Firewall and IDS using IPTables and Snort

Practical Assignment 2 of STI - Report

Tiago Ventura – 2019243695 Sancho Simões – 2019217590

Masters in Intelligent Systems University of Coimbra

Coimbra, 23rd April 2023

Conteúdo

1 Introduction

In today's world, network security has become an integral part of any organization's overall security framework. The increasing number of cyber threats and attacks calls for a strong network security system. The primary objective of this project is to configure a network firewall using IPTables/Netfilter and Snort as an IDS/IPS system. The firewall will be responsible for detecting and reacting to security attacks against services deployed on a protected network.

The network firewall will implement packet filtering, NAT, and intrusion detection mechanisms to react against attacks from hosts on the outside (Internet). The scenario considered for this practical assignment includes the usage of a DMZ and internal networks. The DMZ network is where most of the public services of the organization are placed, while the goal of the internal network is to provide connectivity to users (clients with dynamic IP addresses) while also supporting servers with specific purposes.

The firewall configuration should drop all communications entering the router system, except those required for the normal operation of specific services. Additionally, the firewall configuration should authorize direct communications, connections to the external IP address of the firewall using NAT, and communications from the internal network to the outside (Internet) using NAT.

Moreover, the firewall system should be capable of detecting and reacting to security attacks. The firewall should be able to detect and block attacks, using Snort and IPtables. The attacks that the system should detect and block include two types of SQL injection, two types of DoS (Denial of Service) attacks, and OS fingerprinting attempts.

This report aims to document the configuration of a network firewall using IPTables/Netfilter, including packet filtering, NAT, and integration with Snort as an IDS/IPS system for intrusion detection and prevention/-reaction. The primary goal of this practical assignment is to configure a network firewall that is capable of detecting and reacting to security attacks against services deployed on a protected network. The report provides a general description of the scenario considered for the practical assignment, including the usage of a DMZ and internal networks. Additionally, the report details the requirements for the firewall configuration, such as dropping all communications entering the router system except those required for the normal operation of specific services and enabling the capability to detect and block attacks using Snort and IPTables. Overall, this report provides a comprehensive overview of the configuration of a network firewall using IPTables/Netfilter, as well as the implementation of Snort as an IDS/IPS system for intrusion detection and prevention/reaction.

2 Concepts

2.1 IPTables

IPTables is a Linux-based firewall that provides packet filtering, network address translation (NAT), and other security features.

2.2 Netfilter

Netfilter is a Linux-based packet-filtering framework that allows the manipulation and filtering of network packets. IPTables is built on top of Netfilter.

2.3 Snort

Snort is a free and open-source network intrusion detection and prevention system (IDS/IPS) that can analyze network traffic and alert administrators about potential security threats.

2.4 IDS/IPS

IDS stands for an intrusion detection system, while IPS stands for an intrusion prevention system. IDS/IPS is a security technology that monitors network traffic for signs of security threats and can take automated actions to prevent or stop them.

2.5 DMZ

DMZ stands for demilitarized zone. In the context of network security, a DMZ is a separate network segment that is isolated from the internal network and is used to host public-facing services, such as web servers or email servers.

2.6 NAT

NAT stands for network address translation. It is a technique used to map one IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.

2.7 VPN

VPN stands for a virtual private network. It is a technology that creates a secure, encrypted connection over a less secure network, such as the Internet.

2.8 SSH

SSH stands for secure shell. It is a network protocol that provides secure access to a remote computer.

3 Scenario

The practical assignment will be conducted using virtual machines running CentOS 5 as the operating system. The scenario consists of four virtual machines interconnected via a virtual network. The first virtual machine is a Linux router that connects the internal network to the internet. The second virtual machine is a DNS server that provides name resolution services to the internal network. The third virtual machine is a mail server that handles incoming and outgoing emails. The fourth virtual machine is a client that simulates a user workstation in the internal network. The client will have dynamic IP addressing assigned via DHCP. The router will be configured to perform network address translation (NAT) to allow the client to access the internet.

The objective of the assignment is to configure a network firewall using IPTables/Netfilter that can filter packets, perform NAT, and integrate with Snort for intrusion detection and prevention. The firewall should be capable of detecting and reacting to security attacks against services deployed on the protected network. The firewall will be configured to drop all communications entering the router system except for those required for the normal operation of specific services, including DNS name resolution requests and SSH connections to the router system. The firewall will also be configured to authorize direct communications between networks and to allow connections to external FTP servers, SMTP, POP, and IMAP mail servers, HTTP and HTTPS web servers, and OpenVPN connections to the VPN gateway server.

In addition to packet filtering and NAT using IPTables, Snort will be configured as an IDS/IPS system to enable the firewall to detect and react to security attacks. Snort will be used to detect and block various types of attacks, including SQL injection, DoS attacks, and OS fingerprinting attempts. The final report will describe in detail how these attacks work and how Snort can detect and block them.

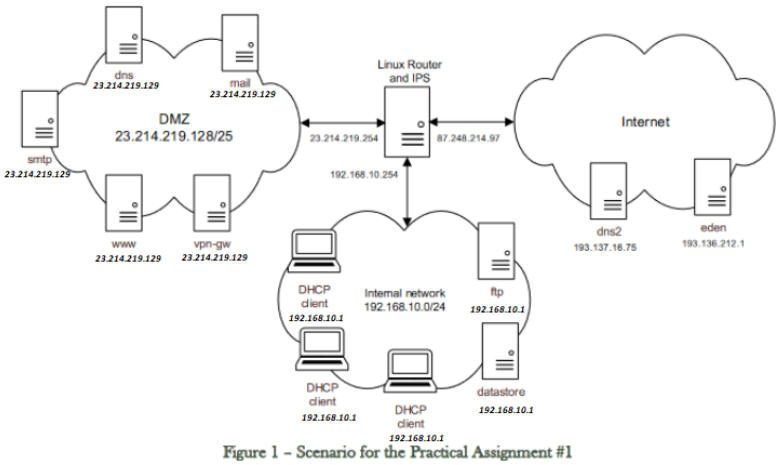

Figure 1 – Scenario for the Practical Assignment #1

4 Machine\Network\IP Configuration

For simplification and resource management, all hosts in the DMZ and internal network are hosted on the same virtual machine and therefore share the same IP address.

DMZ: The DMZ network is assigned the subnet 23.214.219.128/25. This subnet provides 126 usable IP addresses that can be used to assign to the servers and devices in the DMZ network. The default gateway for the DMZ is the IP address of the Linux router interface connected to the DMZ, which is 23.214.219.254.

Internal network: The internal network is assigned the subnet 192.168.10.0/24. This subnet provides 254 usable IP addresses that can be used to assign to the devices in the internal network. The default gateway for the internal network is the IP address of the Linux router interface connected to the internal network, which is 192.168.10.254.

dns2 and eden: The DNS servers, dns2 and eden, are assigned the IP addresses 193.137.16.75 and 193.136.212.1, respectively. These IP addresses are part of the public IP address space and are reachable from the Internet.

Linux router: The Linux router has two interfaces, one connected to the DMZ network with the IP address 23.214.219.254, and another connected to the internal network with the IP address 192.168.10.254. The Linux router also has a public IP address assigned to its interface connected to the Internet, which is 87.248.214.97. This IP address is used for communication with the Internet.

Overall, the IP configuration allows for the different networks to communicate with each other through the Linux router, with appropriate firewall rules in place to control the traffic flow. The DNS servers are reachable from both the DMZ and the internal network, and the Linux router acts as a gateway for both networks to communicate with the Internet.

5 Configuration Files

5.1 Internal Hosts

PATH = /etc/sysconfig/network-scripts/ifcfg-enp0s10

This configuration file defines the network interface enp0s10 and its settings for the internal network. Here is a breakdown of each line:

DEVICE=enp0s10: Specifies the device name for the network interface.

BOOTPROTO=static: Specifies that the IP address for this interface is manually configured, rather than obtained automatically via DHCP.

IPADDR=192.168.10.1: Specifies the IP address for this interface on the internal network.

NETMASK=255.255.255.0: Specifies the subnet mask for the internal network, which defines the range of IP addresses that are part of the network.

ONBOOT=yes: Specifies that this interface should be automatically started at boot time.

5.2 Router - DMZ Network

PATH = /etc/sysconfig/network-scripts/ifcfg-enp0s8

This configuration file defines the network interface enp0s8 and its settings for the router connection. Here is a breakdown of each line:

DEVICE=enp0s8: Specifies the device name for the network interface.

BOOTPROTO=static: Specifies that the IP address for this interface is manually configured, rather than obtained automatically via DHCP.

IPADDR=23.214.219.254: Specifies the IP address for this interface on the router connection.

NETMASK=255.255.255.0: Specifies the subnet mask for the router connection.

ONBOOT=yes: Specifies that this interface should be automatically started at boot time.

5.3 Router - Internal Network

PATH = /etc/sysconfig/network-scripts/ifcfg-enp0s10

This configuration file defines the network interface enp0s10 and its settings for the router connection. Here is a breakdown of each line:

DEVICE=enp0s10: Specifies the device name for the network interface.

BOOTPROTO=static: Specifies that the IP address for this interface is manually configured, rather than obtained automatically via DHCP.

IPADDR=192.168.10.254: Specifies the IP address for this interface on the router connection.

NETMASK=255.255.255.0: Specifies the subnet mask for the router connection.

ONBOOT=yes: Specifies that this interface should be automatically started at boot time.

5.4 Router - Internet

PATH = /etc/sysconfig/network-scripts/ifcfg-enp0s3

This configuration file defines the network interface enp0s10 and its settings for the router connection. Here is a breakdown of each line:

DEVICE=enp0s3: Specifies the device name for the network interface.

BOOTPROTO=static: Specifies that the IP address for this interface is manually configured, rather than obtained automatically via DHCP.

IPADDR=87.248.214.97: Specifies the IP address for this interface on the router connection.

NETMASK=255.255.255.0: Specifies the subnet mask for the router connection.

ONBOOT=yes: Specifies that this interface should be automatically started at boot time.

5.5 DMZ Services

PATH = /etc/sysconfig/network-scripts/ifcfg-enp0s10

This configuration file defines the network interface enp0s10 and its settings for the DMZ services. Here is a breakdown of each line:

DEVICE=enp0s10: Specifies the device name for the network interface.

BOOTPROTO=static: Specifies that the IP address for this interface is manually configured, rather than obtained automatically via DHCP.

IPADDR=23.214.219.129: Specifies the IP address for this interface on the DMZ network.

NETMASK=255.255.255.128: Specifies the subnet mask for the DMZ network.

ONBOOT=yes: Specifies that this interface should be automatically started at boot time.

6 Attacks and Prevention

6.1 SQL Injection Attacks

SQL Injection attacks are one of the most common attacks that can be launched against web applications. The attacker tries to insert malicious SQL commands into the input fields of an application. If the input is not sanitized properly, the attacker can manipulate the database and steal sensitive information or cause damage to the system. To prevent SQL injection attacks, the firewall should be configured to block all SQL injection attempts using Snort. Snort uses a set of rules to detect and block SQL injection attempts by analyzing the incoming packets and blocking any packets that match the patterns in the rules.

6.2 Denial of Service (DoS) Attacks

Denial of Service (DoS) attacks aim to make a service unavailable to its users. In a DoS attack, the attacker floods the target server with a large number of requests or traffic, causing it to become overloaded and unable to respond to legitimate requests. To prevent DoS attacks, the firewall should be configured to block any incoming traffic that exceeds a certain threshold using Snort. Snort can detect and block DoS attacks by analyzing the network traffic and blocking any packets that match the patterns in the rules.

6.3 Operating System (OS) Fingerprinting Attempts

Operating System (OS) fingerprinting is a technique used by attackers to identify the operating system of a target system. The attacker sends a series of packets to the target system and analyzes the responses to determine the operating system running on the system. This information can be used to launch targeted attacks against the system. To prevent OS fingerprinting attempts, the firewall should be configured to block any incoming packets that are attempting to fingerprint the operating system using Snort. Snort can detect and block OS fingerprinting attempts by analyzing the packets and blocking any packets that match the patterns in the rules.

6.4 Cross-site scripting (XSS)

This is a type of attack in which an attacker injects malicious code into a website that is viewed by other users. The injected code can be used to steal sensitive information from users or to manipulate the behavior of the website.

6.5 Cross-site request forgery (CSRF)

In a CSRF attack, an attacker tricks a user into acting on a website without their knowledge or consent. For example, an attacker could send a link to a victim that, when clicked, makes the victim submit a form or perform some other action on a website.

6.6 Man-in-the-middle (MitM) attack

In this type of attack, an attacker intercepts communications between two parties and can read, modify, or inject data into the communication. For example, an attacker could intercept network traffic between a user and a website and steal the user's login credentials.

6.7 Distributed denial-of-service (DDoS) attack

A DDoS attack involves overwhelming a server or network with a large volume of traffic from multiple sources, rendering it unavailable to users. Attackers can use botnets, which are networks of compromised devices, to launch DDoS attacks. In summary, to prevent these attacks, the firewall should be configured to use Snort as an IDS/IPS system. Snort can be configured with rules to detect and block these attacks by analyzing the incoming packets and blocking any packets that match the patterns in the rules. By configuring the firewall with these rules, the system can be protected against a wide range of attacks, ensuring the security and integrity of the system and the data it stores.

7 IPTables Rule Breakdown

We used a lot of iptable rules so we are going to break it down and explain it here,how it works and how it can be used.

Table: The first part of the rule specifies the table that the rule should be added to. The three tables are filter (default), nat, and mangle. The filter table is used for packet filtering and is the most commonly used table.

Chain: The chain is the part of the rule that determines when the rule will be applied. The three built-in chains are INPUT (packets destined for the local system), OUTPUT (packets originating from the local system), and FORWARD (packets that pass through the system). There are also user-defined chains that can be created.

Match: The match is a set of conditions that must be met for the rule to be applied. Examples of matches include source IP address, destination IP address, protocol, port number, etc.

Target: The target is the action that should be taken if the match is successful. Examples of targets include ACCEPT (allow the packet), DROP (discard the packet), and REJECT (discard the packet and send an error message to the sender).

Options: Options are additional parameters that can be used to further define the rule. Examples of options include interface names, IP addresses, and port numbers.

So in general, an iptables rule will consist of a combination of these components to define what packets should be allowed or blocked by the firewall.

8 Firewall Configs with the IPTables Rules

For instance these are our network

```
# iptables -t nat -F

# iptables -F
```

SSH to PC

```
# iptables -A INPUT -p tcp -s 10.0.2.15 -dport ssh -j ACCEPT

# iptables -A INPUT -p tcp -s 127.0.0.1 -dport ssh -j ACCEPT

# iptables -A INPUT -i enp0s3 -p tcp -dport 22 -m state -state NEW,ESTABLISHED
-j ACCEPT

# iptables -A OUTPUT -o enp0s3 -p tcp -sport 22 -m state -state ESTABLISHED
-j ACCEPT
```

DNS name resolution requests sent to the dns and dns2 servers

```
# iptables -A OUTPUT -p udp -d 23.214.219.129 -dport domain -j ACCEPT

# iptables -A INPUT -p udp -s 23.214.219.129 -sport domain -j ACCEPT

# iptables -A OUTPUT -p udp -d 87.248.214.97-dport domain -j ACCEPT

# iptables -A INPUT -p udp -s 87.248.214.97-sport domain -j ACCEPT
```

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

```
# iptables -A INPUT -p tcp -s 192.168.10.1/24 -dport ssh -j ACCEPT

# iptables -A INPUT -p tcp -s 23.214.219.129 -dport ssh -j ACCEPT

# iptables -A OUTPUT -p tcp !  -syn -j ACCEPT

# iptables -I FORWARD 1 -p tcp -d 23.214.219.129 -i enp0s8 -o enp0s10
-dport http -j NFQUEUE -queue-num 0 -queue-bypass
```

```
# iptables -I FORWARD 1 -p tcp -s 23.214.219.129 -i enp0s10 -o enp0s8
-sport http -j NFQUEUE -queue-num 0 -queue-bypass

# iptables -A FORWARD -j NFQUEUE -queue-num 0
```

Domain name resolutions internal network

```
# iptables -A FORWARD -p udp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport domain -j ACCEPT

# iptables -A FORWARD -p udp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport domain -j ACCEPT
```

Domain name resolutions internet

```
# iptables -A FORWARD -p udp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
domain -j ACCEPT

# iptables -A FORWARD -p udp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
domain -j ACCEPT
```

The DNS server should be able to resolve names using DNS servers on the internet

```
# iptables -A FORWARD -p udp -s 23.214.219.129 -i enp0s8 -o enp0s3 -dport
domain -j ACCEPT

# iptables -A FORWARD -p udp -d 23.214.219.129 -o enp0s8 -i enp0s3 -sport
domain -j ACCEPT
```

The DNS and DNS2 servers should be able to synchronize the contents od DNS Zones

```
# iptables -A FORWARD -p tcp -s 23.214.219.129 -d 87.248.214.97-i enp0s8
-o enp0s10 -dport domain -j ACCEPT

# iptables -A FORWARD -p tcp -s 87.248.214.97-d 23.214.219.129 -i enp0s10
-o enp0s8 -sport domain -j ACCEPT

# iptables -A FORWARD -p tcp -s 87.248.214.97-d 23.214.219.129 -i enp0s3
-o enp0s8 -dport domain -j ACCEPT
```

```
# iptables -A FORWARD -p tcp -s 23.214.219.129 -d 87.248.214.97-i enp0s8
-o enp0s3 -sport domain -j ACCEPT
```

The SMTP connections to the smtp server

```
# iptables -A FORWARD -p tcp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport smtp -j ACCEPT

# iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport smtp -j ACCEPT

# iptables -A FORWARD -p tcp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
smtp -j ACCEPT

# iptables -A FORWARD -p tcp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
smtp -j ACCEPT
```

POP and IMAP connections to the mail server IMAP

```
# iptables -A FORWARD -p tcp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport imaps -j ACCEPT

# iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport imaps -j ACCEPT

# iptables -A FORWARD -p tcp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
imaps -j ACCEPT

# iptables -A FORWARD -p tcp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
imaps -j ACCEPT
```

POP

```
# iptables -A FORWARD -p tcp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport pop3s -j ACCEPT

# iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport pop3s -j ACCEPT

# iptables -A FORWARD -p tcp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
pop3s -j ACCEPT
```

```
    # iptables -A FORWARD -p tcp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
pop3s -j ACCEPT
```

HTTP and HTTPs connection to the www server HTTP

```
    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport http -j ACCEPT

    # iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport http -j ACCEPT

    # iptables -A FORWARD -p tcp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
http -j ACCEPT

    # iptables -A FORWARD -p tcp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
http -j ACCEPT
```

HTTPs

```
    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport https -j ACCEPT

    # iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport https -j ACCEPT

    # iptables -A FORWARD -p tcp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
https -j ACCEPT

    # iptables -A FORWARD -p tcp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
https -j ACCEPT
```

OpenVPN connections to the vpn-gw server

```
    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport openvpn -j ACCEPT

    # iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport openvpn -j ACCEPT

    # iptables -A FORWARD -p tcp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
```

```
openvpn -j ACCEPT

      # iptables -A FORWARD -p tcp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
openvpn -j ACCEPT

      # iptables -A FORWARD -p udp -s 192.168.10.1/24 -d 23.214.219.129 -i
enp0s10 -o enp0s8 -dport openvpn -j ACCEPT

      # iptables -A FORWARD -p udp -s 23.214.219.129 -d 192.168.10.1/24 -i
enp0s8 -o enp0s10 -sport openvpn -j ACCEPT

      # iptables -A FORWARD -p udp -d 23.214.219.129 -i enp0s3 -o enp0s8 -dport
openvpn -j ACCEPT

      # iptables -A FORWARD -p udp -s 23.214.219.129 -i enp0s8 -o enp0s3 -sport
openvpn -j ACCEPT
```

VPN clients connected to the gateway (vpn-gw) should able to connect to the PostgreSQL services on the datastore server

```
      # iptables -A FORWARD -p tcp -s 23.214.219.129 -d 192.168.10.1 -i enp0s8
-o enp0s10 -dport postgresql -j ACCEPT

      # iptables -A FORWARD -p tcp -s 192.168.10.1 -d 23.214.219.129 -i enp0s10
-o enp0s8 -sport postgresql -j ACCEPT
```

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

```
      # iptables -A FORWARD -p tcp -d 192.168.10.1 -i enp0s3 -o enp0s10 -dport
ftp -j ACCEPT

      # iptables -A FORWARD -p tcp -s 192.168.10.1 -i enp0s10 -o enp0s3 -sport
ftp-data -j ACCEPT
```

PASSIVE

```
      # iptables -A FORWARD -m state -state RELATED,ESTABLISHED -j ACCEPT

      # modprobe nf_conntrack_ftp

      # modprobe ip_nat_ftp
```

```
      # iptables -t nat -A PREROUTING -p tcp -s 87.248.214.97-d 87.248.214.97
-i enp0s3 -dport ftp -j DNAT -to-destination 192.168.10.1
```

SSH connections to the datastore server, but only if originated at the eden server DNS2

```
      # iptables -A FORWARD -p tcp -s 87.248.214.97-d 192.168.10.1 -i enp0s3
-o enp0s10 -dport ssh -j ACCEPT

      # iptables -A FORWARD -p tcp -s 192.168.10.1 -d 87.248.214.97-i enp0s10
-o enp0s3 !  -syn -j ACCEPT

      # iptables -t nat -A PREROUTING -p tcp -s 87.248.214.97-d 87.248.214.97
-i enp0s3 -dport ssh -j DNAT -to-destination 192.168.10.1
```

EDEN

```
      # iptables -A FORWARD -p tcp -s 87.248.214.91 -d 192.168.10.1 -i enp0s3
-o enp0s10 -dport ssh -j ACCEPT

      # iptables -A FORWARD -p tcp -s 192.168.10.1 -d 87.248.214.91 -i enp0s10
-o enp0s3 !  -syn -j ACCEPT

      # iptables -t nat -A PREROUTING -p tcp -s 87.248.214.91 -d 87.248.214.97
-i enp0s3 -dport ssh -j DNAT -to-destination 192.168.10.1
```

Domain name resolutions using DNS

```
      # iptables -A FORWARD -p udp -s 192.168.10.1/24 -i enp0s10 -o enp0s3
-dport domain -j ACCEPT

      # iptables -A FORWARD -p udp -d 192.168.10.1/24 -i enp0s3 -o enp0s10
-sport domain -j ACCEPT

      # iptables -A FORWARD -p udp -d 192.168.10.1/24 -i enp0s3 -o enp0s10
-sport domain -j ACCEPT

      # iptables -t nat -A POSTROUTING -p udp -s 192.168.10.1/24 -o enp0s3
-dport domain -j SNAT -to-source 87.248.214.97
```

HTTP,HTTPS and SSH connections HTTP

```
    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -i enp0s10 -o enp0s3
-dport http -j ACCEPT

    # iptables -A FORWARD -p tcp -d 192.168.10.1/24 -i enp0s3 -o enp0s10
-sport http -j ACCEPT

    # iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.1/24 -o enp0s3
-dport http -j SNAT -to-source 87.248.214.97
```

HTTPs

```
    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -i enp0s10 -o enp0s3
-dport https -j ACCEPT

    # iptables -A FORWARD -p tcp -d 192.168.10.1/24 -i enp0s3 -o enp0s10
-sport https -j ACCEPT

    # iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.1/24 -o enp0s3
-dport https -j SNAT -to-source 87.248.214.97
```

SSH

```
    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -i enp0s10 -o enp0s3
-dport ssh -j ACCEPT

    # iptables -A FORWARD -p tcp -d 192.168.10.1/24 -i enp0s3 -o enp0s10
-sport ssh -j ACCEPT

    # iptables -t nat -A PREROUTING -p tcp -s 192.168.10.1/24 -d 87.248.214.97
-i enp0s10 -dport ssh -j DNAT -to-destination 87.248.214.90

    # iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.1/24 -o enp0s3
-dport ssh -j SNAT -to-source 87.248.214.97
```

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

```
    # iptables -A FORWARD -p tcp -d 192.168.10.1/24 -i enp0s10 -o enp0s3
-dport ftp -j ACCEPT

    # iptables -A FORWARD -p tcp -s 192.168.10.1/24 -i enp0s3 -o enp0s10
```

```
-sport ftp-data -j ACCEPT
```

PASSIVE

```
# iptables -A FORWARD -m state -state RELATED,ESTABLISHED -j ACCEPT

# modprobe nf_conntrack_ftp

# modprobe ip_nat_ftp

# iptables -t nat -A POSTROUTING -p tcp -s 192.168.10.1/24 -o enp0s3
-dport ftp -j SNAT -to-source 87.248.214.97

# iptables -A FORWARD -j NFQUEUE -queue-num 0

# iptables -P INPUT DROP

# iptables -P OUTPUT DROP

# iptables -P FORWARD DROP

# modprobe nfnetlink_queue
```

The connections were tested using the 'nc' command, and the logs showed successful communication between the internal network, DMZ, and external network for each type of connection (SMTP, POP, IMAP,...) from both internal and external sources.

9 Snort Rules and Usage For Detection and Prevention of Attacks

The second part of this work focuses on preventing malicious attacks on our network, for which we will be using Snort.

Snort was configured according to the provided tutorial, and the following commands were executed to use it in conjunction with PTables:

```
# modprobe nfnetlink queue


# iptables -A FORWARD -j NFQUEUE -queue-num 0
```

To activate Snort in inline mode, the following command was used:

```
# snort -Q -daq nfg -dag-var queue=0 -c /etc/snort/snort.conf -v -l /var/log/snort/
```

This command includes the location of the Snort configuration file as well as the location of the alerts. It also contains the flag -v for sniffer mode and the flag -l to function as a packet logger.

9.1 SQL Injection

SQL Injection is a technique of injecting code via SQL statements through a web page, which can destroy a database and is one of the most common hacking techniques.

This technique usually occurs when an input is requested and an SQL statement is included in that input, which runs on the database without being known.

An example would be SQL injection based on "1=1":

SELECT * FROM Users WHERE UserId = 185 OR 1=1;

This injection is possible when a web page asks, for example, for an input of an ID, and the user enters the ID more creatively.

Userld: |105 OR 1=1

Since the statement will always be true, the entire "Users"database would be revealed if there was nothing to counter it.

Another example would be error-based SQL injection.

This technique involves inserting characters into the input that will generate error messages, such as quote (') or double quote (").

```
drop tcp any any -> any any (msg:"SQL Injection SELECT statement";  flow: to_server, established;
pcre:"/select.from.union.(--|/*|#)/i"; sid:9398; rev:28;)
```

```
drop tcp any any -> any $HTTP_PORTS (msg:"SQL Injection -
Paranoid";flow:to_server,established;pcre:"/(%27)|(')|(--)|(%23)|(#)/i"; classtype:Web-application-
attack; sid:909900;rev:5;)
```

The first SQL injection rule detects the attack that consists of inserting the OR statement along with the equality "1=1", and the second rule detects the use of characters such as the quote ('), detecting both the character itself and its hex equivalent.

9.2 Cross-Site Scripting (XSS) Attacks

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, usually in the form of a browser-side script, to a different end user.

These can be done through HTML opening and closing tags with text or even "img src"tags.

The rules include the appropriate Regex to detect any type of attack that includes the mentioned tags, respectively.

```
drop tcp any any -> any any (msg:"Cross-site Scripting – <body >"; flow:to_server,established;
pcre:"/((%3C)|<)((%62)|b|(%42))((%6F)|o|(%4F))((%64)|d|(%44))((%79)|y|(%59))[^\n]+((%3E)|>)/";
sid:9400; rev:5;)
```

```
drop tcp any any -> any any (msg:"Cross-site Scripting – <img >"; flow:to_server,established;
pcre:"/((%3C)|<)((%69)|i|(%49))((%6D)|m|(%4D))((%67)|g|(%47))[^\n]+((%3E)|>)/"; sid:9401; rev:5;)
```

9.3 DOS Attacks

Denial of Service (DoS) attacks are a common type of cyber attack that aims to disrupt the availability of a network or system by overwhelming it with traffic or requests. Snort, an open-source intrusion detection and prevention system, can be used to detect and prevent DoS attacks by analyzing network traffic and alerting administrators of suspicious activity.

Those are the rules to detect:

```
drop icmp any any -> any any (msg:"ICMP Flood Detected"; threshold: type both, track by_src, count
500, seconds 10; sid:1000001;)
```

```
drop tcp any any -> any any (flags:S; msg:"SYN Flood Detected"; threshold: type both, track by_src,
count 500, seconds 10; sid:1000002;)
```

```
drop udp any any -> any any (msg:"UDP Flood Detected"; threshold: type both, track by_src, count
500, seconds 10; sid:1000003;)
```

Those rules are used for a different type of DOS Attacks: ICMP Flood DoS Attack, SYN Flood DoS Attack, HTTP Flood DoS Attack respectively.

9.4 OS Fingerprinting

OS fingerprinting is an attempt to determine the operating system and software running on a target system. Attackers use this information to identify vulnerabilities in the system, which they can then exploit to gain unauthorized access or disrupt services.

We used Active OS fingerprinting which is a technique used to determine the operating system of a target system by actively sending probes or requests to the target system and analyzing the responses. This technique involves sending packets to the target system and examining the responses to identify unique characteristics of the operating system.

Active OS fingerprinting is performed by tools such as Nmap, which sends a series of probes to the target system to gather information about its operating system. These probes are designed to elicit specific responses from the target system, which can then be analyzed to determine the operating system.

Nmap OS Fingerprinting

The rule is:

```
drop udp any any -> any 137 (msg:"NETBIOS Name Service OS Fingerprinting"; content:"|00 00 00 00
00 00 00 00 00 00 00 00 20 43 4B 41|"; depth:16; reference:arachnids,238; classtype:misc-activity;
sid:12345; rev:1;)
```

This rule detects Nmap OS fingerprinting attempts by monitoring network traffic and alerting administrators when a specific traffic pattern is detected.

We used Nmap to simulate OS Fingerprinting, which is a powerful and widely used tool for network exploration and security auditing. One of its most popular features is the ability to perform OS fingerprinting.

When performing OS fingerprinting with Nmap, the tool sends a series of packets to the target system and analyzes the responses to determine the operating system and software running on the system. Nmap uses a combination of active and passive fingerprinting techniques to gather information

about the target system.

To perform OS fingerprinting with Nmap, the following command can be used:

```
# nmap -O <target IP>
```

In this command, -O specifies that Nmap should perform OS detection on the target system, and <target IP> is the IP address or range of addresses that should be scanned.

When the command is executed, Nmap sends a series of packets to the target system and analyzes the responses to determine the OS and software running on the system. The results of the scan are then displayed in the terminal.

10 PGP Encrypted Delivery

This section it's demonstrated how we encrypted the files and the report using PGP from GnuPG.

10.1 Step 1: Generate a GPG key:

gpg –gen-key

10.2 Step 2:

Follow the prompts to set up the key. We made sure to use a strong passphrase and saved the key pair to a secure location.

10.3 Step 3: Export the public key:

gpg –export –armor <your-email-address> > my-public-key.asc

10.4 Step 4:

Send the public key to the recipient(s) who will be receiving the encrypted and signed files.

10.5 Step 5: Sign the file:

gpg –sign <filename>

10.6 Step 6: Import the recipient key:

gpg –import <recipient_public_key> key.asc

10.7 Step 7: Encrypt the file:

gpg –recipient <recipient_email_address> –encrypt <filename>

10.8 Step 8: Send the file

Attach the encrypted and/or signed file to your email and send it to the recipient(s).

11 Conclusion

In conclusion, the configuration of a network firewall using IPTables/Netfilter and Snort as an IDS/IPS system is essential to protect a network from security attacks. The assignment aimed to configure a firewall capable of detecting and reacting to security attacks against services deployed on a protected network. To achieve this, packet filtering, NAT, and intrusion detection mechanisms were implemented. The network architecture included a DMZ and internal network, and the router interconnecting the various networks ran Linux and supported all the security functionalities described in the assignment.

The firewall configuration included authorizing direct communications between networks for the normal operation of specific services, including DNS name resolution requests, SMTP, POP, and IMAP connections, HTTP and HTTPS connections, and OpenVPN connections. In addition, the firewall configuration for connections to the external IP address of the firewall authorized FTP connections in passive and active modes and SSH connections to the datastore server only if originated at specific servers. The firewall configuration for communications from the internal network to the outside authorized domain name resolutions, HTTP, HTTPS, and SSH connections, and FTP connections in passive and active modes to external FTP servers using NAT.

The second part of the assignment involved enabling the capability to detect and react to security attacks using Snort and IPtables. The firewall was able to detect and block two types of SQL injection, two types of DoS attacks, and OS fingerprinting attempts. It is essential to describe how the attacks work and how Snort can detect and block them to ensure maximum protection.

Overall, the successful completion of this assignment demonstrates the importance of configuring a network firewall using IPTables/Netfilter and Snort as an IDS/IPS system to protect a network from security attacks. It highlights the critical role that firewalls play in network security and the need for continuous monitoring and updates to ensure the highest level of protection.