Lab Manual: Configuring a Firewall using Iptables on a Linux System

Objective:

1. Learn to configure iptables on a Linux system.
2. Create rules to filter incoming, outgoing, and forwarded traffic.
3. Test the firewall's effectiveness against various types of network attacks.

Prerequisites

1. Basic knowledge of Linux commands.
2. Familiarity with networking concepts such as IP addresses, ports, and protocols.
3. A Linux-based system with iptables installed (commonly found on most distributions).

Equipment Required

1. Linux-based system (Ubuntu/Debian/CentOS recommended).
2. Internet access.
3. Another system or virtual machine for generating network traffic and simulating attacks.

Lab Setup

1. Verify iptables Installation:

- Open the terminal on your Linux system and run the following command to check if iptables is installed:
  `sudo iptables --version`
  If not installed, use the following command:
  `sudo apt-get install iptables -y` (For Ubuntu/Debian)
  `sudo yum install iptables -y` (For CentOS/RHEL)

2. View Current iptables Rules:

- Display all current iptables rules to understand existing firewall rules.
  `sudo iptables -L -v -n`

3. Resetting iptables Rules:

- Clear all existing rules to start fresh.
  `sudo iptables -F; sudo iptables -X; sudo iptables -t nat -F; sudo iptables -t nat -X`

Section 1: Configuring Basic iptables Rules

1. 1. Set Default Policies:

Set default policies to drop all incoming and forwarding traffic but allow outgoing traffic:
`sudo iptables -P INPUT DROP`

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

2. 2. Allow Loopback Interface:

Enable traffic on the loopback interface to allow local connections:
`sudo iptables -A INPUT -i lo -j ACCEPT`
`sudo iptables -A OUTPUT -o lo -j ACCEPT`

3. 3. Allow Established and Related Connections:

Permit traffic for established and related connections to maintain connections without interruptions:
`sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`

4. Allow SSH Access:

Allow SSH access on port 22 to ensure remote connection to the system:
`sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT`

Section 2: Creating Rules to Filter Traffic

5. 1. Allow HTTP and HTTPS Traffic:

Permit incoming traffic on ports 80 (HTTP) and 443 (HTTPS):
`sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
`sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT`

6. 2. Block ICMP (Ping) Requests:

Block all incoming ICMP packets to prevent ping attacks:
`sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP`

7. 3. Limit SSH Connections:

Limit SSH connections to prevent brute-force attacks:
`sudo iptables -A INPUT -p tcp --dport 22 -m limit --limit 3/min --limit-burst 3 -j ACCEPT`

Section 3: Testing the Firewall Rules

8. 1. Testing HTTP and HTTPS Access:

From a different machine or browser, access the Linux system's web server. Verify HTTP and HTTPS traffic.

9. 2. Testing SSH Connection:

Attempt an SSH connection to the Linux system, checking rate limiting for multiple rapid attempts.

10. 3. Testing ICMP Blocking:

Ping the Linux system from another machine to verify ICMP echo requests are blocked.

11. 4. Testing with Network Scanning Tools:

Use tools like `nmap` from another system to scan open ports on the Linux machine:
`nmap -sS <IP_Address>`

Section 4: Testing against Network Attacks (Optional)
12. 1. Simulating SYN Flood Attack:

Use `hping3` or a similar tool:
`hping3 -S -p 80 -i u1000 <IP_Address>`

13. 2. Mitigating DoS Attacks:

Limit new connections to mitigate DoS attacks:
`sudo iptables -A INPUT -p tcp --syn -m limit --limit 1/s -j ACCEPT`

14. 3. Testing Port Scanning Prevention:

Add a rule to detect and drop port scanning attempts:
`sudo iptables -N PORTSCAN; sudo iptables -A PORTSCAN -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit-burst 2 -j RETURN; sudo iptables -A PORTSCAN -j DROP; sudo iptables -A INPUT -p tcp --tcp-flags SYN,ACK,FIN,RST RST -j PORTSCAN`

Section 5: Saving and Verifying Configuration
15. 1. Save iptables Rules:

Make sure the rules persist after a system reboot:
`sudo iptables-save | sudo tee /etc/iptables/rules.v4`

16. 2. Reload and Verify:

Reboot the system and verify access restrictions according to the previous tests.

Tasks

1. **Allow Traffic on a Specific Port**:
    - Write an iptables rule to allow traffic on port 8080 (commonly used for web applications).

    - Test by starting a web server on port 8080 and accessing it from another machine.
2. **Block Traffic from a Specific IP**:
    - Add a rule to block all incoming traffic from a specific IP address (e.g., `192.168.1.100`).
    - Verify by pinging from that IP or attempting to connect.
3. **Create a Custom Chain for Logging**:
    - Create a new chain called `LOGGING`.
    - Set up a rule to log dropped packets using this chain.
    - Test by generating some traffic that will be dropped and observe the logs (`/var/log/syslog` or `/var/log/messages`).
4. **Restrict Outgoing SSH Connections**:
    - Configure iptables to only allow outgoing SSH connections to a specific IP address (e.g., `192.168.1.10`).
    - Attempt SSH connections to allowed and blocked IPs to test.
5. **Rate Limit Incoming HTTP Requests**:
    - Implement rate limiting on port 80 to only allow 10 requests per minute.
    - Test this by using a web browser or `curl` to make rapid requests and observe the behavior.

Lab Report

Document each command executed, the observations for each test, and any additional rules you think could enhance security.