



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA



Practical Assignment #2

2023/2024

Master in Informatics Engineering
Information Technology Security

Bruno Sequeira 2020235721, brunosequeira@student.dei.uc.pt
Rui Santos 2020225542, rpsantos@student.dei.uc.pt

May 4, 2024

Contents

1	Introduction	1
2	Architecture	1
3	Software	2
4	Concepts	2
4.1	IPTables	2
4.2	Suricata	3
5	IPTables Configuration	3
5.1	Firewall configuration to protect the router	3
5.1.1	DNS name resolution requests sent to outside servers	3
5.1.2	SSH connections to the router system, if originated at the internal network or at the VPN gateway (vpn-gw).	4
5.2	Firewall configuration to authorize direct communications (without NAT)	4
5.2.1	Domain name resolutions using the dns server	4
5.2.2	The dns server should be able to resolve names using DNS servers on the Internet (dns2 and also others)	5
5.2.3	The dns and dns2 servers should be able to synchronize the contents of DNS zones.	5
5.2.4	SMTP connections to the smtp server	6
5.2.5	POP and IMAP connections to the mail server.	6
5.2.6	HTTP and HTTPS connections to the www server	7
5.2.7	OpenVPN connections to the vpn-gw server	8
5.2.8	VPN clients connected to the gateway (vpn-gw) should be able to connect to all services in the Internal network (assume the gateway does SNAT/MASQUERADING for communications received from clients).	8
5.3	Firewall configuration for connections to the external IP address of the firewall (using NAT)	9
5.3.1	FTP connections (in passive and active modes) to the ftp server	9
5.3.2	SSH connections to the datastore server, but only if originated at the eden or dns2 servers	10
5.4	Firewall configuration for communications from the internal network to the outside (using NAT)	11
5.4.1	Domain name resolutions using DNS	11
5.4.2	HTTP, HTTPS and SSH connections	12
5.4.3	FTP connections (in passive and active modes) to external FTP servers	13
6	Suricata	14
6.1	Suricata configuration	14
6.2	Tests	16
7	Conclusion	18

List of Figures

1	Network Architecture	1
2	Netcat command in the Router UDP and TCP	3
3	Netcat command in the DMZ's dns server UDP and TCP	4
4	SSH from vpn-gw	4
5	SSH from internal network	4
6	Netcat command DMZ's dns server	5
7	Netcat command internal network	5
8	Netcat command DMZ's dns server	5
9	Netcat command external network	5
10	Netcat command DMZ's dns server	6
11	Netcat command external network	6
12	Netcat command DMZ's smtp server	6
13	Netcat command internal network	6
14	Netcat command DMZ's mail server	7
15	Netcat command internal network	7
16	Netcat command DMZ's www server	7
17	Netcat command internal network	7
18	Netcat command DMZ's vpn-gw server	8
19	Netcat command internal network	8
20	Netcat command DMZ's vpn-gw server	8
21	Netcat command internal network (datastore)	8
22	Ftp connection from the external network in active and passive modes	10
23	ssh from the external network	11
24	Netcat command External network	11
25	Netcat command internal network	12
26	Netcat command internal network	12
27	Netcat command external network	12
28	ssh command internal network	13
29	ftp command internal network	13
30	HOME_NET	14
31	af_packet	14
32	suricata.rules path	15
33	Drop Table query being in the sender	16
34	Drop Table in the receiver	16
35	Log - /var/log/suricata/fast.log	16
36	Drop Table query being in the sender	16
37	Drop Table in the receiver	16
38	Log - /var/log/suricata/fast.log	17
39	Drop Table in the sender	17
40	Log - /var/log/suricata/fast.log	17
41	Drop Table in the sender	17
42	Log - /var/log/suricata/fast.log	17
43	nmap command in the sender	17
44	Log - /var/log/suricata/fast.log	18

1 Introduction

In the evolving field of Informatics Engineering, the significance of Information Technology Security cannot be overstated. This practical assignment delves into the intricate architecture and software configurations necessary to safeguard a network infrastructure. By implementing IPTables and Suricata, we aim to establish a secure environment capable of dealing with potential threats and ensuring the integrity of our system.

2 Architecture

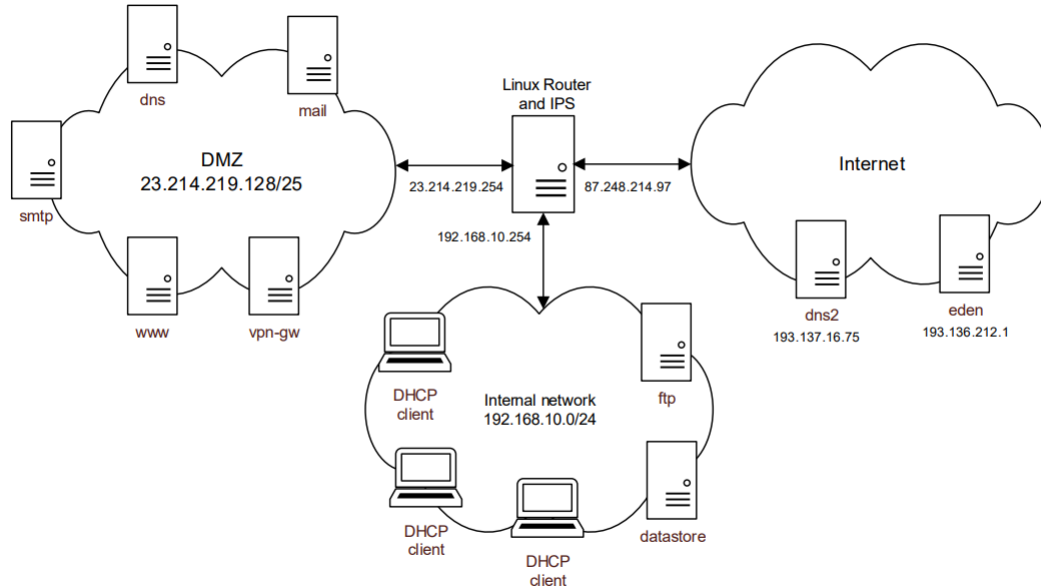


Figure 1: Network Architecture

To do this assignment, he needed three computers, one for the LinuxRouter and IDS, one for the DMZ and another for the internal network. The DMZ have the DNS, mail, smtp, www and vpn-gw servers and the Internal Network computer have DHCP clients, FTP server and a datastore. In each computer we will have a dedicated IP address for each of those services.

Networks:

- DMZ: 23.214.219.218/25
- External: 87.248.214.0/24
- Internal: 192.168.10.0/24

Services IP:

- Linux Router
 - DMZ interface: 23.214.219.254
 - internal network interface: 192.168.10.254
 - external network interface: 87.248.214.97
- DMZ
 - dns: 23.214.219.130
 - mail: 23.214.219.131

- smtp: 23.214.219.132
- www: 23.214.219.133
- vpn-gw: 23.214.219.134
- external network: 87.248.214.89
- Internal Network
 - DHCP Client 1: 192.168.10.2
 - DHCP Client 2: 192.168.10.3
 - DHCP Client 3: 192.168.10.4
 - ftp: 192.168.10.5
 - datastore: 192.168.10.6
 - external network: 87.248.214.90

3 Software

To complete the assignment we used three virtual machines, all CentOS Stream 9. In the Linux Router virtual machine we needed:

- suricata: the technology that we use to detect and block attacks
- iptables: our firewall

On the other machines we needed:

- netcat: to test most of the connections
- vsftpd: to create ftp clients and servers
- nmap: to test OS Fingerprint attacks
- hping3 to test DoS attacks

4 Concepts

4.1 IPTables

IPTables is a firewall that works by matching each packet that crosses the networking interface against a set of rules to decide what to do.

The rules define the characteristics that a packet needs to have to be a match and the action to be taken. Those rules are saved in tables. There are three tables in IPTables, *nat*, *filter* and *mangle*. In this project we will only work with the first two.

The rules inside the tables are grouped in **chains** and users can create chains as needed, but in our case it was not necessary. In the table *filter* there are three default chains: **INPUT**, **OUTPUT** and **FORWARD**. The chain **INPUT** handles the packets addressed to the router, the chain **OUTPUT** handles the traffic created by our router and the chain **FORWARD** handles the traffic that needs to be redirected by the router to another system. In the table *nat* we have the chains **POSTROUTING** and **PREROUTING**. The **PREROUTING** chain is for altering packets before they are routed. It is used for **DNAT** for incoming traffic. The **POSTROUTING** chain is for altering packets as they are leaving the system. It is used for **SNAT** for outgoing traffic.

The rules have actions that define if a packet is dropped/rejected (**DROP**) or if a packet can proceed to his destination (**ACCEPT**).

4.2 Suricata

Suricata is a open source network analysis and threat detection software used by multiple organizations. By default, suricata runs as an Intrusion Detection System (IDS), that means that it only generates alerts and logs of suspicious activity. We are going to configure it to also be in Intrusion Prevention System (IPS) so it can also drop the suspicious traffic. We are also going to use Suricata in inline mode, that is a mode where the packets are sent from IPTables to Suricata using a queue (**NFQUEUE**) and if a packet in the queue have some characteristic that matches one of the suricata's rules it will be dropped. The rules follow a format wich is as follows:

action protocol source(ports) → destination(ports) options

The actions are what determines what happens when a signature matches, the protocol is what protocol does the rule apply, the source and destination are the specification of the source and destination of the traffic and the ports are the ports that the traffic comes in and out.

5 IPTables Configuration

In this section we are going to show how we configured our firewall in order to meet the assignment's requirements.

5.1 Firewall configuration to protect the router

Here we want the firewall to be configured so that it should drop all communications entering the router system except the ones we are going to describe next. But first we need to change the **INPUT** chain policy to **DROP**, which means that the default action for every packet that does not match a rule in this chain is **DROP**.

```
iptables -P INPUT DROP
```

5.1.1 DNS name resolution requests sent to outside servers

Since DNS uses both TCP and UDP protocols we made rules for both of those protocols in order to accept the incoming and outgoing DNS packets.

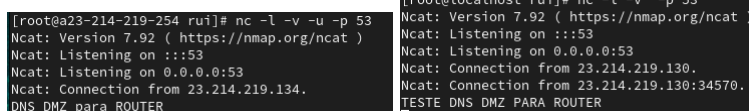
```
iptables -A INPUT -p tcp --dport 53 -j ACCEPT
iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

In order to test the rules we used netcat and in this case we put the following commands:

```
Router UDP: nc -l -v -u -p 53
DMZ (dns server) UDP: nc -v -u 23.214.219.254 53
```

```
Router TCP: nc -l -v -p 53
DMZ (dns server) TCP: nc -v 23.214.219.254 53
```

And the results were:



```
[root@a23-214-219-254 rui]# nc -l -v -u -p 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 23.214.219.134.
DNS DMZ para ROUTER

[root@localhost rui]# nc -l -v -p 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 23.214.219.130.
Ncat: Connection from 23.214.219.130:34570.
TESTE DNS DMZ PARA ROUTER
```

Figure 2: Netcat command in the Router UDP and TCP

Figure 3: Netcat command in the DMZ's dns server UDP and TCP

5.1.2 SSH connections to the router system, if originated at the internal network or at the VPN gateway (vpn-gw).

Here, we want to accept connections from the internal network (192.168.10.0/24) and from the VPN server (23.214.219.134) that use the SSH port.

```
iptables -A INPUT -s 23.214.219.134 -p tcp --dport ssh -j ACCEPT
iptables -A INPUT -s 192.168.10.0/24 -p tcp --dport ssh -j ACCEPT
```

In order to test these we used the ssh command:

DMZ (vpn-gw): `ssh rui@23.214.219.254`
 Internal network: `ssh rui@192.168.10.254`

Figure 4: SSH from vpn-gw

Figure 5: SSH from internal network

5.2 Firewall configuration to authorize direct communications (without NAT)

Here, we want our firewall to drop all communications between networks, which we did with the command:

```
iptables -P FORWARD DROP
```

However, like we did in the subsection before, there are some exceptions that we are going to describe next.

5.2.1 Domain name resolutions using the dns server

```
iptables -A FORWARD -d 23.214.219.130 -p udp --dport domain -j ACCEPT
iptables -A FORWARD -d 23.214.219.130 -p tcp --dport domain -j ACCEPT
iptables -A FORWARD -s 23.214.219.130 -p udp --sport domain -j ACCEPT
iptables -A FORWARD -s 23.214.219.130 -p tcp --sport domain -j ACCEPT
```

In order to test these we used the netcat command:

DMZ (dns server) UDP: `nc -l -v -u -p 53`

Internal Network UDP: nc -v -u 23.214.219.130 53

DMZ (dns server) TCP: nc -l -v -p 53

Internal Network TCP: nc -v 23.214.219.130 53

```
[root@localhost rui]# nc -l -v -u -p 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 192.168.10.5.
teste dns internal para dmz

[root@localhost rui]# nc -l -v -p 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 192.168.10.6.
Ncat: Connection from 192.168.10.6:51568.
teste dns internal para dmz tcp
```

Figure 6: Netcat command DMZ's dns server

```
[root@localhost rui]# nc -v -u 23.214.219.130 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.130:53.
teste dns internal para dmz

[root@localhost rui]# nc -v 23.214.219.130 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.130:53.
teste dns internal para dmz tcp
```

Figure 7: Netcat command internal network

5.2.2 The dns server should be able to resolve names using DNS servers on the Internet (dns2 and also others)

Here we are allowing the dns server to access the Internet to resolve names.

```
iptables -A FORWARD -s 23.214.219.130 -d 87.248.214.0/24 -p udp --dport domain -j ACCEPT
iptables -A FORWARD -s 87.248.214.0/24 -d 23.214.219.130 -p udp --sport domain -j ACCEPT
```

In order to test these we used the netcat command:

DMZ (dns server): nc -v -u 87.248.214.89 53

External Network: nc -l -v -u -p 53

```
[root@localhost rui]# nc -v -u 87.248.214.89 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.89:53.
teste dns dmz e external
```

Figure 8: Netcat command DMZ's dns server

```
[root@localhost rui]# nc -l -v -u 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 23.214.219.130.
teste dns dmz e external
```

Figure 9: Netcat command external network

5.2.3 The dns and dns2 servers should be able to synchronize the contents of DNS zones.

In order to synchronize the contents of DNS zones we can use TCP ports, so the rules will be the same as before but with the port TCP instead of UDP.

```
iptables -A FORWARD -s 23.214.219.130 -d 87.248.214.0/24 -p tcp --dport domain -j ACCEPT
iptables -A FORWARD -s 87.248.214.0/24 -d 23.214.219.130 -p tcp --sport domain -j ACCEPT
```

In order to test these we used the netcat command:

DMZ (dns server): nc -v 87.248.214.89 53
External Network: nc -l -v -p 53

```
[root@localhost rui]# nc -v 87.248.214.89 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.89:53.
teste dns dmz external tcp
```

Figure 10: Netcat command DMZ's dns server

```
[root@localhost rui]# nc -l -v 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 23.214.219.130.
Ncat: Connection from 23.214.219.130:49574.
teste dns dmz external tcp
```

Figure 11: Netcat command external network

5.2.4 SMTP connections to the smtp server

To allow SMTP (Simple Mail Transfer Protocol) to send and receive messages we created the following rules:

```
iptables -A FORWARD -d 23.214.219.132 -p tcp --dport smtp -j ACCEPT
iptables -A FORWARD -s 23.214.219.132 -p tcp --sport smtp -j ACCEPT
```

In order to test these we used the netcat command:

DMZ (smtp server): nc -l -v -p 25
Internal Network: nc -v 23.214.219.132 25

```
[root@localhost rui]# nc -l -v -p 25
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::25
Ncat: Listening on 0.0.0.0:25
Ncat: Connection from 192.168.10.5.
Ncat: Connection from 192.168.10.5:58324.
teste smtp
```

Figure 12: Netcat command DMZ's smtp server

```
[root@localhost rui]# nc -v 23.214.219.132 25
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.132:25.
teste smtp
```

Figure 13: Netcat command internal network

5.2.5 POP and IMAP connections to the mail server.

To allow POP3 (Post Office Protocol 3) and IMAP (Internet Message Access Protocol) we created the following rules:

```
iptables -A FORWARD -d 23.214.219.131 -p tcp --dport pop3 -j ACCEPT
iptables -A FORWARD -s 23.214.219.131 -p tcp --sport pop3 -j ACCEPT
```

```
iptables -A FORWARD -d 23.214.219.131 -p tcp --dport imap -j ACCEPT
iptables -A FORWARD -s 23.214.219.131 -p tcp --sport imap -j ACCEPT
```

To test we use the next netcat commands:

DMZ (mail server) POP3: `nc -l -v -p 110`
Internal Network POP3: `nc -v 23.214.219.131 110`

DMZ (mail server) IMAP: `nc -l -v -p 143`
Internal Network IMAP: `nc -v 23.214.219.131 143`

```
[root@localhost rui]# nc -l -v -p 110
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::110
Ncat: Listening on 0.0.0.0:110
Ncat: Connection from 192.168.10.5.
Ncat: Connection from 192.168.10.5:49128.
teste pop3

[root@localhost rui]# nc -l -v -p 143
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::143
Ncat: Listening on 0.0.0.0:143
Ncat: Connection from 192.168.10.5.
Ncat: Connection from 192.168.10.5:54408.
teste imap
```

Figure 14: Netcat command DMZ's mail server

```
[root@localhost rui]# nc -v 23.214.219.131 110
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.131:110.
teste pop3

[root@localhost rui]# nc -v 23.214.219.131 143
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.131:143.
teste imap
```

Figure 15: Netcat command internal network

5.2.6 HTTP and HTTPS connections to the www server

To allow HTTP and HTTPS we created the following rules:

```
iptables -A FORWARD -d 23.214.219.133 -p tcp --dport http -j ACCEPT
iptables -A FORWARD -s 23.214.219.133 -p tcp --sport http -j ACCEPT
```

```
iptables -A FORWARD -d 23.214.219.133 -p tcp --dport https -j ACCEPT
iptables -A FORWARD -s 23.214.219.133 -p tcp --sport https -j ACCEPT
```

To test we use the next netcat commands:

DMZ (www server) HTTP: `nc -l -v -p 80`
Internal Network HTTP: `nc -v 23.214.219.133 80`

DMZ (www server) HTTPS: `nc -l -v -p 443`
Internal Network HTTPS: `nc -v 23.214.219.133 443`

```
[root@localhost rui]# nc -l -v -p 80
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 192.168.10.5.
Ncat: Connection from 192.168.10.5:48992.
teste http

[root@localhost rui]# nc -l -v -p 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.10.5.
Ncat: Connection from 192.168.10.5:46240.
teste https
```

Figure 16: Netcat command DMZ's www server

```
[root@localhost rui]# nc -v 23.214.219.133 80
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.133:80.
teste http

[root@localhost rui]# nc -v 23.214.219.133 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.133:443.
teste https
```

Figure 17: Netcat command internal network

5.2.7 OpenVPN connections to the vpn-gw server

To enable connections from the OpenVPN (open-source virtual private network) we created the following rules:

```
iptables -A FORWARD -d 23.214.219.134 -p tcp --dport openvpn -j ACCEPT
iptables -A FORWARD -s 23.214.219.134 -p tcp --sport openvpn -j ACCEPT
```

In order to test these we used the netcat command:

DMZ (vpn-gw server): `nc -l -v -p 1194`
Internal Network: `nc -v 23.214.219.134 1194`

```
[root@localhost rui]# nc -l -v -p 1194
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1194
Ncat: Listening on 0.0.0.0:1194
Ncat: Connection from 192.168.10.5.
Ncat: Connection from 192.168.10.5:56226.
teste openvpn
```

Figure 18: Netcat command DMZ's vpn-gw server

```
[root@localhost rui]# nc -v 23.214.219.134 1194
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 23.214.219.134:1194.
teste openvpn
```

Figure 19: Netcat command internal network

5.2.8 VPN clients connected to the gateway (vpn-gw) should be able to connect to all services in the Internal network (assume the gateway does SNAT/MASQUERADING for communications received from clients).

Here we want to allow all users connected to the vpn server to have access to the services in the internal network, so the next rules were created:

```
iptables -A FORWARD -s 23.214.219.134 -d 192.168.10.0/24 -p tcp -j ACCEPT
iptables -A FORWARD -d 23.214.219.134 -s 192.168.10.0/24 -p tcp -j ACCEPT
```

In order to test these we used the netcat command:

DMZ (vpn-gw server): `nc -v 192.168.10.6 1234`
Internal Network (datastore): `nc -l -v -p 1234`

```
[root@localhost rui]# nc -v 192.168.10.6 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.10.6:1234.
Test VPN Client to internal network services
```

Figure 20: Netcat command DMZ's vpn-gw server

```
[root@localhost rui]# nc -l -v -p 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 23.214.219.134.
Ncat: Connection from 23.214.219.134:53948.
Test VPN Client to internal network services
```

Figure 21: Netcat command internal network (datastore)

5.3 Firewall configuration for connections to the external IP address of the firewall (using NAT)

In this part of the assignment we want to authorize all connections originated in the outside and destined to the external IP address. So we are going to use DNAT.

5.3.1 FTP connections (in passive and active modes) to the ftp server

Here we want to configure our firewall in order to allow FTP connections in passive and active modes. The case of FTP is different than what we've done for the other protocols, since with the other protocols we only worry about one port, and in this case we need to create rules to port 20 (ftp), 21 (ftp-data) and 60000:60099 (range of high ports that we defined in the ftp server configuration) and then use DNAT to redirect the ftp connections to the external IP address to the ftp server in the internal network. To do so, we use the following commands:

```
iptables -t nat -A PREROUTING -s 87.248.214.0/24 -d 87.248.214.97 -p tcp --dport ftp
-j DNAT --to-destination 192.168.10.5
iptables -A FORWARD -d 192.168.10.5 -p tcp --dport ftp -j ACCEPT
iptables -A FORWARD -s 192.168.10.5 -p tcp --sport ftp -j ACCEPT
iptables -A FORWARD -d 192.168.10.5 -p tcp --dport ftp-data -j ACCEPT
iptables -A FORWARD -s 192.168.10.5 -p tcp --sport ftp-data -j ACCEPT
iptables -t nat -A PREROUTING -s 87.248.214.0/24 -d 87.248.214.97 -p tcp --dport 60000:60099
-m conntrack --ctstate RELATED,ESTABLISHED -j DNAT --to-destination 192.168.10.5
iptables -A FORWARD -d 87.248.214.97 -p tcp --dport 60000:60099 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -d 87.248.214.97 -p tcp --sport 60000:60099 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
```

In addition to the rules, we need some commands which are:

- `modprobe nf_conntrack_ftp`
- `modprobe nf_nat_ftp`
- `echo 1 > /proc/sys/net/netfilter/nf_conntrack_helper`

In order to test these we created an ftp server in the internal network using vsftp and connected to the router's external ip from a external ip address:

External Network: `ftp 87.248.214.97`

```

[root@https-87-248-214-89 rui]# ftp 87.248.214.97
Connected to 87.248.214.97 (87.248.214.97).
220 (vsFTPd 3.0.5)
Name (87.248.214.97:root): rui
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (87,248,214,97,234,178).
150 Here comes the directory listing.
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Desktop
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Documents
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Downloads
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Pictures
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Public
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Templates
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Videos
-rwxrwxrwx  1 0       0            805 Apr 10 14:12 my_rules
drwxrwxrwx  2 1000    1000          22 Apr 21 17:40 teste
drwx----- 2 1000    1000          6 Apr 21 22:46 testeftp
226 Directory send OK.
ftp> passive
Passive mode off.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Desktop
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Documents
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Downloads
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Pictures
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Public
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Templates
drwxrwxrwx  2 1000    1000          6 Feb 28 01:12 Videos
-rwxrwxrwx  1 0       0            805 Apr 10 14:12 my_rules
drwxrwxrwx  2 1000    1000          22 Apr 21 17:40 teste
drwx----- 2 1000    1000          6 Apr 21 22:46 testeftp
226 Directory send OK.
ftp> █

```

Figure 22: Ftp connection from the external network in active and passive modes

5.3.2 SSH connections to the datastore server, but only if originated at the eden or dns2 servers

Here, we want to create rules that allow connections from the eden and dns2 servers to the datastore server in the internal network using DNAT.

```

iptables -t nat -A PREROUTING -s 193.136.212.1 -d 87.248.214.97 -p tcp --dport ssh
-j DNAT --to-destination 192.168.10.6
iptables -A FORWARD -s 193.136.212.1 -d 192.168.10.6 -p tcp --dport ssh -j ACCEPT
iptables -A FORWARD -s 192.168.10.6 -d 193.136.212.1 -p tcp --sport ssh -j ACCEPT
iptables -t nat -A PREROUTING -s 193.137.16.75 -d 87.248.214.97 -p tcp --dport ssh
-j DNAT --to-destination 192.168.10.6
iptables -A FORWARD -s 193.137.16.75 -d 192.168.10.6 -p tcp --dport ssh -j ACCEPT
iptables -A FORWARD -s 192.168.10.6 -d 193.137.16.75 -p tcp --sport ssh -j ACCEPT

```

In order to test these, we created new rules to simulate eden and dns2 servers (the rules are all the same, the only thing changing is the ip address of the source of the connection):

```

iptables -t nat -A PREROUTING -s 87.248.214.89 -d 87.248.214.97 -p tcp --dport ssh
-j DNAT --to-destination 192.168.10.6
iptables -A FORWARD -s 87.248.214.89 -d 192.168.10.6 -p tcp --dport ssh -j ACCEPT
iptables -A FORWARD -s 192.168.10.6 -d 87.248.214.89 -p tcp --sport ssh -j ACCEPT

```

External Network: `ssh rui@87.248.214.97`

```
[root@localhost rui]# ssh rui@87.248.214.97
rui@87.248.214.97's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed May 1 16:00:51 2024 from 87.248.214.89
[rui@localhost ~]$ ifconfig ens36
ens36: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.6 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::2016:d5ac:b593:60bd prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:ef:a5:85 txqueuelen 1000 (Ethernet)
    RX packets 9391 bytes 1814273 (1.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2364 bytes 203398 (198.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[rui@localhost ~]$
```

Figure 23: ssh from the external network

5.4 Firewall configuration for communications from the internal network to the outside (using NAT)

In this part of the assignment we want to authorize all connections originated in the internal network and destined to the outside. So we are going to use SNAT.

5.4.1 Domain name resolutions using DNS

Here we want to allow dns requests from the internal network to the Internet, using SNAT.

```
iptables -A FORWARD -p udp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport domain -j ACCEPT
iptables -A FORWARD -p udp -d 192.168.10.0/24 -s 87.248.214.0/24 --sport domain -j ACCEPT
iptables -t nat -A POSTROUTING -p udp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport domain -j SNAT --to-source 87.248.214.97
iptables -A FORWARD -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport domain -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.10.0/24 -s 87.248.214.0/24 --sport domain -j ACCEPT
iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport domain -j SNAT --to-source 87.248.214.97
```

To test we use the next netcat commands:

External network UDP: nc -l -v -u -p 53
Internal Network UDP: nc -v -u 87.248.214.89 53

External network TCP: nc -l -v -p 53
Internal Network TCP: nc -v 87.248.214.89 53

```
[root@localhost rui]# nc -l -v -u -p 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 87.248.214.97.
Ncat: Connection from 87.248.214.97:49788.
Teste dns internal para external udp

[rui@localhost rui]# nc -l -v -p 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::53
Ncat: Listening on 0.0.0.0:53
Ncat: Connection from 87.248.214.97.
Ncat: Connection from 87.248.214.97:49788.
teste dns internal para external
```

Figure 24: Netcat command External network

```
[root@localhost rui]# nc -v -u 87.248.214.89 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.89:53.
Teste dns internal para external udp

[root@localhost rui]# nc -v 87.248.214.89 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.89:53.
Teste dns internal para external
```

Figure 25: Netcat command internal network

5.4.2 HTTP, HTTPS and SSH connections

Now we need to create rules to allow http, https and ssh traffic that comes from the internal network to Internet using SNAT.

```
iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.0/24 -d 192.168.93.0/24 --dport
http -j SNAT --to-source 192.168.93.128
iptables -A FORWARD -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport http -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.10.0/24 -s 87.248.214.0/24 --sport http -j ACCEPT
```

```
iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport
https -j SNAT --to-source 87.248.214.97
iptables -A FORWARD -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport https -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.10.0/24 -s 87.248.214.0/24 --sport https -j ACCEPT
```

```
iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport
ssh -j SNAT --to-source 87.248.214.97
iptables -A FORWARD -p tcp -s 192.168.10.0/24 -d 87.248.214.0/24 --dport ssh -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.10.0/24 -s 87.248.214.0/24 --sport ssh -j ACCEPT
```

To test we used the nectat command for http and https and ssh command for ssh:

External network http: nc -l -v -p 80
Internal Network http: nc -v 87.248.214.89 80

External network https: nc -l -v -p 443
Internal Network https: nc -v 87.248.214.89 443

```
[root@localhost rui]# nc -v 87.248.214.89 80
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.89:80.
Teste http internal para external

[root@localhost rui]# nc -v 87.248.214.89 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.89:443.
Teste https internal para external
```

Figure 26: Netcat command internal network

```
[root@localhost rui]# nc -l -v -p 80
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 192.168.10.6.
Ncat: Connection from 192.168.10.6:33748.
Teste http internal para external

[root@localhost rui]# nc -l -v -p 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 87.248.214.97.
Ncat: Connection from 87.248.214.97:41316.
Teste https internal para external
```

Figure 27: Netcat command external network

Internal Network SSH: ssh rui@87.248.214.89

```
[root@localhost Desktop]# ssh rui@87.248.214.89
rui@87.248.214.89's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Apr 27 12:59:01 2024
[rui@localhost ~]$
```

Figure 28: ssh command internal network

5.4.3 FTP connections (in passive and active modes) to external FTP servers

In this topic we want to allow ftp connections from the internal network to an external network ftp server.

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -d 87.248.214.0/24 -p tcp --dport
ftp -j SNAT --to-source 87.248.214.97
iptables -A FORWARD -d 87.248.214.0/24 -p tcp --dport ftp -j ACCEPT
iptables -A FORWARD -s 87.248.214.0/24 -p tcp --sport ftp -j ACCEPT
iptables -A FORWARD -s 87.248.214.0/24 -p tcp --sport ftp-data -j ACCEPT
iptables -A FORWARD -d 87.248.214.0/24 -p tcp --dport ftp-data -j ACCEPT
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -d 87.248.214.0/24 -p tcp --dport
60000:60099 -m conntrack --ctstate RELATED,ESTABLISHED -j SNAT --to-source 87.248.214.97
iptables -A FORWARD -d 87.248.214.0/24 -p tcp --dport 60000:60099 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -d 87.248.214.0/24 -p tcp --sport 60000:60099 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --dport 60000:60099 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -d 192.168.10.0/24 -p tcp --sport 60000:60099 -m conntrack --ctstate RELATED,ESTABLISHED
-j ACCEPT
```

In order to test these we created an ftp server in the external network using vsftpd and connected from the internal network:

Internal Network: ftp 87.248.214.89

```
[root@localhost rui]# ftp 87.248.214.89
Connected to 87.248.214.89 (87.248.214.89).
220 (vsFTPd 3.0.5)
Name (87.248.214.89:root): rui
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
ftp> ls
227 Entering Passive Mode (87,248,214,89,234,155).
150 Here comes the directory listing.
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Desktop
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Documents
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Downloads
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Pictures
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Public
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Templates
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Videos
-rw-r--r--  1 0       0           895 Apr 23 14:22 my_rules
226 Directory send OK.
ftp> passive
Passive mode off.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Desktop
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Documents
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Downloads
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Pictures
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Public
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Templates
drwxr-xr-x  2 1000    1000          6 Feb 26 14:50 Videos
-rw-r--r--  1 0       0           895 Apr 23 14:22 my_rules
226 Directory send OK.
ftp>
```

Figure 29: ftp command internal network

6 Suricata

The last requirement of this assignment is to enable in the firewall system, the capability to detect and react to security attacks. To be more specific, we want to be able to detect and react to the following attacks:

- Two types of SQL injection.
- Two types of DoS (Denial of Service) attacks.
- OS fingerprinting attempts.

6.1 Suricata configuration

TO configure suricata we need to add our rules to a file and specify in the suricata.yaml file where the rules are. In addition to that, in the same yaml file we need to change the capture settings in the `af_packet` section, and the network information in `HOME_NET`.

```
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,192.168.10.0/24, 87.248.214.0/24, 23.214.219.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
```

Figure 30: HOME_NET

```
# Linux high speed capture support
af-packet:
  - interface: ens33
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    tpacket-v3: yes

  - interface: ens36
    cluster-id: 98
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    tpacket-v3: yes

  - interface: ens37
    cluster-id: 97
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    tpacket-v3: yes

  - interface: ens38
    cluster-id: 96
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    tpacket-v3: yes
```

Figure 31: af_packet

```
##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /etc/suricata/rules

rule-files:
- suricata.rules
```

Figure 32: suricata.rules path

After this we need to add one rule to IPTables in order to have all the traffic go through a queue and be analyzed in Suricata.

```
iptables -I FORWARD -j NFQUEUE --queue-num 0
```

And now we add the rules to the suricata.rules file:

- SQL injection:

```
drop tcp any any -> any any (msg:"SQL Injection type drop table detected"; content:"drop table"; sid:1; rev:1;)
```

```
drop tcp any any -> any any (msg: "SQL Injection type 1=1 detected"; pcre: "/OR+( )+[0-9]+( )*+=( )*[0-9]+/"; sid:2;)
```

The first rule is to detect and drop a message containing a "drop table" query. The second one detects if a query has the "OR 1=1" using a regular expression.

- DOS attacks:

```
alert udp any any -> 192.168.10.0/24 any (msg:"ALERT UDP Flood"; threshold:type both, track by_dst, count 15, seconds 3; sid:1000001; rev:1;)
```

```
alert tcp any any -> 192.168.10.0/24 any (msg:"ALERT TCP Flood"; threshold:type both, track by_dst, count 15, seconds 3; sid:1000005; rev:1;)
```

Here we are detecting both TCP and UDP flooding attacks.

- OS Fingerprint:

```
alert tcp any any -> any any (msg:"OS Fingerprinting Detected: NMAP"; flags:FPU;reference: url,nmap.org; sid:6; rev:1;)
```

This rule is to detect OS fingerprinting attacks via NMAP with the flag -sX.

All the rules follow the syntax mentioned in 4.2. We used **alert** and **drop actions** that are generate an alert when the conditions are met and drop the packet and generate an alert respectively.

We used both **TCP** and **UDP** protocols, however each rule uses `or tcp or udp`, so if the rule uses **tcp** and the attack is performed using **udp** it will not be detected.

The source ip and source ports were always **any** which means that the rules apply to all incoming IP's and ports. About the destination ports and IP's we also used **any** except in the DOS attacks where we specify a range of IP's.

Then we have the **msg** that is what appears when Suricata detects that type of attack, the **sid** that is the rule ID, **rev** that is the revision number, the **pcre** which means Perl Compatible Regular Expressions that is used to identify regular expressions, in our case the expression `"/OR+()+[0-9]+()*+=()*[0-9]+/"` is used to identify **OR 1=1**, the **content** which have a string to verify if it is in the message's content. The other parameters like **flags** and **threshold** are more specific to some types of rules.

6.2 Tests

In order to test those rules we need to start suricata and connect it to the queue (NFQUEUE) using the command:

```
sudo suricata -c /etc/suricata/suricata.yaml -q 0
```

To test SQL Injection we used netcat:

- Drop Table attack

```
[root@localhost rui]# nc 192.168.10.5 1234
siga testar
supostamente a proxima linha nao vai passar
drop table secret;
```

Figure 33: Drop Table query being in the sender

```
[root@localhost rui]# nc -l -v 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 87.248.214.89.
Ncat: Connection from 87.248.214.89:40198.
siga testar
supostamente a proxima linha nao vai passar
```

Figure 34: Drop Table in the receiver

```
[root@a23-214-219-254 rui]# sudo tail -1 /var/log/suricata/fast.log
04/27/2024-17:50:09.974505 [Drop] [**] [1:1:1] SQL Injection type drop table detected [**] [Classification: (null)] [Priority: 3] {TCP} 87.248.214.89:40198 -> 192.168.10.5:1234
[root@a23-214-219-254 rui]#
```

Figure 35: Log - /var/log/suricata/fast.log

- 1=1 attack

```
[root@localhost rui]# nc 192.168.10.5 1234
este deve passar
o proximo ja nao
select username, password from users where userid=2 OR 1=1;
^C
[root@localhost rui]# nc 192.168.10.5 1234
esta mensagem deve passar
a seguinte ja nao
select username, password from users where userid=2 OR 1=1;
```

Figure 36: Drop Table query being in the sender

```
[root@localhost rui]# nc -l -v 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 87.248.214.89.
Ncat: Connection from 87.248.214.89:55814.
esta mensagem deve passar
a seguinte ja nao
```

Figure 37: Drop Table in the receiver

```
[root@a23-214-219-254 rui]# sudo tail -1 /var/log/suricata/fast.log
04/27/2024-17:46:03.416395 [Drop] [**] [1:2:0] SQL Injection type 1=1 detected [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:55814 -> 192.168.10.5:1234
[root@a23-214-219-254 rui]#
```

Figure 38: Log - /var/log/suricata/fast.log

To test DOS attacks we used hping3 command:

- UDP flood:

hping3 -S -p 80 -2 --flood 192.168.10.6

```
[root@localhost rui]# hping3 -S -p 80 -2 --flood 192.168.10.6
HPING 192.168.10.6 (ens37 192.168.10.6): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 39: Drop Table in the sender

```
[root@localhost rui]# sudo tail -5 /var/log/suricata/fast.log
05/01/2024-18:19:12.518771 [**] [1:4:1] ALERT UDP Flood [**] [Classification: (null)] [Priority: 3] (UDP) 87.248.214.89:33194 -> 192.168.10.6:80
05/01/2024-18:19:14.353715 [**] [1:4:1] ALERT UDP Flood [**] [Classification: (null)] [Priority: 3] (UDP) 87.248.214.89:21295 -> 192.168.10.6:80
05/01/2024-18:19:18.510033 [**] [1:4:1] ALERT UDP Flood [**] [Classification: (null)] [Priority: 3] (UDP) 87.248.214.89:9737 -> 192.168.10.6:80
05/01/2024-18:19:20.050807 [**] [1:4:1] ALERT UDP Flood [**] [Classification: (null)] [Priority: 3] (UDP) 87.248.214.89:55231 -> 192.168.10.6:80
05/01/2024-18:19:23.150214 [**] [1:4:1] ALERT UDP Flood [**] [Classification: (null)] [Priority: 3] (UDP) 87.248.214.89:17297 -> 192.168.10.6:80
[root@localhost rui]#
```

Figure 40: Log - /var/log/suricata/fast.log

- TCP flood:

hping3 -S -p 80 --flood 192.168.10.6

```
[root@localhost rui]# hping3 -S -p 80 --flood 192.168.10.6
HPING 192.168.10.6 (ens37 192.168.10.6): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 41: Drop Table in the sender

```
[root@localhost rui]# sudo tail -5 /var/log/suricata/fast.log
05/01/2024-18:17:05.490623 [**] [1:5:1] ALERT TCP Flood [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:2370 -> 192.168.10.6:80
05/01/2024-18:17:07.814765 [**] [1:5:1] ALERT TCP Flood [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:9877 -> 192.168.10.6:80
05/01/2024-18:17:10.282927 [**] [1:5:1] ALERT TCP Flood [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:24130 -> 192.168.10.6:80
05/01/2024-18:17:13.807740 [**] [1:5:1] ALERT TCP Flood [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:8725 -> 192.168.10.6:80
05/01/2024-18:17:16.271411 [**] [1:5:1] ALERT TCP Flood [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:23907 -> 192.168.10.6:80
[root@localhost rui]#
```

Figure 42: Log - /var/log/suricata/fast.log

To test OS fingerprint attacks we use nmap:

nmap -sX --system-dns -p 80 192.168.10.6

```
[root@https-87-248-214-89 rui]# nmap -sX --system-dns -p 80 192.168.10.6
Starting Nmap 7.92 ( https://nmap.org ) at 2024-05-01 18:39 WEST
Nmap scan report for 192.168.10.6
Host is up (0.0021s latency).

PORT      STATE SERVICE
80/tcp    closed http

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Figure 43: nmap command in the sender

```
[root@localhost rui]# sudo tail -1 /var/log/suricata/fast.log
05/01/2024-18:46:23.403035  [**] [1:6:1] OS Fingerprinting Detected: NMAP Xmas scan [**] [Classification: (null)] [Priority: 3] (TCP) 87.248.214.89:34470 -> 192.168.10.6:80
[root@localhost rui]#
```

Figure 44: Log - /var/log/suricata/fast.log

7 Conclusion

To conclude, this assignment has successfully demonstrated the practical application of IPTables and Suricata in creating a secure network environment. With it we learn how to create IPTables rules and create rules to detect sql injection, dos and os fingerprint attacks. To validate our configuration we did a set of tests and with that we believe that the assignment requirements were successfully implemented.

References

- [1] Information Tecnology Security Slides
- [2] Jorge Granjal, Segurança Prática em Sistemas e Redes com Linux, FCA, 2017
- [3] Setup an OpenVPN server with certificate and two-factor authentication on CentOS 7 Retrieved March 8, 2023, from <https://nethack.ch/2016/12/08/setup-an-openvpn-server-with-certificate-and-two-factor-authentication-on-centos-7/>
- [4] How the iptables firewall works, Retrieved April 22, 2024 from <https://www.digitalocean.com/community/tutorials/how-the-iptables-firewall-works>
- [5] Suricata Rules Format Retrieved April 27, 2024 from <https://docs.suricata.io/en/suricata-6.0.3/rules/intro.html>
- [6] Fix FTP access from Linux client PC with firewall enabled Retrieved April 25 from <https://winaero.com/fix-ftp-access-from-linux-client-pc-with-firewall-enabled/>
- [7] iptables settings in vsftpd active /standby mode for Linux instances Retrieved April 25 from <https://www.alibabacloud.com/help/en/ecs/iptables-settings-in-vsftpd-active-or-standby-mode-for-linux-instances>
- [8] TCP FIN, NULL, and Xmas Scans Retrieved April 29, 2024 from <https://nmap.org/book/scan-methods-null-fin-xmas-scan.html>
- [9] DOS Flood With hping3 Retrieved April 29, 2024 from <https://linuxhint.com/hping3/>
- [10] DDoS Attack Detection with Suricata Retrieved April 29 from <https://medium.com/@mshulkhan/detection-attack-using-suricata-1-5ea7b2f62551>
- [11] Sql Injection Retrieved April 29 from https://www.w3schools.com/sql/sql_injection.asp