HW08

Riccardo Zucchelli 1984963

October 2024

1 Introduction

This homework is based on **iptables**, its use, configuration and its possible scenarios. iptables is a utility program that implements netfilter module, which is the underlying firewall framework, to allow users to configure a Linux firewall which can be both a packet and a session filter. This is possible by organizing its filters into tables, each containing chain of rules, with each table associating different kind of packet processing. The predefined chains for the table *filter*, which is our domain of interest, are:

- **PREROUTING**: Packets will enter this chain before a routing decision is made.
- **INPUT**: Packet is going to be locally delivered. It does not have anything to do with processes having an opened socket.
- **FORWARD**: All packets that have been routed and were not for local delivery will traverse this chain.
- **OUTPUT**: Packets sent from the machine itself will be visiting this chain.
- **POSTROUTING**: Routing decision has been made. Packets enter this chain just before handing them off to the hardware.

A packet will traverse the specific chain(s) of interest, until one of these things happen:

- A rule matches the packet, so iptables will apply the rule to it (which can be ACCEPT, DROP, or a call to another chain).
- A user-defined chain has been called, and the packet traversed it all without having any hit, so it calls the RETURN rule and gets back to the previous chain.
- End of chain is reached: at that point the default policy will be used.

2 Testing option chosen

In order to ensure the safety and reliability of our network configuration tests, we will conduct all experiments in a virtualized Linux Container (LXC) environment rather than directly on a production system or using Docker. This approach is facilitated by the Proxmox Hypervisor, which runs on a small and compact desktop PC. Using LXC offers significant advantages in terms of flexibility, efficiency, and manageability, making it an ideal choice for our needs.

Unlike Docker, which is designed primarily for application isolation, LXC provides a more comprehensive virtualization approach that closely mimics the behavior of a full Linux system. This allows us to test complex configurations, such as iptables rules, in an environment that mirrors a real-world system without the overhead of managing a full virtual machine, all while using the lightweight nature of them to restart or reloaded quickly. This efficiency not only saves time but also enhances productivity, especially when dealing with repetitive tasks or troubleshooting. Additionally, since iptables rules are flushed upon system reboot, testing them in an isolated and responsive container ensures that even unexpected outcomes will not affect the host system, making this approach a more controlled environment for experimenting with different configurations, validating their behavior, and ensuring their reliability before applying them in production.

Another advantage of using LXC on Proxmox is the platform's robust web-based management interface. This interface allows for easy control and monitoring of containers, even from computationally limited devices such as older laptops or tablets, and so eliminating the need for resource-intensive tools, ensuring that our setup remains accessible and manageable from virtually any device.

3 Scenarios chosen

For the purpose of better explaining the capabilities of what can be done with a proper configurations we'll be displaying three possible scenarios:

- The customer service department is being created: we are installing iptables on every single client. The person who works in this department will need to have access to the web, use a Remote Desktop Protocol program to access client's pc and have Skype. A user-defined chain will be defined such that customer service pcs are correctly secured against foreign traffic.

- A company wants to expose its mail server to the public but it needs to protect it from security threats. The boss wants to access the server with a very old email client application (which he incessantly praises for its old looks) so the firewall needs to allow email traffic on ports 25, 110 and 143 but only from his trusted ip address. Other users should be allowed to only use IMAPS and SMTPS.

- We want to be able to create an internet cafe, where each computer has its own firewall. Guests should be able to browse the internet freely, but should be prevented from accessing the internal network and other PC. DNS traffic should be allowed, and other protocols should be blocked (apart from Instant Messaging applications).

Default rules

Before starting any of the rules creation it's important to create some of the default rules needed for basic access to the network and internet. We'll also define a new high-security network interface, wg0, that can be used to remote-in a pc and fix it, in case of any sort of problem. We will assume that traffic is passed to the interface after being properly authenticated.

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP

iptables -A INPUT -i lo0 -j ACCEPT
iptables -A OUTPUT -o lo0 -j ACCEPT

iptables -A INPUT -i wg0 -p tcp --sport 22 -m state --state
    NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o wg0 -m state --state ESTABLISHED,RELATED -j
    ACCEPT

iptables -A INPUT -i eth0 -p icmp -j ACCEPT
iptables -A OUTPUT -o eth0 -p icmp -j ACCEPT

iptables -A OUTPUT -p udp --dport 53 --sport 1024:65535 -j ACCEPT
iptables -A INPUT -p udp --sport 53 --dport 1024:65535 -j ACCEPT
```

This is a very basic setup that grants very limited access to the internet by allowing ICMP traffic through it, and have the possibility to request data from a DNS server.

Scenario 1

We'll create chain called **CS** for this, that allows only the bare minimum traffic to flow.

```
iptables -N CS

# Blocks access to internal network (PUT ON TOP)
iptables -I CS -o eth0 -d 10.0.0.0/8 -j DROP


# allow to start connection over port 80 and 443, and to receive
    already established traffic
```

```
iptables -A CS -o eth0 -p tcp -m multiport --dports 80,443 -m state
    --state NEW,ESTABLISHED -j ACCEPT
iptables -A CS -i eth0 -p tcp --sport 1024:65535 -m state --state
    ESTABLISHED,RELATED -j ACCEPT

# Allows to connect to a RDP client, but doesn't allow a new
    connection back
iptables -A CS -o eth0 -p tcp --dport 3389 -j ACCEPT

# Allow Skype communication to pass through
iptables -A CS -o eth0 -p udp -m multiport --dports 3478,3480 -j
    ACCEPT
iptables -A CS -i eth0 -p udp -m multiport --sports 3478,3480 -j
    ACCEPT

# Append the user-defined chain on the INPUT and OUTPUT chain
iptables -A INPUT -j CS
iptables -A OUTPUT -j CS
```

Scenario 2

We'll create another chain called **MS** for this

```
iptables -n MS

iptables -I MS -o eth0 -d 10.0.0.0/8 -j DROP
# Allow the boss's ip address to access mail server over insecure
    ports. Generally block it since nobody should ever use these
    services over insecure channels
iptables -A MS -i eth0 -p tcp -m multiports --sports 25,110,143 -s
    10.0.0.1 -j ACCEPT

# Open the ports for allowing IMAP and SMTP over secure SSL
    connections, which for security reasons is the only way any
    client should connect.
iptables -A MS -i eth0 -p tcp -m multiports --sports 993,995,465 -j
    ACCEPT

# Allow the server to send packets to already opened TCP connections
    (if the server would open a connection with the outside world
    would be a massive red flag!)
iptables -A MS -o eth0 -p tcp --dport 1024:65535 -m state --state
    ESTABLISHED -j ACCEPT

iptables -A INPUT -j MS
iptables -A OUTPUT -j MS
```

Scenario 3

We'll create the last chain called **IC** for this

```
iptables -N IC
# Block access to the internal network
iptables -I IC -o eth0 -d 10.0.0.0/8 -j DROP

# allow to start connection over port 80 and 443, and to receive
    already established traffic
iptables -A CS -o eth0 -p tcp -m multiport --dports 80,443 -m state
    --state NEW,ESTABLISHED -j ACCEPT
iptables -A CS -i eth0 -p tcp --sport 1024:65535 -m state --state
    ESTABLISHED,RELATED -j ACCEPT

# Allow Discord UDP packets (needed for voice calls) to pass
iptables -A CS -i eth0 -p udp --sport 50000:65535 -j ACCEPT
iptables -A CS -o eth0 -p udp --dport 50000:65535 -j ACCEPT

# Allows Skype to contact other employees and the client itself
iptables -A CS -o eth0 -p udp -m multiport --dports 3478,3480 -j
    ACCEPT
iptables -A CS -i eth0 -p udp -m multiport --sports 3478,3480 -j
    ACCEPT

iptables -A INPUT -j CS
iptables -A OUTPUT -j CS
```

Generally speaking, iptables can be perfectly adapted to many scenarios with relative ease. It's very important to be as thorough as possible with permitting just the bare minimum. Firewalls, as any other toolset of software, aren't impenetrable and can be victims of discovered exploits, but most of the attacks are able to work around one because of a flaw in its rules; once a firewall gets deployed to a workplace, it can get more and more complicated, especially if new rules are added, or they can outright create conflicts difficult to debug.