Host Firewall

Student: Mohammed JBILOU
ID: 7702249
Course: Cybersecurity
Institution: Keimyung University
Github reposiroty:
https://github.com/Molaryy/KMU_cybersecurity_assignment_2025

Table of Contents

1. Objective
2. Implementation Overview
3. Default Deny Firewall Policy
4. Allowing Required Services
5. Rule Persistence
6. Verification and Testing with nmap
7. Security Best Practices Implemented
8. Conclusion

1 - Objective

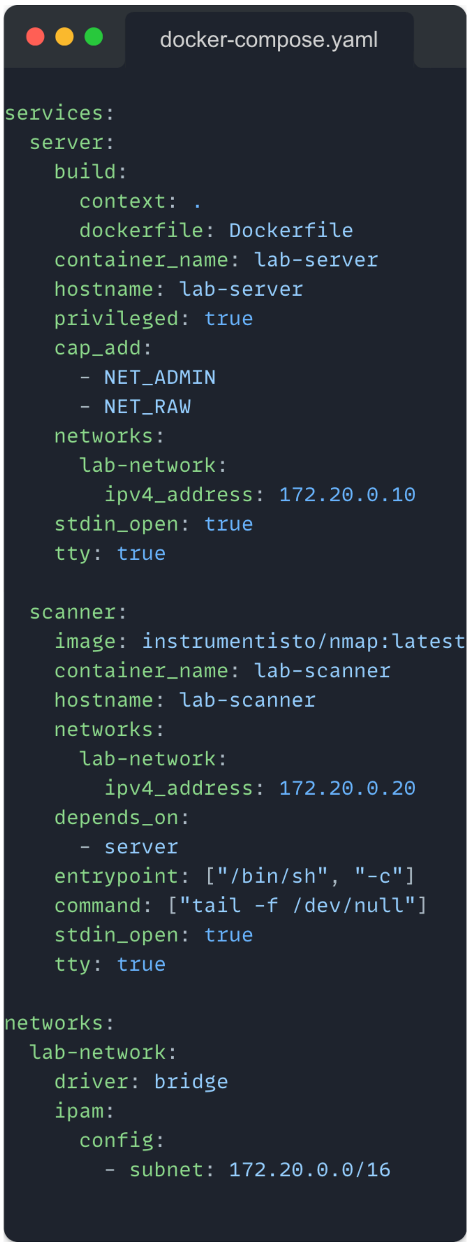
The objective of this lab was to implement a host based firewall on a Linux server using iptables. The firewall should enforce a default deny policy and then selectively allow only required traffic such as SSH and HTTPS. The configuration must persist across container restarts and must be verified through active scanning with nmap from a separate machine.

2 - Implementation Overview

A multi container lab environment was deployed using Docker Compose.
 The environment consisted of two containers:

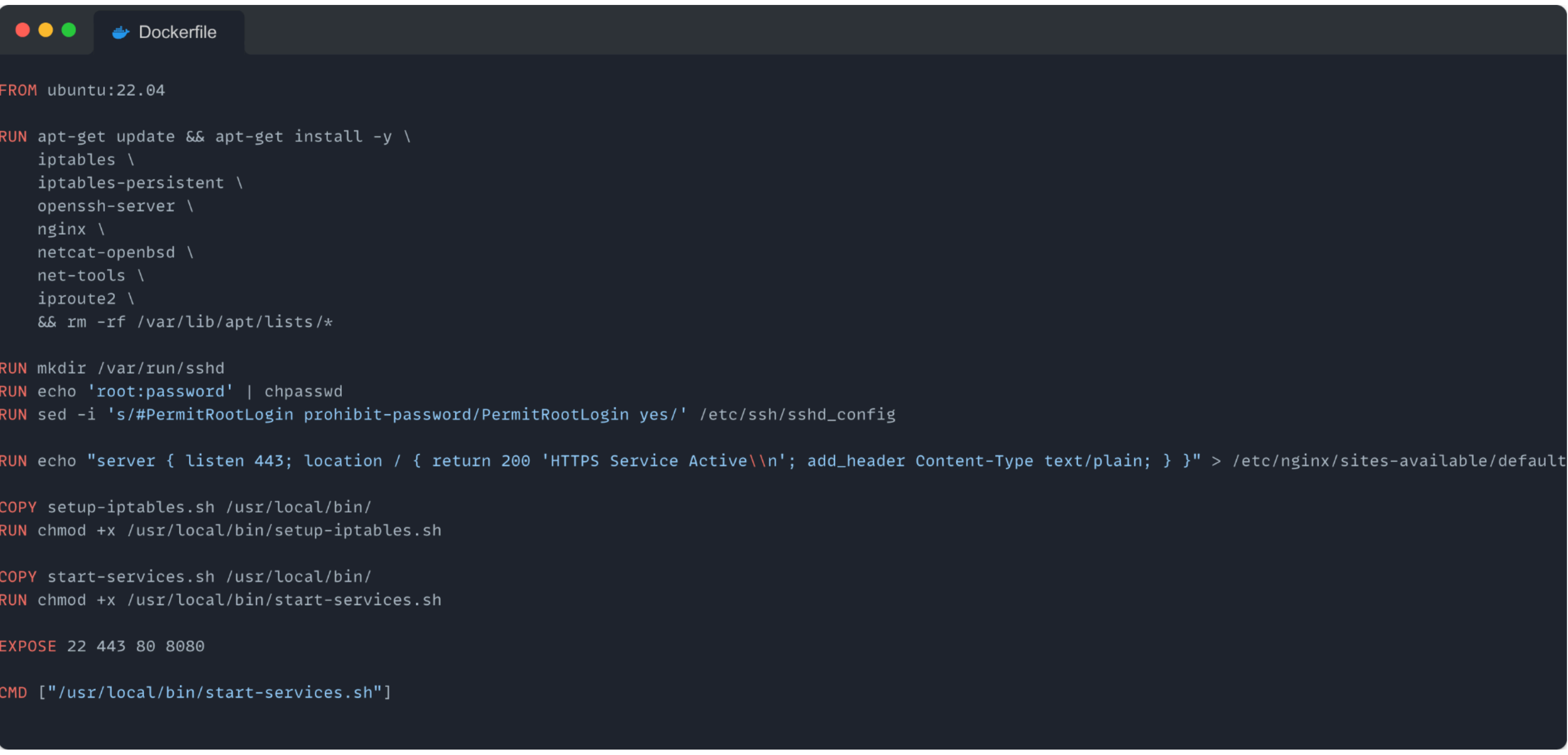
- `lab-server`: Ubuntu based server running SSH and Nginx with the ability to configure iptables rules
- `lab-scanner`: A separate container running nmap used for scanning and verification

The full environment definition is shown below.



```yaml
services:
  server:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: lab-server
    hostname: lab-server
    privileged: true
    cap_add:
      - NET_ADMIN
      - NET_RAW
    networks:
      lab-network:
        ipv4_address: 172.20.0.10
    stdin_open: true
    tty: true

  scanner:
    image: instrumentisto/nmap:latest
    container_name: lab-scanner
    hostname: lab-scanner
    networks:
      lab-network:
        ipv4_address: 172.20.0.20
    depends_on:
      - server
    entrypoint: ["/bin/sh", "-c"]
    command: ["tail -f /dev/null"]
    stdin_open: true
    tty: true

networks:
  lab-network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/16
```

The server image installs all necessary packages such as iptables, nginx, sshd, and tools required for debugging and networking.

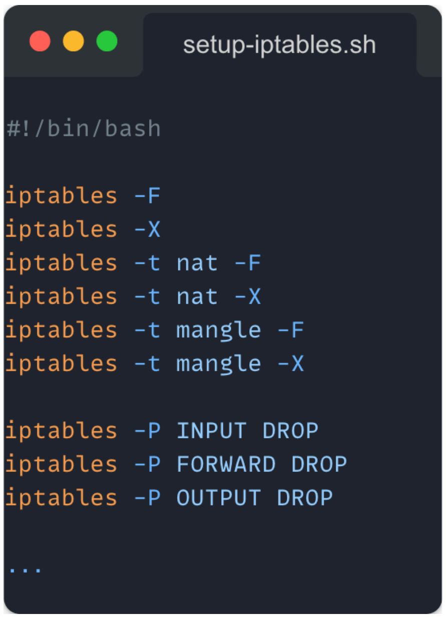


```dockerfile
FROM ubuntu:22.04

RUN apt-get update && apt-get install -y \
    iptables \
    iptables-persistent \
    openssh-server \
    nginx \
    netcat-openbsd \
    net-tools \
    iproute2 \
    && rm -rf /var/lib/apt/lists/*

RUN mkdir /var/run/sshd
RUN echo 'root:password' | chpasswd
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

RUN echo "server { listen 443; location / { return 200 'HTTPS Service Active\\n'; add_header Content-Type text/plain; } }" > /etc/nginx/sites-available/default

COPY setup-iptables.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/setup-iptables.sh

COPY start-services.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/start-services.sh

EXPOSE 22 443 80 8080

CMD ["/usr/local/bin/start-services.sh"]
```

3 - Default Deny Firewall Policy

The first step was to flush any existing firewall rules and apply a strict baseline configuration. All default chain policies were set to DROP for inbound, outbound, and forwarding traffic.

This ensures the server operates with a least privilege security posture.



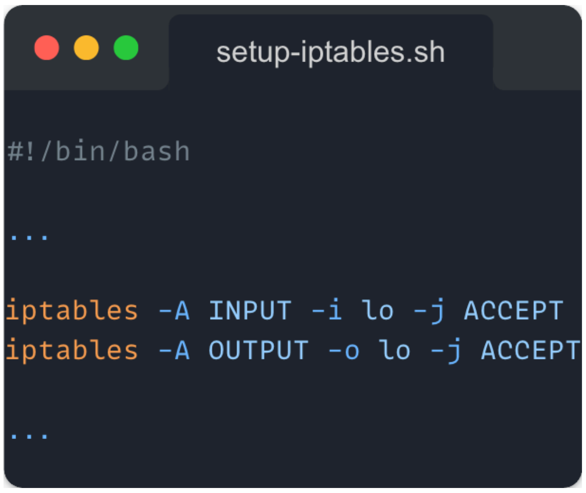
```
#!/bin/bash

iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

...
```

Running this script results in a system where no traffic is allowed unless explicitly permitted.

4 - Allowing Required Services

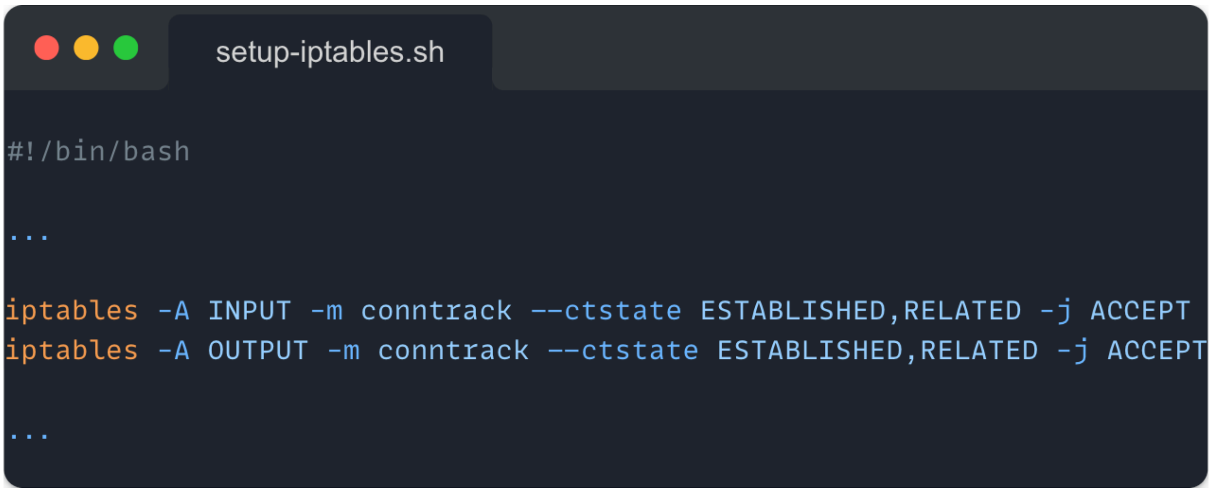
After defining the global deny rules, specific exceptions were added.

Allow loopback traffic



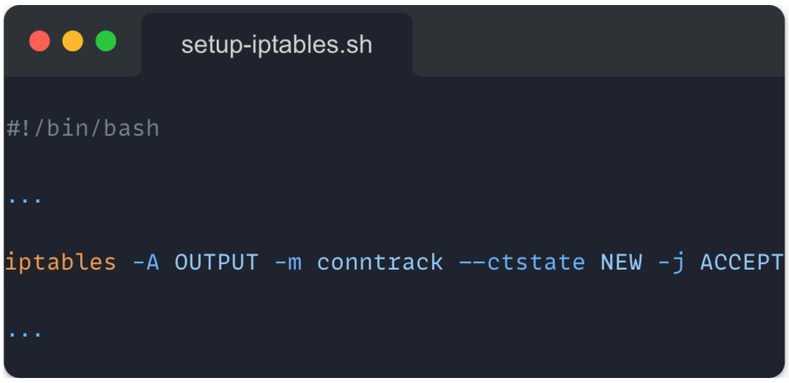
```
#!/bin/bash

...

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

...
```

Allow established and related connections



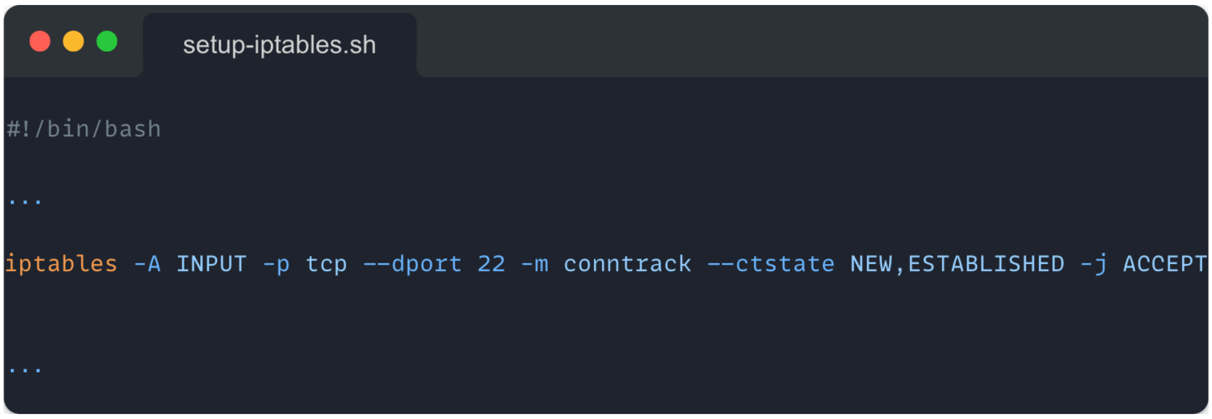
```
#!/bin/bash

...

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

...
```

Allow new outgoing connections



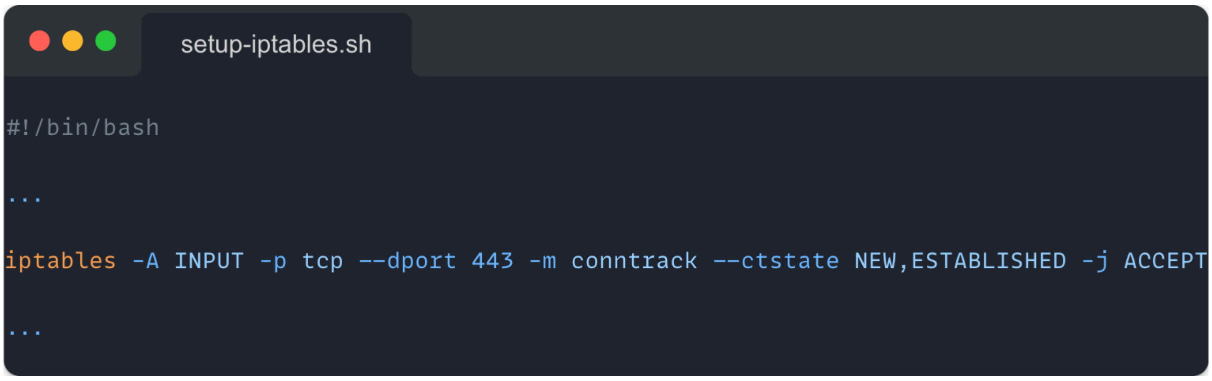
```
#!/bin/bash

...

iptables -A OUTPUT -m conntrack --ctstate NEW -j ACCEPT

...
```

Permit SSH on port 22



```
#!/bin/bash

...

iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

...
```

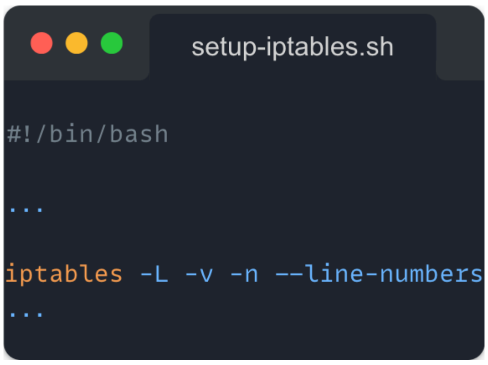
Permit HTTPS on port 443



setup-iptables.sh

```bash
#!/bin/bash

...

iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

...
```

Output rules after configuration



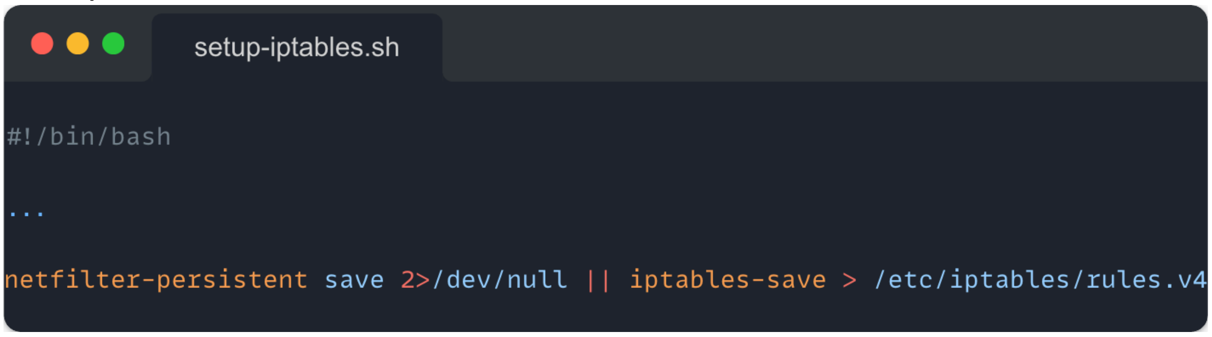
setup-iptables.sh

```bash
#!/bin/bash

...

iptables -L -v -n --line-numbers
...
```

5 - Rule Persistence

After installing `iptables-persistent`, the rule set is saved automatically when the script is executed.

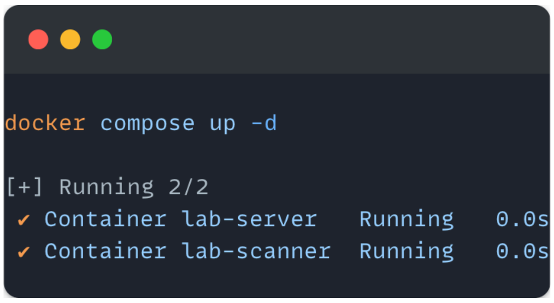
Example:



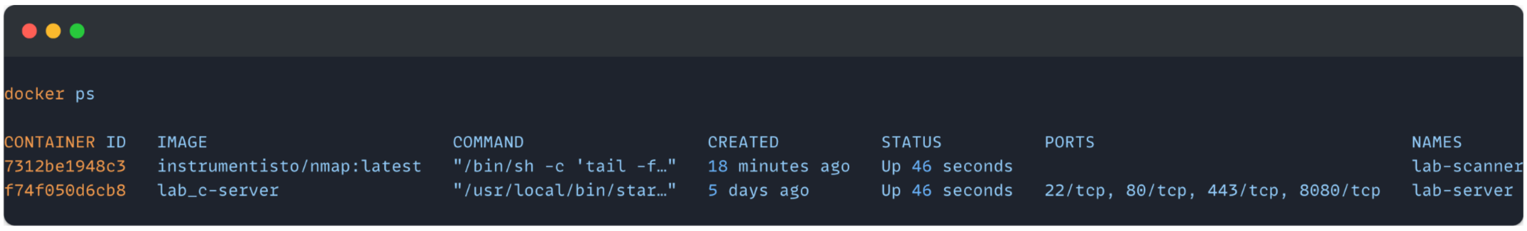
```bash
#!/bin/bash

...

netfilter-persistent save 2>/dev/null || iptables-save > /etc/iptables/rules.v4
```

6 - Verification and Testing with nmap

Start containers:



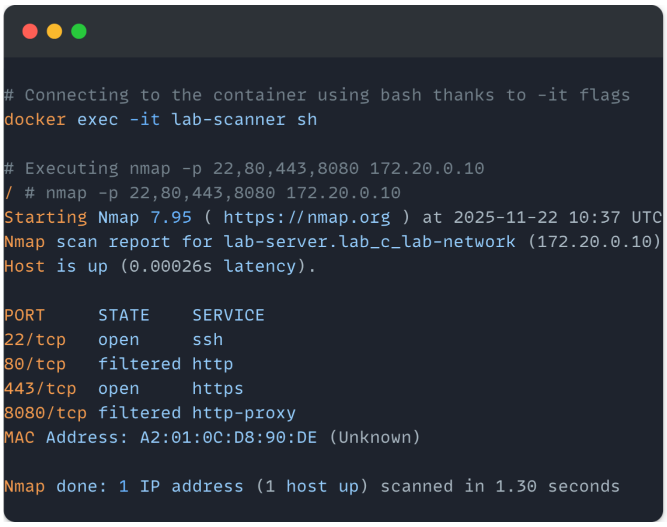
```
docker compose up -d

[+] Running 2/2
 ✔ Container lab-server   Running   0.0s
 ✔ Container lab-scanner  Running   0.0s
```



```
docker ps

CONTAINER ID   IMAGE                        COMMAND                  CREATED          STATUS          PORTS                                 NAMES
7312be1948c3   instrumentisto/nmap:latest   "/bin/sh -c 'tail -f…"   18 minutes ago   Up 46 seconds                                         lab-scanner
f74f050d6cb8   lab_c-server                 "/usr/local/bin/star…"   5 days ago       Up 46 seconds   22/tcp, 80/tcp, 443/tcp, 8080/tcp     lab-server
```

To validate the firewall configuration, the second container `lab-scanner` was used for active scanning.

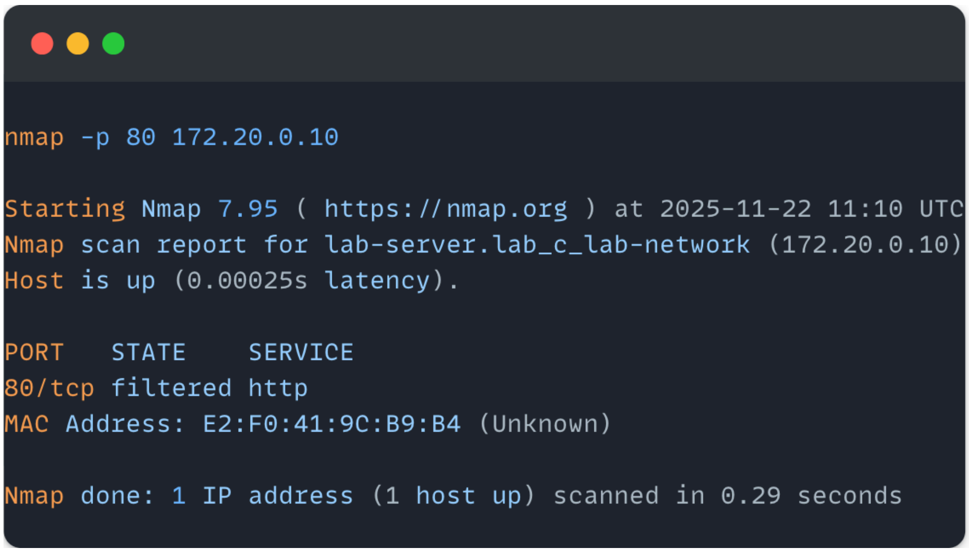
Scan all ports



```
# Connecting to the container using bash thanks to -it flags
docker exec -it lab-scanner sh

# Executing nmap -p 22,80,443,8080 172.20.0.10
/ # nmap -p 22,80,443,8080 172.20.0.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-22 10:37 UTC
Nmap scan report for lab-server.lab_c_lab-network (172.20.0.10)
Host is up (0.00026s latency).

PORT     STATE    SERVICE
22/tcp   open     ssh
80/tcp   filtered http
443/tcp  open     https
8080/tcp filtered http-proxy
MAC Address: A2:01:0C:D8:90:DE (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.30 seconds
```

Expected results:

- Port 22: open (SSH)
- Port 433: open (HTTPS)
- Other common ports such as 80 or 8080 should not be accessible



```
nmap -p 80 172.20.0.10

Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-22 11:10 UTC
Nmap scan report for lab-server.lab_c_lab-network (172.20.0.10)
Host is up (0.00025s latency).

PORT   STATE    SERVICE
80/tcp filtered http
MAC Address: E2:F0:41:9C:B9:B4 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

These scanning tests confirm that only authorized services are reachable while all other traffic is denied.

7 - Security Best Practices Implemented

- Complete deny by default policy
- Allow only specific and required traffic
- Preservation of firewall rules across reboots
- Logging and inspection of current rules
- Use of connection state tracking to prevent unnecessary open rules
- Testing from an external system to ensure the firewall is functioning as expected

8 - Conclusion

This lab successfully demonstrated implementation of a host based firewall using iptables. The rules follow a security focused approach where all access is denied unless explicitly allowed. Required services such as SSH and HTTPS were enabled and functional, and all other ports were confirmed to be blocked using nmap scanning. The solution is persistent, self contained, and can be expanded for more complex service configurations in a production environment.