packet loss, time 21492ms
```

Figure 25: Ping to the Graylog by Client-ext-1.

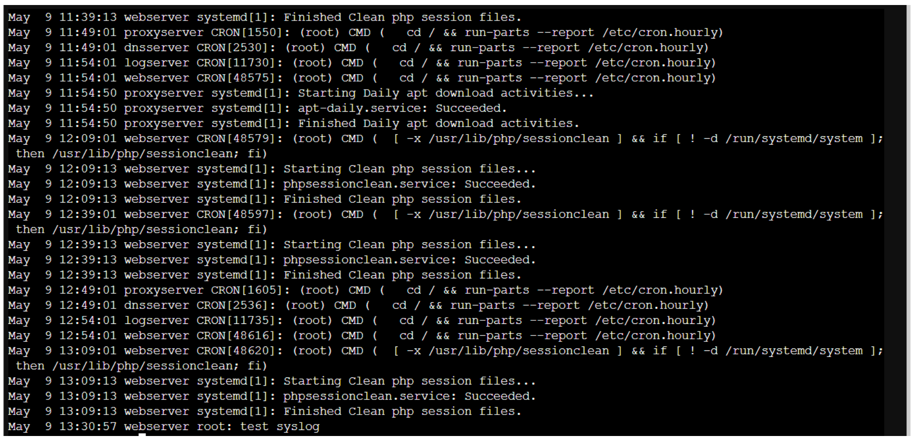
5. This point was tested the use of the syslog and logcollector services on the Logserver by all hosts in the network except those in the Client network. To test this we used the command *logger "text"*.



```
root@webserver:/# logger "test syslog"
```

Figure 26: Command *logger "test syslog"*.

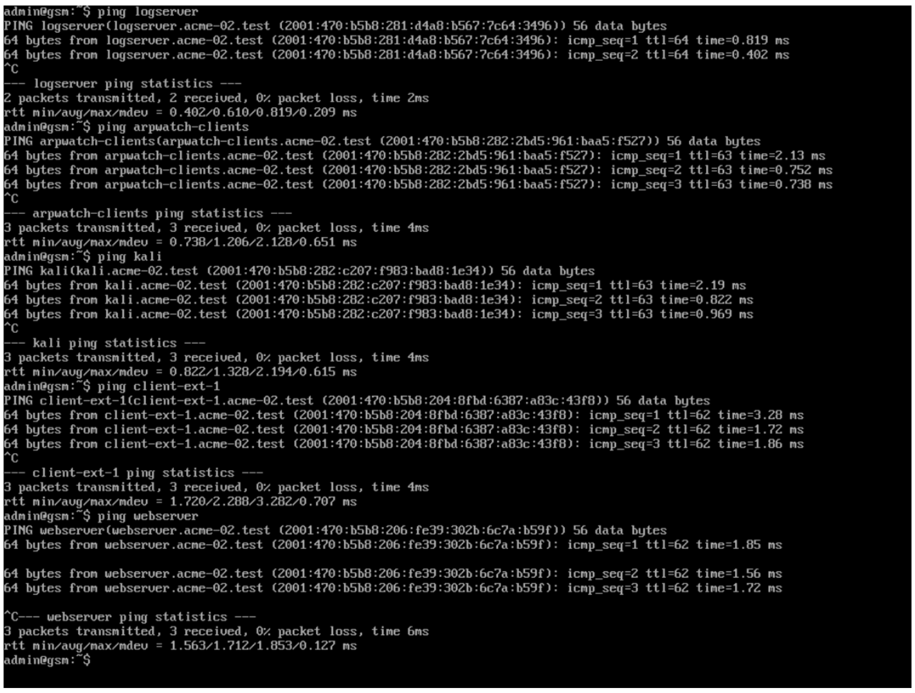
After that, we verified that the log we had just sent was present in the file */var/log/syslog* of the Logserver and Graylog hosts.



```
May  9 11:39:13 webserver systemd[1]: Finished Clean php session files.
May  9 11:49:01 proxyserver CRON[1550]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 11:49:01 dnsserver CRON[2530]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 11:54:01 logserver CRON[11730]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 11:54:01 webserver CRON[48575]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 11:54:50 proxyserver systemd[1]: Starting Daily apt download activities...
May  9 11:54:50 proxyserver systemd[1]: apt-daily.service: Succeeded.
May  9 11:54:50 proxyserver systemd[1]: Finished Daily apt download activities.
May  9 12:09:01 webserver CRON[48579]: (root) CMD (  [ -x /usr/lib/php/sessionclean ] && if [ ! -d /run/systemd/system ];
 then /usr/lib/php/sessionclean; fi)
May  9 12:09:13 webserver systemd[1]: Starting Clean php session files...
May  9 12:09:13 webserver systemd[1]: phpsessionclean.service: Succeeded.
May  9 12:09:13 webserver systemd[1]: Finished Clean php session files.
May  9 12:39:01 webserver CRON[48597]: (root) CMD (  [ -x /usr/lib/php/sessionclean ] && if [ ! -d /run/systemd/system ];
 then /usr/lib/php/sessionclean; fi)
May  9 12:39:13 webserver systemd[1]: Starting Clean php session files...
May  9 12:39:13 webserver systemd[1]: phpsessionclean.service: Succeeded.
May  9 12:39:13 webserver systemd[1]: Finished Clean php session files.
May  9 12:49:01 proxyserver CRON[1605]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 12:49:01 dnsserver CRON[2536]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 12:54:01 logserver CRON[11735]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 12:54:01 webserver CRON[48616]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
May  9 13:09:01 webserver CRON[48620]: (root) CMD (  [ -x /usr/lib/php/sessionclean ] && if [ ! -d /run/systemd/system ];
 then /usr/lib/php/sessionclean; fi)
May  9 13:09:13 webserver systemd[1]: Starting Clean php session files...
May  9 13:09:13 webserver systemd[1]: phpsessionclean.service: Succeeded.
May  9 13:09:13 webserver systemd[1]: Finished Clean php session files.
May  9 13:30:57 webserver root: test syslog
```

Figure 27: /var/log/syslog of the Logserver.

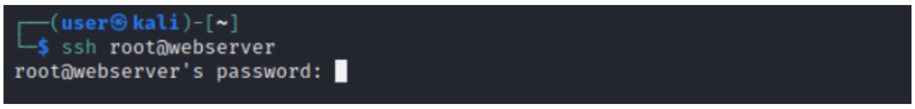
6. This point was tested trying to ping all ACME's hosts from the Greenbone.



```
admin@gsm:~$ ping logserver
PING logserver(logserver.acme-02.test (2001:470:b5b8:281:d4a8:b567:7c64:3496)) 56 data bytes
64 bytes from logserver.acme-02.test (2001:470:b5b8:281:d4a8:b567:7c64:3496): icmp_seq=1 ttl=64 time=0.819 ms
64 bytes from logserver.acme-02.test (2001:470:b5b8:281:d4a8:b567:7c64:3496): icmp_seq=2 ttl=64 time=0.402 ms
^C
--- logserver ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 0.402/0.610/0.819/0.209 ms
admin@gsm:~$ ping arpwatch-clients
PING arpwatch-clients(arpwatch-clients.acme-02.test (2001:470:b5b8:282:2bd5:961:baa5:f527)) 56 data bytes
64 bytes from arpwatch-clients.acme-02.test (2001:470:b5b8:282:2bd5:961:baa5:f527): icmp_seq=1 ttl=63 time=2.13 ms
64 bytes from arpwatch-clients.acme-02.test (2001:470:b5b8:282:2bd5:961:baa5:f527): icmp_seq=2 ttl=63 time=0.752 ms
64 bytes from arpwatch-clients.acme-02.test (2001:470:b5b8:282:2bd5:961:baa5:f527): icmp_seq=3 ttl=63 time=0.738 ms
^C
--- arpwatch-clients ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4ms
rtt min/avg/max/mdev = 0.738/1.206/2.128/0.651 ms
admin@gsm:~$ ping kali
PING kali(kali.acme-02.test (2001:470:b5b8:282:c207:f983:bad8:1e34)) 56 data bytes
64 bytes from kali.acme-02.test (2001:470:b5b8:282:c207:f983:bad8:1e34): icmp_seq=1 ttl=63 time=2.19 ms
64 bytes from kali.acme-02.test (2001:470:b5b8:282:c207:f983:bad8:1e34): icmp_seq=2 ttl=63 time=0.822 ms
64 bytes from kali.acme-02.test (2001:470:b5b8:282:c207:f983:bad8:1e34): icmp_seq=3 ttl=63 time=0.969 ms
^C
--- kali ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4ms
rtt min/avg/max/mdev = 0.822/1.328/2.194/0.615 ms
admin@gsm:~$ ping client-ext-1
PING client-ext-1(client-ext-1.acme-02.test (2001:470:b5b8:204:8fbd:6387:a83c:43f8)) 56 data bytes
64 bytes from client-ext-1.acme-02.test (2001:470:b5b8:204:8fbd:6387:a83c:43f8): icmp_seq=1 ttl=62 time=3.28 ms
64 bytes from client-ext-1.acme-02.test (2001:470:b5b8:204:8fbd:6387:a83c:43f8): icmp_seq=2 ttl=62 time=1.72 ms
64 bytes from client-ext-1.acme-02.test (2001:470:b5b8:204:8fbd:6387:a83c:43f8): icmp_seq=3 ttl=62 time=1.86 ms
^C
--- client-ext-1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4ms
rtt min/avg/max/mdev = 1.720/2.288/3.282/0.707 ms
admin@gsm:~$ ping webserver
PING webserver(webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f)) 56 data bytes
64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f): icmp_seq=1 ttl=62 time=1.85 ms

64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f): icmp_seq=2 ttl=62 time=1.56 ms
64 bytes from webserver.acme-02.test (2001:470:b5b8:206:fe39:302b:6c7a:b59f): icmp_seq=3 ttl=62 time=1.72 ms

^C--- webserver ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6ms
rtt min/avg/max/mdev = 1.563/1.712/1.853/0.127 ms
admin@gsm:~$
```

Figure 28: Ping to all hosts by Greenbone.

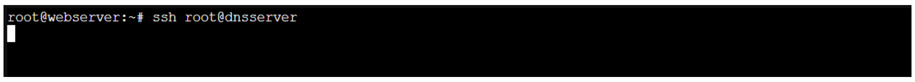
7. This point was tested performing an ssh access from the Client network hosts (in this case from Kali) to other hosts of the network. We can see that the ssh request from the Client network hosts is working, since the password is requested:



```
┌──(user㉿kali)-[~]
└─$ ssh root@webserver
root@webserver's password: 
```

Figure 29: ssh to webserver from Kali.

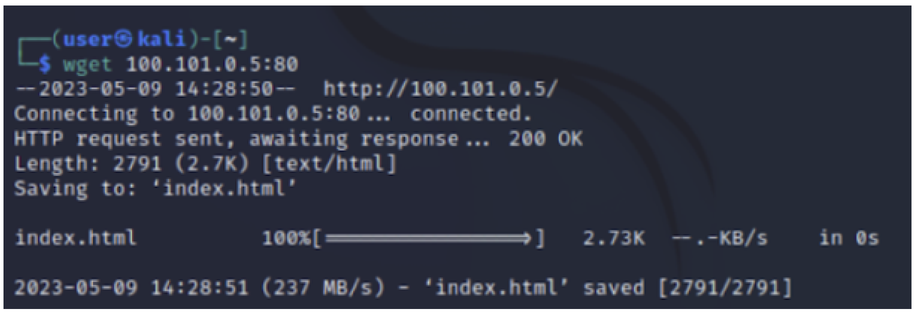
Instead the webserver (in the DMZ) can't use ssh:



```
root@webserver:~# ssh root@dnsserver
```

Figure 30: ssh connection to dnsserver from webserver.

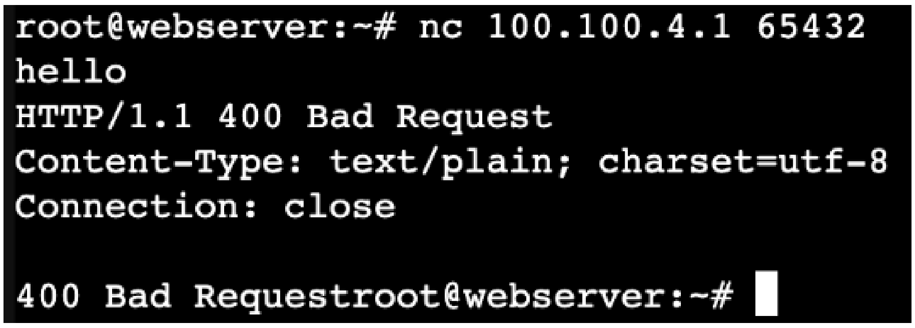
8. This point was tested via the *wget* command to an Internet address on port 80.



```
┌──(user㉿kali)-[~]
└─$ wget 100.101.0.5:80
--2023-05-09 14:28:50--  http://100.101.0.5/
Connecting to 100.101.0.5:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 2791 (2.7K) [text/html]
Saving to: 'index.html'

index.html          100%[=================>]   2.73K  --.-KB/s    in 0s

2023-05-09 14:28:51 (237 MB/s) - 'index.html' saved [2791/2791]
```

Figure 31: Access to external web services (HTTP/HTTPS).

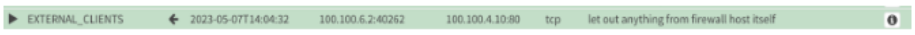
9. This point was tested using nc 100.100.4.1 on port 65432 and checking that it was redirected to port 80 of fantasticcoffee.



```
root@webserver:~# nc 100.100.4.1 65432
hello
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=utf-8
Connection: close

400 Bad Requestroot@webserver:~#
```

Figure 32: nc 100.100.4.1 on port 65432

We've also seen it from the log:



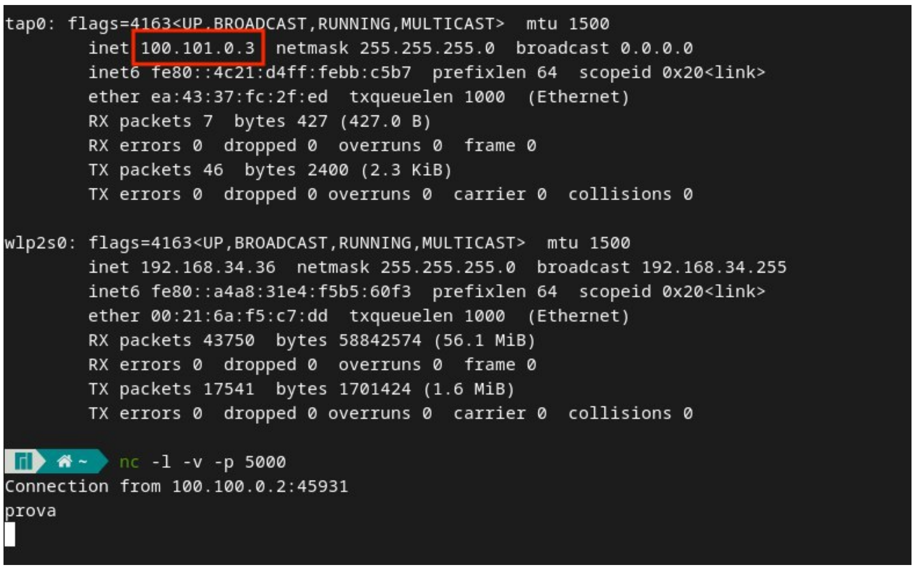
```
▶ EXTERNAL_CLIENTS   ← 2023-05-07T14:04:32   100.100.6.2:40262   100.100.4.10:80   tcp   let out anything from firewall host itself
```

Figure 33: Log Files

So the traffic from the webserver is redirected to 100.100.4.10 on port 80.

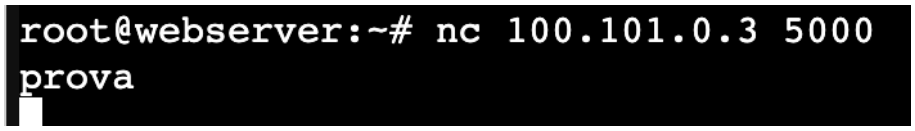
10. Point 10 was tested checking that specifying the source in the firewall rules actually worked.

11. Point 11 was tested via a ncat from the webserver to an internet host: we see that the host does not see the address 100.100.6.2 of the web server but the address 100.100.0.2, which is of the WAN interface of the main firewall.



```
tap0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 100.101.0.3  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::4c21:d4ff:febb:c5b7  prefixlen 64  scopeid 0x20<link>
        ether ea:43:37:fc:2f:ed  txqueuelen 1000  (Ethernet)
        RX packets 7  bytes 427 (427.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 46  bytes 2400 (2.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.34.36  netmask 255.255.255.0  broadcast 192.168.34.255
        inet6 fe80::a4a8:31e4:f5b5:60f3  prefixlen 64  scopeid 0x20<link>
        ether 00:21:6a:f5:c7:dd  txqueuelen 1000  (Ethernet)
        RX packets 43750  bytes 58842574 (56.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17541  bytes 1701424 (1.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

 ~  nc -l -v -p 5000
Connection from 100.100.0.2:45931
prova
```

Figure 34: Netcat command in the internet host



```
root@webserver:~# nc 100.101.0.3 5000
prova
```

Figure 35: Netcat command in the webserver

12. Point 12 was tested by the command *ping -s 1300 -i 0.01 100.100.6.2* from a WAN host.

13. Point 13 didn't require testing.

5 Final remarks

The implementation of some policies required by the assignment was a little bit challenging, in particular the points related to the dns implementation and on the log services. We also needed to interpret some policies and decided what was the better choice.

We've also implemented IPv6, which required to adapt some of the rules that we have already done for the first assignment. Anyway, the implementation of IPv6 gived us the opportunity to better understand its properties and configurations.