1 2 9 0
UNIVERSIDADE D
COIMBRA

Information Technology Security

Network Protection

Francisco Catarino Mendes - 2019222823
Leonardo Oliveira Pereira - 2020239125
Department of Informatics Engineering
University of Coimbra

Introduction

This assignment was made mainly to acquire knowledge on how to configure a network firewall capable of detecting and reacting to attacks against services deployed on a protected network. For this purpose, the firewall is going to implement packet filtering, NAT, and intrusion detection, as well as mechanisms to react against attacks from hosts on the outside (Internet). Figure 1 illustrates the scenario considered for this practical assignment:

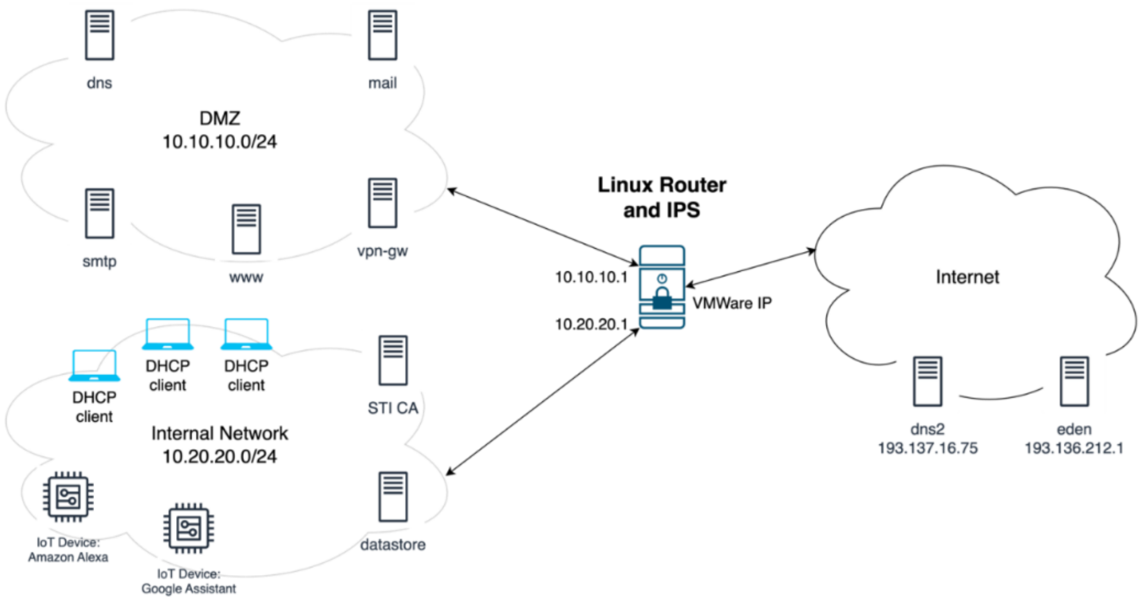


Figure 1: Scenario of the Assignment

Overall, the main goals are to configure a network firewall using NFTables/Netfilter (filtering, NAT, and integration with Snort/Suricata), to configure Snort/Suricata as an IDS/IPS system (intrusion detection and prevention/reaction), and to audit any configuration performed in the firewall and on the IDS/IPS.

1 Configuration of the Networks

As seen in Figure 1, the usage of a DMZ and an internal network must be mentioned. The DMZ network is where most of the public services of the organization are placed (services that are accessible from the outside). The objective of the internal network is to provide connectivity to users (clients with dynamic IP addresses) and support servers with specific purposes. The router interconnecting the various networks runs Linux and supports all the security functionalities described in the assignment. For all systems, IP addresses are assigned, as appropriate, shown below:

1. **DMZ:**
   a) **vpn-gw -** 10.10.10.100
   b) **mail -** 10.10.10.100
   c) **dns -** 10.10.10.100
   d) **smtp -** 10.10.10.100
   e) **www -** 10.10.10.100
2. **Internal Network:**
   a) **STI CA -** 10.20.20.100
   b) **datastore- -** 10.20.20.100
3. **Linux Router:**
   a) **eth0 -** IP attributed by the internet
   b) **eth1 -** 10.20.20.1
   c) **eth2 -** 10.10.10.1
4. **Internet :** IP attributed by the internet

Basically, all services are together in the same IP address. Regarding external target machines, for testing purposes, the ones used were the DNS of Universidade do Minho, ”193.137.16.75”, and the DNS corresponding to Eden, ”193.136.212.1”.

2 Packet filtering and NAT using NFtables

In this section, the various firewall configurations are going to be talked about and demonstrated, including configurations to protect the router, authorize direct communications, for connections to the external IP address of the firewall, and for communications from the internal network to the outside.

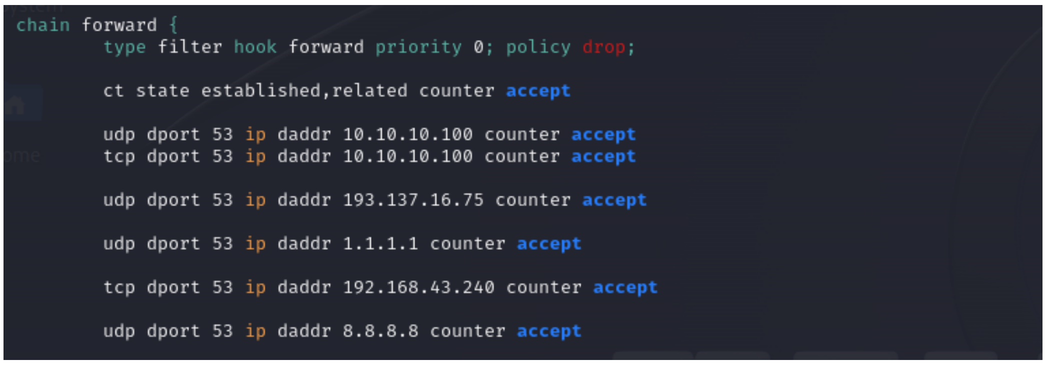
2.1 Firewall configurations to protect the router

Here, the firewall configurations will drop all communications entering the router system, except those required for the normal operation of the following services:

A - DNS name resolution requests sent to outside servers;

B - SSH connections to the router system, if originating at the internal network or the VPN gateway (vpn-gw). The SSH connections are also protected with the Port Knocking mechanism.

Figures 2 and 3 show those firewall configurations, respectively:

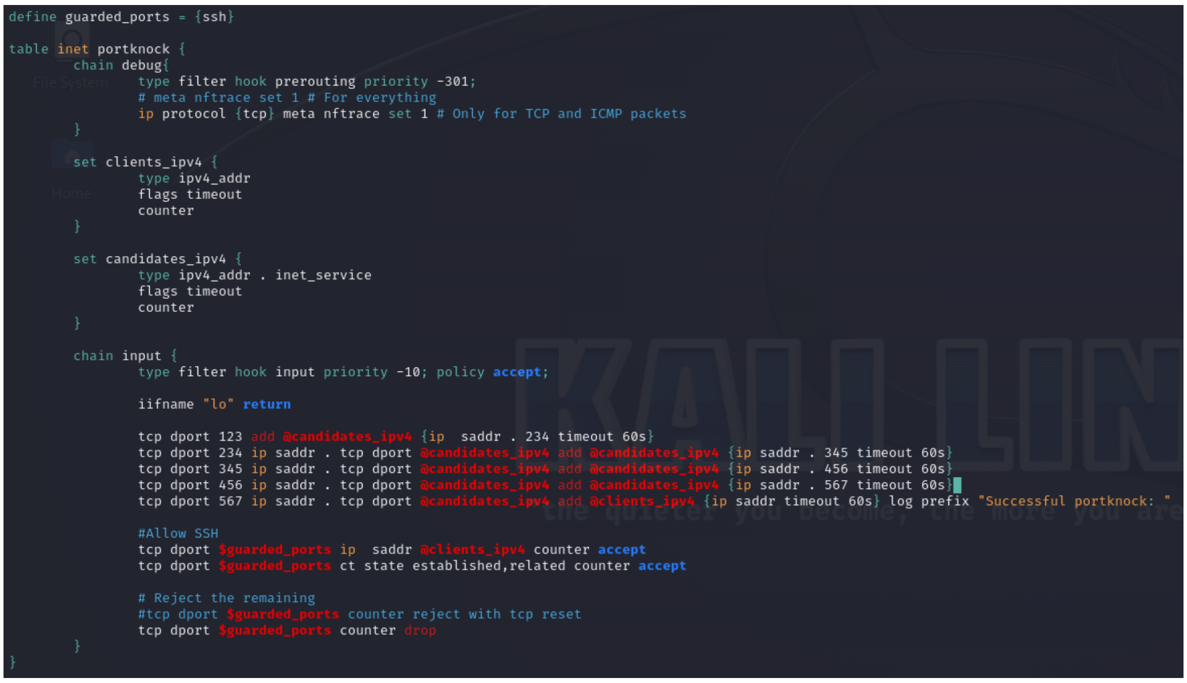


```
chain forward {
        type filter hook forward priority 0; policy drop;

        ct state established,related counter accept

        udp dport 53 ip daddr 10.10.10.100 counter accept
        tcp dport 53 ip daddr 10.10.10.100 counter accept

        udp dport 53 ip daddr 193.137.16.75 counter accept

        udp dport 53 ip daddr 1.1.1.1 counter accept

        tcp dport 53 ip daddr 192.168.43.240 counter accept

        udp dport 53 ip daddr 8.8.8.8 counter accept
```

Figure 2: Rules for DNS name resolution

As one can see in Figure 2, the rule **udp dport 53 ip daddr 8.8.8.8 counter accept** effectively allows UDP-based DNS requests from your network to be forwarded to Google's DNS server at 8.8.8.8, while also keeping track of how many requests are made. This indicates a controlled approach to managing DNS traffic, allowing requests only from trusted sources.

Additionally, in Figure 3, the Port Knocking process is essentially shown. Port-specific rules add the source address to the candidates_ipv4 set with a 60-second timeout. This is characteristic of port knocking where access to further ports/services is contingent on accessing specific ports in a particular sequence. The TCP ports 123, 234, 345, 456, and 567 are being used as part of said sequence to add an IP to a more privileged set, allowing SSH access. Traffic that reaches the guarded ports (including SSH) without meeting criteria (not in guarded_ports) is either rejected with TCP reset or dropped. There is a definition for guarded_ports which includes the SSH port, among potentially others, for which access is tightly controlled, requiring prior successful ”knocking” on the designated sequence of ports.



```
define guarded_ports = {ssh}

table inet portknock {
        chain debug{
                type filter hook prerouting priority -301;
                # meta nftrace set 1 # For everything
                ip protocol {tcp} meta nftrace set 1 # Only for TCP and ICMP packets
        }

        set clients_ipv4 {
                type ipv4_addr
                flags timeout
                counter
        }

        set candidates_ipv4 {
                type ipv4_addr . inet_service
                flags timeout
                counter
        }

        chain input {
                type filter hook input priority -10; policy accept;

                iifname "lo" return

                tcp dport 123 add @candidates_ipv4 {ip  saddr . 234 timeout 60s}
                tcp dport 234 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 {ip saddr . 345 timeout 60s}
                tcp dport 345 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 {ip saddr . 456 timeout 60s}
                tcp dport 456 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 {ip saddr . 567 timeout 60s}
                tcp dport 567 ip saddr . tcp dport @candidates_ipv4 add @clients_ipv4 {ip saddr timeout 60s} log prefix "Successful portknock: "

                #Allow SSH
                tcp dport $guarded_ports ip  saddr @clients_ipv4 counter accept
                tcp dport $guarded_ports ct state established,related counter accept

                # Reject the remaining
                #tcp dport $guarded_ports counter reject with tcp reset
                tcp dport $guarded_ports counter drop
        }
}
```

Figure 3: SSH connections with the Port Knocking mechanism

2.2 Firewall configurations to authorize direct communications (without NAT)

In this part, the firewall configuration will drop all communications between networks, except the ones required for the normal operation of the following services:

a) **C -** Domain name resolutions using the dns server;

b) **D -** The dns server must resolve names using DNS servers on the Internet (dns2 and 1.1.1.1);

c) **E -** The dns and dns2 servers must be able to synchronize the contents of DNS zones;

d) **F -** SMTP connections to the smtp server;

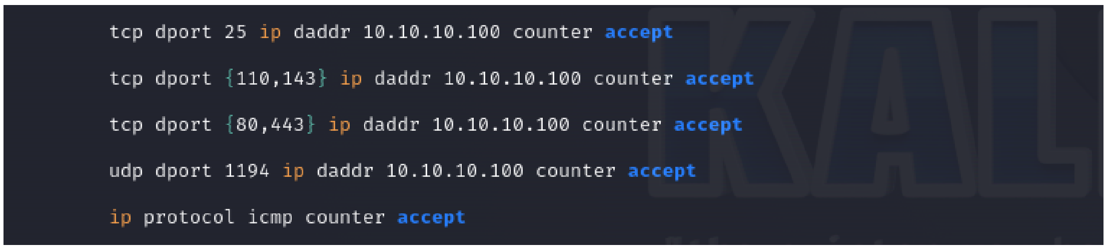
e) **G -** POP and IMAP connections to the mail server;

f) **H -** HTTP and HTTPS connections to the www server;

g) **I -** OpenVPN connections to the vpn-gw server.

For the dns server ones, the configurations can be seen in Figure 2. The rules **udp dport 53 ip daddr 10.10.10.100 counter accept** and **tcp dport 53 ip daddr 10.10.10.100 counter accept** are associated with domain name resolutions using the dns server, while **udp dport 53 ip daddr 1.1.1.1 counter accept** and **udp dport 53 ip daddr 193.137.16.75 counter accept** refer to the ability of the dns server to resolve names using the DNS of Universidade do Minho (dns2) and ”1.1.1.1”.

As for the rest of the requirements, Figure 4 displays the configurations made towards those goals:



```
tcp dport 25 ip daddr 10.10.10.100 counter accept

tcp dport {110,143} ip daddr 10.10.10.100 counter accept

tcp dport {80,443} ip daddr 10.10.10.100 counter accept

udp dport 1194 ip daddr 10.10.10.100 counter accept

ip protocol icmp counter accept
```

Figure 4: Rules to authorize direct communications (without NAT)

These rules specifically allow TCP traffic to ports 25 - SMTP, 110 - POP, 143 - IMAP, 80 - HTTP, 443 - HTTPS, and 3306 - MySQL, as well as UDP traffic to port 1194, which is used by OpenVPN, all directed to the IP ”10.10.10.100”.

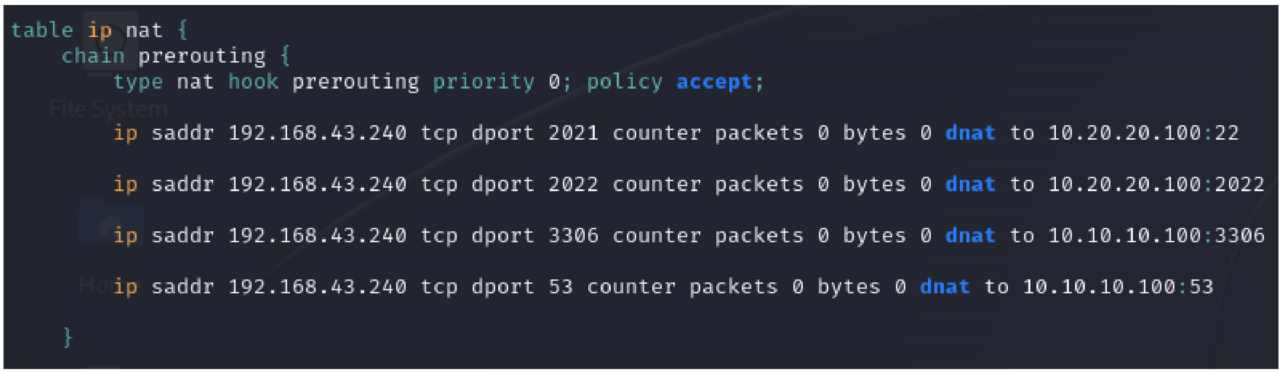
2.3 Firewall configurations for connections to the external IP address of the firewall (using NAT)

In this section, the connections originated on the outside (Internet) and destined to the external IP address (Linux Router) of the firewall will be authorized and treated according to the following requirements:

J - SSH connections towards port 2021 of the Linux Router must be redirected to the SSH port of the datastore server and must be limited to 2 simultaneous connections;

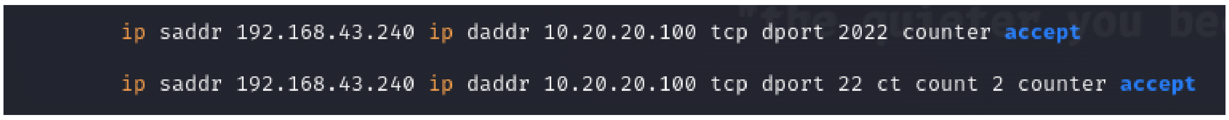
K - Connections towards port 2022 of the Linux Router must be redirected to port 2022 of the STI CA server (internal network) and must only be allowed from a single IP address on the Internet.

From the rules illustrated in Figures 5 and 6, some answers can be provided:



```
table ip nat {
    chain prerouting {
        type nat hook prerouting priority 0; policy accept;

        ip saddr 192.168.43.240 tcp dport 2021 counter packets 0 bytes 0 dnat to 10.20.20.100:22

        ip saddr 192.168.43.240 tcp dport 2022 counter packets 0 bytes 0 dnat to 10.20.20.100:2022

        ip saddr 192.168.43.240 tcp dport 3306 counter packets 0 bytes 0 dnat to 10.10.10.100:3306

        ip saddr 192.168.43.240 tcp dport 53 counter packets 0 bytes 0 dnat to 10.10.10.100:53

    }
```

Figure 5: NAT configurations - redirects



```
ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 2022 counter accept

ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 22 ct count 2 counter accept
```

Figure 6: Rules to limit number of possible connections

As one can see, Figure 5 shows the rule relative to redirecting SSH connections towards port 2021 of the router to port 22 of the datastore server inside the internal network, with **ip saddr 192.168.43.240 tcp dport 2021 dnat to 10.20.20.100:22**, while Figure 6 shows how those connections are limited to 2 simultaneous connections, with **ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 22 ct count 2 counter accept**.

Furthermore, Figure 5 also has the rule referring to redirecting connections towards port 2022 of the router to port 2022 of the internal network, which goes by **ip saddr 192.168.43.240 tcp dport 2022 dnat to 10.20.20.100:2022**, while Figure 6 presents how those connections are only allowed from a single IP address, with the rule **ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 2022 counter accept**.

2.4 Firewall configurations for communications from the internal network to the outside (using NAT)

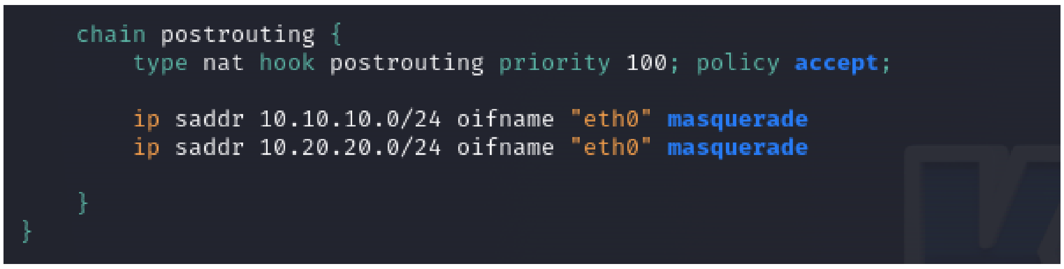
Regarding communications from the internal network to the outside, the following communication must be authorized using NAT:

L - Domain name resolutions using DNS.

M - HTTP, HTTPS, and SSH connections from devices with dynamic IP addresses (DHCP clients).

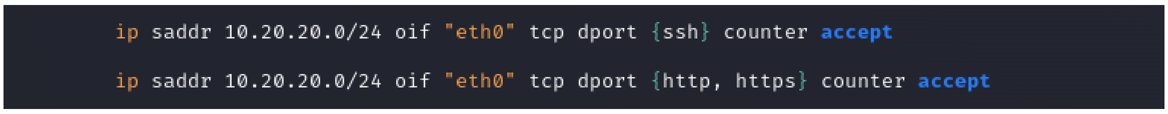
About the DNS part, once again the rule **udp dport 53 ip daddr 8.8.8.8 counter accept** is used successfully when trying to connect from the internal network to the outside (internet). This will be more explicit in the Tests section.

As for HTTP, HTTPS, and SSH connections, Figures 7 and 8 help when trying to provide a solution:



```
    chain postrouting {
        type nat hook postrouting priority 100; policy accept;

        ip saddr 10.10.10.0/24 oifname "eth0" masquerade
        ip saddr 10.20.20.0/24 oifname "eth0" masquerade

    }
}
```

Figure 7: Presenting the internal networks to the outside



```
ip saddr 10.20.20.0/24 oif "eth0" tcp dport {ssh} counter accept

ip saddr 10.20.20.0/24 oif "eth0" tcp dport {http, https} counter accept
```

Figure 8: HTTP, HTTPS, and SSH connections from devices with dynamic IP addresses

From the rule **ip saddr 10.20.20.0/24 oifname ”eth0” masquerade”** in Figure 7 and the rules from Figure 8, one can assume that HTTP, HTTPS, and SSH connections from the internal network to the outside are indeed working, by showing to the internet the IP address of the internal network, which possesses the DHCP clients, and allowing traffic for those 3 services originating from any device within the internal network range.

3 Intrusion detection and prevention (IDS/IPS)

In this second phase of this assignment, the objective is to enable, in the firewall system, the capability to detect and react to attacks. When an attack is successfully detected, the firewall must be able to block it. Thus, intrusion detection and prevention is going to be implemented in the firewall considering the following requirements:

- The tool used for this purpose is **Suricata**; **-** The system should be able to detect and block one type of SQL injection, DoS attack, and Brute Force attack.

Before explaining the procedure, a definition of these attacks must be provided, for context and better understanding:

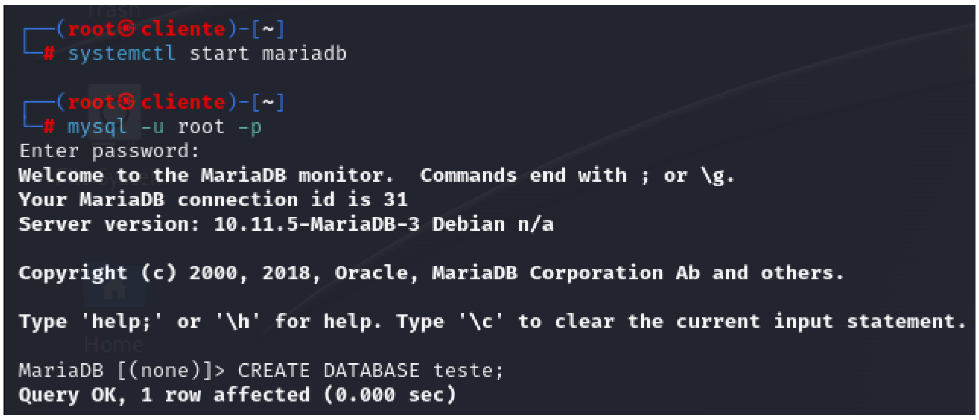
SQL Injection - a type of security vulnerability that occurs in an application’s data layer, where an attacker can inject malicious SQL code into a query. This happens when input data provided by a user is improperly sanitized or directly used in the construction of SQL statements. This allows the attacker to manipulate the underlying database, which can lead to unauthorized data access, modification, or deletion, potentially compromising the entire database.

Denial of Service (DoS) attack - a type of cyber attack where the adversary aims to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting the services of a host connected to the Internet. This is typically accomplished by overwhelming the target with a flood of incoming traffic, requests, or messages, causing the system to become overloaded and thus preventing legitimate requests from being attended.

Brute Force attack - a type of cyber attack where an attacker systematically checks all possible passwords or keys until the correct one is found. Essentially, the attacker attempts to gain access to a system or encrypted data by trying every possible combination of credentials or decryption keys.

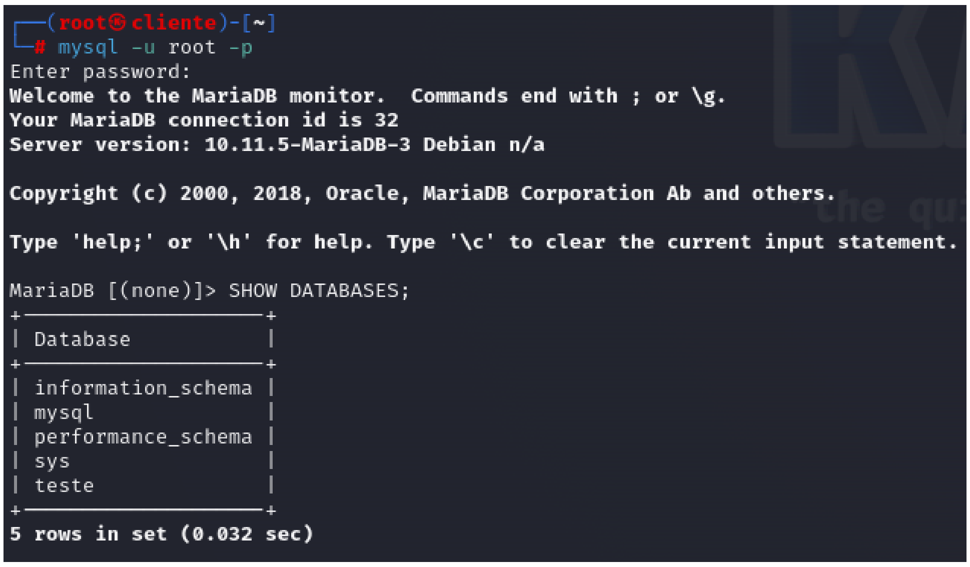
3.1 SQL Injection Attack

To simulate the SQL Injection, first, the service **mariadb** was installed and a test database was created. Figures 9 and 10 illustrate the commands:



```
┌──(root㉿cliente)-[~]
└─# systemctl start mariadb

┌──(root㉿cliente)-[~]
└─# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.5-MariaDB-3 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE teste;
Query OK, 1 row affected (0.000 sec)
```

Figure 9: Creation of the test database



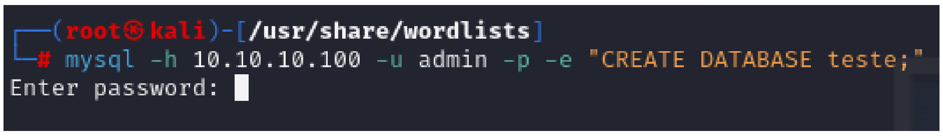
```
┌──(root㉿cliente)-[~]
└─# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.11.5-MariaDB-3 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| teste              |
+--------------------+
5 rows in set (0.032 sec)
```

Figure 10: Proof of existence of the database

Then, two new rules were added to the firewall, one in the prerouting chain (Figure 5) which goes by **ip saddr 192.168.43.240 tcp dport 3306 dnat to 10.10.10.100:3306**, and one to the forward chain, **ip saddr 192.168.43.240 ip daddr 10.10.10.100 tcp dport 3306 counter accept**. These rules redirect traffic from the router to the mysql service in the DMZ (port 3306) while also allowing the traffic to go through. This is needed to test the defenses implemented.

Afterwards, a defensive rule was created in Suricata, to detect and block/drop this type of attack: **drop mysql any any -> any any (msg:"SQL Injection attempt"; content:"DROP DATABASE"; nocase; sid:1000001; rev:1;)**. It drops MySQL traffic if it contains the phrase "DROP DATABASE", which is indicative of an attempt to perform an SQL injection attack by deleting a database.

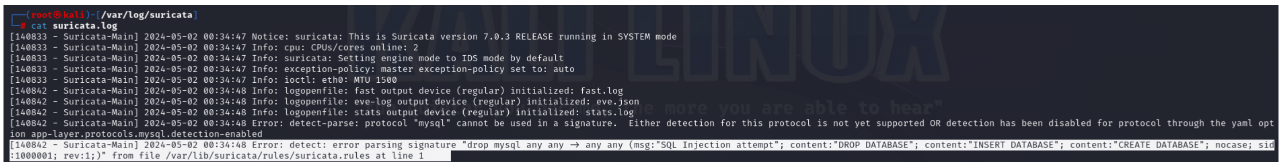
To test the efficiency of all these procedures, the following attack was performed, exhibited by Figure 11:



```
┌──(root㉿kali)-[/usr/share/wordlists]
└─# mysql -h 10.10.10.100 -u admin -p -e "CREATE DATABASE teste;"
Enter password: 
```

Figure 11: SQL Injection Attack

The attack was successfully stopped and detected, proven by the information contained in the Suricata logs and displayed in Figure 12:



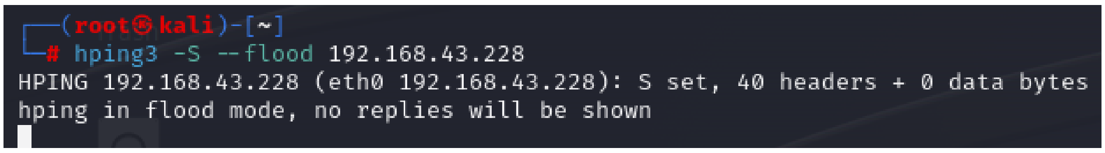
```
┌──(root㉿kali)-[/var/log/suricata]
└─# cat suricata.log
[140833 - Suricata-Main] 2024-05-02 00:34:47 Notice: suricata: This is Suricata version 7.0.3 RELEASE running in SYSTEM mode
[140833 - Suricata-Main] 2024-05-02 00:34:47 Info: cpu: CPUs/cores online: 2
[140833 - Suricata-Main] 2024-05-02 00:34:47 Info: suricata: Setting engine mode to IDS mode by default
[140833 - Suricata-Main] 2024-05-02 00:34:47 Info: exception-policy: master exception-policy set to: auto
[140833 - Suricata-Main] 2024-05-02 00:34:47 Info: ioctl: eth0: MTU 1500
[140842 - Suricata-Main] 2024-05-02 00:34:48 Info: logopenfile: fast output device (regular) initialized: fast.log
[140842 - Suricata-Main] 2024-05-02 00:34:48 Info: logopenfile: eve-log output device (regular) initialized: eve.json
[140842 - Suricata-Main] 2024-05-02 00:34:48 Info: logopenfile: stats output device (regular) initialized: stats.log
[140842 - Suricata-Main] 2024-05-02 00:34:48 Error: detect-parse: protocol "mysql" cannot be used in a signature.  Either detection for this protocol is not yet supported OR detection has been disabled for protocol through the yaml opt
ion app-layer.protocols.mysql.detection-enabled
[140842 - Suricata-Main] 2024-05-02 00:34:48 Error: detect: error parsing signature "drop mysql any any → any any (msg:"SQL Injection attempt"; content:"DROP DATABASE"; content:"INSERT DATABASE"; content:"CREATE DATABASE"; nocase; sid
:1000001; rev:1;)" from file /var/lib/suricata/rules/suricata.rules at line 1
```

Figure 12: SQL Injection attempt detected and blocked

3.2 Denial of Service Attack

For this attack, the tool **hping3** was used, which is going to flood the router with TCP-SYN packets to try a denial of service. In order to battle this attack, another security rule was added to Suricata: **drop tcp any any -> any any (msg:"Possible TCP DoS"; flags:S; flow:stateless; threshold:type both, track by_dst, count 100, seconds 1; sid:1000002; rev:1;)**. It drops TCP traffic if there are more than 100 connection attempts with the SYN flag set (flags:S) towards the same destination within one second.

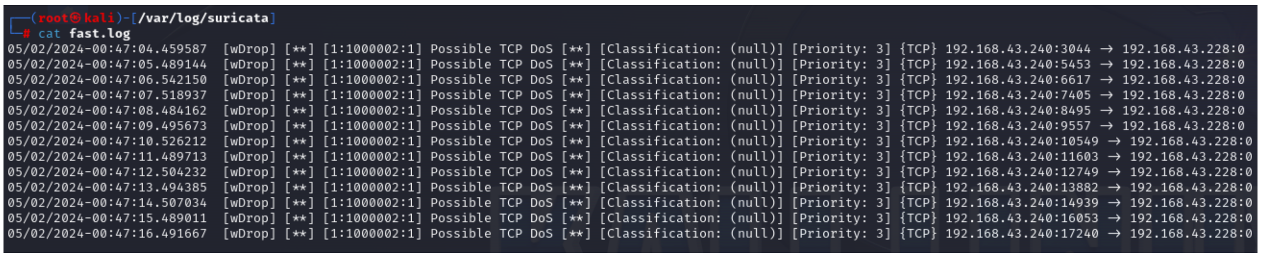
Figure 13 shows the realization of the DoS attempt:



```
┌──(root㉿kali)-[~]
└─# hping3 -S --flood 192.168.43.228
HPING 192.168.43.228 (eth0 192.168.43.228): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 13: Denial of Service Attack - Flooding

Once again, the attack was successfully blocked and detected, with the evidence present in the Suricata logs presented in Figure 14:



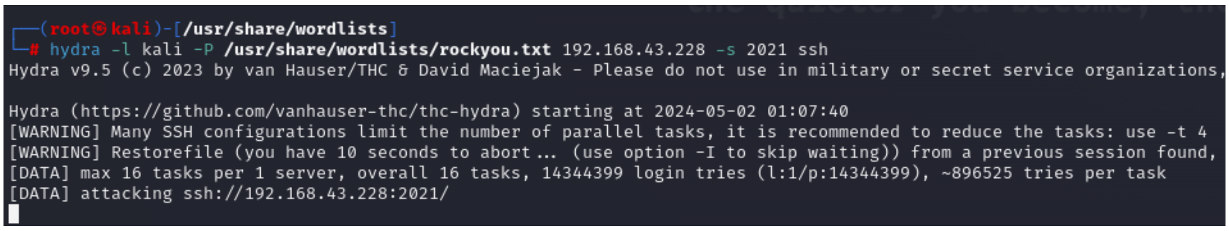
```
┌──(root㉿kali)-[/var/log/suricata]
└─# cat fast.log
05/02/2024-00:47:04.459587  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:3044 → 192.168.43.228:0
05/02/2024-00:47:05.489144  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:5453 → 192.168.43.228:0
05/02/2024-00:47:06.542150  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:6617 → 192.168.43.228:0
05/02/2024-00:47:07.518937  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:7405 → 192.168.43.228:0
05/02/2024-00:47:08.484162  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:8495 → 192.168.43.228:0
05/02/2024-00:47:09.495673  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:9557 → 192.168.43.228:0
05/02/2024-00:47:10.526212  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:10549 → 192.168.43.228:0
05/02/2024-00:47:11.489713  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:11603 → 192.168.43.228:0
05/02/2024-00:47:12.504232  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:12749 → 192.168.43.228:0
05/02/2024-00:47:13.494385  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:13882 → 192.168.43.228:0
05/02/2024-00:47:14.507034  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:14939 → 192.168.43.228:0
05/02/2024-00:47:15.489011  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:16053 → 192.168.43.228:0
05/02/2024-00:47:16.491667  [wDrop] [**] [1:1000002:1] Possible TCP DoS [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.43.240:17240 → 192.168.43.228:0
```

Figure 14: DoS attempt detected and blocked

3.3 Brute Force attack

Regarding the last attack to be tested, which is a brute force attack, the tool selected was Hydra. Hydra contains wordlists of potential passwords that can be used in commands to try and brute force a login. In this scenario, the list selected was the rockyou.txt list, and the attack is going to be directed to port 2021 of the SSH service. Once more, a rule to stop this attack from going through was added to the Suricata rules: **drop tcp any any -> any any (msg:"Possible SSH brute force attack"; flow:to_server,established; flags: PSH+ACK; content:"Failed password"; threshold: type both, track by_src, count 10, seconds 60; sid:1000004; rev:1;)**. It drops TCP traffic directed towards an SSH server (traffic flowing to the server and on an established connection) if the payload contains the phrase "Failed password" more than 10 times in 60 seconds from the same source.

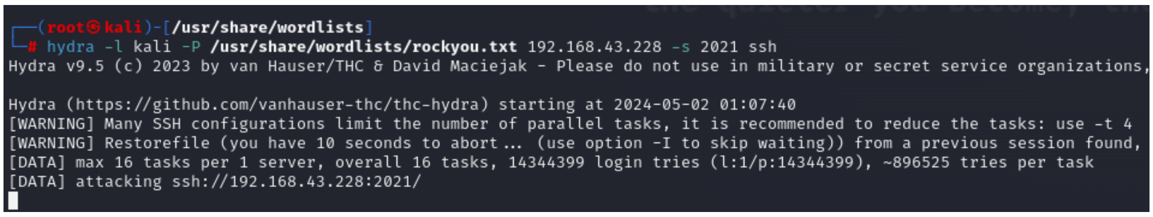
As for the attack, Figure 15 illustrates the command executed:



```
┌──(root㉿kali)-[/usr/share/wordlists]
└─# hydra -l kali -P /usr/share/wordlists/rockyou.txt 192.168.43.228 -s 2021 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-02 01:07:40
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found,
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.43.228:2021/
█
```

Figure 15: Brute Force attack detected and blocked

Finally, it must be mentioned that this attack was also blocked and detected with great success, and the proof can be encountered in the Suricata logs displayed in Figure 16:



```
┌──(root㉿kali)-[/usr/share/wordlists]
└─# hydra -l kali -P /usr/share/wordlists/rockyou.txt 192.168.43.228 -s 2021 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-02 01:07:40
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found,
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.43.228:2021/
█
```

Figure 16: Brute Force attack

4 Audit Events

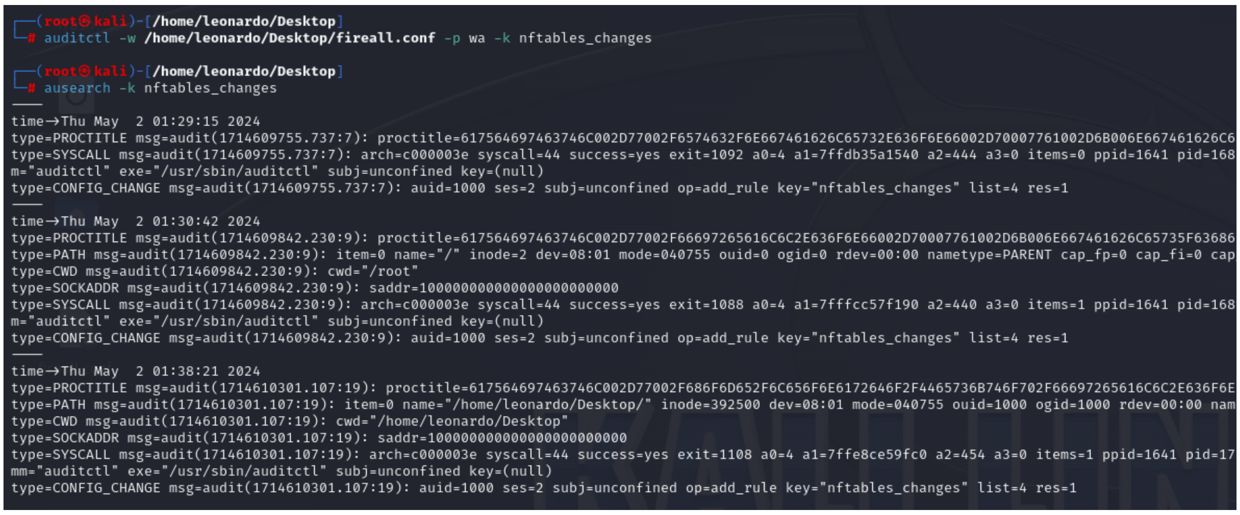
To end the assignment, with the help of the audit facilities in Linux, the goal is to find any modification in the security files of the firewall and IDS/IPS. Any attempt to modify the permissions and any attempt to read these configuration files has to be audited.

For that to happen the tool **auditd** was used. By using its options:

1. **-w**, which is used to specify a watch for a particular file or directory;
2. **-p wa**, where -p specifies the permissions or operations to monitor on the watched file (in this case ”w” stands for write, and ”a” stands for attribute changes), meaning the audit rule will log any write operations (such as modifications to the file) and attribute changes (like changes to ownership or permissions);
3. **-k**, which allows adding a key or label to the audit rule, making it easier to search the audit logs;

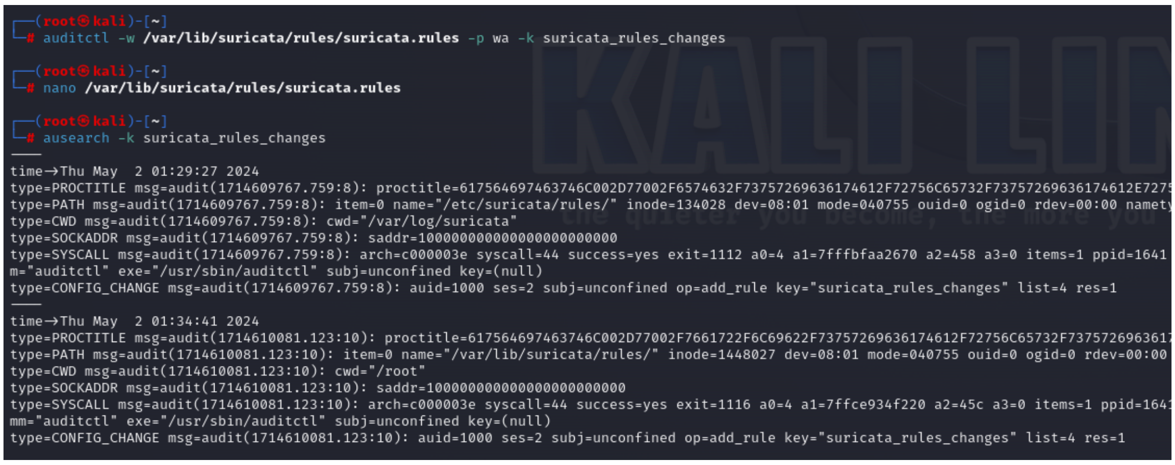
Any attempt to modify permissions or read the content of the Suricata and firewall files will be audited accordingly, improving security, as it is critical to ensure that firewall and other network security settings are not altered maliciously or without proper oversight.

The examples of this process are illustrated in Figures 17 and 18, for the firewall and the Suricata files, respectively:



```
┌──(root㉿kali)-[/home/leonardo/Desktop]
└─# auditctl -w /home/leonardo/Desktop/fireall.conf -p wa -k nftables_changes

┌──(root㉿kali)-[/home/leonardo/Desktop]
└─# ausearch -k nftables_changes
----
time->Thu May  2 01:29:15 2024
type=PROCTITLE msg=audit(1714609755.737:7): proctitle=617564697463746C002D77002F6574632F6E667461626C65732E636F6E66002D70007761002D6B006E667461626C6
type=SYSCALL msg=audit(1714609755.737:7): arch=c000003e syscall=44 success=yes exit=1092 a0=4 a1=7ffdb35a1540 a2=444 a3=0 items=0 ppid=1641 pid=168
m="auditctl" exe="/usr/sbin/auditctl" subj=unconfined key=(null)
type=CONFIG_CHANGE msg=audit(1714609755.737:7): auid=1000 ses=2 subj=unconfined op=add_rule key="nftables_changes" list=4 res=1
----
time->Thu May  2 01:30:42 2024
type=PROCTITLE msg=audit(1714609842.230:9): proctitle=617564697463746C002D77002F66697265616C6C2E636F6E66002D70007761002D6B006E667461626C65735F63686
type=PATH msg=audit(1714609842.230:9): item=0 name="/" inode=2 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00 nametype=PARENT cap_fp=0 cap_fi=0 cap
type=CWD msg=audit(1714609842.230:9): cwd="/root"
type=SOCKADDR msg=audit(1714609842.230:9): saddr=100000000000000000000000
type=SYSCALL msg=audit(1714609842.230:9): arch=c000003e syscall=44 success=yes exit=1088 a0=4 a1=7fffcc57f190 a2=440 a3=0 items=1 ppid=1641 pid=168
m="auditctl" exe="/usr/sbin/auditctl" subj=unconfined key=(null)
type=CONFIG_CHANGE msg=audit(1714609842.230:9): auid=1000 ses=2 subj=unconfined op=add_rule key="nftables_changes" list=4 res=1
----
time->Thu May  2 01:38:21 2024
type=PROCTITLE msg=audit(1714610301.107:19): proctitle=617564697463746C002D77002F686F6D652F6C656F6E6172646F2F4465736B746F702F66697265616C6C2E636F6E6E
type=PATH msg=audit(1714610301.107:19): item=0 name="/home/leonardo/Desktop/" inode=392500 dev=08:01 mode=040755 ouid=1000 ogid=1000 rdev=00:00 nam
type=CWD msg=audit(1714610301.107:19): cwd="/home/leonardo/Desktop"
type=SOCKADDR msg=audit(1714610301.107:19): saddr=100000000000000000000000
type=SYSCALL msg=audit(1714610301.107:19): arch=c000003e syscall=44 success=yes exit=1108 a0=4 a1=7ffe8ce59fc0 a2=454 a3=0 items=1 ppid=1641 pid=17
mm="auditctl" exe="/usr/sbin/auditctl" subj=unconfined key=(null)
type=CONFIG_CHANGE msg=audit(1714610301.107:19): auid=1000 ses=2 subj=unconfined op=add_rule key="nftables_changes" list=4 res=1
```

Figure 17: Auditing the Firewall rules file



```
┌──(root㉿kali)-[~]
└─# auditctl -w /var/lib/suricata/rules/suricata.rules -p wa -k suricata_rules_changes

┌──(root㉿kali)-[~]
└─# nano /var/lib/suricata/rules/suricata.rules

┌──(root㉿kali)-[~]
└─# ausearch -k suricata_rules_changes
----
time->Thu May  2 01:29:27 2024
type=PROCTITLE msg=audit(1714609767.759:8): proctitle=617564697463746C002D77002F6574632F73757269636174612F72756C65732F73757269636174612E7275
type=PATH msg=audit(1714609767.759:8): item=0 name="/etc/suricata/rules/" inode=134028 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00 namety
type=CWD msg=audit(1714609767.759:8): cwd="/var/log/suricata"
type=SOCKADDR msg=audit(1714609767.759:8): saddr=100000000000000000000000
type=SYSCALL msg=audit(1714609767.759:8): arch=c000003e syscall=44 success=yes exit=1112 a0=4 a1=7fffbfaa2670 a2=458 a3=0 items=1 ppid=1641
m="auditctl" exe="/usr/sbin/auditctl" subj=unconfined key=(null)
type=CONFIG_CHANGE msg=audit(1714609767.759:8): auid=1000 ses=2 subj=unconfined op=add_rule key="suricata_rules_changes" list=4 res=1
----
time->Thu May  2 01:34:41 2024
type=PROCTITLE msg=audit(1714610081.123:10): proctitle=617564697463746C002D77002F7661722F6C69622F73757269636174612F72756C65732F737572696361
type=PATH msg=audit(1714610081.123:10): item=0 name="/var/lib/suricata/rules/" inode=1448027 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1714610081.123:10): cwd="/root"
type=SOCKADDR msg=audit(1714610081.123:10): saddr=100000000000000000000000
type=SYSCALL msg=audit(1714610081.123:10): arch=c000003e syscall=44 success=yes exit=1116 a0=4 a1=7ffce934f220 a2=45c a3=0 items=1 ppid=164
mm="auditctl" exe="/usr/sbin/auditctl" subj=unconfined key=(null)
type=CONFIG_CHANGE msg=audit(1714610081.123:10): auid=1000 ses=2 subj=unconfined op=add_rule key="suricata_rules_changes" list=4 res=1
```

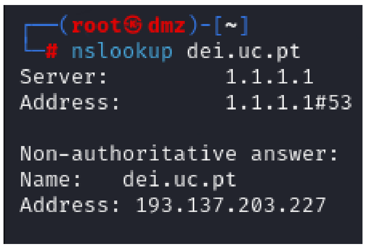
Figure 18: Auditing the Suricata rules file

In both Figures, one can indeed verify that whenever some modification is made inside those files, the logs work accordingly, giving information in the ”type” field.

5 Firewall Tests

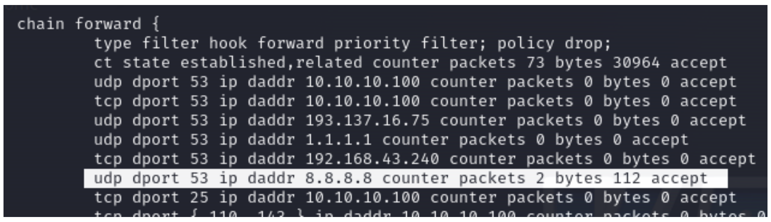
This section works as an appendix, and it is here that all the tests and proof relative to what was said about the firewall configurations reside. For organization purposes, the indexes will go from A to M, according to what was attributed inside the subsections earlier.

A - DNS name resolution requests sent to outside servers



```
┌──(root㉿dmz)-[~]
└─# nslookup dei.uc.pt
Server:         1.1.1.1
Address:        1.1.1.1#53

Non-authoritative answer:
Name:   dei.uc.pt
Address: 193.137.203.227
```

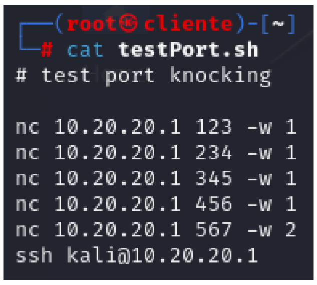
Figure 19: DNS name resolution request



```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 73 bytes 30964 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 2 bytes 112 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0
```

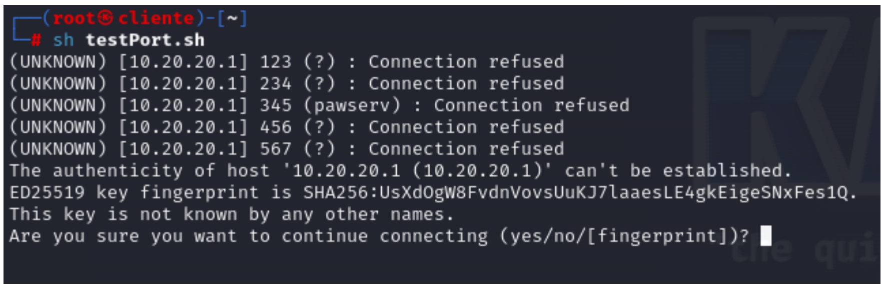
Figure 20: Corresponding rule in the NFTable

B - SSH connections to the router system, if originating at the internal network or the VPN gateway (vpn-gw). The SSH connections are also protected with the Port Knocking mechanism



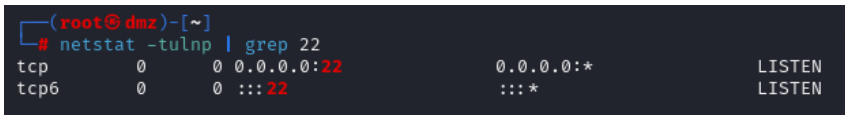
```
┌──(root㉿cliente)-[~]
└─# cat testPort.sh
# test port knocking

nc 10.20.20.1 123 -w 1
nc 10.20.20.1 234 -w 1
nc 10.20.20.1 345 -w 1
nc 10.20.20.1 456 -w 1
nc 10.20.20.1 567 -w 2
ssh kali@10.20.20.1
```

Figure 21: Script of the Port Knocking



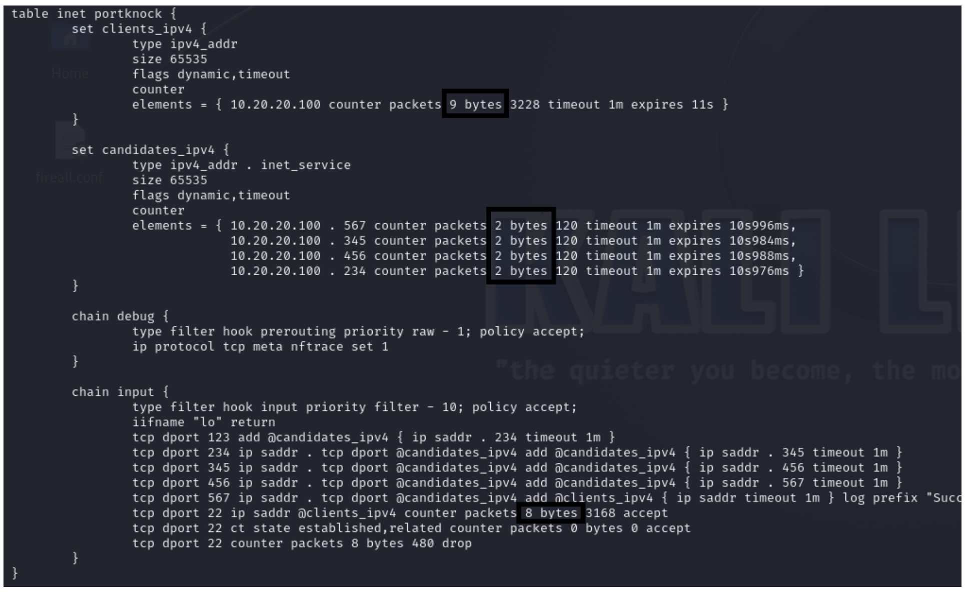
```
┌──(root㉿cliente)-[~]
└─# sh testPort.sh
(UNKNOWN) [10.20.20.1] 123 (?) : Connection refused
(UNKNOWN) [10.20.20.1] 234 (?) : Connection refused
(UNKNOWN) [10.20.20.1] 345 (pawserv) : Connection refused
(UNKNOWN) [10.20.20.1] 456 (?) : Connection refused
(UNKNOWN) [10.20.20.1] 567 (?) : Connection refused
The authenticity of host '10.20.20.1 (10.20.20.1)' can't be established.
ED25519 key fingerprint is SHA256:UsXdOgW8FvdnVovsUuKJ7laaesLE4gkEigeSNxFes1Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Figure 22: Executing the Port Knocking



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 22
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

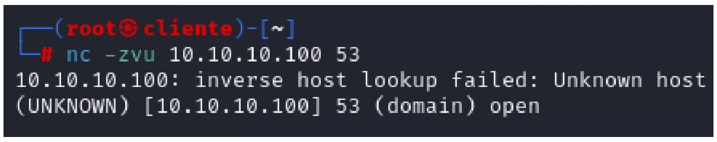
Figure 23: State of the port when receiving the request



```
table inet portknock {
        set clients_ipv4 {
                type ipv4_addr
                size 65535
                flags dynamic,timeout
                counter
                elements = { 10.20.20.100 counter packets 9 bytes 3228 timeout 1m expires 11s }
        }

        set candidates_ipv4 {
                type ipv4_addr . inet_service
                size 65535
                flags dynamic,timeout
                counter
                elements = { 10.20.20.100 . 567 counter packets 2 bytes 120 timeout 1m expires 10s996ms,
                             10.20.20.100 . 345 counter packets 2 bytes 120 timeout 1m expires 10s984ms,
                             10.20.20.100 . 456 counter packets 2 bytes 120 timeout 1m expires 10s988ms,
                             10.20.20.100 . 234 counter packets 2 bytes 120 timeout 1m expires 10s976ms }
        }

        chain debug {
                type filter hook prerouting priority raw - 1; policy accept;
                ip protocol tcp meta nftrace set 1
        }

        chain input {
                type filter hook input priority filter - 10; policy accept;
                iifname "lo" return
                tcp dport 123 add @candidates_ipv4 { ip saddr . 234 timeout 1m }
                tcp dport 234 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 { ip saddr . 345 timeout 1m }
                tcp dport 345 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 { ip saddr . 456 timeout 1m }
                tcp dport 456 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 { ip saddr . 567 timeout 1m }
                tcp dport 567 ip saddr . tcp dport @candidates_ipv4 add @clients_ipv4 { ip saddr timeout 1m } log prefix "Succ
                tcp dport 22 ip saddr @clients_ipv4 counter packets 8 bytes 3168 accept
                tcp dport 22 ct state established,related counter packets 0 bytes 0 accept
                tcp dport 22 counter packets 8 bytes 480 drop
        }
}
```

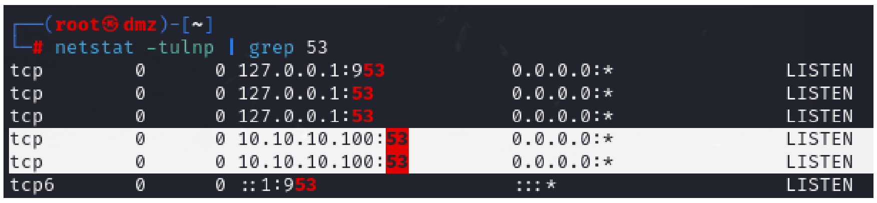
Figure 24: Corresponding rules in the NFTable

C - Domain name resolutions using the dns server



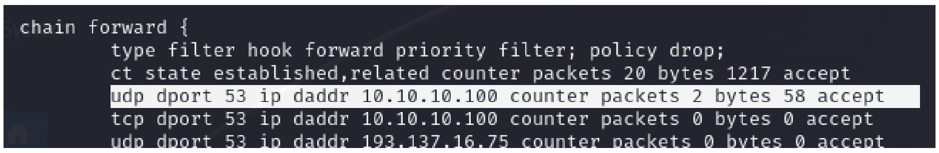
```
┌──(root㉿cliente)-[~]
└─# nc -zvu 10.10.10.100 53
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 53 (domain) open
```

Figure 25: Command to initiate connection



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 53
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.10.10.100:53         0.0.0.0:*               LISTEN
tcp        0      0 10.10.10.100:53         0.0.0.0:*               LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

Figure 26: State of the port when receiving the request

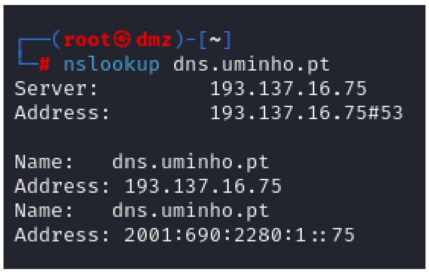


```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 20 bytes 1217 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 2 bytes 58 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
```

Figure 27: Corresponding rule in the NFTable

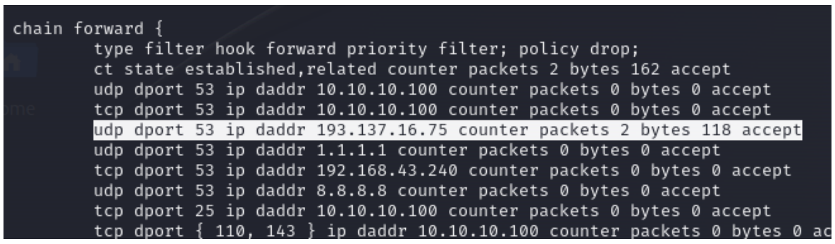
D - The dns server must resolve names using DNS servers on the Internet (dns2 and 1.1.1.1)

DNS of Universidade do Minho (dns2)



```
┌──(root㉿dmz)-[~]
└─# nslookup dns.uminho.pt
Server:         193.137.16.75
Address:        193.137.16.75#53

Name:   dns.uminho.pt
Address: 193.137.16.75
Name:   dns.uminho.pt
Address: 2001:690:2280:1::75
```

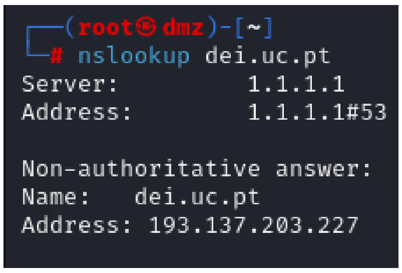
Figure 28: DNS name resolution request



```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 2 bytes 162 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 2 bytes 118 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 0 bytes 0 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 ac
```

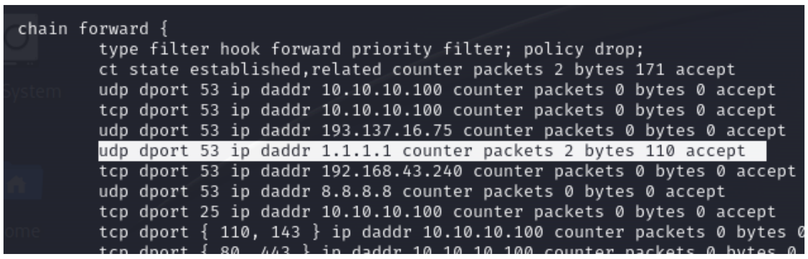
Figure 29: Corresponding rule in the NFTable

Server 1.1.1.1



```
┌──(root㉿dmz)-[~]
└─# nslookup dei.uc.pt
Server:         1.1.1.1
Address:        1.1.1.1#53

Non-authoritative answer:
Name:   dei.uc.pt
Address: 193.137.203.227
```

Figure 30: DNS name resolution request



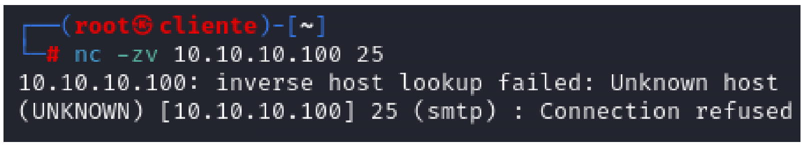
```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 2 bytes 171 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 2 bytes 110 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 0 bytes 0 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0
```

Figure 31: Corresponding rule in the NFTable

E - The dns and dns2 servers should be able to synchronize the contents of DNS zones

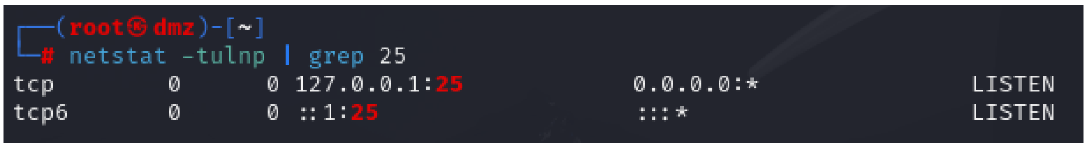
Not implemented/working.

F - SMTP connections to the smtp server



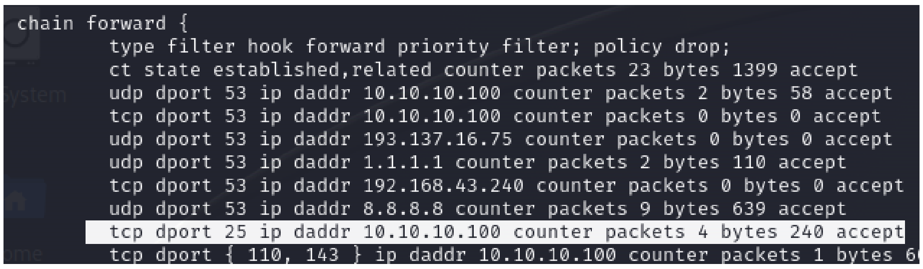
```
┌──(root㉿cliente)-[~]
└─# nc -zv 10.10.10.100 25
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 25 (smtp) : Connection refused
```

Figure 32: Command to initiate connection



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 25
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 ::1:25                  :::*                    LISTEN
```

Figure 33: State of the port when receiving the request



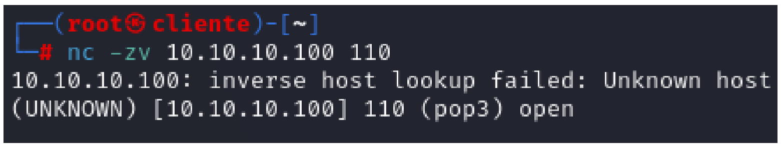
```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 23 bytes 1399 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 2 bytes 58 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 2 bytes 110 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 9 bytes 639 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 4 bytes 240 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 1 bytes 6
```

Figure 34: Corresponding rule in the NFTable

For the smtp server, the connection appears to be refused, which does not make sense as the port is open (Figure 33) and the rule is made to accept the traffic/connections (Figure 34).

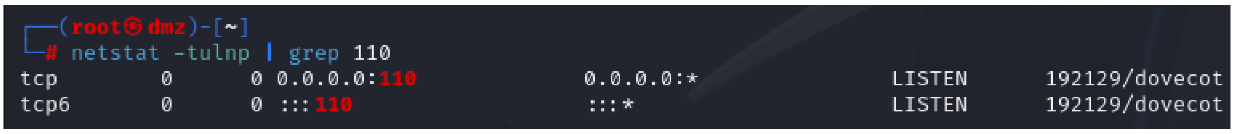
G - POP and IMAP connections to the mail server

POP connection



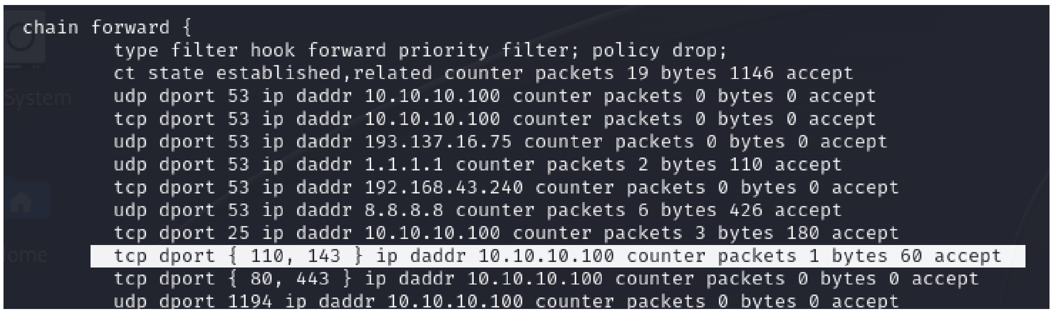
```
┌──(root㉿cliente)-[~]
└─# nc -zv 10.10.10.100 110
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 110 (pop3) open
```

Figure 35: Command to initiate connection



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 110
tcp        0      0 0.0.0.0:110            0.0.0.0:*               LISTEN      192129/dovecot
tcp6       0      0 :::110                 :::*                    LISTEN      192129/dovecot
```

Figure 36: State of the port when receiving the request



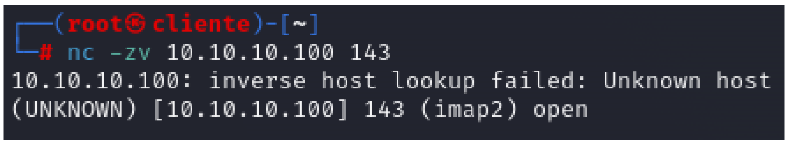
```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 19 bytes 1146 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 2 bytes 110 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 6 bytes 426 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 3 bytes 180 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 1 bytes 60 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
```

Figure 37: Corresponding rule in the NFTable

.

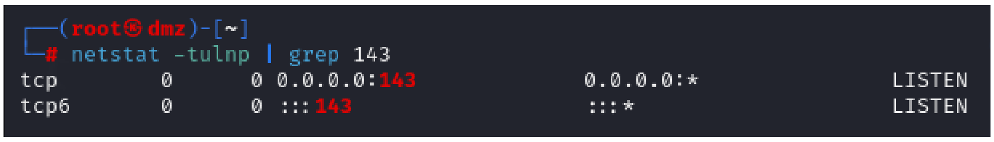
IMAP connection

.



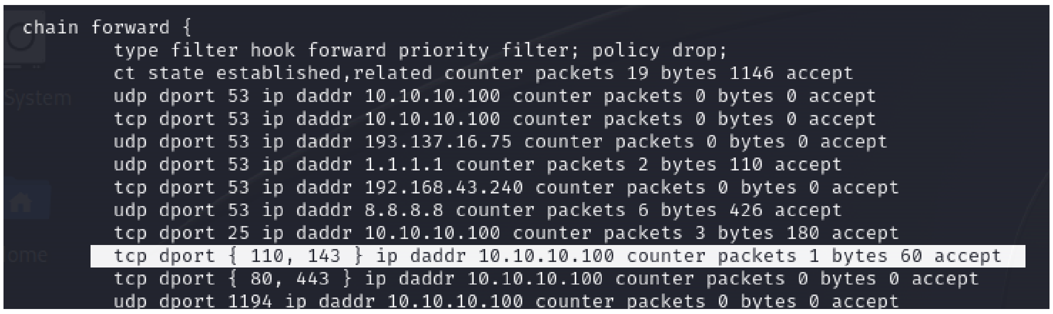
```
┌──(root㉿cliente)-[~]
└─# nc -zv 10.10.10.100 143
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 143 (imap2) open
```

Figure 38: Command to initiate connection



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 143
tcp        0      0 0.0.0.0:143             0.0.0.0:*               LISTEN
tcp6       0      0 :::143                  :::*                    LISTEN
```

Figure 39: State of the port when receiving the request

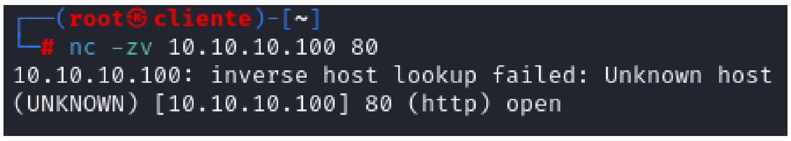


```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 19 bytes 1146 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 2 bytes 110 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 6 bytes 426 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 3 bytes 180 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 1 bytes 60 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
```

Figure 40: Corresponding rule in the NFTable

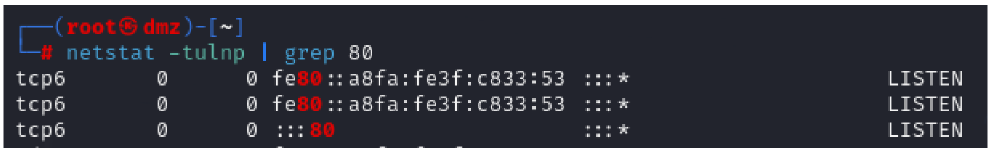
H - HTTP and HTTPS connections to the www server

HTTP connection



```
┌──(root㉿cliente)-[~]
└─# nc -zv 10.10.10.100 80
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 80 (http) open
```

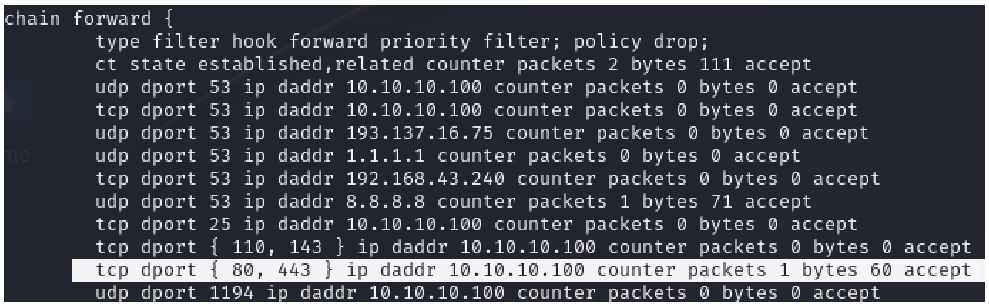
Figure 41: Command to initiate connection



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 80
tcp6       0      0 fe80::a8fa:fe3f:c833:53 :::*                    LISTEN
tcp6       0      0 fe80::a8fa:fe3f:c833:53 :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
```

Figure 42: State of the port when receiving the request

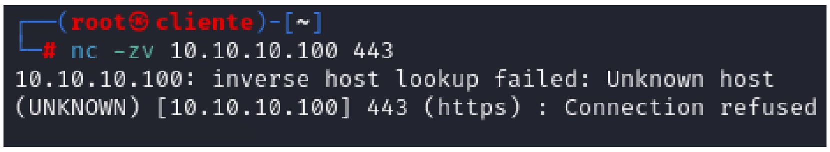
.



```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 2 bytes 111 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 1 bytes 71 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 1 bytes 60 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
```

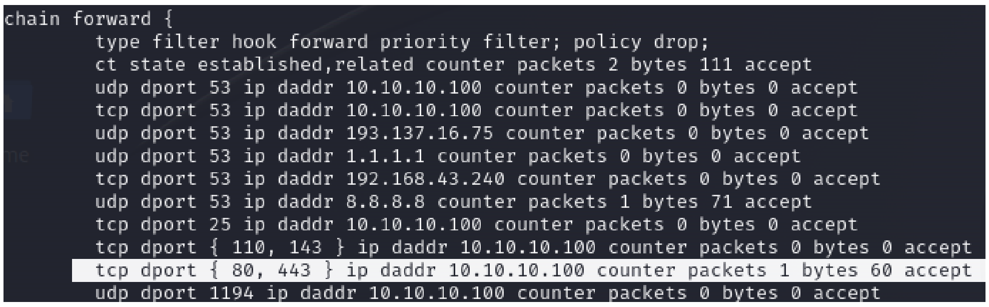
Figure 43: Corresponding rule in the NFTable

HTTPS connection



```
┌──(root㉿cliente)-[~]
└─# nc -zv 10.10.10.100 443
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 443 (https) : Connection refused
```

Figure 44: Command to initiate connection

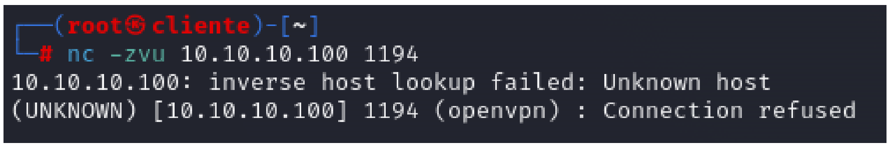


```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 2 bytes 111 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 1 bytes 71 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 1 bytes 60 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
```

Figure 45: Corresponding rule in the NFTable

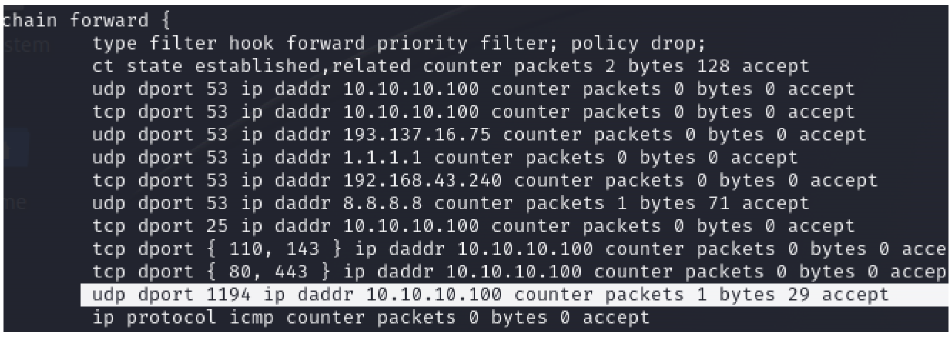
Here in the case of HTTPS, not only the connection appears to be refused, the ports also do not show up. We do not understand why, as the rule is built so that the connections are accepted.

I - OpenVPN connections to the vpn-gw server



```
┌──(root㉿cliente)-[~]
└─# nc -zvu 10.10.10.100 1194
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 1194 (openvpn) : Connection refused
```

Figure 46: Command to initiate connection

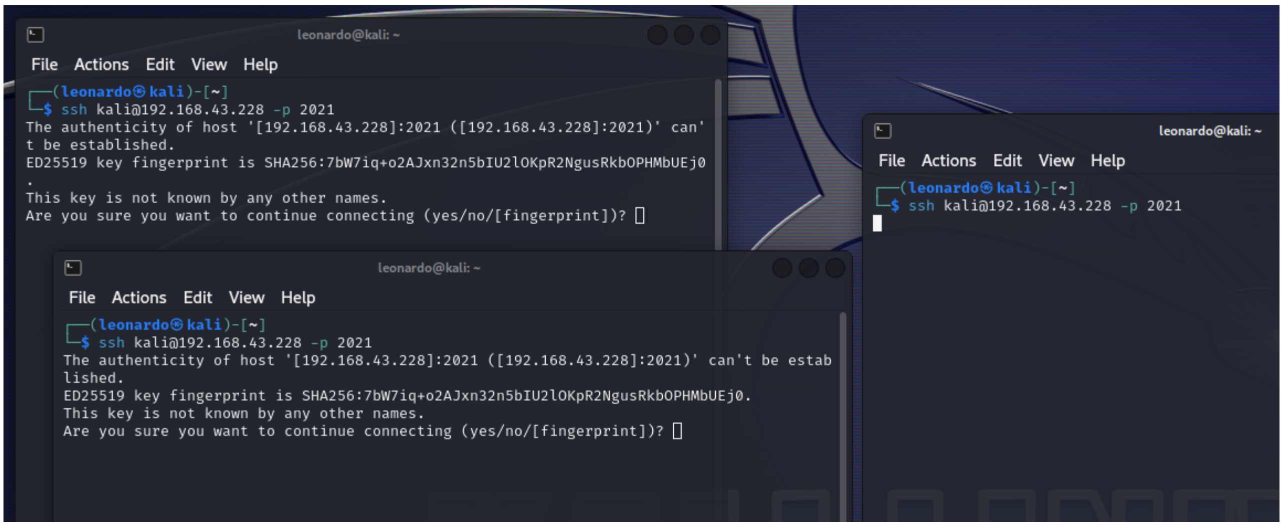


```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 2 bytes 128 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 1 bytes 71 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 acce
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accep
        udp dport 1194 ip daddr 10.10.10.100 counter packets 1 bytes 29 accept
        ip protocol icmp counter packets 0 bytes 0 accept
```

Figure 47: Corresponding rule in the NFTable

Once more, not only the connection appears to be refused, but also the ports also do not show up. We do not understand why, as the rule is built so that the connections are accepted.

J - SSH connections towards port 2021 of the Linux Router must be redirected to the SSH port of the datastore server and must be limited to 2 simultaneous connections

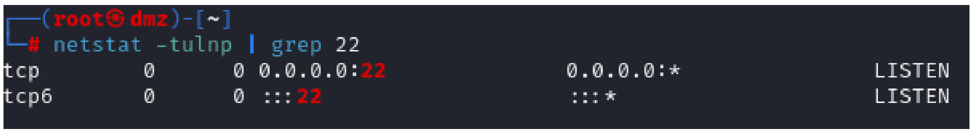


```
┌──(leonardo㉿kali)-[~]
└─$ ssh kali@192.168.43.228 -p 2021
The authenticity of host '[192.168.43.228]:2021 ([192.168.43.228]:2021)' can'
t be established.
ED25519 key fingerprint is SHA256:7bW7iq+o2AJxn32n5bIU2lOKpR2NgusRkbOPHMbUEj0
.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

```
┌──(leonardo㉿kali)-[~]
└─$ ssh kali@192.168.43.228 -p 2021
The authenticity of host '[192.168.43.228]:2021 ([192.168.43.228]:2021)' can't be estab
lished.
ED25519 key fingerprint is SHA256:7bW7iq+o2AJxn32n5bIU2lOKpR2NgusRkbOPHMbUEj0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

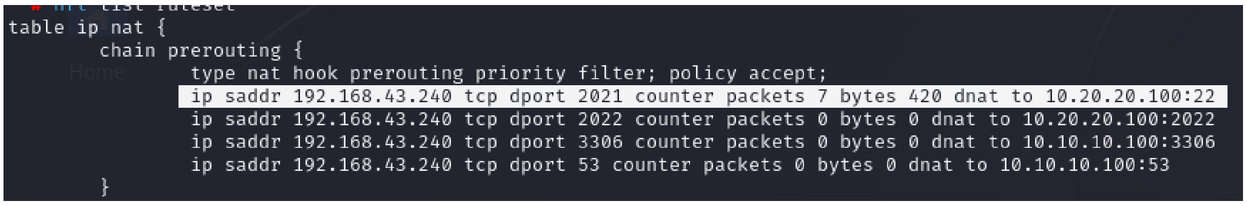
```
┌──(leonardo㉿kali)-[~]
└─$ ssh kali@192.168.43.228 -p 2021
```

Figure 48: Proof that 3 connections simultaneously do not work



```
┌──(root㉿dmz)-[~]
└─# netstat -tulnp | grep 22
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

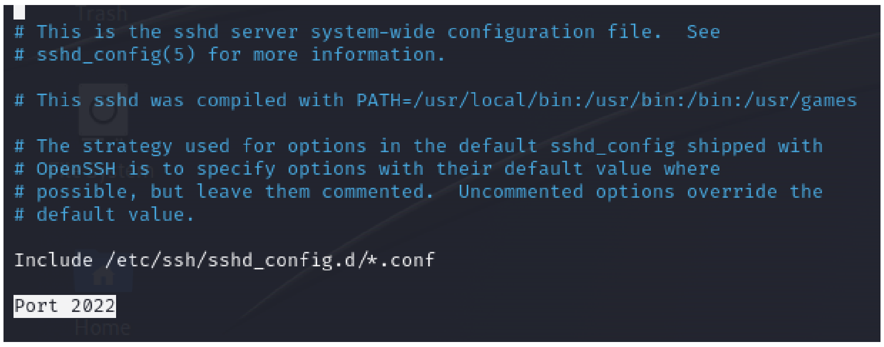
Figure 49: State of the port when receiving the request



```
table ip nat {
        chain prerouting {
                type nat hook prerouting priority filter; policy accept;
                ip saddr 192.168.43.240 tcp dport 2021 counter packets 7 bytes 420 dnat to 10.20.20.100:22
                ip saddr 192.168.43.240 tcp dport 2022 counter packets 0 bytes 0 dnat to 10.20.20.100:2022
                ip saddr 192.168.43.240 tcp dport 3306 counter packets 0 bytes 0 dnat to 10.10.10.100:3306
                ip saddr 192.168.43.240 tcp dport 53 counter packets 0 bytes 0 dnat to 10.10.10.100:53
        }
```

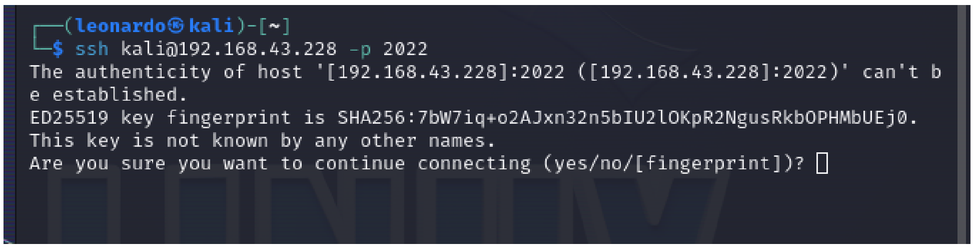
Figure 50: Corresponding rule in the NFTable

K - Connections towards port 2022 of the Linux Router must be redirected to port 2022 of the STI CA server (internal network) and must only be allowed from a single IP address on the Internet



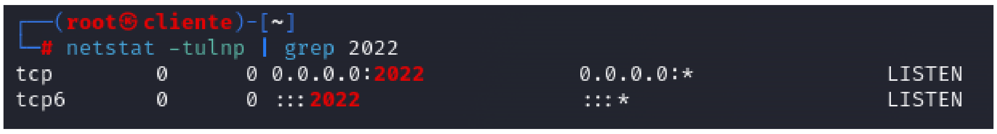
```
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 2022
```

Figure 51: Configuration needed before making the connection



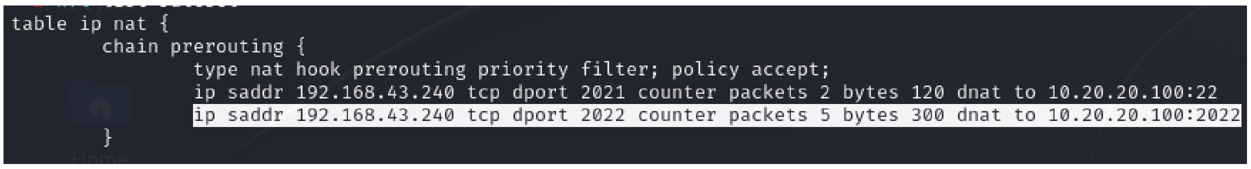
```
┌──(leonardo㉿kali)-[~]
└─$ ssh kali@192.168.43.228 -p 2022
The authenticity of host '[192.168.43.228]:2022 ([192.168.43.228]:2022)' can't b
e established.
ED25519 key fingerprint is SHA256:7bW7iq+o2AJxn32n5bIU2lOKpR2NgusRkbOPHMbUEj0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Figure 52: Command to initiate connection



```
┌──(root㉿cliente)-[~]
└─# netstat -tulnp | grep 2022
tcp        0      0 0.0.0.0:2022            0.0.0.0:*               LISTEN
tcp6       0      0 :::2022                 :::*                    LISTEN
```

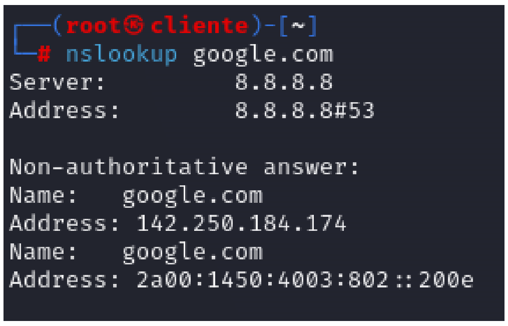
Figure 53: State of the port when receiving the request



```
table ip nat {
        chain prerouting {
                type nat hook prerouting priority filter; policy accept;
                ip saddr 192.168.43.240 tcp dport 2021 counter packets 2 bytes 120 dnat to 10.20.20.100:22
                ip saddr 192.168.43.240 tcp dport 2022 counter packets 5 bytes 300 dnat to 10.20.20.100:2022
        }
```

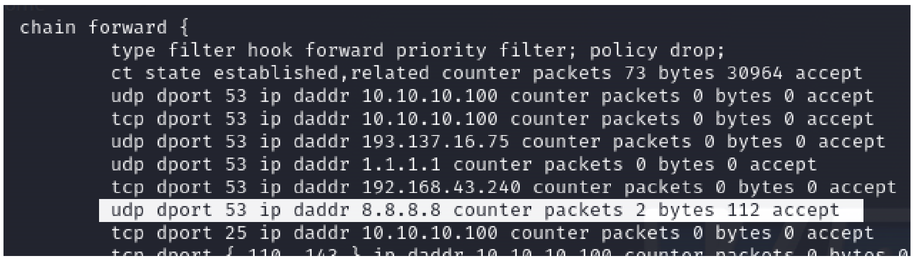
Figure 54: Corresponding rule in the NFTable

L - Domain name resolutions using DNS



```
┌──(root㉿cliente)-[~]
└─# nslookup google.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.184.174
Name:   google.com
Address: 2a00:1450:4003:802::200e
```

Figure 55: DNS name resolution request

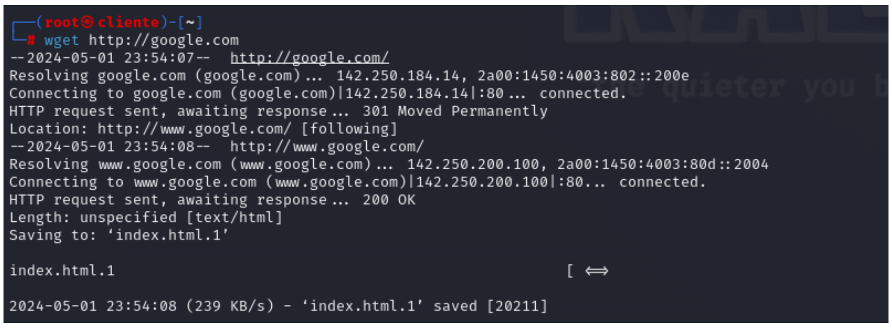


```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 73 bytes 30964 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 2 bytes 112 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0
```

Figure 56: Corresponding rule in the NFTable

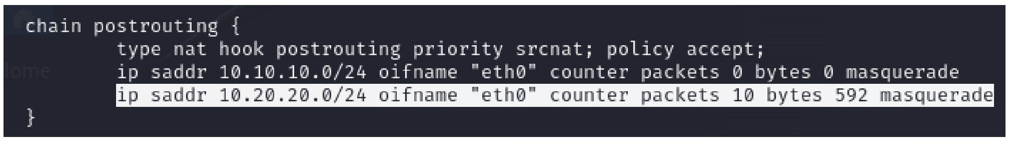
M - HTTP, HTTPS, and SSH connections from devices with a dynamic IP address (DHCP clients)

HTTP connection



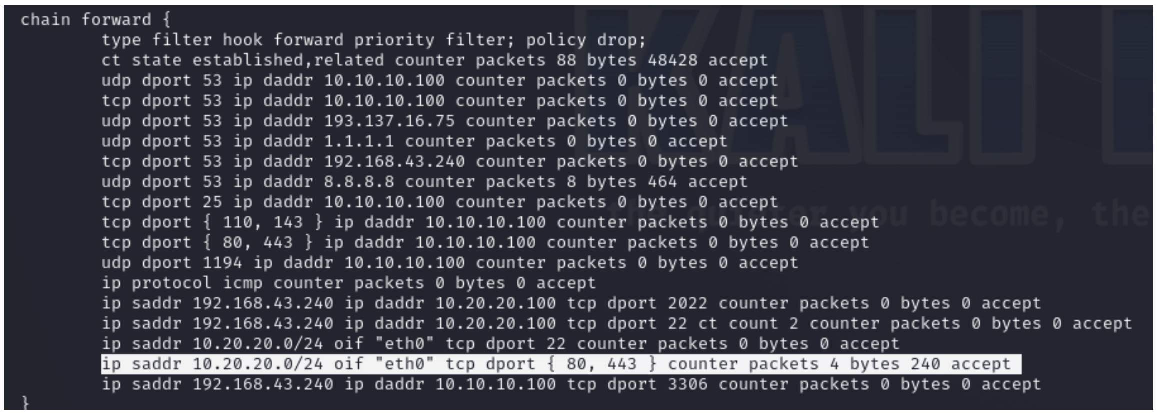
```
┌──(root㉿cliente)-[~]
└─# wget http://google.com
--2024-05-01 23:54:07--  http://google.com/
Resolving google.com (google.com)... 142.250.184.14, 2a00:1450:4003:802::200e
Connecting to google.com (google.com)|142.250.184.14|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.google.com/ [following]
--2024-05-01 23:54:08--  http://www.google.com/
Resolving www.google.com (www.google.com)... 142.250.200.100, 2a00:1450:4003:80d::2004
Connecting to www.google.com (www.google.com)|142.250.200.100|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html.1'

index.html.1                                   [ ⟺

2024-05-01 23:54:08 (239 KB/s) - 'index.html.1' saved [20211]
```

Figure 57: Command to start the web page download (HTTP)



```
chain postrouting {
        type nat hook postrouting priority srcnat; policy accept;
        ip saddr 10.10.10.0/24 oifname "eth0" counter packets 0 bytes 0 masquerade
        ip saddr 10.20.20.0/24 oifname "eth0" counter packets 10 bytes 592 masquerade
}
```

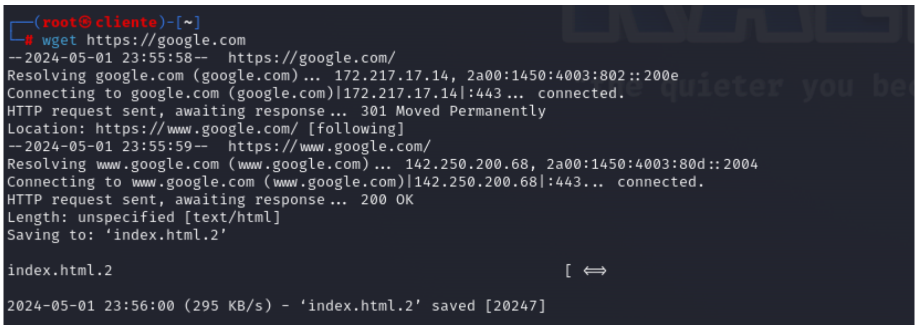
Figure 58: Corresponding rule in the NFTable (1)



```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 88 bytes 48428 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 8 bytes 464 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        ip protocol icmp counter packets 0 bytes 0 accept
        ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 2022 counter packets 0 bytes 0 accept
        ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 22 ct count 2 counter packets 0 bytes 0 accept
        ip saddr 10.20.20.0/24 oif "eth0" tcp dport 22 counter packets 0 bytes 0 accept
        ip saddr 10.20.20.0/24 oif "eth0" tcp dport { 80, 443 } counter packets 4 bytes 240 accept
        ip saddr 192.168.43.240 ip daddr 10.10.10.100 tcp dport 3306 counter packets 0 bytes 0 accept
}
```

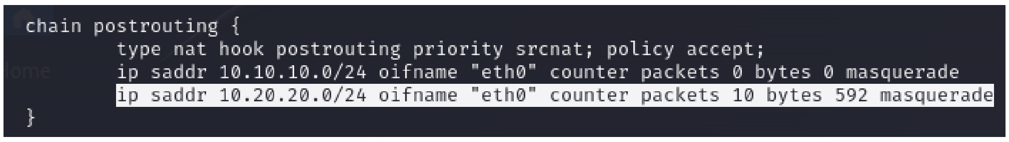
Figure 59: Corresponding rule in the NFTable (2)

HTTPS connection



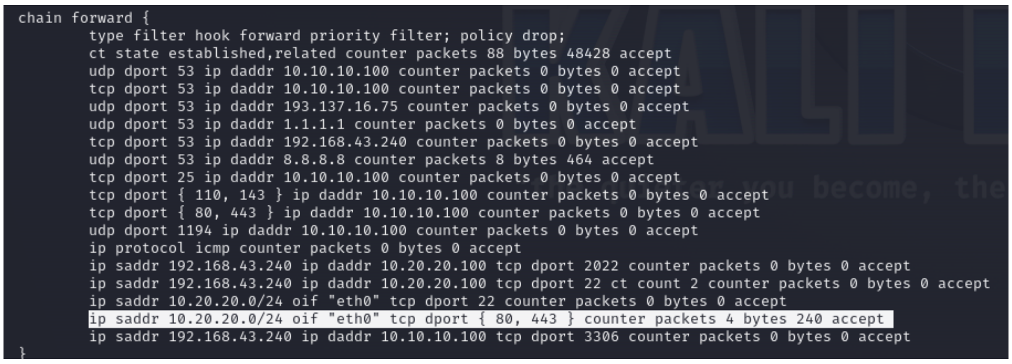
```
┌──(root㉿cliente)-[~]
└─# wget https://google.com
--2024-05-01 23:55:58--  https://google.com/
Resolving google.com (google.com)... 172.217.17.14, 2a00:1450:4003:802::200e
Connecting to google.com (google.com)|172.217.17.14|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.google.com/ [following]
--2024-05-01 23:55:59--  https://www.google.com/
Resolving www.google.com (www.google.com)... 142.250.200.68, 2a00:1450:4003:80d::2004
Connecting to www.google.com (www.google.com)|142.250.200.68|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘index.html.2’

index.html.2                                            [ <=>

2024-05-01 23:56:00 (295 KB/s) - ‘index.html.2’ saved [20247]
```

Figure 60: Command to start the web page download (HTTPS)



```
chain postrouting {
        type nat hook postrouting priority srcnat; policy accept;
        ip saddr 10.10.10.0/24 oifname "eth0" counter packets 0 bytes 0 masquerade
        ip saddr 10.20.20.0/24 oifname "eth0" counter packets 10 bytes 592 masquerade
}
```

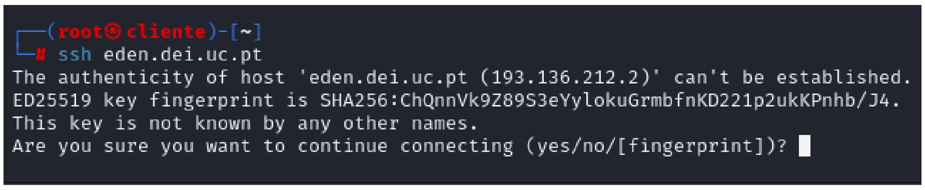
Figure 61: Corresponding rule in the NFTable (1)



```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 88 bytes 48428 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 8 bytes 464 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        ip protocol icmp counter packets 0 bytes 0 accept
        ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 2022 counter packets 0 bytes 0 accept
        ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 22 ct count 2 counter packets 0 bytes 0 accept
        ip saddr 10.20.20.0/24 oif "eth0" tcp dport 22 counter packets 0 bytes 0 accept
        ip saddr 10.20.20.0/24 oif "eth0" tcp dport { 80, 443 } counter packets 4 bytes 240 accept
        ip saddr 192.168.43.240 ip daddr 10.10.10.100 tcp dport 3306 counter packets 0 bytes 0 accept
}
```

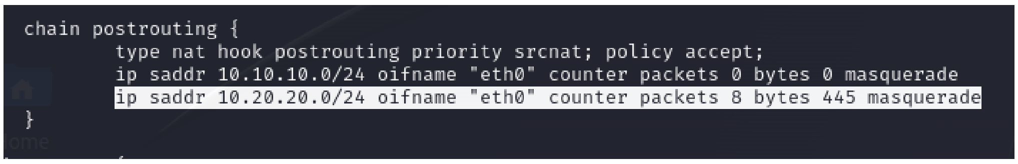
Figure 62: Corresponding rule in the NFTable (2)

SSH connection



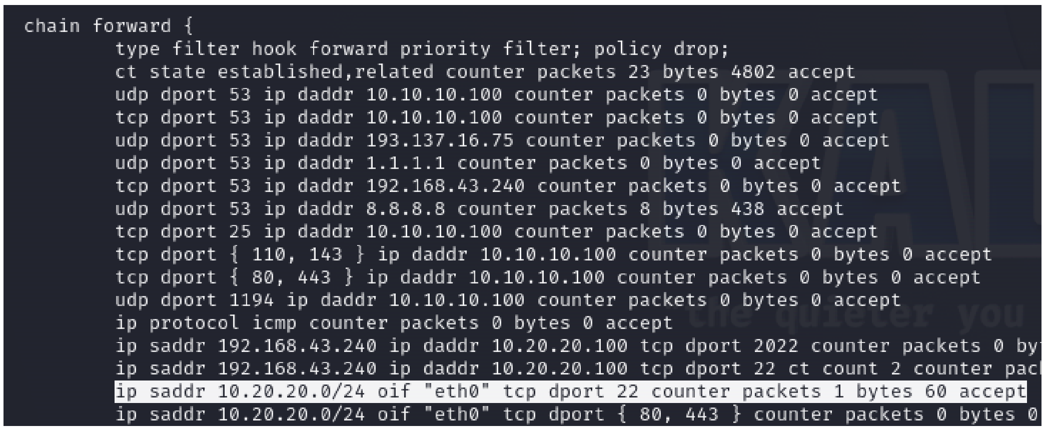
```
┌──(root㉿cliente)-[~]
└─# ssh eden.dei.uc.pt
The authenticity of host 'eden.dei.uc.pt (193.136.212.2)' can't be established.
ED25519 key fingerprint is SHA256:ChQnnVk9Z89S3eYylokuGrmbfnKD221p2ukKPnhb/J4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Figure 63: Command to initiate connection



```
chain postrouting {
        type nat hook postrouting priority srcnat; policy accept;
        ip saddr 10.10.10.0/24 oifname "eth0" counter packets 0 bytes 0 masquerade
        ip saddr 10.20.20.0/24 oifname "eth0" counter packets 8 bytes 445 masquerade
}
```

Figure 64: Corresponding rule in the NFTable (1)



```
chain forward {
        type filter hook forward priority filter; policy drop;
        ct state established,related counter packets 23 bytes 4802 accept
        udp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 193.137.16.75 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 1.1.1.1 counter packets 0 bytes 0 accept
        tcp dport 53 ip daddr 192.168.43.240 counter packets 0 bytes 0 accept
        udp dport 53 ip daddr 8.8.8.8 counter packets 8 bytes 438 accept
        tcp dport 25 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 110, 143 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        tcp dport { 80, 443 } ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        udp dport 1194 ip daddr 10.10.10.100 counter packets 0 bytes 0 accept
        ip protocol icmp counter packets 0 bytes 0 accept
        ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 2022 counter packets 0 by
        ip saddr 192.168.43.240 ip daddr 10.20.20.100 tcp dport 22 ct count 2 counter pac
        ip saddr 10.20.20.0/24 oif "eth0" tcp dport 22 counter packets 1 bytes 60 accept
        ip saddr 10.20.20.0/24 oif "eth0" tcp dport { 80, 443 } counter packets 0 bytes 0
```

Figure 65: Corresponding rule in the NFTable (2)

Conclusion

To conclude, in this assignment, a lot of knowledge was gathered about Firewall configurations, mainly how NFTables function, and IDS/IPS's, mainly Suricata. Also, other concepts were worked on, like network fundaments, research skills, and problem-solving abilities.

It was also enlightening to learn about the audit tools Linux has, in order for us to be as prepared as possible when facing the security of real-life systems.