

Experiment 10: Implementing Firewall Using Iptables

Bhuvi Ghosh
60009210191

Aim:

The objective of this experiment is to gain hands-on experience in implementing a firewall using iptables on a Linux-based system. Participants will learn how to configure iptables rules to control incoming and outgoing network traffic, enhance network security, and protect the system from unauthorized access.

Equipment/Requirements:

Linux-based system (e.g., Ubuntu, CentOS)
Access to the terminal or SSH for command-line interface
Basic understanding of networking concepts
Administrative privileges (sudo access)

Background:

Iptables is a powerful tool for configuring the Linux kernel firewall, which controls network traffic on a system. It can be used to set up rules to allow or block traffic based on various criteria such as source IP, destination IP, ports, and protocols.

Procedures:

Step 1: Verify iptables Installation

Ensure that iptables is installed on your system. If not, install it using the package manager appropriate for your Linux distribution (e.g., apt, yum).

```
sudo apt-get install iptables
```

Step 2: View Current iptables Rules

Check the current iptables rules on your system using the following command

```
sudo iptables -L
```

Step 3: Define Firewall Rules

Create and implement firewall rules based on your network security requirements.

For example:

```
# Allow incoming SSH (replace <your_ip> with your actual IP address)
sudo iptables -A INPUT -p tcp --dport 22 -s <your_ip> -j ACCEPT
```

```
# Allow incoming HTTP traffic
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
# Drop all other incoming traffic
sudo iptables -A INPUT -j DROP
```

Step 4: Save iptables Rules

Save the iptables rules to ensure they persist after a system reboot.

```
sudo service iptables save
```

Step 5: Test Firewall Rules

Test the firewall rules by attempting to access the system from a remote location. Ensure that allowed traffic is permitted, and blocked traffic is denied.

Step 6: Monitor Firewall Activity

Use the following command to monitor live firewall activity:

```
sudo iptables -L -v
```

Expected Outcomes:

```
[twt @ barnan_arnav]$sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination
ACCEPT     all  --  anywhere              anywhere
ACCEPT     tcp  --  anywhere              anywhere             multiport dports ssh,telnet,domain
DROP       all  --  192.168.1.10          anywhere

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination
ACCEPT     all  --  anywhere              anywhere
ACCEPT     tcp  --  anywhere              anywhere             multiport sports ssh,telnet,domain
DROP       all  --  anywhere              192.168.1.10
[twt @ barnan_arnav]$
```

```
[tw@ barman_arnav]$ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.045 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.151 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.053 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.045 ms
^C
--- 127.0.0.1 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10249ms
rtt min/avg/max/mdev = 0.029/0.060/0.151/0.029 ms
[tw@ barman_arnav]$
```

Conclusion:

This experiment provides valuable hands-on experience in configuring and managing a firewall using iptables. Participants should now have a better understanding of how to enhance network security by controlling incoming and outgoing traffic on a Linux-based system.