University of Balamand

Faculty of Engineering

Computer Engineers

FireWall with IP tables

Submitted to:

Dr. Nayla Greige

Submitted by:

Christelle Oueiss

Salam Lababidi

CPEN310: CyberSecurity Lab Project

December 6, 2024

ABSTRACT

In an era where digital connectivity governs every facet of modern life, the need for robust and intelligent network security mechanisms is paramount. This project delves into the creation of a minimalist yet powerful firewall using iptables, a cornerstone of Linux-based network defense. By harnessing the intricacies of packet filtering and traffic management, this endeavor aims to sculpt a dynamic shield that adapts to evolving network landscapes. At its heart, this firewall embodies simplicity and precision, blocking unauthorized access while ensuring uninterrupted communication for trusted connections. By configuring rules that allow only essential services like SSH and HTTP, and enabling granular control over trusted IPs, the project not only fortifies system security but also fosters an environment of trust and controlled access. Through a meticulous learning curve, we unravel the anatomy of iptables commands—exploring how chains, tables, and rules converge to orchestrate traffic flow. The firewall is rigorously tested against a spectrum of scenarios, evaluating its resilience and adaptability to real-world challenges. By the end of this journey, we demonstrate that even the most fundamental tools, when wielded with intent and innovation, can forge an unyielding line of defense in the digital frontier.

TABLE OF CONTENTS

ABSTRACT.. **2**

1. Introduction: **4**

2. Literature Review **4**

3. What is a FireWall with IP tables:.. **5**

3.1 Definition of Firewall.. **5**

3.2 What is a FireWall using iptables: **5**

4. Technical Methods and Tools:.. **6**

4.1 Packet Filtering Process.. **6**

4.2 Key Components..6

- **Tables**: Organize rules based on their functionality. Common tables include: 6

1. FILTER: The default table used for packet filtering (e.g., accept, drop, reject). 6

2. NAT: Used for altering packet addresses (e.g., for port forwarding). 6

3. **MANGLE**: Used for modifying packet headers. 6

4.3 Rule Management..7

5. Real World Implications **7**

6. Implementation:.. **8**

7. Discussion:..**11**

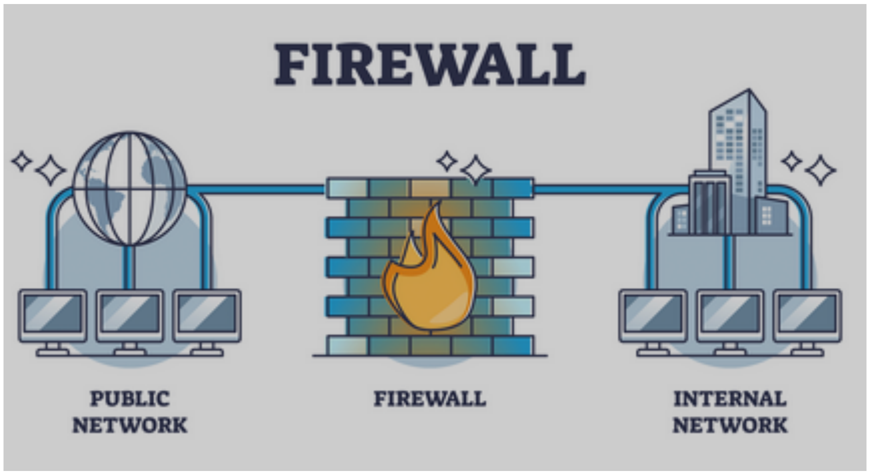
8. Checking:..**11**

10. Recommendations **13**

11. Conclusion..**14**

1. **Introduction:**

Network security is a critical pillar in safeguarding systems from unauthorized access and malicious threats. Firewalls act as the first line of defense, filtering traffic to ensure only trusted data reaches the system. This project focuses on implementing a simple yet effective firewall using iptables, a powerful tool in Linux. By learning the core commands and configuring rules to manage traffic based on IP addresses, ports, and protocols, this project provides a hands-on approach to securing Linux systems. With a focus on precision and adaptability, this firewall aims to balance security with accessibility, ensuring seamless operation for essential services while protecting against vulnerabilities.



2. **Literature Review**

Firewalls have been a cornerstone of network security since their inception in the late 1980s, evolving alongside the growth of the internet. Early firewalls were simple packet filters, scrutinizing data packets based on predefined rules. Over time, they advanced to stateful inspection and application-layer firewalls, offering more robust and dynamic protection. Linux's `iptables`, introduced with the Netfilter framework, revolutionized open-source firewall technology by providing a flexible, command-line interface for managing network traffic. It enabled administrators to define granular rules for packet filtering, logging, and NAT (Network Address Translation). As a lightweight and powerful tool, iptables remains widely used in modern network security, offering unparalleled control over traffic flow in Linux environments. This project builds upon this rich history by leveraging iptables to craft a simple yet effective firewall that embodies the principles of modern cybersecurity.

3. What is a FireWall with IP tables:

3.1 Definition of Firewall

A firewall is a critical network security device or software tool designed to monitor and control incoming and outgoing traffic based on predefined security rules. Acting as a barrier between trusted internal networks and untrusted external networks, such as the internet, a firewall helps prevent unauthorized access while permitting legitimate communication. Firewalls operate at different levels, from basic packet filtering—inspecting data packets based on IP addresses, ports, and protocols—to advanced stateful inspection and application-layer monitoring, which analyze the context and behavior of traffic. By enforcing access control policies and filtering malicious data, firewalls play a vital role in securing systems, mitigating cyber threats, and maintaining the integrity of network communications.

3.2 What is a FireWall using iptables:

iptables is a powerful command-line tool used in Linux-based systems for configuring and managing firewall rules. As part of the Netfilter framework, it enables administrators to define rules that control the flow of network traffic into, out of, and through a system. These rules operate on various layers of the network stack, filtering traffic based on criteria such as IP addresses, ports, protocols, and connection states.

With iptables , users can create custom chains of rules that determine how packets are processed, dropped, or accepted. It supports a wide range of functionalities, including packet filtering, Network Address Translation (NAT), and logging. This flexibility allows `iptables` to serve as an effective tool for implementing robust firewalls tailored to specific security needs.

In this project, iptables is used to configure a basic firewall that blocks all incoming traffic by default, allowing only essential services such as SSH (port 22) and HTTP (port 80), while selectively permitting trusted IPs. This approach demonstrates how `iptables` provides granular control over traffic flow, ensuring security without compromising accessibility.

4. Technical Methods and Tools:

The iptables firewall operates by defining rules within specific tables and chains that dictate how network packets are processed. Below is a breakdown of the technical methods and tools that enable its functionality:

4.1 Packet Filtering Process

iptables inspects packets as they traverse the system and determines their fate based on predefined rules. Each rule specifies criteria such as source/destination IP, port, protocol, and connection state. Packets are processed by matching them against these rules sequentially

4.2 Key Components

- **Tables**:

  Organize rules based on their functionality. Common tables include:

  1. **FILTER**: The default table used for packet filtering (e.g., accept, drop, reject).
  2. **NAT**: Used for altering packet addresses (e.g., for port forwarding).
  3. **MANGLE**: Used for modifying packet headers.
- **Chains**: Define the path packets follow within a table:
  1. **INPUT**: Manages packets destined for the local machine.
  2. **OUTPUT**: Handles packets leaving the local machine.
  3. **FORWARD**: Processes packets routed through the machine (e.g., for NAT or routing).
- **Targets**: Specify the action to take on a packet:
  1. **ACCEPT**: Allow the packet to pass.
  2. **DROP**: Silently discard the packet.
  3. **REJECT**: Discard the packet and send an error message.

4.3 Rule Management

iptables provides commands to manage rules in chains:

- -A (Append): Adds a rule to the end of a chain.
- -I (Insert): Adds a rule at a specific position.
- -D (Delete): Removes a rule from a chain.
- -L (List): Displays the current rules.
- -F (Flush): Clears all rules in a chain.

5. Real World Implications

WannaCry Ransomware Attack (2017)

In May 2017, the WannaCry ransomware attack spread globally, infecting over 230,000 computers in more than 150 countries. The malware exploited a vulnerability in Microsoft's Windows operating system to propagate through networks, encrypting files on infected systems and demanding ransom payments in Bitcoin. Organizations ranging from hospitals to transportation systems faced crippling disruptions.

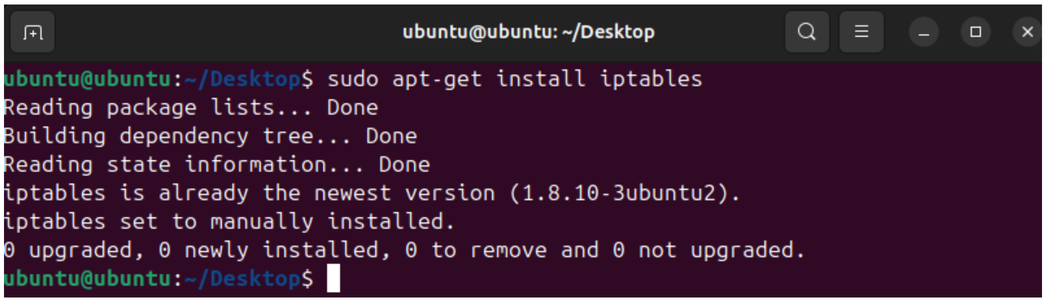
One of the critical ways organizations mitigated the impact of WannaCry was using firewalls. Network administrators quickly configured firewalls to block the specific ports (445 and 139) used by the malware's exploit, known as EternalBlue, to spread. By restricting traffic on these vulnerable ports, firewalls effectively contained the ransomware's propagation within networks and prevented further infection.

For example, the UK's National Health Service (NHS), one of the most affected entities, relied on firewall adjustments to halt the spread of the malware within its systems. Organizations that had pre-configured firewalls to block unnecessary ports or restricted access to trusted IPs were able to largely avoid the attack, demonstrating the critical role of firewalls in defending against large-scale cyber threats.

This incident highlights how proactive firewall configurations, such as those achievable with tools like iptables, can be the difference between a widespread disaster and a contained security breach.

6. Implementation:

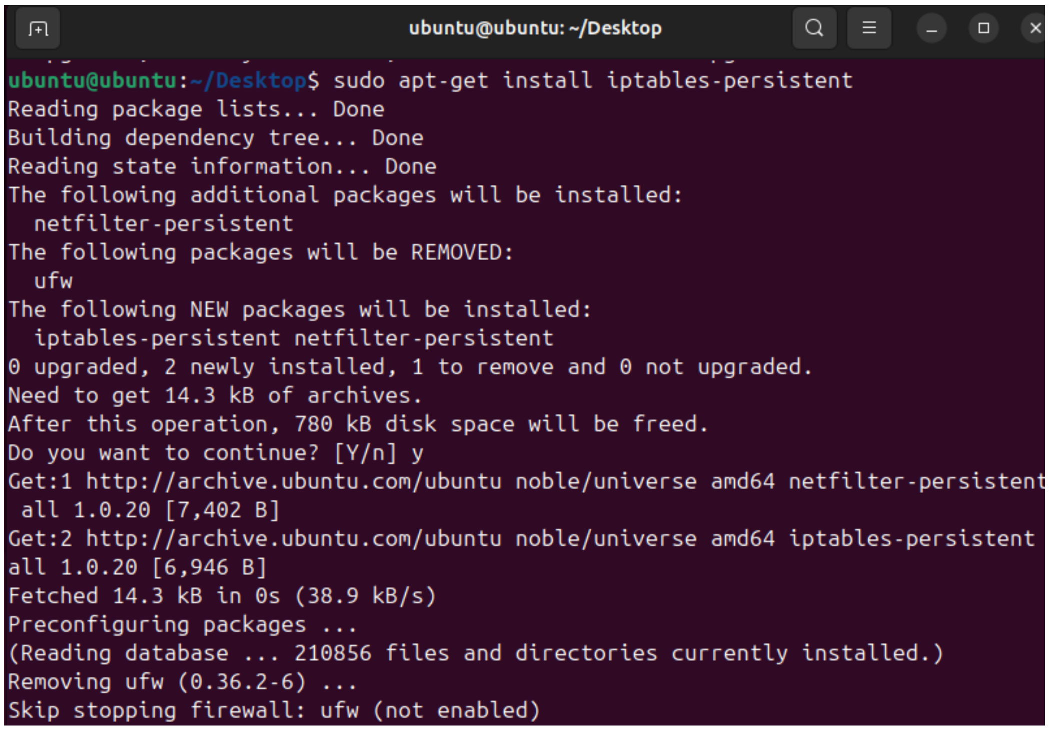
1- Check if iptables Exits:



```
ubuntu@ubuntu:~/Desktop$ sudo apt-get install iptables
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables is already the newest version (1.8.10-3ubuntu2).
iptables set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ubuntu:~/Desktop$
```

In this step, the iptables package was verified as already installed on the system using the sudo apt-get install iptables command. The output confirms that the latest version (1.8.10-3ubuntu2) is already manually installed, with no further updates or installations required.

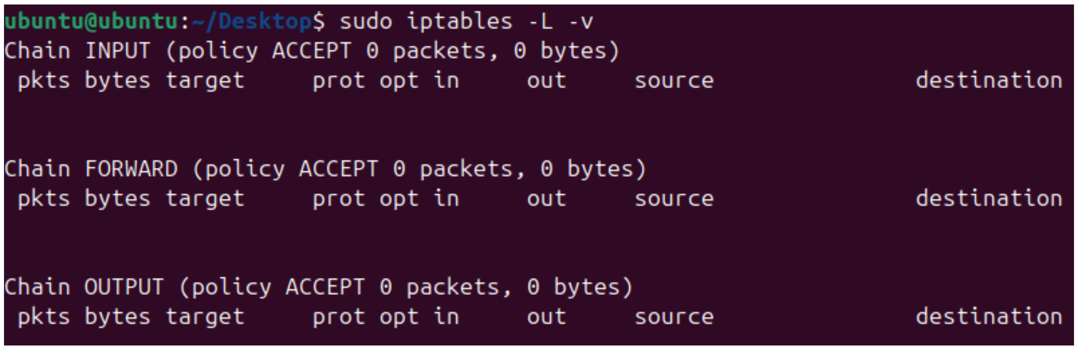
2- Install iptables (If Needed)



```
ubuntu@ubuntu:~/Desktop$ sudo apt-get install iptables-persistent
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  netfilter-persistent
The following packages will be REMOVED:
  ufw
The following NEW packages will be installed:
  iptables-persistent netfilter-persistent
0 upgraded, 2 newly installed, 1 to remove and 0 not upgraded.
Need to get 14.3 kB of archives.
After this operation, 780 kB disk space will be freed.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 netfilter-persistent
 all 1.0.20 [7,402 B]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 iptables-persistent
all 1.0.20 [6,946 B]
Fetched 14.3 kB in 0s (38.9 kB/s)
Preconfiguring packages ...
(Reading database ... 210856 files and directories currently installed.)
Removing ufw (0.36.2-6) ...
Skip stopping firewall: ufw (not enabled)
```

In this step, the iptables-persistent package was installed using the sudo apt-get install iptables-persistent command. During the process, the netfilter-persistent tool was also installed to manage persistent firewall rules.

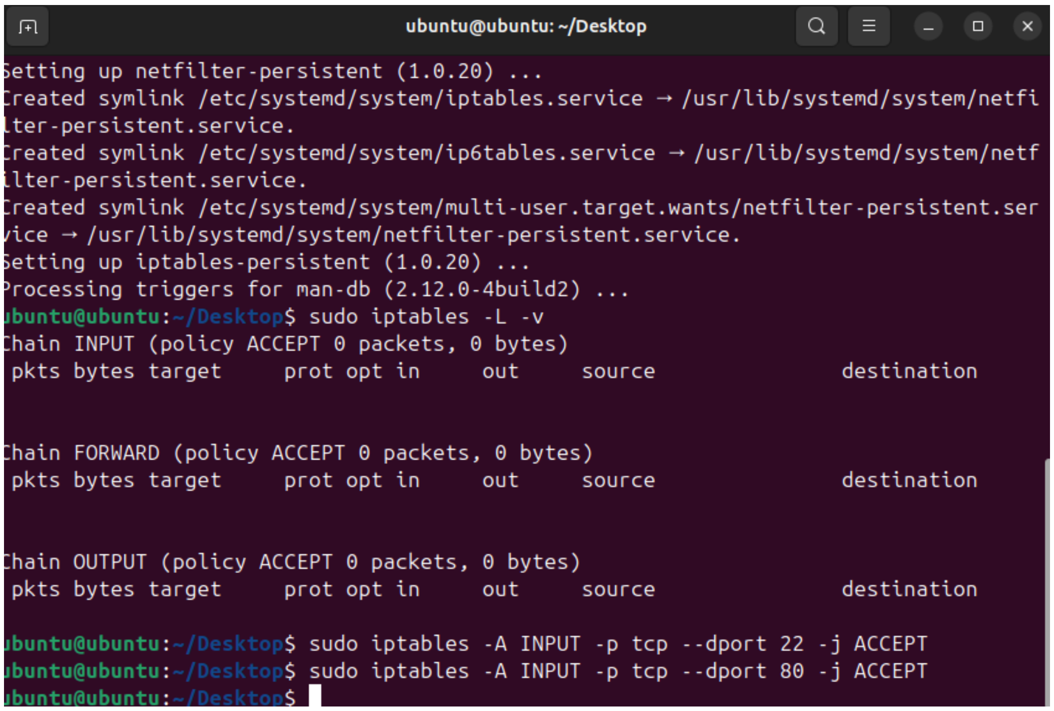
3- Check Existing iptables Rules



```
ubuntu@ubuntu:~/Desktop$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

In this step, the command sudo iptables -L -v was executed to display the current firewall rules in verbose mode. The output shows that all default policies (INPUT, FORWARD, and OUTPUT chains) are set to ACCEPT, meaning no traffic is being filtered, and all incoming, forwarded, and outgoing traffic is allowed by default. No specific rules have been added yet.

4- Allow HTTP (Port 80) and SSH (Port 22)



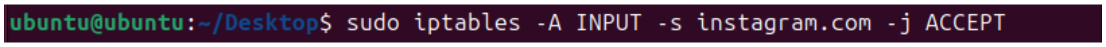
```
etting up netfilter-persistent (1.0.20) ...
reated symlink /etc/systemd/system/iptables.service → /usr/lib/systemd/system/netfi
lter-persistent.service.
reated symlink /etc/systemd/system/ip6tables.service → /usr/lib/systemd/system/netf
ilter-persistent.service.
reated symlink /etc/systemd/system/multi-user.target.wants/netfilter-persistent.ser
ice → /usr/lib/systemd/system/netfilter-persistent.service.
etting up iptables-persistent (1.0.20) ...
rocessing triggers for man-db (2.12.0-4build2) ...
buntu@ubuntu:~/Desktop$ sudo iptables -L -v
hain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


hain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


hain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

buntu@ubuntu:~/Desktop$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
buntu@ubuntu:~/Desktop$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
buntu@ubuntu:~/Desktop$
```

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT: This allows incoming TCP traffic on port 22, which is used for SSH connections, ensuring remote access to the system.

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT: This allows incoming TCP traffic on port 80, which is used for HTTP, enabling the system to serve or receive web traffic.

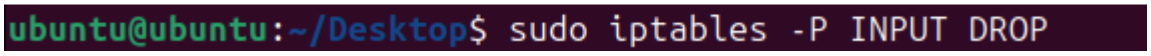
These rules ensure that the system is accessible via SSH and HTTP while other incoming traffic remains unrestricted for now

5- Allow Traffic from a specific IP

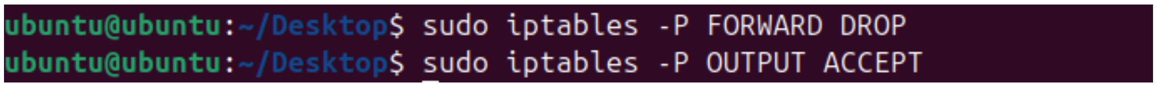


```
ubuntu@ubuntu:~/Desktop$ sudo iptables -A INPUT -s instagram.com -j ACCEPT
```

6- Block Unnecessary traffic and allow outgoing requests



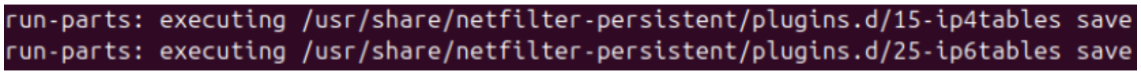
```
ubuntu@ubuntu:~/Desktop$ sudo iptables -P INPUT DROP
```



```
ubuntu@ubuntu:~/Desktop$ sudo iptables -P FORWARD DROP
ubuntu@ubuntu:~/Desktop$ sudo iptables -P OUTPUT ACCEPT
```

This command sets the default policy for the `INPUT` chain to `DROP`, meaning all incoming traffic will be blocked unless explicitly allowed by another rule. It is a strong security measure to deny unauthorized access by default.

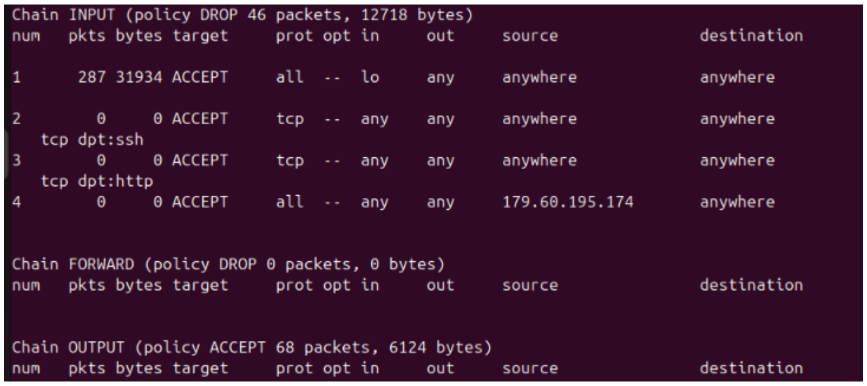
7- Save FireWall rules Persistently :Sudo netfliter-persistent save



```
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
```

The command `sudo netfilter-persistent save` saves the current `iptables` rules to be persistent across system reboots. The output indicates that both IPv4 (`15-ip4tables`) and IPv6 (`25-ip6tables`) rules were successfully saved by the `netfilter-persistent` service.

7. **Discussion:**

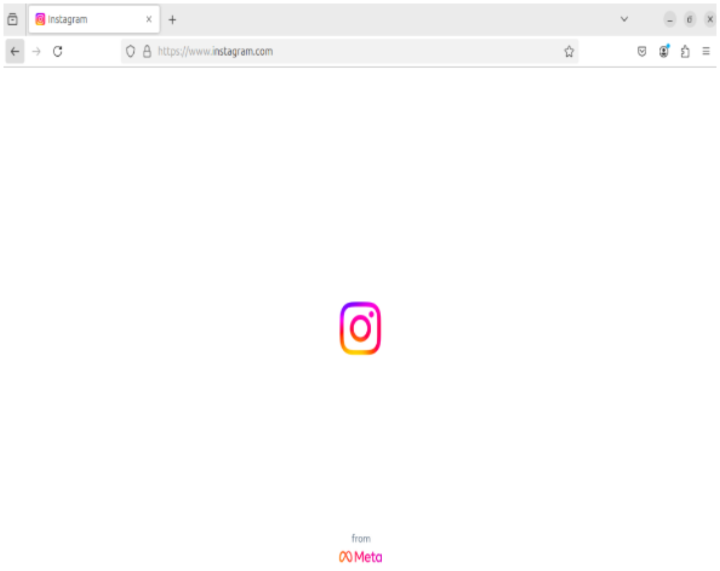


```
Chain INPUT (policy DROP 46 packets, 12718 bytes)
num   pkts bytes target     prot opt in     out     source               destination

1      287 31934 ACCEPT     all  --  lo     any     anywhere             anywhere

2        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere
   tcp dpt:ssh
3        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere
   tcp dpt:http
4        0     0 ACCEPT     all  --  any    any     179.60.195.174       anywhere


Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 68 packets, 6124 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

The output confirms that the steps we followed earlier to configure the firewall are correctly reflected in the current iptables configuration. The rules are in place, and the default policies are properly set. The packet and byte counts show that the loopback traffic (Rule 1) is active, while the other rules are ready to process traffic but haven't yet matched any packets. This indicates the firewall is operational and enforcing the configured rules.

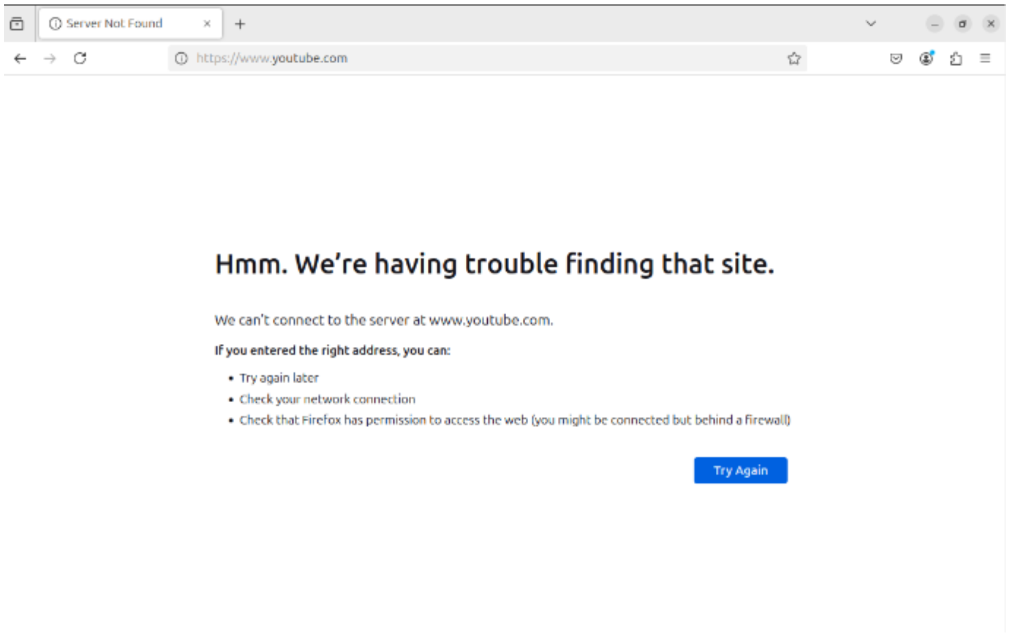
8. **Checking:**

1. Checking if specific IP traffic is being allowed



Restricting access to a specific IP address (Instagram) and limiting traffic to essential ports like 80 (HTTP) and 22 (SSH) enhances the system's security by allowing only trusted sources to connect. This controlled access minimizes exposure to unauthorized connections and reduces the risk of potential cyberattacks. By keeping unnecessary ports closed and only allowing traffic for critical services, the system's attack surface is significantly reduced. This ensures that malicious traffic is blocked while maintaining functionality for legitimate users, striking a balance between accessibility and robust security.

2. Checking if all other IP traffic is being blocked



Hmm. We're having trouble finding that site.

We can't connect to the server at www.youtube.com.

If you entered the right address, you can:

- Try again later
- Check your network connection
- Check that Firefox has permission to access the web (you might be connected but behind a firewall)

Blocking unnecessary traffic greatly improves the security of the system by eliminating unauthorized access attempts. By restricting access to only essential ports (80 for HTTP and 22 for SSH), the system is shielded from unwanted external connections and potential vulnerabilities. This strategy minimizes the risk of exploitation by ensuring that only legitimate traffic is allowed through, effectively reducing the system's exposure to attacks. Any attempt to access services outside the permitted rules is automatically blocked, highlighting the firewall's capability to restrict unauthorized access and safeguard the system against potential threats.

9. Challenges and Countermeasures

Configuring and maintaining firewalls, such as those using `iptables`, presents several challenges. A common issue is the complexity of crafting precise rules, which can lead to misconfigurations, inadvertently blocking legitimate traffic or leaving vulnerabilities exposed. Firewalls can also struggle with high traffic volumes, leading to performance bottlenecks. Additionally, they may be bypassed by sophisticated attacks like encrypted malware or internal threats, which operate within trusted networks.

To counter these challenges, several measures can be implemented. Regular audits and testing of firewall rules ensure configurations remain effective and up to date. Automating rule generation with tools like Ansible can reduce human error, while integrating firewalls with Intrusion Detection Systems (IDS) adds a layer of monitoring to detect and respond to anomalous behavior. For performance optimization, load balancers can distribute traffic, and deep packet inspection can analyze encrypted data without compromising security. By combining robust configurations with proactive management, firewalls can remain effective against evolving threats.

10. Recommendations

1- **Integration with Monitoring Tools**

Future implementations could include integrating iptables with network monitoring tools, such as Fail2Ban or Suricata, to enable real-time threat detection and automatic response to suspicious activities.

2- Automated Rule Management

Employing configuration management tools like Ansible or Puppet to automate firewall rule updates can minimize human error and ensure consistent security policies across multiple systems.

3- **Enhanced Logging and Analytics**
   Incorporating advanced logging mechanisms, such as centralized log management using tools like ELK Stack, can provide deeper insights into traffic patterns and potential threats.
4- **Scalability and Performance Optimization**
   Future development could focus on optimizing iptables for high-traffic environments by combining it with performance enhancers like load balancers or transitioning to more scalable solutions such as nftables.
5- **Security Hardening and Threat Modeling**
   Regularly reviewing and updating firewall rules based on emerging threats, combined with simulated attack scenarios, can enhance the overall security posture of the system.

These recommendations ensure that firewalls remain effective and adaptive, capable of addressing the evolving landscape of cybersecurity challenges.

11. Conclusion

Firewalls are an indispensable tool in safeguarding networks from unauthorized access and malicious threats. Through this project, we demonstrated how iptables can be used to design a robust yet straightforward firewall, capable of filtering traffic and securing Linux-based systems. By mastering the configuration of chains, tables, and rules, this project highlights the importance of precise traffic control in achieving an optimal balance between security and functionality. The practical implementation and testing of the firewall underscore its adaptability to real-world challenges, proving its value in protecting systems across diverse environments.