



NETWORK SECURITY REPORT

B00151368 James Foley
& B00142777 Alexander
Aregbesola



Table of Contents

1. INTRODUCTION TO FIREWALLS	2
1.1 INTRODUCTION TO NFTables	2
1.2 REPLACEMENT OF TABLES	3
1.3 INTRODUCTION TO PFSense	3
2. NMAP.....	4
2.1 WHAT IS NMAP?.....	4
2.2 PURPOSE: NETWORK RECONNAISSANCE & OPEN PORT SCANNING	4
2.3 HOW ATTACKERS USE NMAP TO IDENTIFY WEAKNESSES	4
3. SSH	5
3.1 WHAT IS SSH?	5
3.2 WHY SSH IS A COMMON ATTACK VECTOR	5
4. NETWORK SETUP FOR FIREWALL TESTING	6
4.1 NFTables NETWORK	6
4.2 PFSense NETWORK	6
4.2.1 NETWORK CONFIGURATION	7
5. FIREWALL TESTING	8
5.1 NFTables	8
5.1.1 TEST 1: NMAP SCAN BEFORE AND AFTER RULES	8
5.1.2 FIREWALL RULE IMPLEMENTATION	9
5.1.3 TEST 2: SSH ACCESS BEFORE AND AFTER RULES.....	11
5.1.4 IMPLEMENTATION OF SSH BLOCKING RULE	12
5.2 PFSense	14
5.2.1 TEST 1: NMAP SCAN BEFORE AND AFTER RULES	14
5.2.2 FIREWALL RULE IMPLEMENTATION IN PFSense	14
5.2.3 TEST 2: SSH ACCESS BEFORE AND AFTER RULES.....	15
6. COMPARISON OF NFTables AND PFSense	16
6.1 SUMMARY.....	16
7. FINAL CONCLUSION	17
8. BIBLIOGRAPHY	18

1. Introduction to Firewalls

A firewall is a crucial network security tool that monitors and controls incoming and outgoing network traffic based on predefined security rules. It is a barrier between a trusted internal network and untrusted external networks, such as the Internet, to prevent unauthorised access and cyber threats (Cisco, 2024).

There are several types of firewalls, including packet filtering firewalls, stateful inspection firewalls, proxy firewalls, and next-generation firewalls (NGFWs). Modern firewalls often include intrusion prevention, deep packet inspection, and AI-powered threat detection to enhance security (Cisco, 2024).

We used **nfTables** and **pfSense** to test and compare the firewall performance of this project. These firewalls were chosen to evaluate how different firewall solutions handle network security. We analysed their ability to filter traffic and protect against threats by implementing rules and running tests.

1.1 Introduction to nfTables

nfTables configuration was done by editing **nftables.conf**, which allows for loading only the necessary content. The IP filter table, **ingress**, was designed to be the first reader of incoming packets based on predefined rules. The Table IP filter handles all incoming packets with an ingress hook, the first activated process when a packet enters the system. This setup ensures early traffic filtering, allowing only packets with valid source and destination IP addresses and ports to proceed while dropping all others. (Hypponen, 2021)

The IP filter table is structured to include a base chain with an ingress hook for each network interface. Since ingress filtering operates per interface, a separate chain is required for each interface. In this specific setup, the server has only one base chain, called **enp0s3**, since it is the only interface present in the system. (Hypponen, 2021)

The **inet table** consists of three major chains: input, output, forward, and a non-base chain, **inet ban**. The input chain processes incoming packets, but only after they have passed through the ingress hook of the IP table. The output chain processes outbound packets, while the forward chain handles packets that must be forwarded to another host. The inet ban chain is designed to drop traffic exceeding a predefined rate limit, ensuring that excessive or malicious traffic is controlled effectively. (Hypponen, 2021)

The net IP filter tables primarily filter packets at an early stage, whereas the inet table processes packets that have already passed through ingress filtering. Since the inet table works later, it can handle more complex filtering operations like connection tracking. The base chains of the inet table are configured with a priority value of zero, enforcing a default deny policy, meaning that only explicitly

permitted packets are allowed to proceed, thereby securing the system effectively. (Hypponen, 2021)

1.2 Replacement of tables

nfTables is widely recognised as the successor to **ipTables**, offering a more efficient and streamlined approach to firewall management. Unlike ipTables, which required multiple independent rule sets, nfTables consolidates filtering functions (including IP, IPv6, and ARP filtering) into a single command-line tool, significantly simplifying rule management. This increases flexibility to set up firewall rules while maintaining efficiency and security. (Olivia et al., 2025)

nfTables now supports Network Address Translation (NAT), a significant enhancement. NAT enables mapping many private IP addresses to a single public IP address, thereby keeping address space. NAT automatically blocks outbound connections, improving security and limiting unnecessary access. (Olivia et al., 2025)

The move from ipTables to nfTables is inspired by improved speed, simpler syntax, and better interoperability with contemporary networking tools. By addressing scalability concerns and rule optimisation, nfTables has become the preferred choice for many network administrators, particularly in enterprise environments where security and performance are critical. (Olivia et al., 2025)

1.3 Introduction to pfSense

pfSense is a free and open-source firewall and router operating system. It is designed to provide enterprise-level networking capabilities while remaining accessible for home users and IT professionals. Unlike many off-the-shelf routers, pfSense can be installed on standard computer hardware, making it a flexible and cost-effective alternative to proprietary solutions (TekLager, 2024).

pfSense is primarily used as a firewall and router but also supports additional features such as DHCP services, DNS resolution, VPN hosting, and intrusion detection. Users can install extra security tools like Snort for intrusion detection and Squid for web filtering through its built-in Package Manager. This makes pfSense a preferred choice for home networks, businesses, and large corporate environments (TekLager, 2024).

One key reason for using pfSense over commercial routers is its security and flexibility. Many off-the-shelf routers receive limited security updates, leaving networks vulnerable to cyber threats. In contrast, pfSense is regularly updated, ensuring that security vulnerabilities are patched swiftly. Its web-based interface allows users to configure settings without needing to edit configuration files manually, making it accessible to those with limited networking experience (TekLager, 2024).

Since 2004, pfSense has built an effective community, including online documentation and tutorials to assist users in troubleshooting problems. This open-source firewall ensures that pfSense develops into a reliable and cost-effective networking solution (TekLager, 2024).

2. Nmap

2.1 What is Nmap?

Nmap (Network Mapper) is an open-source network discovery and security auditing tool. It helps administrators scan networks to identify active hosts, open ports, and running services. While network administrators use Nmap for legitimate security assessments, it is also commonly used by attackers to find vulnerabilities in a system (Shivanandhan, 2020).

2.2 Purpose: Network Reconnaissance & Open Port Scanning

Nmap scans IP addresses and ports to determine which devices are online and which services they are running. This helps administrators understand their network security posture. Key functions of Nmap scans include (Shivanandhan, 2020):

- **Finding active hosts:** Identifies computers, servers, and devices within a network.
- **Open port scanning:** Detects which ports are open and what services are listening (e.g., SSH on port 22, HTTP on port 80).
- **Service detection:** Determining which applications are running, including their versions, helps find outdated or vulnerable software. This is critical for both defenders and attackers:
 - **Administrators** use Nmap to identify security gaps and improve firewall rules.
 - **Hackers** use it for reconnaissance, gathering data before launching attacks.

2.3 How Attackers Use Nmap to Identify Weaknesses

Attackers frequently rely on Nmap to detect security vulnerabilities in a network. Some common ways they use Nmap scans include (Shivanandhan, 2020):

- **Detecting open ports:** An attacker may attempt brute-force login attacks if a system has SSH (port 22) open.
- **Fingerprinting running services:** Attackers can identify outdated or unpatched applications vulnerable to exploits by analysing software versions.

- **Firewall evasion techniques:** If firewalls are misconfigured, attackers can use stealth scanning techniques to avoid detection and gain access.

3. SSH

3.1 What is SSH?

Secure Shell (SSH) is a cryptographic network protocol that enables secure remote access and management of systems over an unsecured network (*Gillis et al., 2024*)

System administrators commonly use it to configure and control remote machines securely, replacing older, less secure protocols like Telnet. SSH encrypts data transmissions, preventing unauthorised interception. SSH operates on port 22 by default, making it a well-known target for cyberattacks if not correctly secured (*Gillis et al., 2024*)

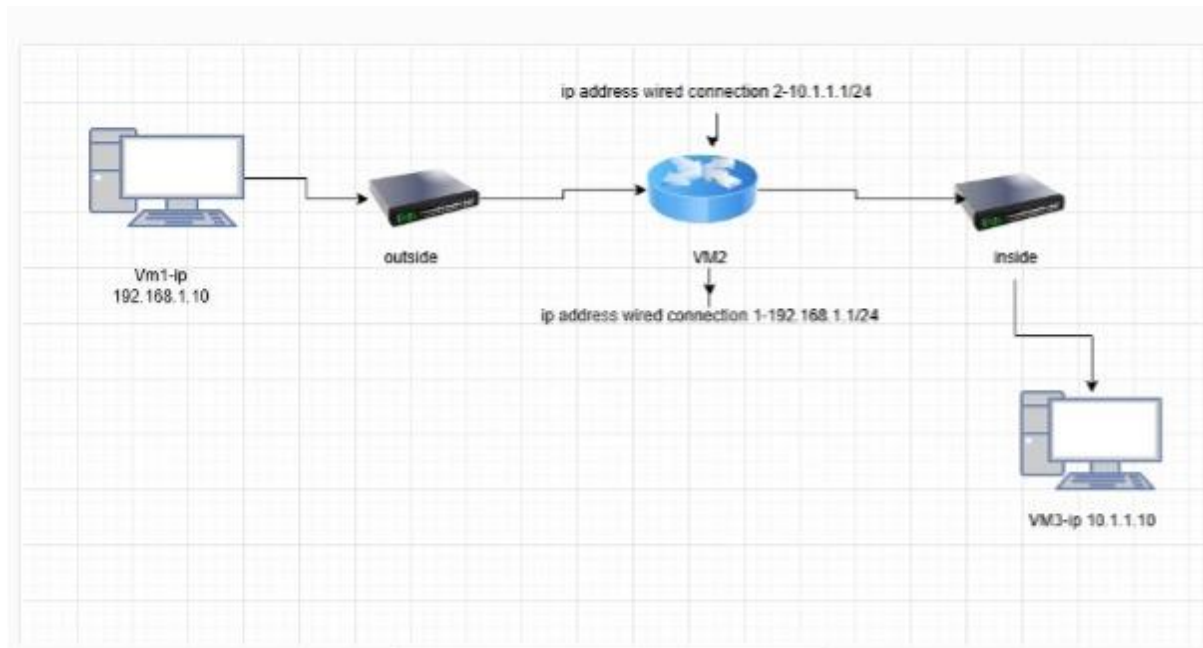
3.2 Why SSH is a Common Attack Vector

Because SSH provides direct access to critical systems, attackers frequently target it. One of the most common attack methods is brute-force credential guessing, where automated tools repeatedly attempt to log in using common usernames and passwords (*Gillis et al., 2024*)

If SSH is poorly configured—such as using default passwords or allowing unrestricted login attempts—attackers can gain unauthorised access. Additionally, leaving port 22 open to the internet without extra security measures significantly increases the risk of brute-force attacks and exploits (*Gillis et al., 2024*)

4. Network Setup for Firewall Testing

4.1 nfTables Network



This visual presentation shows three virtual machines connected to the same network. The switches in the image above represent the network adapters for VM1 and VM3.

- VM1, VM2, and VM3 are in the internal network.
- These switches allow each virtual machine to communicate.
- VM2 acts as the man-in-the-middle, forwarding traffic through the network adapters labelled “outside” and “inside” in the image above.

4.2 PFSense Network

To conduct this firewall testing project, a virtualised system was created using VirtualBox, which hosts three key components:

1. pfSense Firewall – The network security application that acts as the firewall.
2. Xubuntu (Target Machine) – A Linux-based machine representing the internal network.
3. Kali Linux (Attacker Machine) – The penetration testing system used to perform network scanning and SSH login attempts.

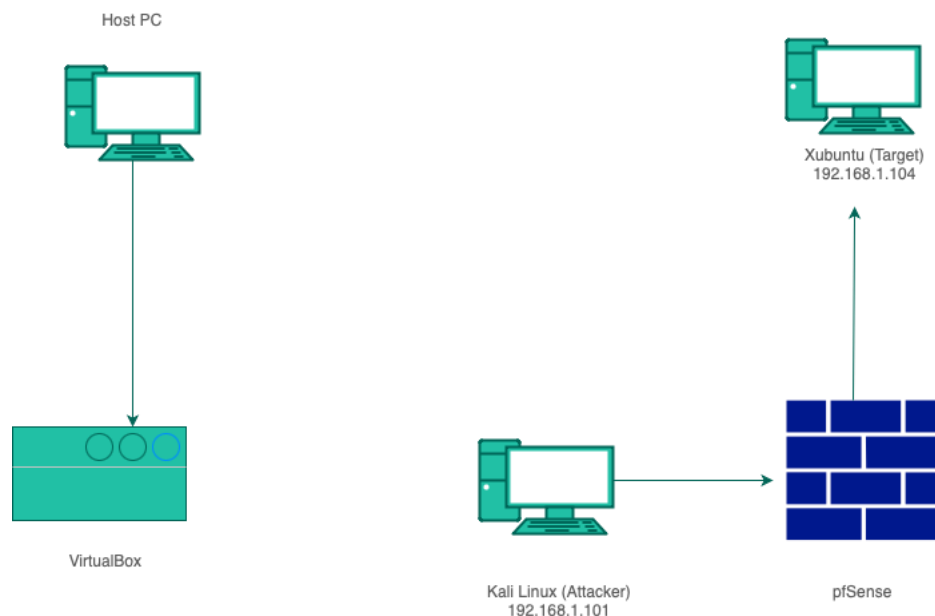
This setup allows for controlled testing of firewall rules, ensuring that security measures such as blocking Nmap scans and SSH access can be adequately implemented and evaluated.

4.2.1 Network Configuration

The network is made up of an internal network maintained by pfSense. Each computer is provided with an IP address:

- pfSense Firewall acts as a gateway, testing traffic between devices.
- Kali Linux (Attacker): 192.168.1.101 - Used to attack the Xubuntu VM.
- The target IP address for Xubuntu is 192.168.1.104, which is protected by pfSense.
- Host PC: Runs VirtualBox and manages the virtualised environment.

The network is visualised in the diagram below:



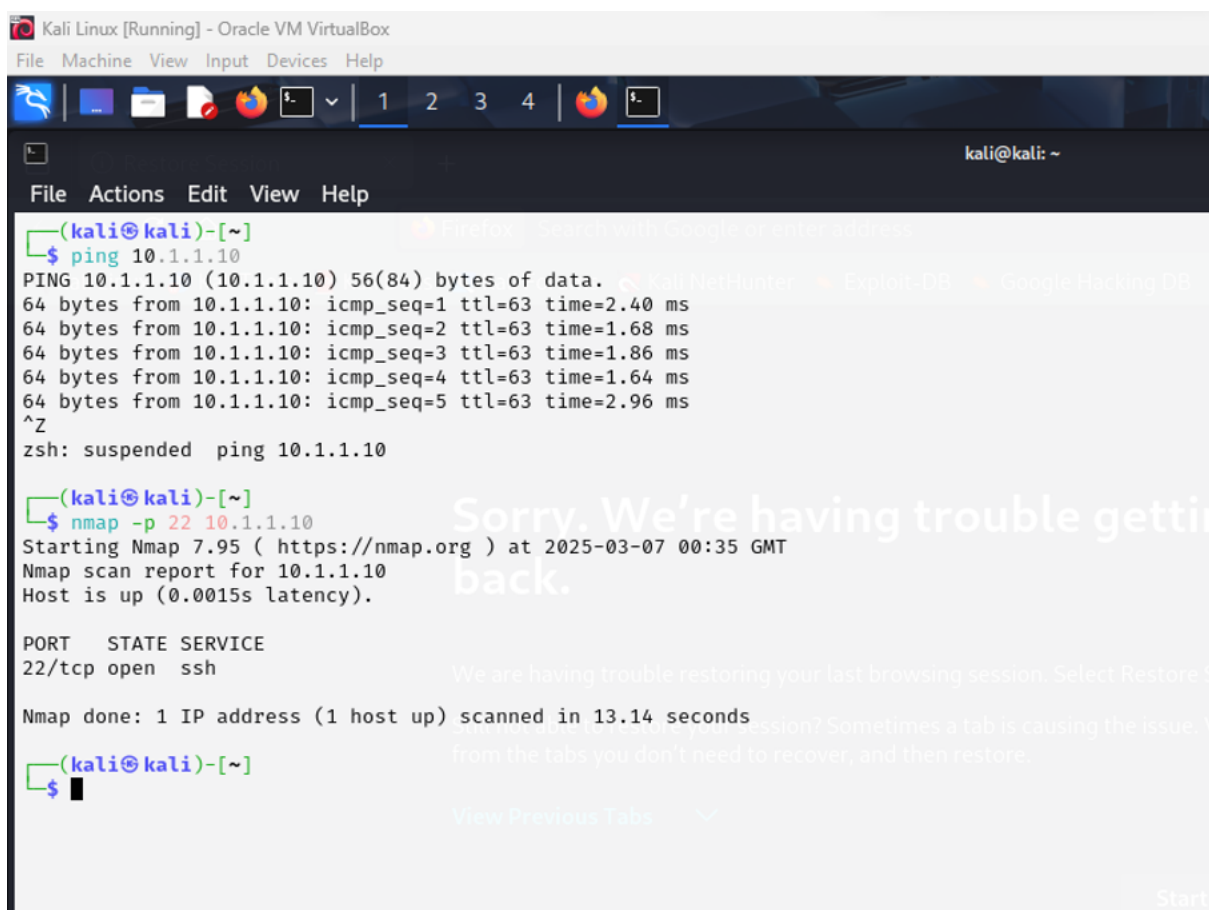
5. Firewall Testing

5.1 NfTables

5.1.1 Test 1: Nmap Scan Before and After Rules

Initial Scan Results (Open Ports)

My first test was an Nmap scan to see if VM1 was able to ping VM3 before adding any rules.



The screenshot shows a Kali Linux terminal window titled "Kali Linux [Running] - Oracle VM VirtualBox". The terminal displays the following commands and output:

```
(kali㉿kali)-[~]
$ ping 10.1.1.10
PING 10.1.1.10 (10.1.1.10) 56(84) bytes of data:
64 bytes from 10.1.1.10: icmp_seq=1 ttl=63 time=2.40 ms
64 bytes from 10.1.1.10: icmp_seq=2 ttl=63 time=1.68 ms
64 bytes from 10.1.1.10: icmp_seq=3 ttl=63 time=1.86 ms
64 bytes from 10.1.1.10: icmp_seq=4 ttl=63 time=1.64 ms
64 bytes from 10.1.1.10: icmp_seq=5 ttl=63 time=2.96 ms
^Z
zsh: suspended ping 10.1.1.10

(kali㉿kali)-[~]
$ nmap -p 22 10.1.1.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-07 00:35 GMT
Nmap scan report for 10.1.1.10
Host is up (0.0015s latency).

PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 1 IP address (1 host up) scanned in 13.14 seconds

(kali㉿kali)-[~]
$
```

5.1.2 Firewall Rule Implementation

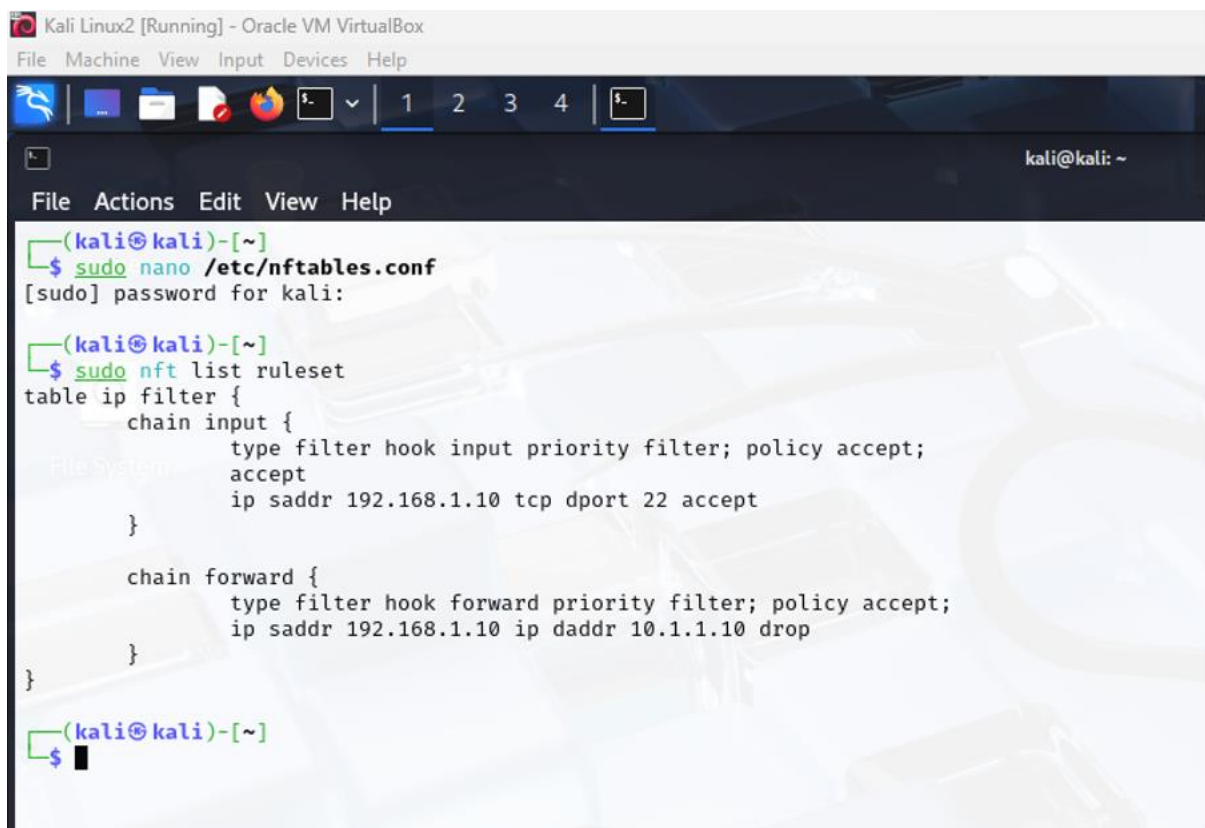
After that, I decided to implement rules on VM2, as VM2 is my rule-setting machine.

- The filter table is the pinnacle where all the rules are held, as displayed in the image.
- The input policy is set to accept, meaning it can intake any rule applied.
- The forward policy is also on accept, which allows the rule to be forwarded to the target.

The implemented rules:

1. The rule states that IP 192.168.1.10 can access and see that port 22 is open, whereas IP 10.1.1.10 cannot see it.
2. The second rule blocks all traffic from 192.168.1.10 to 10.1.1.10 (VM1 to VM3), preventing VM1 from pinging VM3.

The rule configuration is displayed below.



```
Kali Linux2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

(kali@kali)-[~]
$ sudo nano /etc/nftables.conf
[sudo] password for kali:

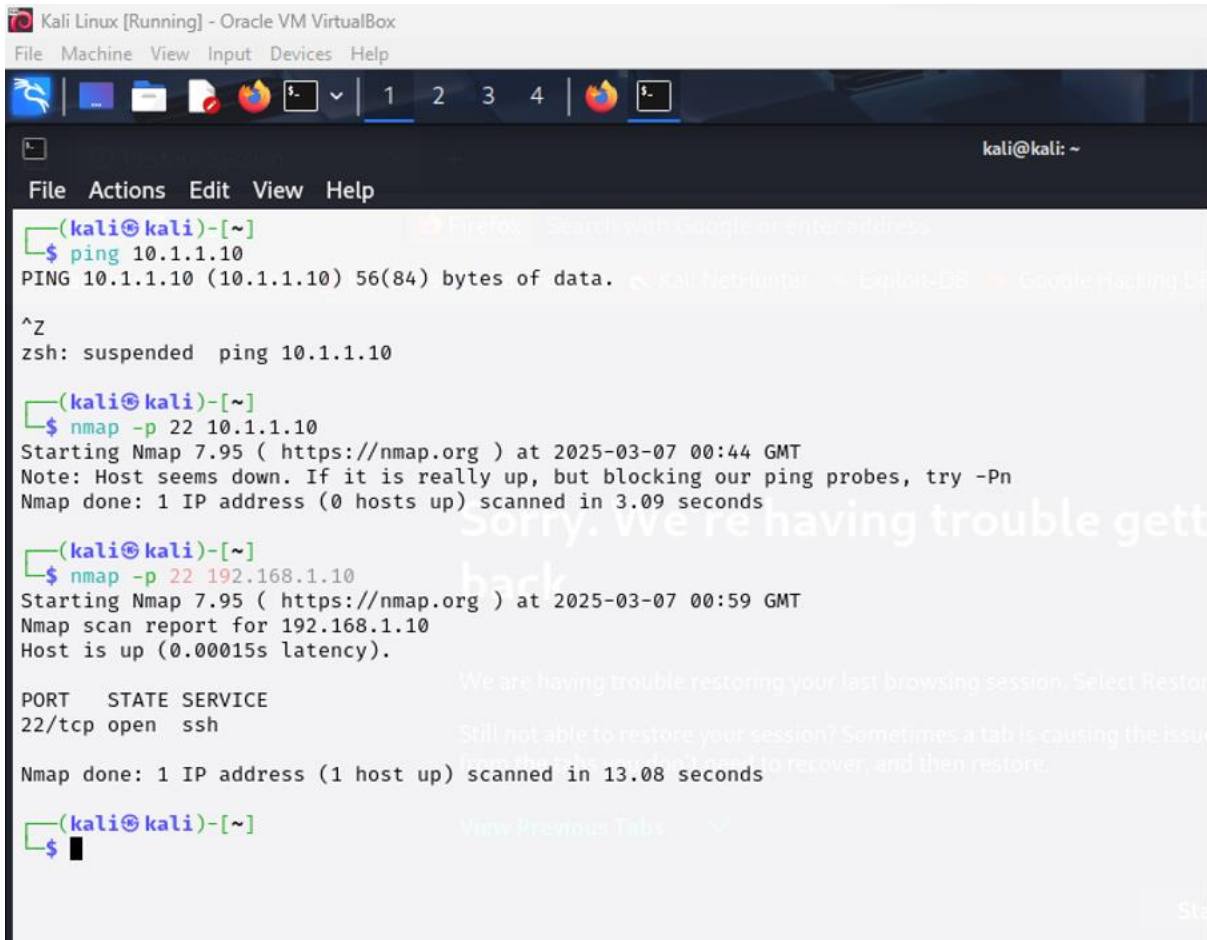
(kali@kali)-[~]
$ sudo nft list ruleset
table ip filter {
    chain input {
        type filter hook input priority filter; policy accept;
        accept
        ip saddr 192.168.1.10 tcp dport 22 accept
    }

    chain forward {
        type filter hook forward priority filter; policy accept;
        ip saddr 192.168.1.10 ip daddr 10.1.1.10 drop
    }
}
```

Post-Rule Scan Results

The new rule guarantees that the IP address 192.168.1.10 can access port 22, while IP 10.1.1.10 is blocked.

- The image shows that VM1 cannot see what is open on port 22 on VM3 using the IP address 10.1.1.10.
- VM1 can no longer ping VM3 due to the firewall rule.



```
Kali Linux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ ping 10.1.1.10
PING 10.1.1.10 (10.1.1.10) 56(84) bytes of data.
^Z
zsh: suspended ping 10.1.1.10

(kali@kali)-[~]
$ nmap -p 22 10.1.1.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-07 00:44 GMT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.09 seconds

(kali@kali)-[~]
$ nmap -p 22 192.168.1.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-07 00:59 GMT
Nmap scan report for 192.168.1.10
Host is up (0.00015s latency).

PORT      STATE SERVICE
22/tcp    open  ssh

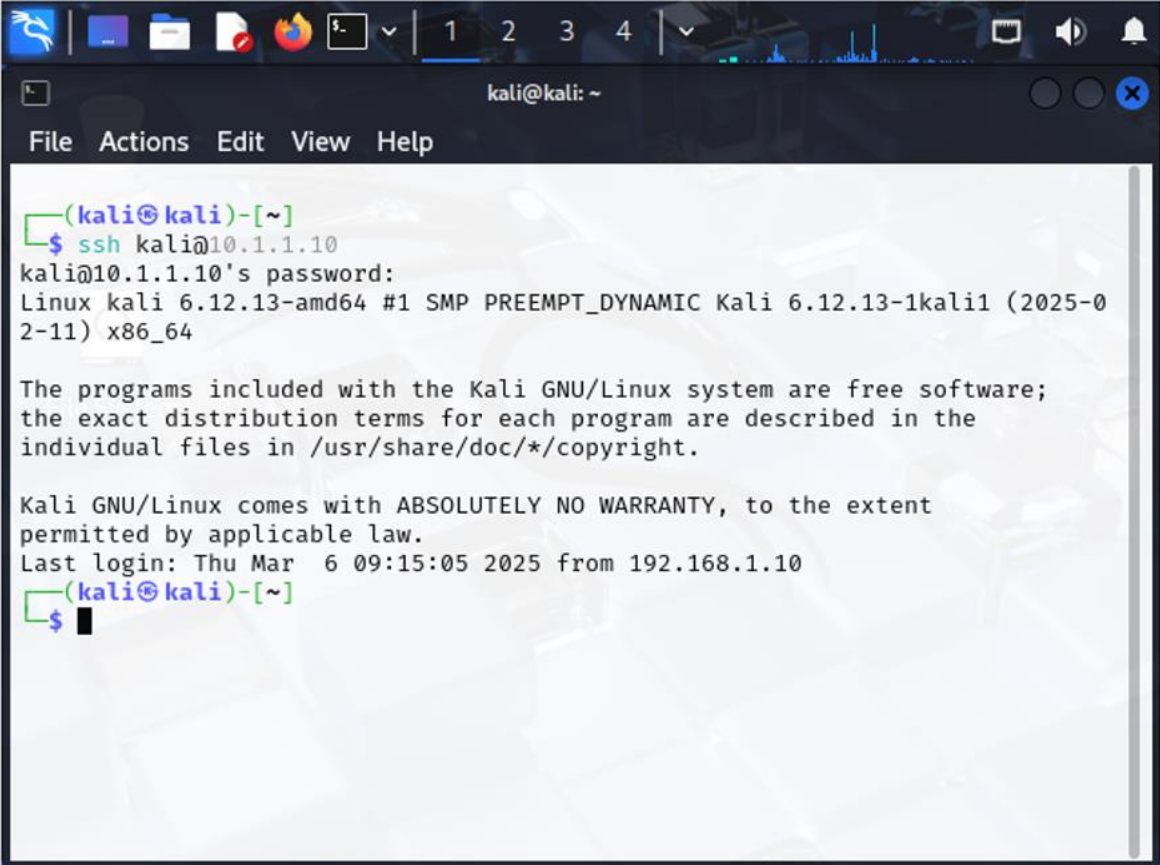
Nmap done: 1 IP address (1 host up) scanned in 13.08 seconds

(kali@kali)-[~]
$
```

5.1.3 Test 2: SSH Access Before and After Rules

Attempt to Log in Before Firewall Rule

Before the rules were set, VM1 was able to log in remotely via SSH to VM3 using VM3's IP address (10.1.1.10).



The screenshot shows a terminal window titled 'kali@kali: ~'. The terminal output is as follows:

```
(kali@kali)-[~]  
$ ssh kali@10.1.1.10  
kali@10.1.1.10's password:  
Linux kali 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1 (2025-02-11) x86_64  
  
The programs included with the Kali GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Thu Mar 6 09:15:05 2025 from 192.168.1.10  
(kali@kali)-[~]  
$
```

5.1.4 Implementation of SSH Blocking Rule

New rules were set on VM2:

1. The first rule allows traffic within the same network, meaning VM1 can still ping VM3.
2. The second rule blocks SSH remote login using 10.1.1.10, as port 22 is now blocked. This will result in a connection refused error when attempting to SSH.

```
(kali㉿kali)-[~]  
$ sudo nft list ruleset  
table ip filter {  
    chain input {  
        type filter hook input priority filter; policy accept;  
        accept  
        ip saddr 192.168.1.10 icmp type echo-request accept  
    }  
  
    chain forward {  
        type filter hook forward priority filter; policy accept;  
        ip saddr 192.168.1.10 ip daddr 10.1.1.10 tcp dport 22 reject  
    }  
}  
  
(kali㉿kali)-[~]  
$
```

Post-Rule Results (Blocked Login)

To test the rule, I attempted to log in from VM1 to VM3 using the IP address of VM3.

- VM1 was still able to ping VM3 since the rule allows internal network traffic.
- However, SSH access from IP 10.1.1.10 to VM3 was completely blocked, and the connection was refused on port 22.

```
(kali㉿kali)-[~]  
$ ping 10.1.1.10  
PING 10.1.1.10 (10.1.1.10) 56(84) bytes of data:  
64 bytes from 10.1.1.10: icmp_seq=1 ttl=63 time=1.81 ms  
64 bytes from 10.1.1.10: icmp_seq=2 ttl=63 time=2.10 ms  
64 bytes from 10.1.1.10: icmp_seq=3 ttl=63 time=1.82 ms  
64 bytes from 10.1.1.10: icmp_seq=4 ttl=63 time=2.16 ms  
64 bytes from 10.1.1.10: icmp_seq=5 ttl=63 time=2.91 ms  
64 bytes from 10.1.1.10: icmp_seq=6 ttl=63 time=3.20 ms  
64 bytes from 10.1.1.10: icmp_seq=7 ttl=63 time=1.71 ms  
64 bytes from 10.1.1.10: icmp_seq=8 ttl=63 time=2.75 ms  
^Z  
zsh: suspended  ping 10.1.1.10  
  
(kali㉿kali)-[~]  
$ ssh kali@10.1.1.10  
ssh: connect to host 10.1.1.10 port 22: Connection refused  
  
(kali㉿kali)-[~]  
$ █
```

5.2 PFSense

5.2.1 Test 1: Nmap Scan Before and After Rules

Initial Nmap Scan Results (Before Firewall Rules)

To assess the system's security posture before implementing firewall rules, a Nmap scan was conducted from the Kali Linux machine against the Xubuntu target machine (192.168.1.104). The results showed that port 22 (SSH) was open, indicating that the system was accessible for remote login attempts.

```
(james@kali)-[~]
$ nmap -p 22 192.168.1.104
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-08 02:59 GMT
Nmap scan report for 192.168.1.104
Host is up (0.0012s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:04:BC:ED (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

5.2.2 Firewall Rule Implementation in pfSense

To restrict access, a new rule was added in pfSense to block incoming SSH traffic on port 22 to the Xubuntu machine. This rule prevents unauthorised users from scanning and accessing the SSH service. The rule was configured as follows:

- Action: Block
- Interface: LAN
- Protocol: TCP
- Source: Any
- Destination: 192.168.1.104
- Destination Port: 22 (SSH)

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	1 / 4.29 MIB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	✗ 0 / 0 B	IPv4 TCP	*	*	192.168.1.104	22 (SSH)	*	none		Block Port 22 (NMAP)	
<input type="checkbox"/>	✓ 6 / 11.39 MIB	IPv4 *	LAN net	*	*	*	*	none		Default allow LAN to any rule	
<input type="checkbox"/>	✓ 0 / 0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	

Post-Rule Nmap Scan Results

After applying the firewall rule, another Nmap scan was performed to verify if the rule successfully blocked SSH access. The results confirmed that port 22 was now closed, meaning the firewall was effectively preventing external access.

```
PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:04:BC:ED (Oracle VirtualBox virtual NIC)
```

5.2.3 Test 2: SSH Access Before and After Rules

SSH Login Attempt Before Firewall Rule

Before implementing the firewall rule, an SSH connection was attempted from the Kali Linux machine to the Xubuntu target. As expected, the login was successful since port 22 was open, allowing remote access.

```
(james@kali)-[~]
$ ssh vmuser@192.168.1.104
vmuser@192.168.1.104's password:
Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

106 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

Last login: Sat Mar  8 02:36:52 2025 from 192.168.1.101
vmuser@vmuser-VirtualBox:~$
```

Post-Rule SSH Access Attempt

To verify the effectiveness of the rule, another SSH login attempt was made. This time, the connection was denied, confirming that pfSense successfully blocked SSH access.

```
(james@kali)-[~]
$ ssh vmuser@192.168.1.104
ssh: connect to host 192.168.1.104 port 22: Connection refused
```


6. Comparison of nftables and pfSense

Both **nftables** and **pfSense** were tested to evaluate their effectiveness in handling **network security**, specifically in blocking **Nmap scans** and **SSH access**. While both firewalls successfully restricted unauthorised access, they differ in **implementation, usability, and flexibility**.

Criteria	nftables	pfSense
Configuration & Use	Command-line-based: It requires manual rule writing and configuration via nftables.conf. It provides more flexibility but requires advanced knowledge.	GUI-based interface: Rules can be configured easily through the web interface without needing command-line expertise.
Firewall Rule Management	It uses chains and tables for rule processing and supports fine-grained rule customisation. Requires familiarity with rule syntax.	Intuitive rule management with a structured interface; rules can be created, modified, and reordered easily.
Flexibility	Highly flexible; allows for complex rule sets and scripting. Ideal for users comfortable with Linux networking.	Limited to GUI options, but offers pre-configured features like VPN, IDS/IPS, and NAT, making setup easier.
Performance	Efficient in handling large rule sets integrated into the Linux kernel.	Slightly higher overhead due to GUI and additional features but optimised for ease of use.
Best Use Case	It is ideal for advanced users, system administrators, or environments requiring custom firewall rules and scripting.	It is best for users who prefer ease of use, quick deployment, and a centralized firewall solution with built-in features.

6.1 Summary

From this comparison, **nftables** is better suited for users who require a highly customisable firewall with scripting capabilities. At the same time, **pfSense** is more accessible for those who prefer a user-friendly GUI with built-in security features.

7. Final Conclusion

This project successfully evaluated **nfTables** and **pfSense** as firewall solutions, testing their effectiveness in blocking Nmap scans and restricting SSH access. Both firewalls demonstrated their ability to secure a network but with key differences in usability and implementation.

Ultimately, the choice between **nfTables** and **pfSense** depends on the user's specific needs, whether they prefer a command-line-driven, highly customisable firewall (**nfTables**) or an intuitive, structured approach to network security (**pfSense**). Both solutions are effective, reinforcing the importance of firewalls in protecting networks from potential threats.

8. Bibliography

Cisco, 2024. What is a firewall? Cisco. Available at:

<https://www.cisco.com/site/uk/en/learn/topics/security/what-is-a-firewall.html>

[Accessed 7 March 2024].

Gillis, A.S., Loshin, P. & Cobb, M., 2024. What is SSH (Secure Shell) and How Does It Work? TechTarget. Available at:

<https://www.techtarget.com/searchsecurity/definition/Secure-Shell>

[Accessed 7 March 2024].

Hypponen, M., 2024. Firewall Security and Network Protection. Tampere University. Available at:

<https://trepo.tuni.fi/bitstream/handle/10024/131119/Hypp%C3%B6nenMikko.pdf?sequence=2> [Accessed 7 March 2024].

Manish, S., 2020. What is Nmap and How to Use it – A Tutorial for the Greatest Scanning Tool of All Time. FreeCodeCamp. Available at:

<https://www.freecodecamp.org/news/what-is-nmap-and-how-to-use-it-a-tutorial-for-the-greatest-scanning-tool-of-all-time/> [Accessed 7 March 2024].

Olivia, J., Mohandes, D. & Dariush, K., 2024. Replacement of iptables with nftables. Université catholique de Louvain. Available at:

https://dial.uclouvain.be/downloader/downloader.php?pid=thesis%3A35548&datastream=PDF_01&cover=cover-mem [Accessed 7 March 2024].

TekLager, 2024. pfSense - Introduction to the Most Powerful Router Operating System.

TekLager. Available at: <https://teklager.se/en/pfsense-introduction-open-source-router-firewall/> [Accessed 7 March 2024].