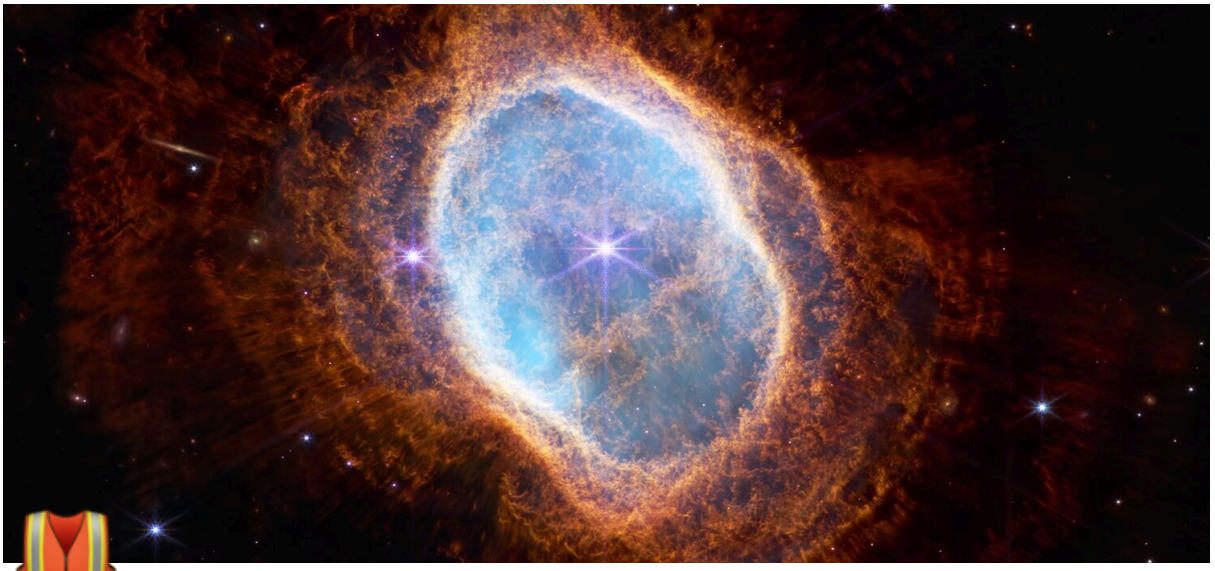

Tutorial Worksheet D

Exercise 1

IPTables is a user-space utility program that allows administrators to configure the Linux
kernel firewall implemented within the Netfilter framework. It defines rules for filtering,
modifying, and forwarding network packets (IPv4) based on tables, chains, and rules. You
earn a Linux-based firewall that you can configure using IPTables. Write IPTables rules
for the following specifications to secure your internal network:

Drops all incoming/forwarded traffic by default (allow outbound only).

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

Block all traffic from IP address 192.114.2.2 to your internal network.

```
iptables -A INPUT -s 192.114.2.2 -j DROP
```

Blocks all traffic from address range 192.114.0.0/16 to your internal network.

```
iptables -A INPUT -s 192.114.0.0/16 -j DROP
```

Allow SSH connection from only one trusted IP address 10.0.0.5.

```
iptables -A INPUT -p tcp --dport 22 -s 10.0.0.5  -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j DROP
```

Limit SYN flood attack.

```
iptables -A INPUT -p tcp --syn -m limit --limit 1/s  -j ACCEPT
```

Block all ICMP echo requests (Ping).

```
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Block all traffic from private IP addresses

```
# Assuming eth0 is the public-facing interface
iptables -A INPUT -s 10.0.0.0/8 -j DROP
iptables -A INPUT -s 172.16.0.0/12 -j DROP
iptables -A INPUT -s 192.168.0.0/16 -j DROP
```

Restrict outgoing traffic to be HTTP and HTTPs only (Whitelist)

```
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -j DROP
```

Block FTP and Telnet access

```
iptables -A INPUT -p tcp --dport 21 -j DROP     # FTP
iptables -A INPUT -p tcp --dport 23 -j DROP     # Telnet
```

Block a specific MAC Address

```
iptables -A INPUT -m mac --mac-source 00:BB:23:4D:EE:6F -j DROP
```

Allow a specific MAC Address

```
iptables -A INPUT -m mac --mac-source 11:22:33:44:55:66 -j ACCEPT
iptables -A INPUT -j DROP
```

Block application-level HTTP flooding

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 100/minute -j ACCEPT
```

Block port scanning

```
iptables -N PortScanning
iptables -A PortScanning -m recent --name Attacker --set -j DROP
iptables -A INPUT -m recent --name Attacker --update --seconds 60 --hitcou
```

Block invalid packets

```
iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
# Using connection tracking module, drop packets that are classified as INVAl
# (corrupted or unexpected packets)
```

Block Tor exit nodes

```
iptables -A INPUT -m set --match-set tor-exit-nodes src -j DROP

# tor-exit-nodes: a predefined IP set containing exit nodes of the Tor network,
# often used by attackers to anonymize their traffic
```

Exercise 2

A ⇒ Problem:

Allowing **all outbound traffic** from inside allows attackers to exfiltrate data or establish reverse TCP connections.

Issue: All traffic is allowed outbound from inside → data leaks + reverse TCP attacks.

Solution: Restrict outbound traffic to only necessary ports (HTTP, HTTPS, DNS).

```
iptables -A OUTPUT -s 192.168.1.0/24 -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -s 192.168.1.0/24 -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -s 192.168.1.0/24 -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -s 192.168.1.0/24 -p tcp -j DROP
```

B ⇒ Problem:

All inbound traffic from Zone 2 (Z2) to Zone 1 (Z1) is allowed. If Z2 is compromised, attacker can pivot to Z1 freely.

Solution: Allow only specific protocols (e.g., SSH) from Z2 to Z1.

```
iptables -A INPUT -s 131.159.20.0/24 -d 192.168.1.0/24 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -s 131.159.20.0/24 -d 192.168.1.0/24 -p tcp -j DROP
```

C ⇒ Problem:

Inbound SSH is allowed from everywhere, exposing to brute force attacks.

Solution: Drop inbound SSH connections from IPs **not** in the trusted network.

```
iptables -A INPUT -p tcp --dport 22 -s ! 131.159.20.0/24 -j DROP
```

D ⇒ Problem:

All established traffic is allowed without restriction—even if it comes from malicious sessions.

Solution: Restrict established and related traffic only to secure, whitelisted services (HTTP, HTTPS, DNS).

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -p tcp --dport 443 -j ACCEPT
```

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -p udp --dport 53 -j ACCEPT
```

RELATED ⇒ packet that trigger another packet (echo-request trigger echo-res