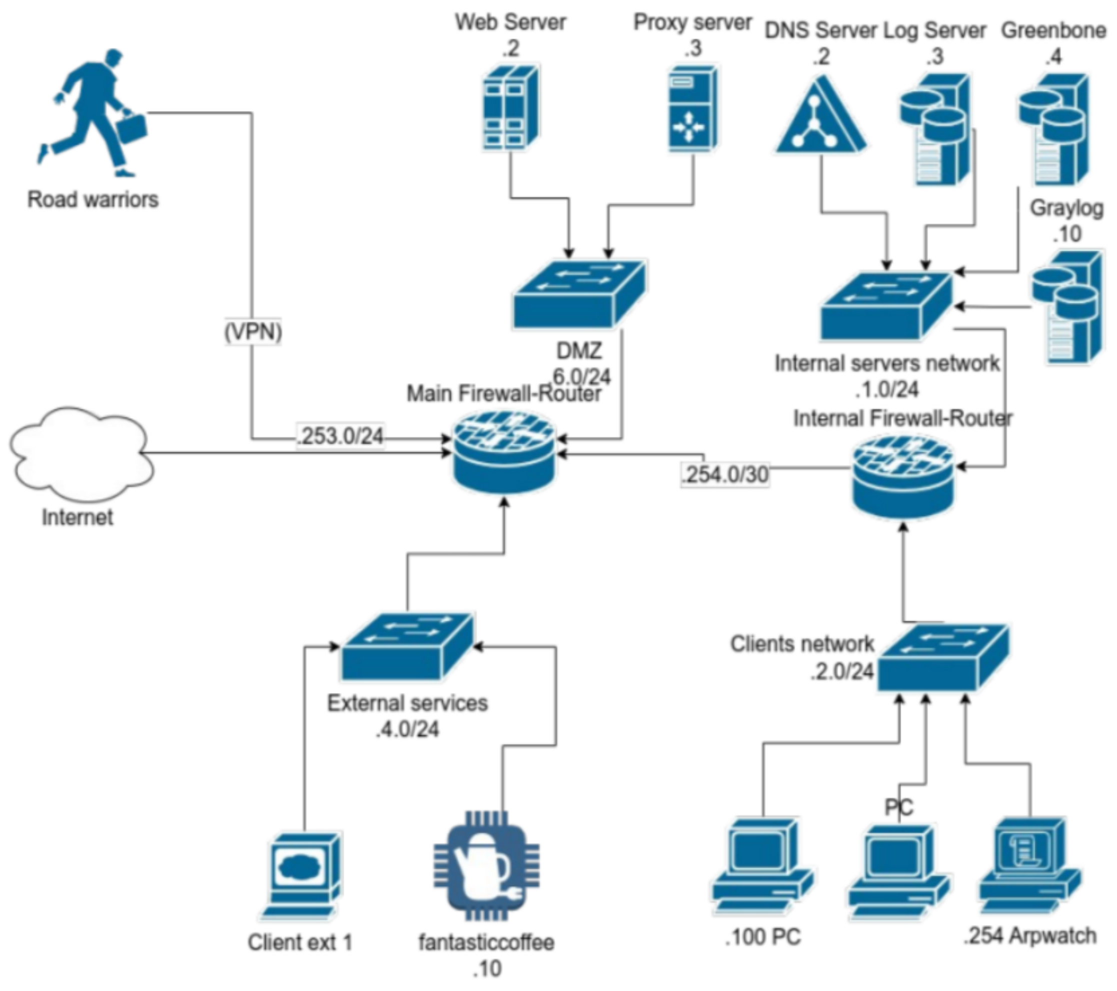
Assignment 2 Group [redacted]
Student names and numbers:

- [redacted]
- [redacted]
- [redacted]
- [redacted]

The network infrastructure in which we will perform the assignments requested is the following:

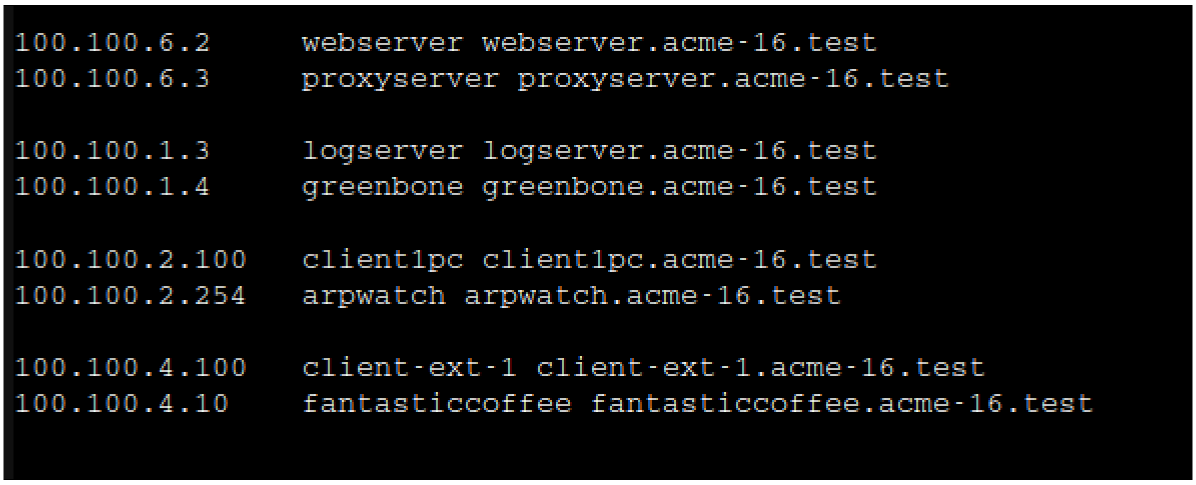


All hosts must use the internal DNS Server as a DNS resolver.

We installed dnsmasq on the DNS server and edited his configuration file /etc/dnsmasq.conf with the following changes:
Uncomment bogus/priv
Uncomment local=/localnet/
local=/acme-16.test/
Uncomment listen-address
listen-address=100.100.1.2,127.0.0.1
Uncomment no-dhcp-interface
no-dhcp-interface=eth0
Uncomment bind-interfaces
Uncomment expand-hosts

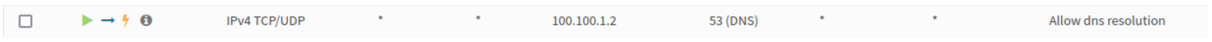
Uncomment domain
domain=acme-16.test

These changes allow the dns server to listen on his interface eth0 (whose ip is 100.100.1.2) and serve as a dns resolver for every host that requests this service. Now we have to tell the DNS what to resolve. To do so we edited his /etc/hosts file inserting all the hosts names we wanted to be resolvable. The content of /etc/host is the following:



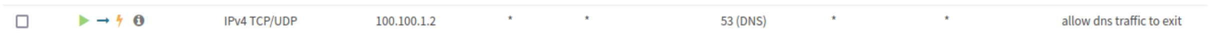
```
100.100.6.2     webserver webserver.acme-16.test
100.100.6.3     proxyserver proxyserver.acme-16.test

100.100.1.3     logserver logserver.acme-16.test
100.100.1.4     greenbone greenbone.acme-16.test

100.100.2.100   client1pc client1pc.acme-16.test
100.100.2.254   arpwatch arpwatch.acme-16.test

100.100.4.100   client-ext-1 client-ext-1.acme-16.test
100.100.4.10    fantasticcoffee fantasticcoffee.acme-16.test
```

At this point simply restart the dnsmasq service with the command *systemctl restart dnsmasq.service* and the DNS will be up and running correctly. We now just have to edit the hosts' resolv.conf files to tell them that every dns query must be requested to the DNS server. To do that we could edit every resolv.conf file by hand but this is a tedious process and we can automate a part of it. To automatically execute this action and make it persistent after every reboot of the host, we edited in the opnsense gui of both routers the sections: Services->DHCPv4->EXTERNAL_CLIENTS and Services->DHCPv4->CLIENTS by adding the ip 100.100.1.2 in the DNS boxes and adding the same ip in System->Settings->general. This allows opnsense to overwrite the resolv.conf file of all the machines in the interfaces modified at every reboot. We should do it in the interfaces DMZ and Servers too but we noticed that for these subnets this solution does not work. We supposed that the servers above have a different kind of startup configuration so we simply edited by hand their resolv.conf. We thought that for these services this is not a problem because they should be always up.
Now it's necessary that we let the dns queries go through the routers without being blocked by the firewalls. To do so add the following rule on all the interfaces we want to allow the traffic towards the dns server:



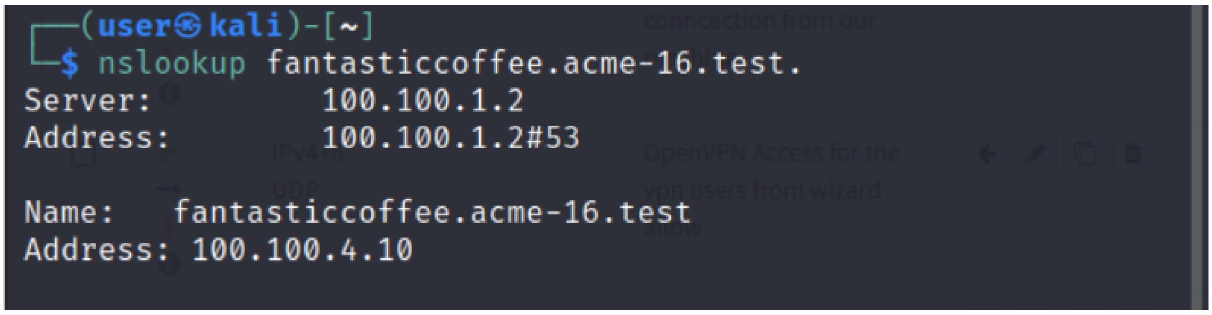
IPv4 TCP/UDP	*	*	100.100.1.2	53 (DNS)	*	*	Allow dns resolution

and in the interface Server insert the following rule to allow dns traffic to exit from the Servers' subnet:



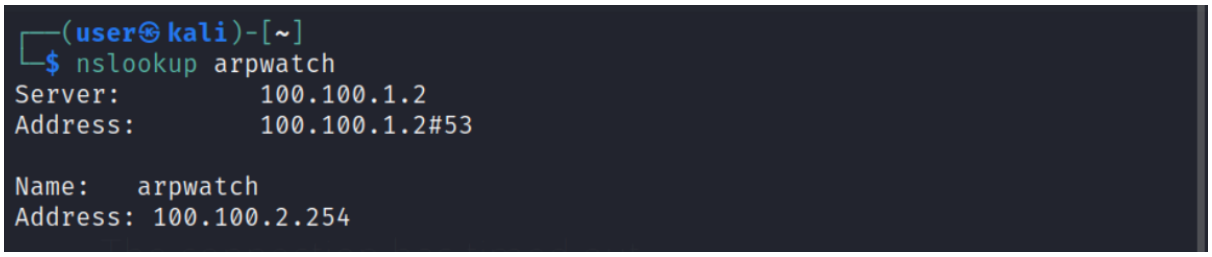
IPv4 TCP/UDP	100.100.1.2	*	*	53 (DNS)	*	*	allow dns traffic to exit

To test this we used the command nslookup to discover which was the dns resolver used. In these two pictures we made the request both from Clients and External networks:



```
┌──(user㉿kali)-[~]
└─$ nslookup fantasticcoffee.acme-16.test.
Server:         100.100.1.2
Address:        100.100.1.2#53

Name:   fantasticcoffee.acme-16.test
Address: 100.100.4.10
```

Screenshot taken from 100.100.2.100

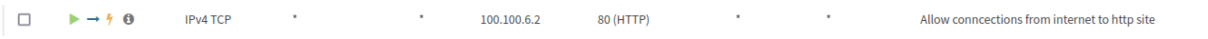


```
┌──(user㉿kali)-[~]
└─$ nslookup arpwatch
Server:         100.100.1.2
Address:        100.100.1.2#53

Name:   arpwatch
Address: 100.100.2.254
```

Screenshot taken from 100.100.4.100

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

To do this we simply added the following inbound rule in Firewall->Rules->Wan

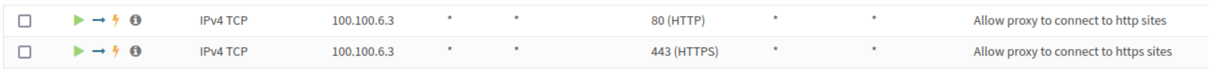


☐	▶ → ϟ ℹ	IPv4 TCP	*	*	100.100.6.2	80 (HTTP)	*	*	Allow conncections from internet to http site

This allows connections from the users of the internet (coming from the Wan interface of the main router) to the HTTP websites hosted on the web server in the DMZ. Since the default inbound rule is Drop, every other connection attempt gets denied.

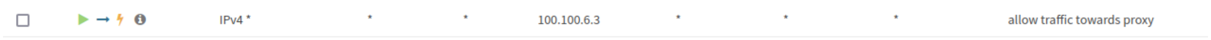
The proxy service provided in the DMZ has to be accessible only from the hosts of the Acme network. However, the proxy needs internet access.

To do so we simply added the following rules in Firewall->Rules->DMZ



☐	▶ → ϟ ℹ	IPv4 TCP	100.100.6.3	*	*	80 (HTTP)	*	*	Allow proxy to connect to http sites
☐	▶ → ϟ ℹ	IPv4 TCP	100.100.6.3	*	*	443 (HTTPS)	*	*	Allow proxy to connect to https sites

These two rules allow packets coming from the proxy server to exit from the interface and travel towards the internet.
To allow every host of the network to connect to the proxy server we simply add the following rule on every interface of both routers (EXTERNAL_CLIENTS, CLIENTS and SERVERS):

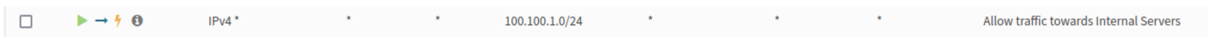


☐	▶ → ϟ ℹ	IPv4 *	*	*	100.100.6.3	*	*	*	allow traffic towards proxy

Everything from the internet is already dropped by default because opnsense drops everything not explicitly allowed.

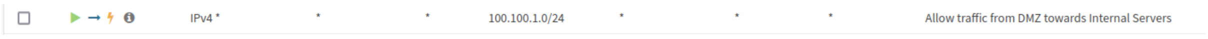
Beside the DNS resolver, the other services in the Internal server network have to be accessible only by hosts of Client and DMZ networks.

We added this rule in Firewall->Rules->CLIENTS



☐	▶ → ϟ ℹ	IPv4 *	*	*	100.100.1.0/24	*	*	*	Allow traffic towards Internal Servers

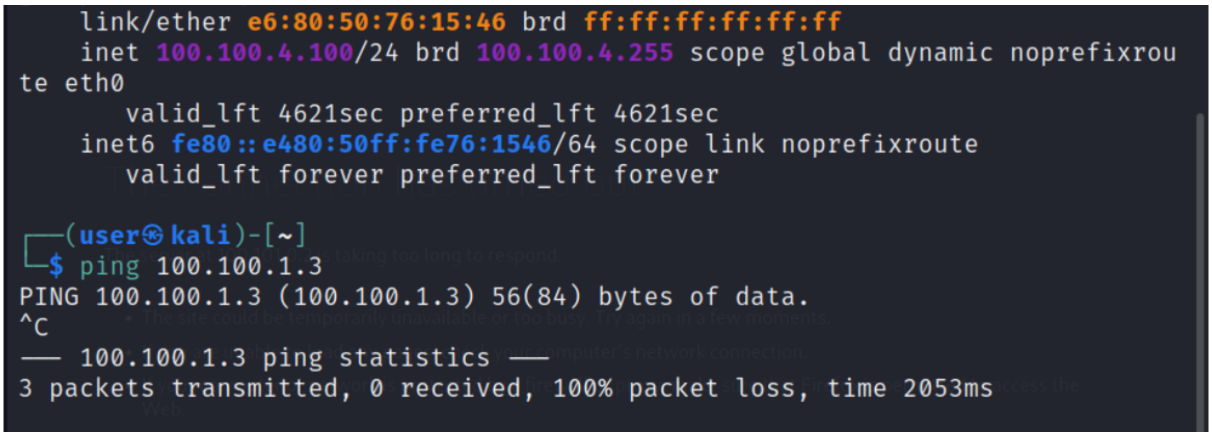
and this one in Firewall->Rules->DMZ



IPv4 *   *   *   100.100.1.0/24   *   *   *   Allow traffic from DMZ towards Internal Servers

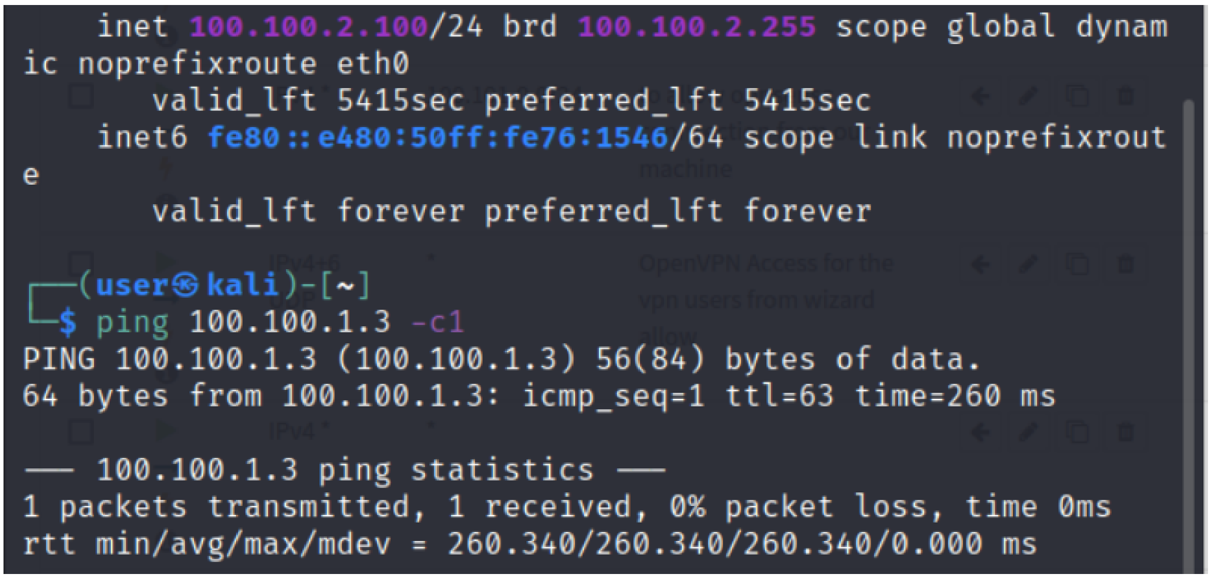
These rules allow all kind of traffic from Clients and Dmz to go towards the Internal Server while External Clients cannot reach the Internal Servers since the default inbound rule in Opnsense is “drop” so every attempt to connect from External Clients to Internal Server is immediately dropped by the interface EXTERNAL_CLIENTS of the main firewall.
We can also delete the previous rules that allow dns traffic to exit from CLIENTS and DMZ interfaces because the above rules contain the DNS ones.

To test these rules, we tried to ping from client_ext_1(ip: 100.100.4.100), which is in the External services network, without any success, as expected.



```
    link/ether e6:80:50:76:15:46 brd ff:ff:ff:ff:ff:ff
    inet 100.100.4.100/24 brd 100.100.4.255 scope global dynamic noprefixrou
te eth0
       valid_lft 4621sec preferred_lft 4621sec
    inet6 fe80::e480:50ff:fe76:1546/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(user㉿kali)-[~]
└─$ ping 100.100.1.3
PING 100.100.1.3 (100.100.1.3) 56(84) bytes of data.
^C
── 100.100.1.3 ping statistics ──
3 packets transmitted, 0 received, 100% packet loss, time 2053ms
```

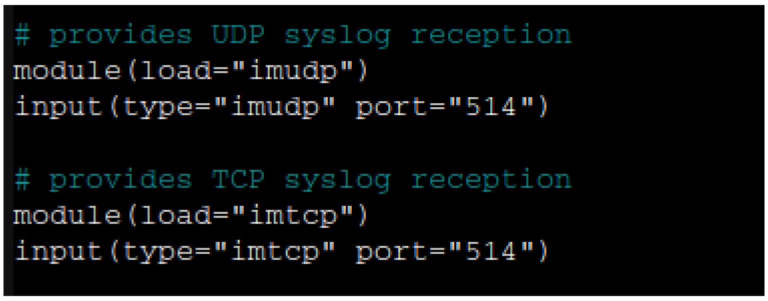
Then we did the same from the Clients network, and it worked as expected:



```
    inet 100.100.2.100/24 brd 100.100.2.255 scope global dynam
ic noprefixroute eth0
       valid_lft 5415sec preferred_lft 5415sec
    inet6 fe80::e480:50ff:fe76:1546/64 scope link noprefixrout
e
       valid_lft forever preferred_lft forever

┌──(user㉿kali)-[~]
└─$ ping 100.100.1.3 -c1
PING 100.100.1.3 (100.100.1.3) 56(84) bytes of data.
64 bytes from 100.100.1.3: icmp_seq=1 ttl=63 time=260 ms

── 100.100.1.3 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 260.340/260.340/260.340/0.000 ms
```

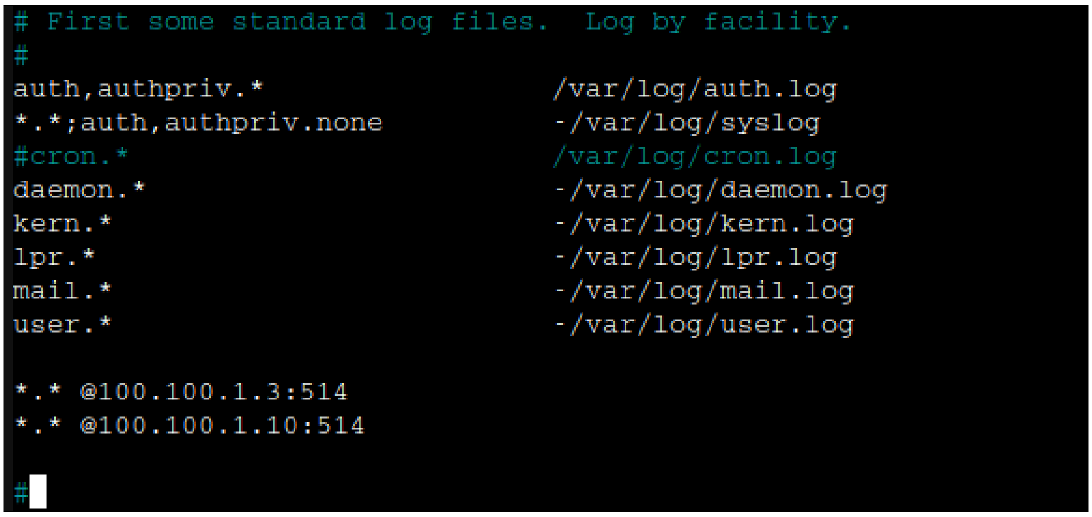
All the hosts (but the Client network hosts) have to use the syslog and the log collector services on the Log server (syslog) and on Graylog server.

To configure the log server to receive logs from clients using rsyslog, we modified the /etc/rsyslog.conf file on the log server and add the following lines:



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

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

The above lines enable rsyslog to listen for incoming TCP and UDP syslog messages on port 514.
To configure the clients to send logs to the log server, we modified the /etc/rsyslog.conf file on each client and add the following lines:



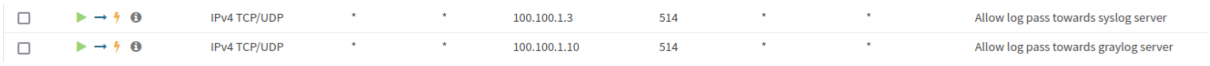
```
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log

*.* @100.100.1.3:514
*.* @100.100.1.10:514

#
```

The above line forwards all syslog messages (*.*) to the log servers' IP address on port 514.

Once we have made the changes to the rsyslog.conf files on the log server and clients, we restarted the rsyslog service on each system for the changes to take effect:

systemctl restart rsyslog

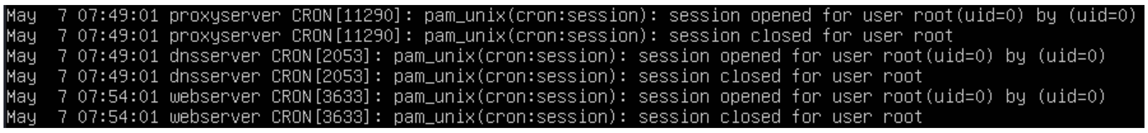
The exact same thing must be done for the graylog server. Now we simply have to allow logs to pass through the routers. Add the following firewall rules on EXTERNAL_CLIENTS to allow the log sending towards the syslog server and the graylog server:



☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	*	*	100.100.1.3	514	*	*	Allow log pass towards syslog server
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	*	*	100.100.1.10	514	*	*	Allow log pass towards graylog server

The above rules are not needed on the DMZ and CLIENTS interfaces because there we allowed all the traffic towards SERVERS in the previous point.
Note: Greenbone is a log analyzer so it won't need to collect and send logs to the others log servers so we didn't edit his rsyslog.conf file.

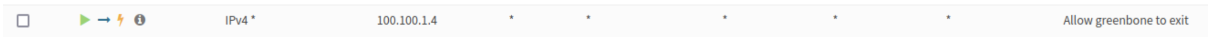
To test that all works fine we generated events on all the machines by simply requesting access to the root user and verifying that these attempts of connection are recorded in /var/log/ of the Log server and Graylog server. For example the following image shows some root sessions we opened on the proxyserver, the dns server and the webserver and as we can see all of them are stored in /var/log/auth.log of log server and graylog server.



```
May  7 07:49:01 proxyserver CRON[11290]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
May  7 07:49:01 proxyserver CRON[11290]: pam_unix(cron:session): session closed for user root
May  7 07:49:01 dnsserver CRON[2053]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
May  7 07:49:01 dnsserver CRON[2053]: pam_unix(cron:session): session closed for user root
May  7 07:54:01 webserver CRON[3633]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
May  7 07:54:01 webserver CRON[3633]: pam_unix(cron:session): session closed for user root
```

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

Simply add the following rule to allow exiting traffic from Greenbone in the SERVERS interface of the internal router

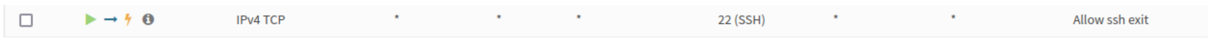


☐	▶ → ϟ ℹ	IPv4 *	100.100.1.4	*	*	*	*	*	Allow greenbone to exit

To test this we can simply verify that the Greenbone server reaches every other machine in the network without being blocked and that is the case. In fact we can reach every webpage or attempt to connect with ssh to any machine.

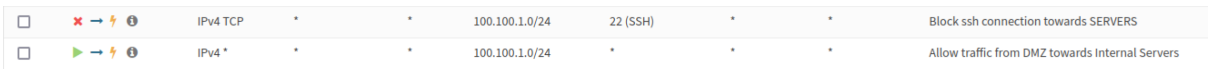
All network hosts have to be managed via ssh, only from hosts within the Client network.

We added an inbound rule on CLIENTS to let ssh connection to exit (destination all network):



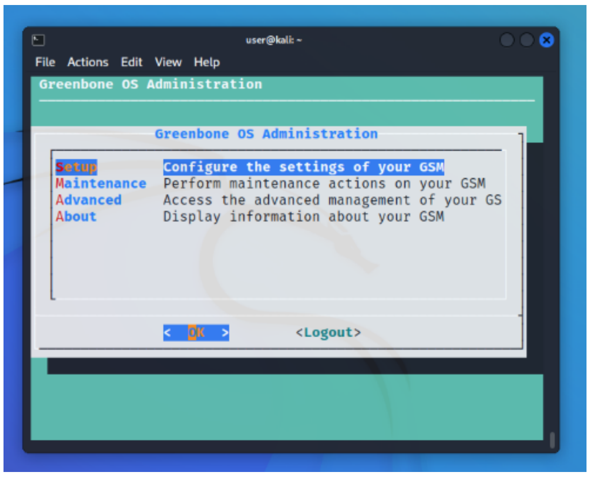
☐	▶ → ϟ ℹ	IPv4 TCP	*	*	*	22 (SSH)	*	*	Allow ssh exit

and an inbound rule on DMZ blocking exiting ssh connections with SERVERS as destination interface (this rule is mandatory because DMZ has the “allow all connection towards SERVERS” for the previous point). So now the rules on the DMZ interface become:



☐	✗ → ϟ ℹ	IPv4 TCP	*	*	100.100.1.0/24	22 (SSH)	*	*	Block ssh connection towards SERVERS
☐	▶ → ϟ ℹ	IPv4 *	*	*	100.100.1.0/24	*	*	*	Allow traffic from DMZ towards Internal Servers

Note that we have to make sure the block rule is above the “allow all” so we make sure that ssh packets are already blocked before being evaluated by the “allow all” rule.
For EXTERNAL_CLIENTS instead we allowed just the log pass towards SERVERS so we don’t have to block explicitly ssh (since it is already blocked by default).
To test this we enabled the ssh service on greenbone (the only one in which we have admin access and not root) and we established an ssh connection from kali.acme:

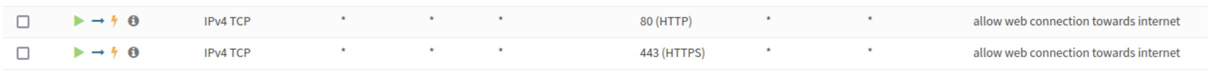


```
Greenbone OS Administration

Setup        Configure the settings of your GSM
Maintenance  Perform maintenance actions on your GSM
Advanced     Access the advanced management of your GS
About        Display information about your GSM
```

While trying the same from any other machine outside CLIENTS, it fails.

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

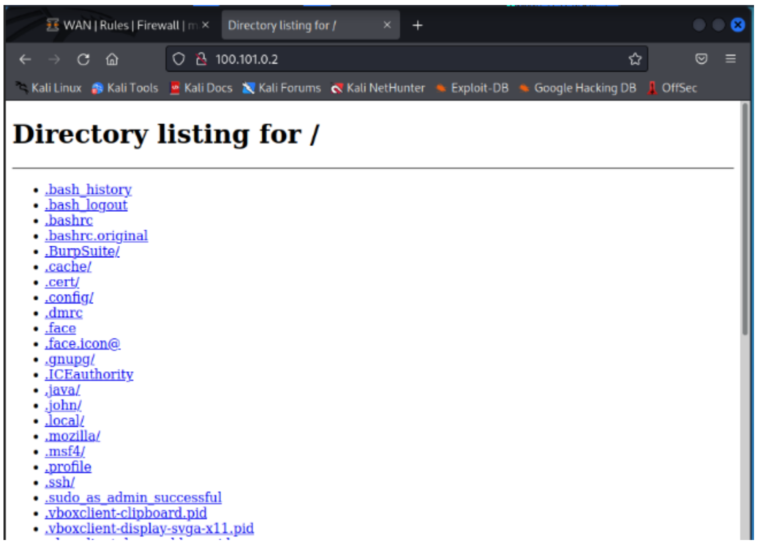
We added the following rules to the CLIENTS interface:



☐	▶ → ϟ ❶	IPv4 TCP	*	*	*	80 (HTTP)	*	*		allow web connection towards internet
☐	▶ → ϟ ❶	IPv4 TCP	*	*	*	443 (HTTPS)	*	*		allow web connection towards internet

These allow every attempt to connect to a web service both within or without the network to pass.

To test this we opened a web server on our personal kali machine (100.101.0.2) with the command *python3 -m http.server 80* and we requested the connection on this site from kali.acme (result is shown in the next image). This allows us to verify that hosts inside the network can reach outside websites. Obviously every attempt to connect to internal websites succeeded too.
Every other attempt of connection from CLIENTS using a different protocol from HTTP or HTTPS is immediately blocked in the CLIENTS interface since the default inbound rule is "drop".

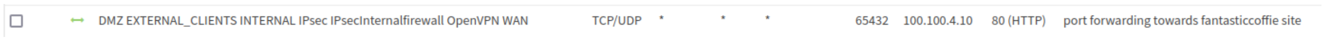


Directory listing for /

- .bash_history
- .bash_logout
- .bashrc
- .bashrc.original
- .BurpSuite/
- .cache/
- .cert/
- .config/
- .dmrc
- .face
- .face.icon@
- .gnupg/
- .ICEauthority
- .java/
- .john/
- .local/
- .mozilla/
- .msf4/
- .profile
- .ssh/
- .sudo_as_admin_successful
- .vboxclient-clipboard.pid
- .vboxclient-display-svga-x11.pid

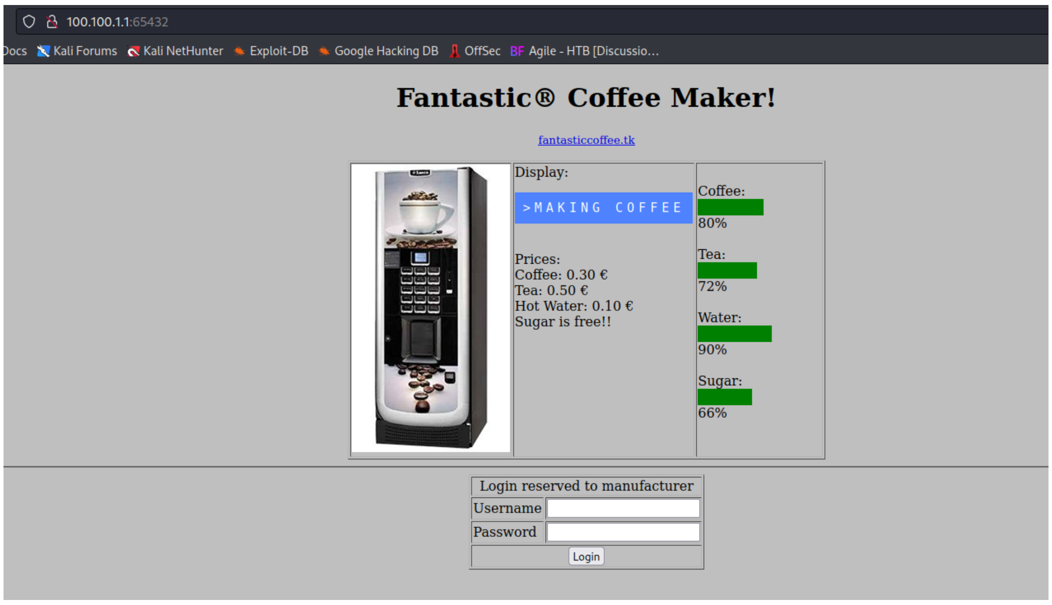
Any packet received by the Main Firewall on port 65432 should be redirected to port 80 of the fantasticcoffee host.

This is a simple port forwarding rule. In fact we added a rule on the main router in Firewall->NAT->PortForwarding



☐	⟷	DMZ EXTERNAL_CLIENTS INTERNAL IPsec IPsecInternalfirewall OpenVPN WAN	TCP/UDP	*	*	*	65432	100.100.4.10	80 (HTTP)	port forwarding towards fantasticcoffie site

This allows every packet entering in any interface of the router coming from any source and port 65432 as destination port to be redirected to the host 100.100.4.10 (fantasticcoffe machine) on the port 80 where its website is hosted.
To test this we can request from the web browser of our machine to reach 100.100.x.x:65432 (it's enough that the packet goes through the main firewall) and we get redirected to the fantastic coffee site. This is because as soon as we pass through the main router, the port forwarding rule is triggered and we get redirected.


100.100.1.1:65432

Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Agile - HTB [Discussio...

Fantastic® Coffee Maker!

fantasticcoffee.tk

Display:

>MAKING COFFEE

Prices:
Coffee: 0.30 €
Tea: 0.50 €
Hot Water: 0.10 €
Sugar is free!!

Coffee: 80%
Tea: 72%
Water: 90%
Sugar: 66%

Login reserved to manufacturer

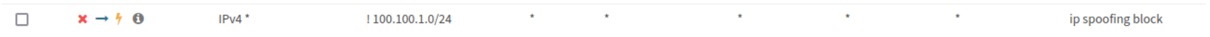
Username	
Password	

Login

This obviously can be done from any machine in the acme network.

The firewalls should protect against IP address spoofing.

To do so we blocked the exit of every packet having the source address different from the one of the source interface on which the rule is evaluated. This means that in SERVERS for example, we inserted at the top of all the rules, a block rule for all the packets having the source different from 100.100.1.0/24 (which is the ip of the subnet linked to SERVERS interface of the internal router). The rule is the following:



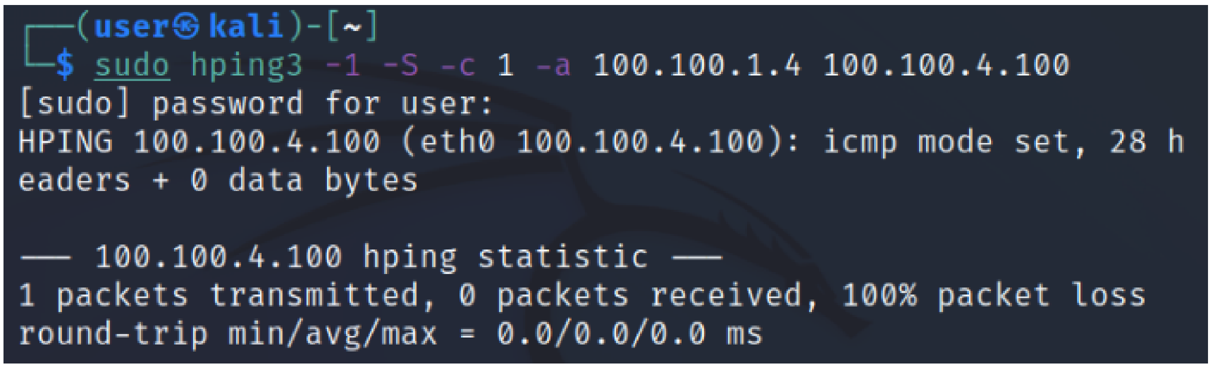
☐	✖ → ϟ ℹ	IPv4 *	! 100.100.1.0/24	*	*	*	*	*	ip spoofing block

Obviously this rule has been inserted on every interface of both the routers modifying the source ip accordingly as we described above.

To test this we tried to ping from kali.acme (in the CLIENTS subnet) client_ext_1 as if we were Greenbone (so outside the CLIENTS subnet) with the command:

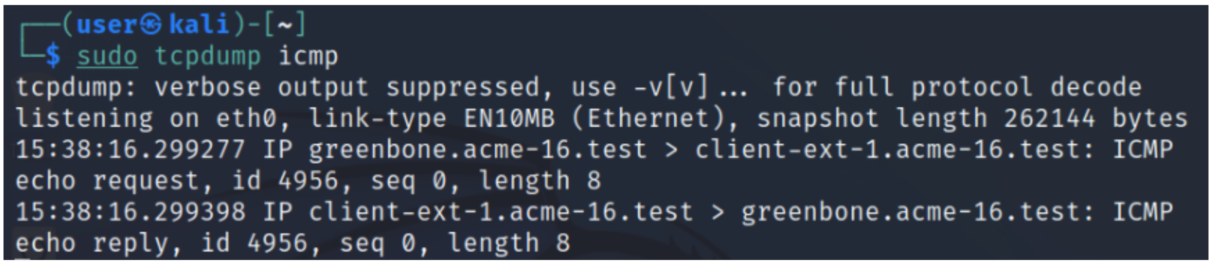
hping3 -1 -c 1 -a 100.100.1.4 100.100.4.100

This command will send one (-c 1) icmp packet (-1 flag) to 100.100.4.100 with 100.100.1.4 as source. Executing it without the above firewall rules this is the result:



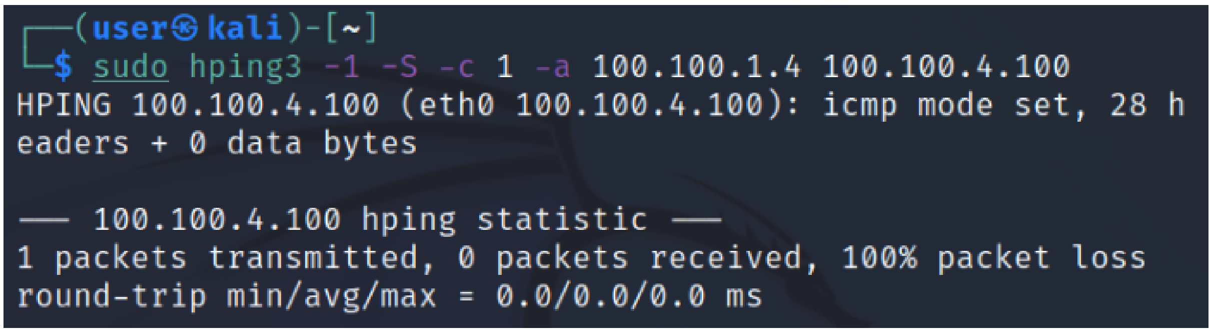
```
┌──(user㉿kali)-[~]
└─$ sudo hping3 -1 -S -c 1 -a 100.100.1.4 100.100.4.100
[sudo] password for user:
HPING 100.100.4.100 (eth0 100.100.4.100): icmp mode set, 28 h
eaders + 0 data bytes

--- 100.100.4.100 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

One packet is transmitted but 0 packets received. Let's do a tcpdump on the target (100.100.4.100) to verify what happened.



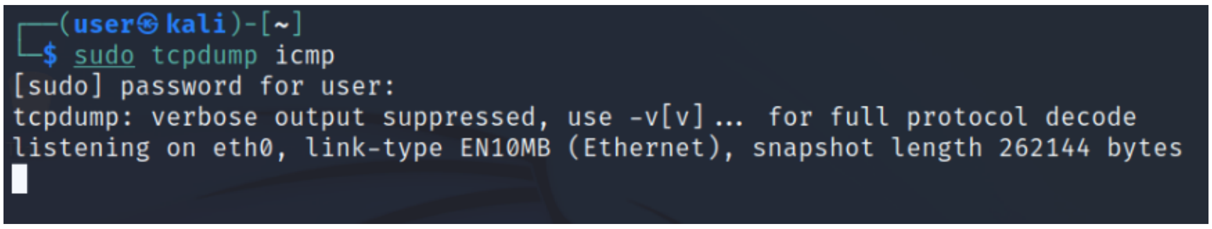
```
┌──(user㉿kali)-[~]
└─$ sudo tcpdump icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:38:16.299277 IP greenbone.acme-16.test > client-ext-1.acme-16.test: ICMP
echo request, id 4956, seq 0, length 8
15:38:16.299398 IP client-ext-1.acme-16.test > greenbone.acme-16.test: ICMP
echo reply, id 4956, seq 0, length 8
```

As we can see the target received the icmp request with the sender Greenbone and it will send to Greenbon a reply. That's why we cannot see a reply on kali.acme and we have only transmitted packets. We executed an ip spoofing attack correctly.
If we insert the above firewall rules instead this attack is immediately blocked by the nearest router to the attacker. Let's see it.



```
┌──(user㉿kali)-[~]
└─$ sudo hping3 -1 -S -c 1 -a 100.100.1.4 100.100.4.100
HPING 100.100.4.100 (eth0 100.100.4.100): icmp mode set, 28 h
eaders + 0 data bytes

─── 100.100.4.100 hping statistic ───
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The packet is apparently still sent to the target but if we check the icmp packets received on the target host we can verify that it was successfully blocked by the firewall:

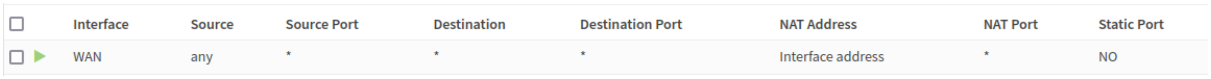


```
┌──(user㉿kali)-[~]
└─$ sudo tcpdump icmp
[sudo] password for user:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
█
```

Note: this kind of solution blocks ip spoofing attacks when the attacker tries to imitate an ip outside of a specific subnet. We could potentially still be able to imitate ip addresses of the same subnet we are in but defenses against this kind of attacks are far more complicated and probably not achievable with a simple firewall in opnsense.

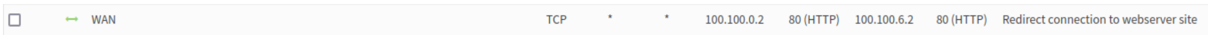
All the internal hosts should use the public IP address of the Main Firewall to exit towards the Internet.

The NAT rule on the WAN interface is already implemented in opnsense:



☐	Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port
☐ ▶	WAN	any	*	*	*	Interface address	*	NO

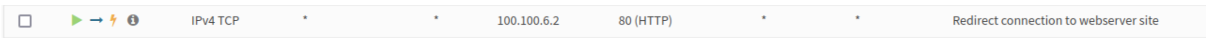
In fact this allows every host inside the network to use the ip of the Wan interface of the Main router (100.100.0.2). One thing that must be done is to open the website to the public. To do this we can just implement a local port forwarding redirecting all the traffic received by the Wan interface on port 80 to port 80 of the web server:



☐	⟷ WAN	TCP	*	*	100.100.0.2	80 (HTTP)	100.100.6.2	80 (HTTP)	Redirect connection to webserver site

To test this we can simply request 100.100.0.2:80 and see that we get redirected to the webpage of the web server.

Note: as soon as we create the port forwarding, this corresponding rule on the Wan interface is automatically generated:

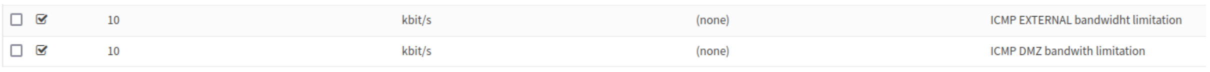


☐	▶ → ⚡ ℹ	IPv4 TCP	*	*	100.100.6.2	80 (HTTP)	*	*	Redirect connection to webserver site

This means that the previous rule which allowed connection from the internet to the web server is redundant and we can delete it.

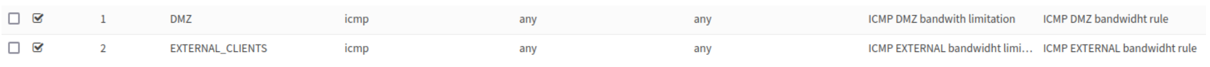
The rate of ICMP echo request packets should be limited to 10 Kbit/s.

The only way for us to somewhat limit the bandwidth of a specific protocol traffic in the network is to use the shaper tool in the firewall section on opnsense. This allows us to create a pipe associated with a specific bandwidth and a rule which is able to link a pipe with any protocol on a specific interface of the router. So to satisfy the request the assignment we created a pipe and his respective rule for each interface we want to limit ICMP packets on. On firewall->Sharper of the main router we created two pipes:



☐	✓	10	kbit/s	(none)	ICMP EXTERNAL bandwidht limitation
☐	✓	10	kbit/s	(none)	ICMP DMZ bandwith limitation

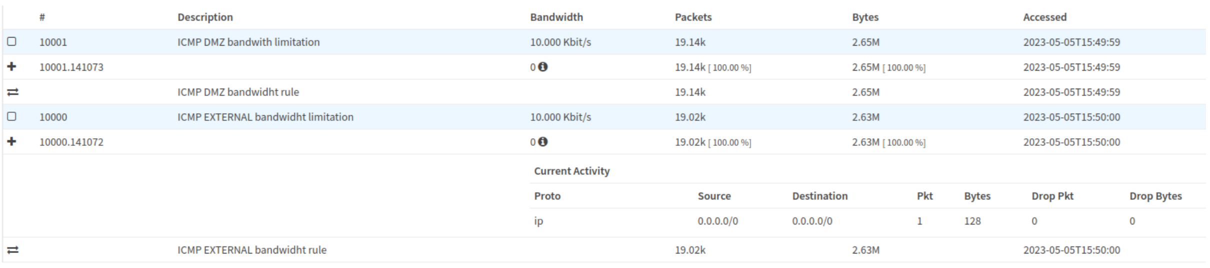
which they limit to 10 Kbp/s the bandwidth and two rules:



☐	✓	1	DMZ	icmp	any	any	ICMP DMZ bandwith limitation	ICMP DMZ bandwidht rule
☐	✓	2	EXTERNAL_CLIENTS	icmp	any	any	ICMP EXTERNAL bandwidht limi...	ICMP EXTERNAL bandwidht rule

which they assign the above pipes to ICMP protocol and to the interfaces DMZ and EXTERNAL_CLIENTS. The exact same things must be done on the internal router but on the interfaces CLIENTS and SERVERS. Every other interface can be left as it is because the ICMP traffic will be generated only from the above interfaces modified.

To test the correct functioning of the rules above we tried to ping from client_ext_1 the proxyserver (that was reachable for the previous rule even by pings) and we verified that the packets were slowed down by the bandwidth assigned. Removing the limitation we noted the difference in speed of the packets. We can also verify the status of the pipe by going on Shaper->Status:



	#	Description	Bandwidth	Packets	Bytes	Accessed
☐	10001	ICMP DMZ bandwith limitation	10.000 Kbit/s	19.14k	2.65M	2023-05-05T15:49:59
+	10001.141073		0 ℹ	19.14k [100.00 %]	2.65M [100.00 %]	2023-05-05T15:49:59
⇄		ICMP DMZ bandwidht rule		19.14k	2.65M	2023-05-05T15:49:59
☐	10000	ICMP EXTERNAL bandwidht limitation	10.000 Kbit/s	19.02k	2.63M	2023-05-05T15:50:00
+	10000.141072		0 ℹ	19.02k [100.00 %]	2.63M [100.00 %]	2023-05-05T15:50:00
⇄		ICMP EXTERNAL bandwidht rule		19.02k	2.63M	2023-05-05T15:50:00

Current Activity

Proto	Source	Destination	Pkt	Bytes	Drop Pkt	Drop Bytes
ip	0.0.0.0/0	0.0.0.0/0	1	128	0	0

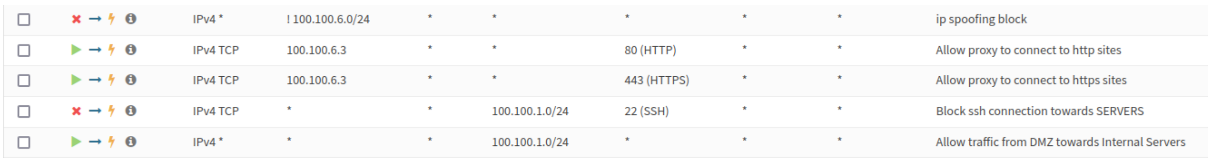
Anything that is not explicitly allowed has to be denied.

Since the default inbound rule is block, everything that has not been explicitly allowed is immediately dropped by the closest interface of the source of the traffic.

To clarify, let's recap the final configuration by printing all the rules of the interfaces.

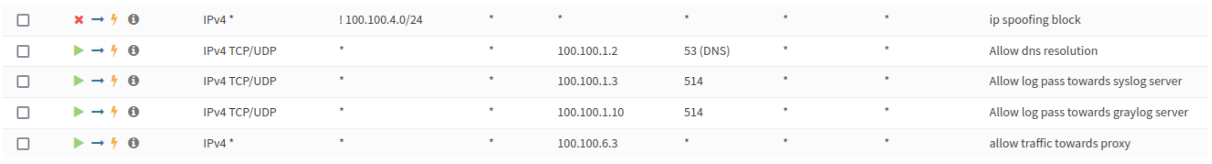
Main Firewall:

DMZ



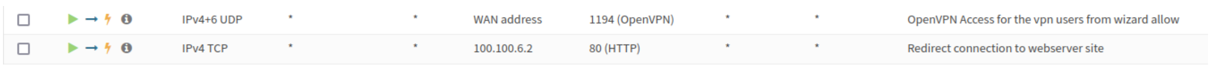
☐	✖ → ⚡ ℹ	IPv4 *	! 100.100.6.0/24	*	*	*	*	*	ip spoofing block
☐	▶ → ⚡ ℹ	IPv4 TCP	100.100.6.3	*	*	80 (HTTP)	*	*	Allow proxy to connect to http sites
☐	▶ → ⚡ ℹ	IPv4 TCP	100.100.6.3	*	*	443 (HTTPS)	*	*	Allow proxy to connect to https sites
☐	✖ → ⚡ ℹ	IPv4 TCP	*	*	100.100.1.0/24	22 (SSH)	*	*	Block ssh connection towards SERVERS
☐	▶ → ⚡ ℹ	IPv4 *	*	*	100.100.1.0/24	*	*	*	Allow traffic from DMZ towards Internal Servers

EXTERNAL_CLIENTS



☐	✖ → ⚡ ℹ	IPv4 *	! 100.100.4.0/24	*	*	*	*	*	ip spoofing block
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	*	*	100.100.1.2	53 (DNS)	*	*	Allow dns resolution
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	*	*	100.100.1.3	514	*	*	Allow log pass towards syslog server
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	*	*	100.100.1.10	514	*	*	Allow log pass towards graylog server
☐	▶ → ⚡ ℹ	IPv4 *	*	*	100.100.6.3	*	*	*	allow traffic towards proxy

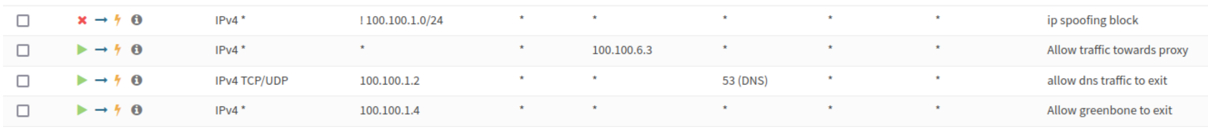
WAN



☐	▶ → ⚡ ℹ	IPv4+6 UDP	*	*	WAN address	1194 (OpenVPN)	*	*	OpenVPN Access for the vpn users from wizard allow
☐	▶ → ⚡ ℹ	IPv4 TCP	*	*	100.100.6.2	80 (HTTP)	*	*	Redirect connection to webserver site

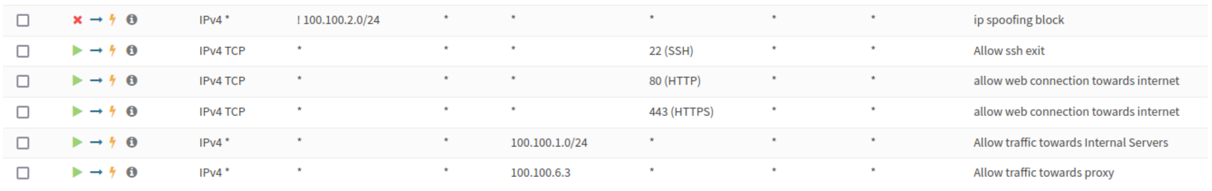
Internal Firewall:

SERVERS



☐	✖ → ⚡ ℹ	IPv4 *	! 100.100.1.0/24	*	*	*	*	*	ip spoofing block
☐	▶ → ⚡ ℹ	IPv4 *	*	*	100.100.6.3	*	*	*	Allow traffic towards proxy
☐	▶ → ⚡ ℹ	IPv4 TCP/UDP	100.100.1.2	*	*	53 (DNS)	*	*	allow dns traffic to exit
☐	▶ → ⚡ ℹ	IPv4 *	100.100.1.4	*	*	*	*	*	Allow greenbone to exit

CLIENTS



☐	✖ → ⚡ ℹ	IPv4 *	! 100.100.2.0/24	*	*	*	*	*	ip spoofing block
☐	▶ → ⚡ ℹ	IPv4 TCP	*	*	*	22 (SSH)	*	*	Allow ssh exit
☐	▶ → ⚡ ℹ	IPv4 TCP	*	*	*	80 (HTTP)	*	*	allow web connection towards internet
☐	▶ → ⚡ ℹ	IPv4 TCP	*	*	*	443 (HTTPS)	*	*	allow web connection towards internet
☐	▶ → ⚡ ℹ	IPv4 *	*	*	100.100.1.0/24	*	*	*	Allow traffic towards Internal Servers
☐	▶ → ⚡ ℹ	IPv4 *	*	*	100.100.6.3	*	*	*	Allow traffic towards proxy