Lab Project - 9

Objective:

Linux environment management lab

DURATION: 4 - 5 Hourse

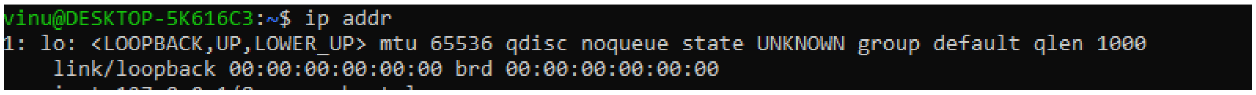
Pre-requisites

1. A Linux machine (Ubuntu, CentOS, Debian, or any other distribution).
2. Administrative privileges (root or sudo).
3. Basic knowledge of network services and firewall configuration.

Lab Task Steps

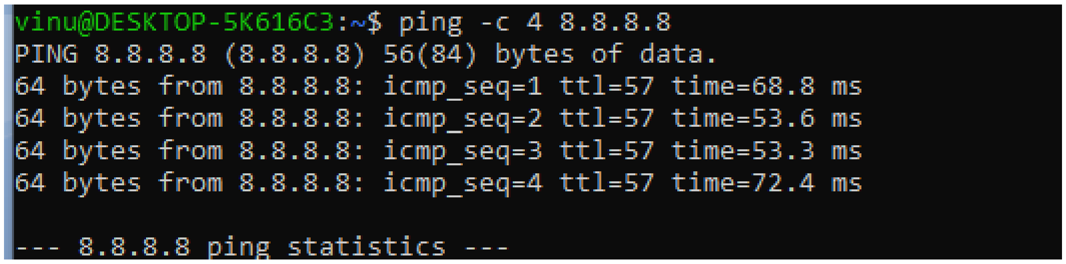
Part 1: Setup and Verify Networking Configuration

1. **Check network interfaces:**
   - Run `ip addr` or `ifconfig` to identify your network interfaces. Typically, your interface may be `eth0`, `ens33`, `enp0s3`, etc.



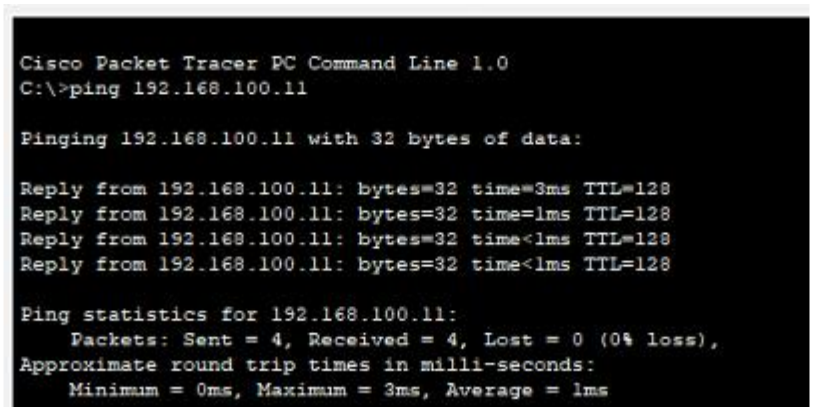
```
vinu@DESKTOP-5K616C3:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

2. **Verify connectivity:**
   - Make sure you can reach the internet and other machines in your local network:
     - `ping 8.8.8.8` (Google DNS server)



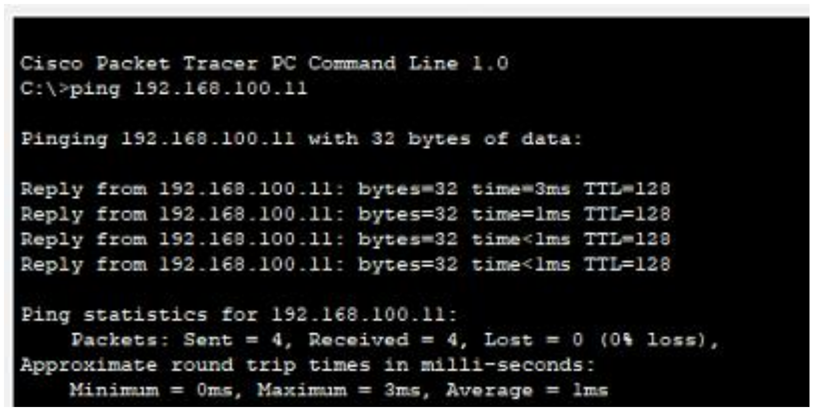
```
vinu@DESKTOP-5K616C3:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=68.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=53.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=57 time=53.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=57 time=72.4 ms

--- 8.8.8.8 ping statistics ---
```

     - `ping <your_gateway_ip>`



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.100.11

Pinging 192.168.100.11 with 32 bytes of data:

Reply from 192.168.100.11: bytes=32 time=3ms TTL=128
Reply from 192.168.100.11: bytes=32 time=1ms TTL=128
Reply from 192.168.100.11: bytes=32 time<1ms TTL=128
Reply from 192.168.100.11: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.100.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

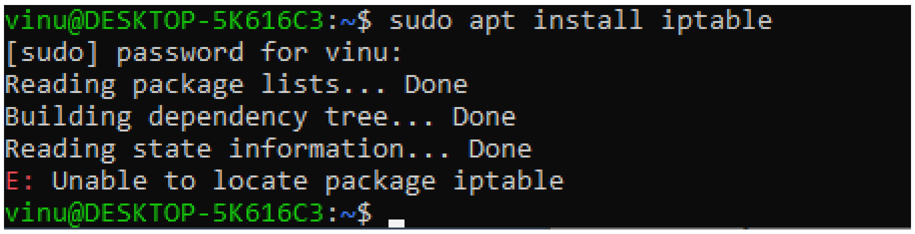
     - `ping <another_machine_in_the_network>`



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.100.11

Pinging 192.168.100.11 with 32 bytes of data:

Reply from 192.168.100.11: bytes=32 time=3ms TTL=128
Reply from 192.168.100.11: bytes=32 time=1ms TTL=128
Reply from 192.168.100.11: bytes=32 time<1ms TTL=128
Reply from 192.168.100.11: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.100.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

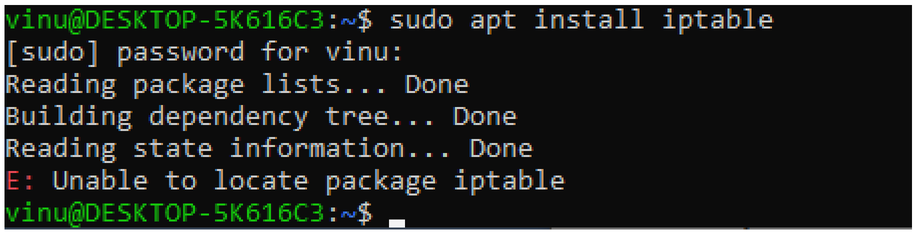
3.) Confirm that `iptables` is installed:

- Run `iptables --version` to ensure that `iptables` is installed. If not, install it:
  - `sudo apt install iptables` (on Ubuntu/Debian)



```
vinu@DESKTOP-5K616C3:~$ sudo apt install iptable
[sudo] password for vinu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package iptable
vinu@DESKTOP-5K616C3:~$
```

  - `sudo yum install iptables` (on CentOS/RHEL)

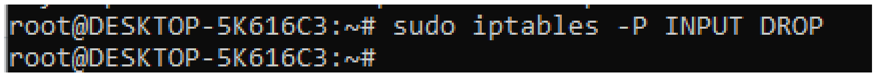


```
vinu@DESKTOP-5K616C3:~$ sudo apt install iptable
[sudo] password for vinu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package iptable
vinu@DESKTOP-5K616C3:~$
```

Part 2: Configure a Basic Firewall

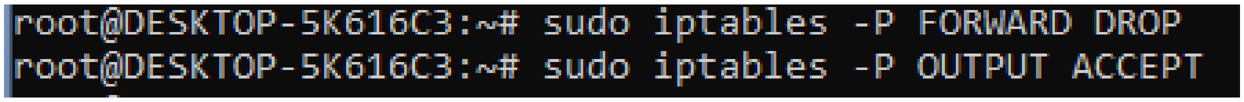
1. **Set default policies:**
   - By default, we want to deny all incoming traffic and allow outgoing traffic. This can be set as follows:

```
2. sudo iptables -P INPUT DROP       # Block all incoming
   traffic
```



```
root@DESKTOP-5K616C3:~# sudo iptables -P INPUT DROP
root@DESKTOP-5K616C3:~#
```

```
3. sudo iptables -P FORWARD DROP     # Block forwarding
   sudo iptables -P OUTPUT ACCEPT    # Allow all outgoing
   traffic
```

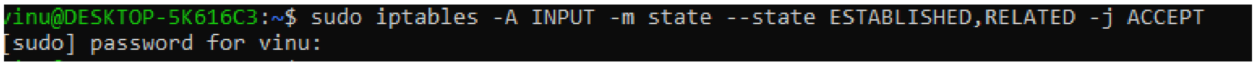


```
root@DESKTOP-5K616C3:~# sudo iptables -P FORWARD DROP
root@DESKTOP-5K616C3:~# sudo iptables -P OUTPUT ACCEPT
```

2.) Allow established connections:

- To maintain established connections (like active SSH sessions), you need to allow the related and established connections:

```
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED
-j ACCEPT
```

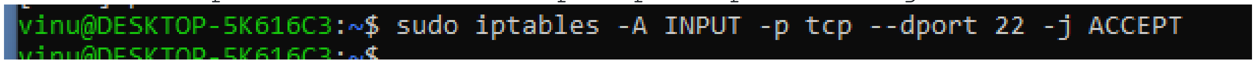


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
[sudo] password for vinu:
```

3.)Allow SSH access (port 22):

- You need to allow SSH traffic to connect remotely to the system. This is done by allowing inbound traffic on port 22:

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

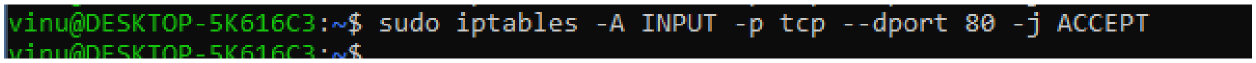


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

4.)Allow HTTP/HTTPS (ports 80, 443):

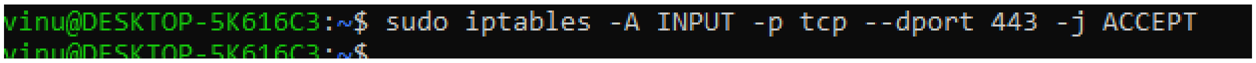
If your server will serve web pages, open HTTP and HTTPS ports:

1.
   ```
   sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT   #
   Allow HTTP
   ```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

```
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT   #
Allow HTTPS
```

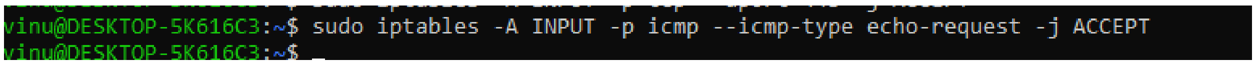


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

5.) Allow ICMP (ping) requests:

- You can allow ICMP traffic for ping functionality:

```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j
ACCEPT
```

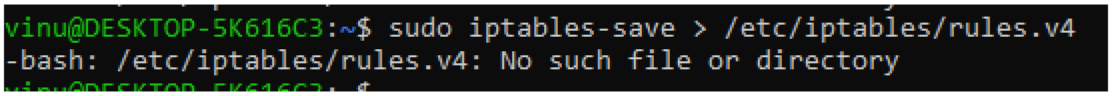


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
vinu@DESKTOP-5K616C3:~$ _
```

6.)Save your rules:

- To ensure the firewall rules persist after reboot, save the iptables rules:

```
sudo iptables-save > /etc/iptables/rules.v4   # For
Debian/Ubuntu
```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables-save > /etc/iptables/rules.v4
-bash: /etc/iptables/rules.v4: No such file or directory
```

```
sudo service iptables save                          # For
CentOS/RHEL
```

Part 3: Test Firewall Rules

1. **Test SSH connection:**
   - From another machine, try to SSH into your Linux server. It should work if port 22 is open.
   - `ssh user@<server-ip>`

- **"Connection refused"**
  - Ensure SSH is installed and running on the server:

    ```sh
    sudo systemctl status ssh
    ```

    If it's inactive, start it:

    ```sh
    sudo systemctl start ssh
    ```

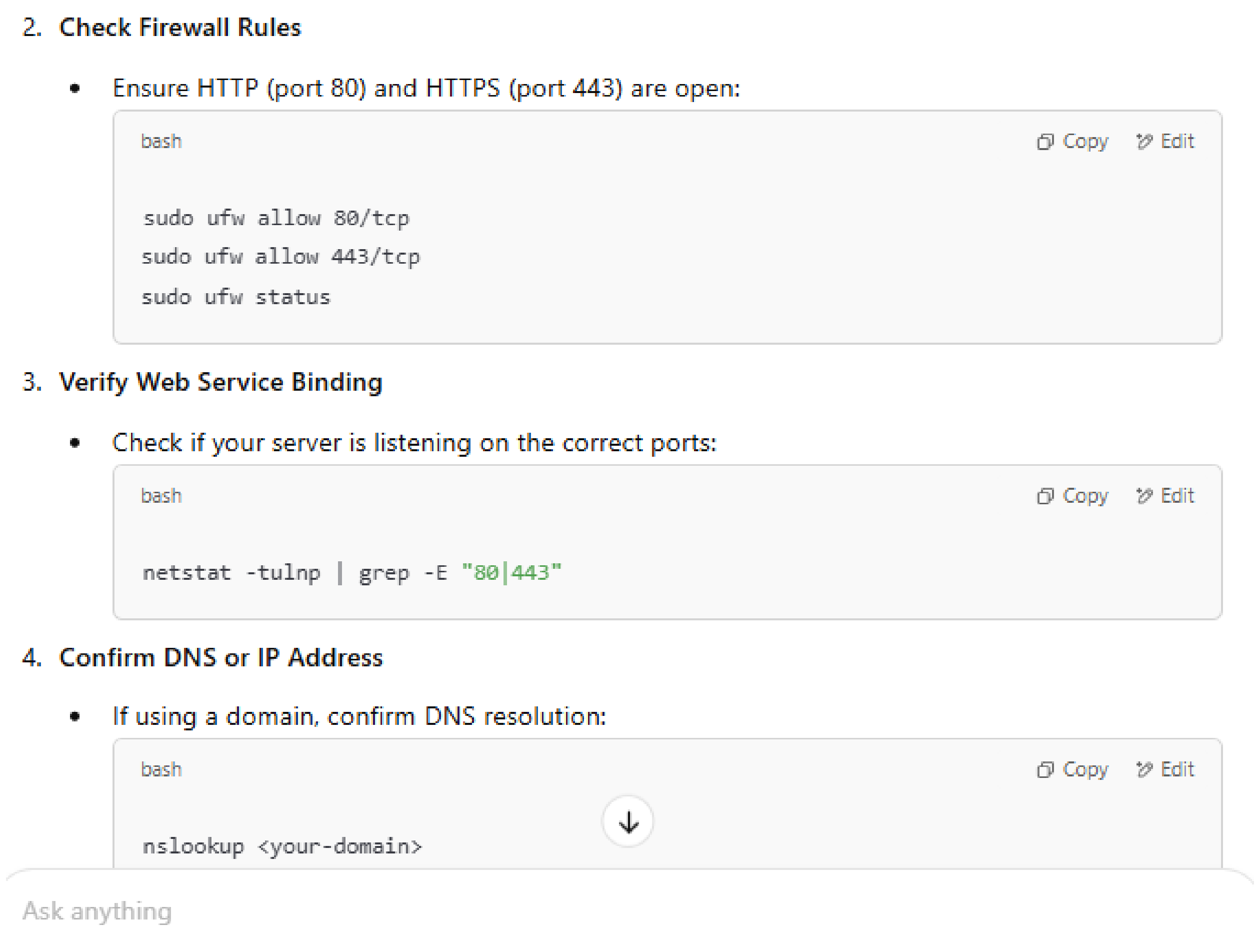
- **"Permission denied (publickey, password)"**
  - Check if the correct SSH key is being used:

    ```sh
    ssh -v user@<server-ip>
    ```

  - If using key authentication, make sure your `~/.ssh/authorized_keys` file contains the correct

2.) **Test web server access (if applicable):**

- If you allowed HTTP/HTTPS, try accessing the server from a browser:
  - `http://<server-ip>` for HTTP
  - `https://<server-ip>` for HTTPS (if SSL is configured)



2. **Check Firewall Rules**
   - Ensure HTTP (port 80) and HTTPS (port 443) are open:

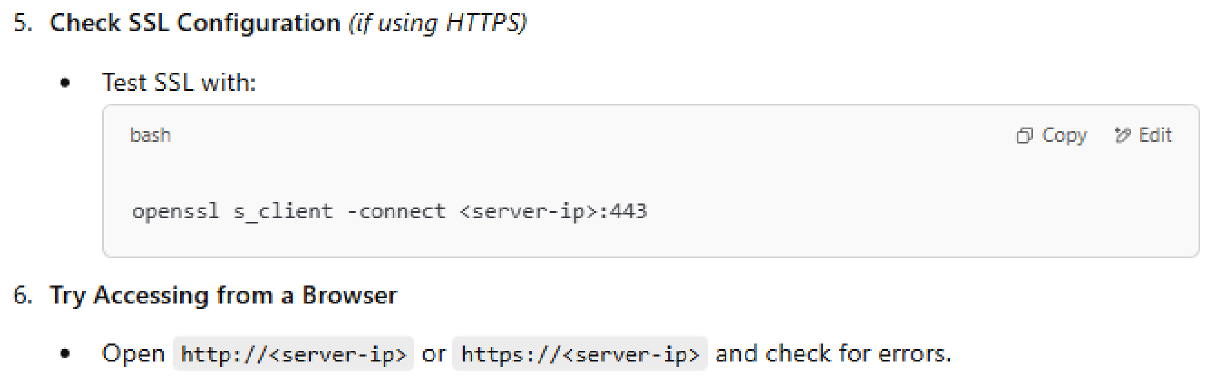
```bash
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw status
```

3. **Verify Web Service Binding**
   - Check if your server is listening on the correct ports:

```bash
netstat -tulnp | grep -E "80|443"
```

4. **Confirm DNS or IP Address**
   - If using a domain, confirm DNS resolution:

```bash
nslookup <your-domain>
```


5. **Check SSL Configuration** *(if using HTTPS)*
   - Test SSL with:

```bash
openssl s_client -connect <server-ip>:443
```

6. **Try Accessing from a Browser**
   - Open `http://<server-ip>` or `https://<server-ip>` and check for errors.

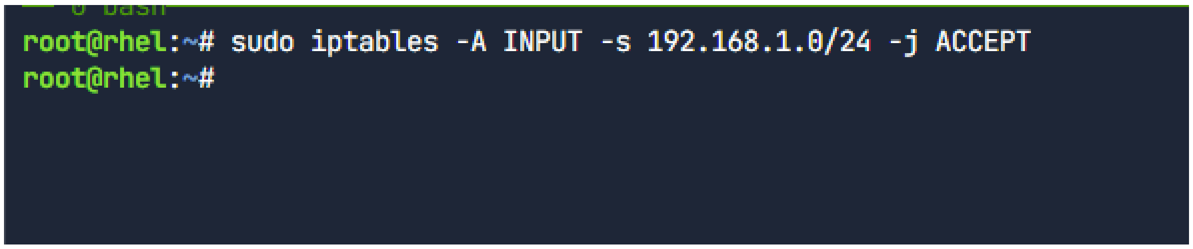
3.)Test ping:

- From another machine, ping the server to ensure ICMP traffic is allowed.
  - `ping <server-ip>`

Part 4: Enhance Security (Optional)

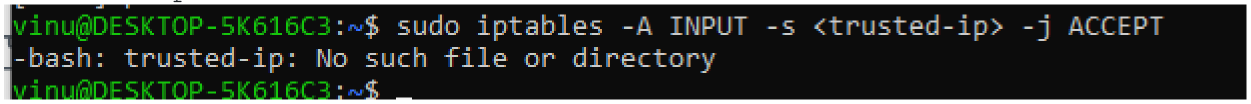
1. **Block all incoming traffic by default, but allow specific IP ranges:**
   - You can restrict access to the server to specific IP ranges, for example:

```
2. sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT    # Allow local network
```



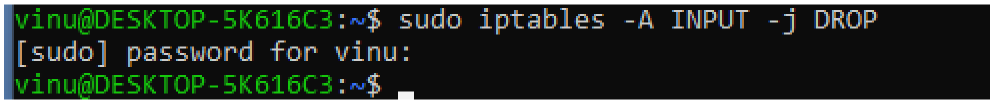
```
root@rhel:~# sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
root@rhel:~#
```

```
3. sudo iptables -A INPUT -s <trusted-ip> -j ACCEPT        # Allow a specific trusted IP
```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -s <trusted-ip> -j ACCEPT
-bash: trusted-ip: No such file or directory
vinu@DESKTOP-5K616C3:~$ _
```

```
sudo iptables -A INPUT -j DROP                             # Block everything else
```

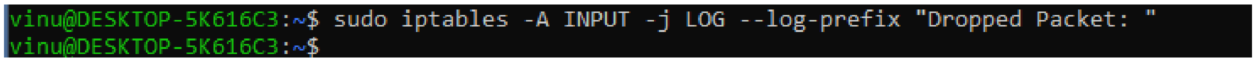


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -j DROP
[sudo] password for vinu:
vinu@DESKTOP-5K616C3:~$
```

2.)Log dropped packets:

- You can enable logging to monitor dropped packets:

```
sudo iptables -A INPUT -j LOG --log-prefix "Dropped Packet: 
"
```

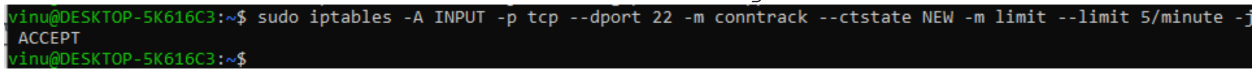


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -j LOG --log-prefix "Dropped Packet: "
vinu@DESKTOP-5K616C3:~$
```

3.)Rate limiting (Optional):

- To prevent brute-force attacks on services like SSH, you can limit the number of connections:

```
sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --
ctstate NEW -m limit --limit 5/minute -j ACCEPT
```

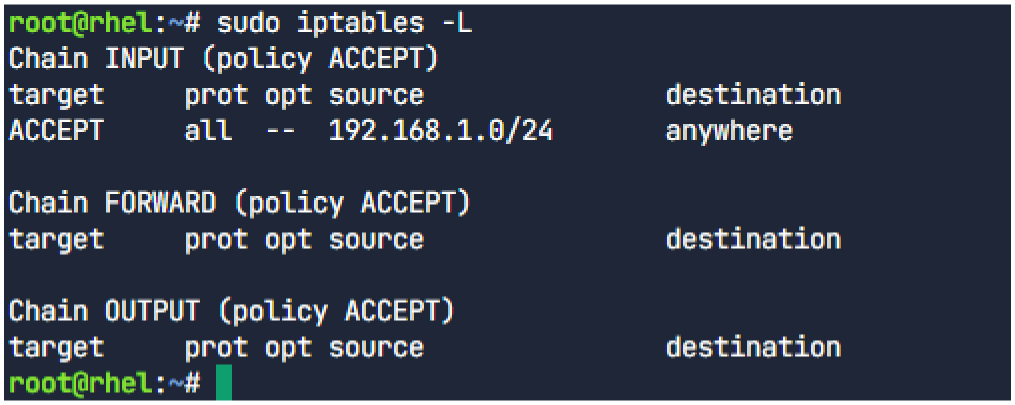


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m limit --limit 5/minute -j
 ACCEPT
vinu@DESKTOP-5K616C3:~$
```

Part 5: Monitor and Manage Firewall

1. **View current firewall rules:**
   - Check the current rules using:

```
sudo iptables -L
```

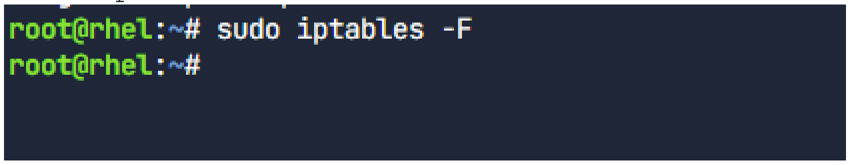


```
root@rhel:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  192.168.1.0/24       anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@rhel:~#
```

2.)Flush all rules (reset firewall):

- If you want to reset the firewall to its default state (deny all traffic):

```
sudo iptables -F
```

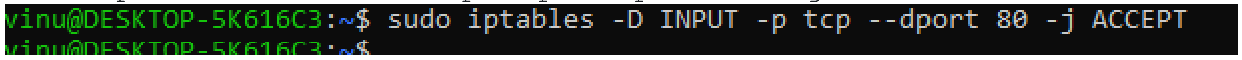


```
root@rhel:~# sudo iptables -F
root@rhel:~#
```

3.)Delete a specific rule:

- If you need to delete a specific rule, use:

```
sudo iptables -D INPUT -p tcp --dport 80 -j ACCEPT
```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables -D INPUT -p tcp --dport 80 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

Lab Task: Configuring Port Blocking, IP Allowance, IP Range, and Protocol Allowance using `iptables`

Objective:

- Block/allow specific ports.
- Permit traffic from specific IP addresses.
- Allow traffic only from certain IP ranges.
- Allow/deny specific network protocols.

Prerequisites:

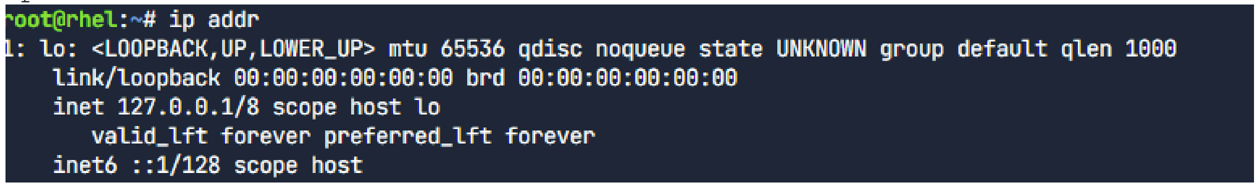
1. A Linux machine (Ubuntu, CentOS, or any distribution).
2. Administrative privileges (root or sudo).
3. Basic networking knowledge and the ability to use the terminal.

Steps for the Lab Task:

1. Verify Current Networking and Firewall Configuration

1. **Check IP addresses and network interfaces:** Run the following command to get a list of network interfaces and their IP addresses.

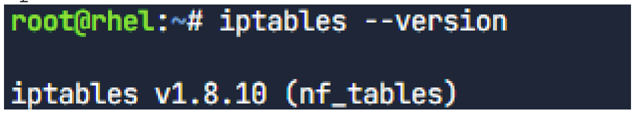
```
ip addr
```



```
root@rhel:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

2.) Verify if `iptables` is installed: Check if `iptables` is available on your system.

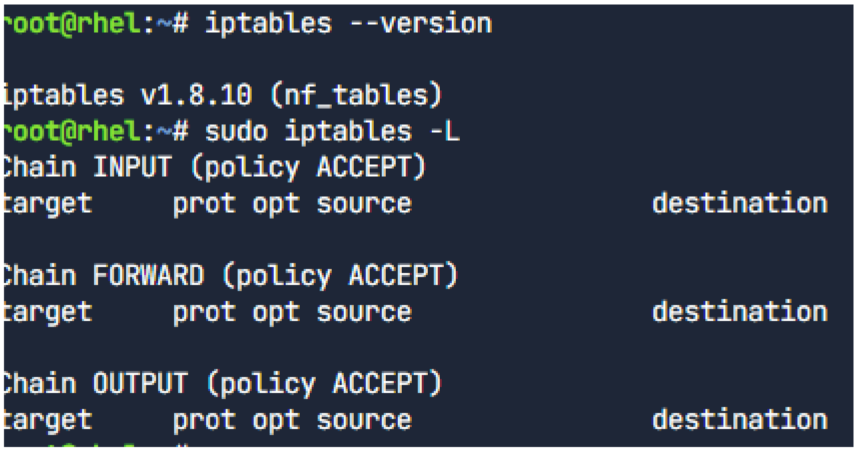
```
iptables –version
```



```
root@rhel:~# iptables --version

iptables v1.8.10 (nf_tables)
```

3.)Check current `iptables` rules: List all existing rules in the firewall.

```
sudo iptables -L
```



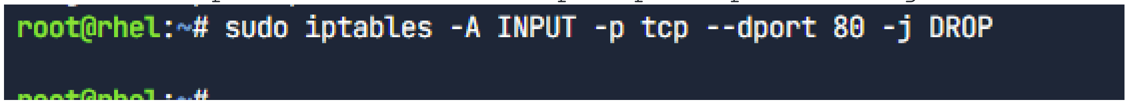
```
oot@rhel:~# iptables --version

iptables v1.8.10 (nf_tables)
oot@rhel:~# sudo iptables -L
hain INPUT (policy ACCEPT)
arget     prot opt source               destination

hain FORWARD (policy ACCEPT)
arget     prot opt source               destination

hain OUTPUT (policy ACCEPT)
arget     prot opt source               destination
```

2. Block Specific Ports

You can block incoming traffic on specific ports using `iptables`.

1. **Block incoming traffic on port 80 (HTTP):**
   - This will block all HTTP traffic from reaching your server.

```
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
```

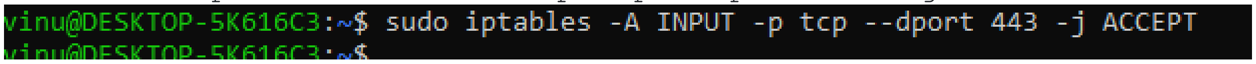


```
root@rhel:~# sudo iptables -A INPUT -p tcp --dport 80 -j DROP
```

2.)Block incoming traffic on port 443 (HTTPS):

- This will block all HTTPS traffic.

```
sudo iptables -A INPUT -p tcp --dport 443 -j DROP
```

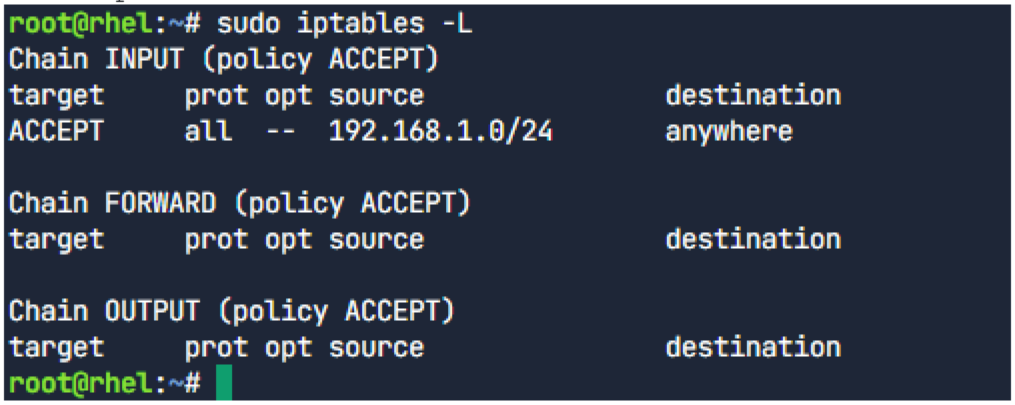


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

3.)Verify the changes:

- List the rules to ensure that the ports are blocked.

```
sudo iptables -L
```

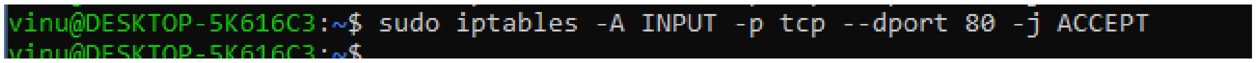


```
root@rhel:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  192.168.1.0/24       anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@rhel:~#
```

4.) Test port blocking:

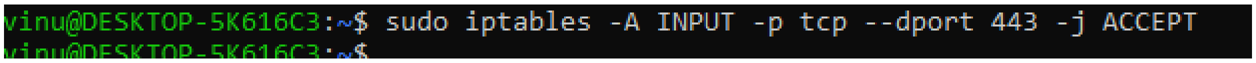
- From a different machine, try to access the blocked port using `curl` or a browser.
- You should not be able to reach the server on ports 80 or 443.

2.
```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT    # Allow HTTP
```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

```
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT  # Allow HTTPS
```



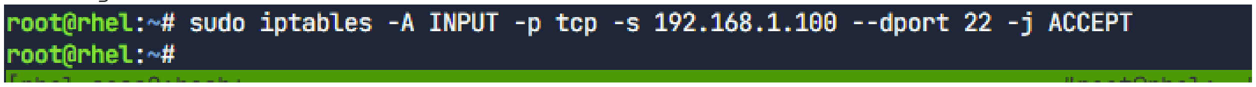
```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
vinu@DESKTOP-5K616C3:~$
```

3. Allow Traffic from Specific IP Addresses

You can allow or deny traffic based on specific IP addresses.

1. **Allow SSH (port 22) only from a specific IP address (e.g., `192.168.1.100`):**

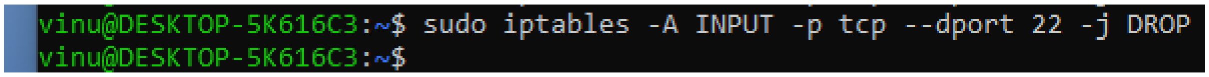
```
sudo iptables -A INPUT -p tcp -s 192.168.1.100 --dport 22 -
j ACCEPT
```



```
root@rhel:~# sudo iptables -A INPUT -p tcp -s 192.168.1.100 --dport 22 -j ACCEPT
root@rhel:~#
```

4.)Block SSH access from all other IP addresses:

```
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```

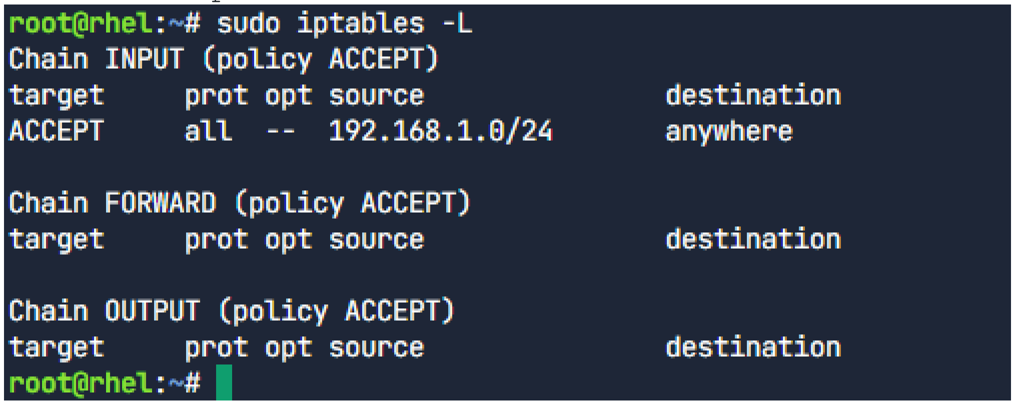


```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
vinu@DESKTOP-5K616C3:~$
```

5.)Verify the changes:

- List the rules again to confirm the changes.

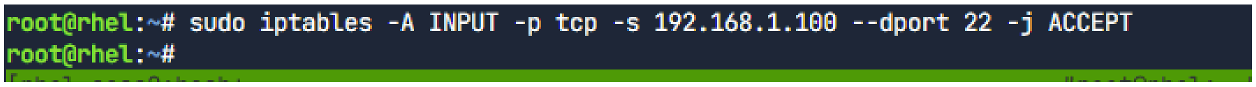
```
sudo iptables -L
```



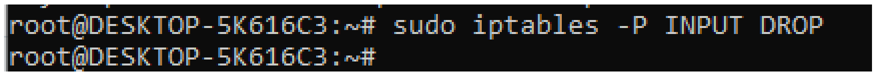
```
root@rhel:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  192.168.1.0/24       anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@rhel:~#
```

Test the configuration:

- Try to SSH into the server from `192.168.1.100` — it should work.
- Try from any other IP — it should be blocked.



```
root@rhel:~# sudo iptables -A INPUT -p tcp -s 192.168.1.100 --dport 22 -j ACCEPT
root@rhel:~#
```



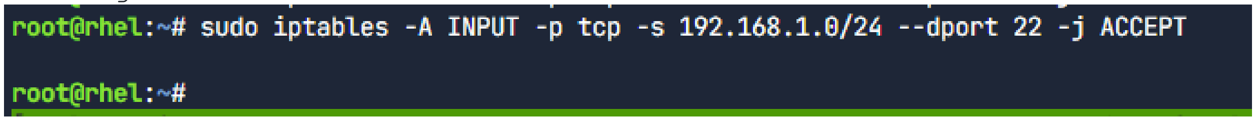
```
root@DESKTOP-5K616C3:~# sudo iptables -P INPUT DROP
root@DESKTOP-5K616C3:~#
```

4. Allow Traffic from a Specific IP Range

You can also allow traffic from a specific range of IPs. For example, if you want to allow access to your server from a range of IP addresses within the `192.168.1.0/24` subnet:

1. **Allow traffic from the IP range `192.168.1.0/24`:**

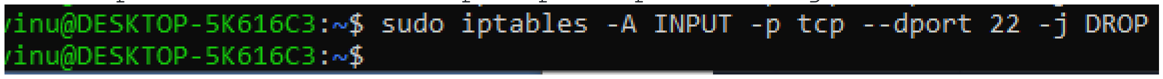
   ```
   sudo iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 22
   -j ACCEPT
   ```



```
root@rhel:~# sudo iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 22 -j ACCEPT

root@rhel:~#
```

2.)Block traffic from all other IP ranges:

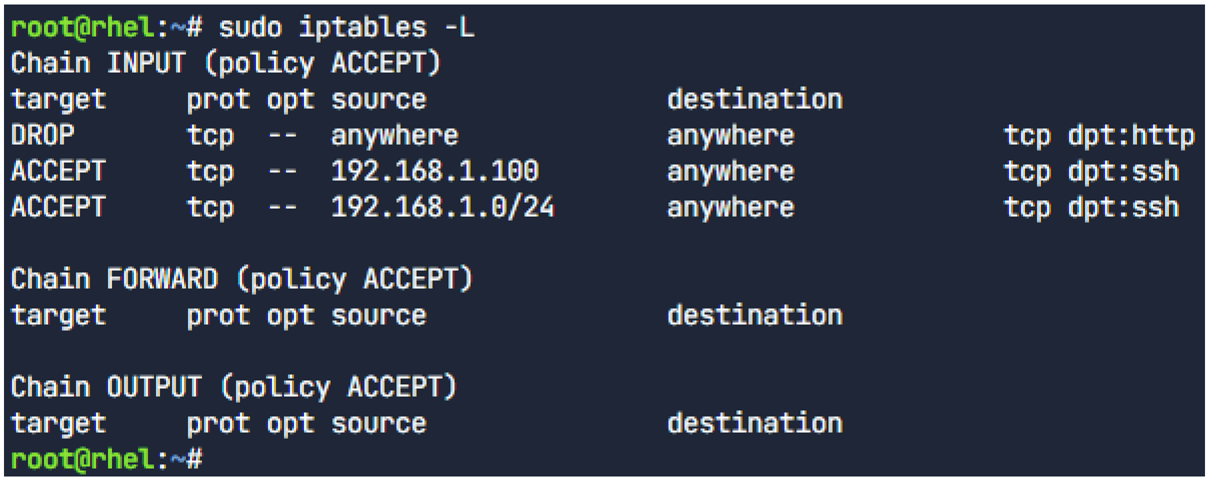
```
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
vinu@DESKTOP-5K616C3:~$
```

3.)Verify the rules:

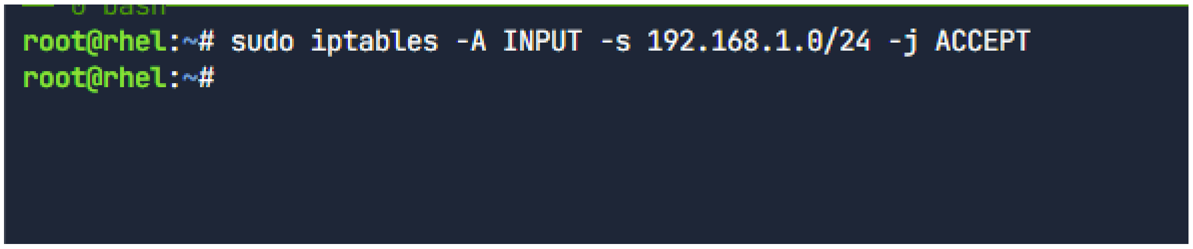
```
sudo iptables -L
```



```
root@rhel:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere             tcp dpt:http
ACCEPT     tcp  --  192.168.1.100        anywhere             tcp dpt:ssh
ACCEPT     tcp  --  192.168.1.0/24       anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@rhel:~#
```

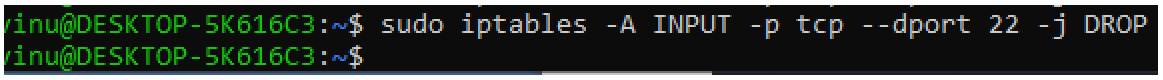
4.)Test the configuration:

- Try accessing the server from an IP within the `192.168.1.0/24` range — it should work.



```
root@rhel:~# sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
root@rhel:~#
```

- Try accessing from an outside range — it should be blocked.



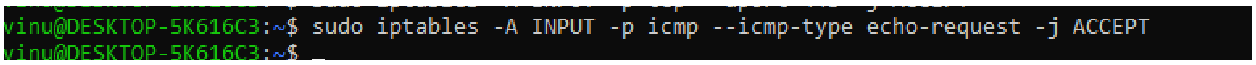
```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
vinu@DESKTOP-5K616C3:~$
```

5. Allow Specific Protocols (TCP, UDP, ICMP)

You can allow or block specific network protocols (e.g., TCP, UDP, ICMP).

1. **Allow all incoming ICMP (Ping) requests:**

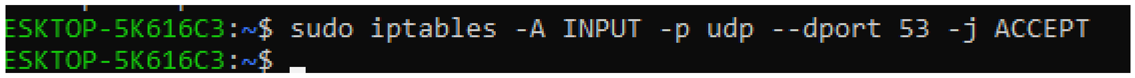
```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j
ACCEPT
```



```
vinu@DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
vinu@DESKTOP-5K616C3:~$ _
```

2.Allow incoming UDP traffic on port 53 (DNS):

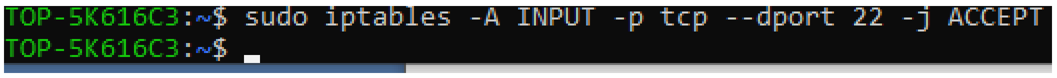
```
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
```



```
ESKTOP-5K616C3:~$ sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
ESKTOP-5K616C3:~$ _
```

3.)Allow incoming TCP traffic on port 22 (SSH):

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

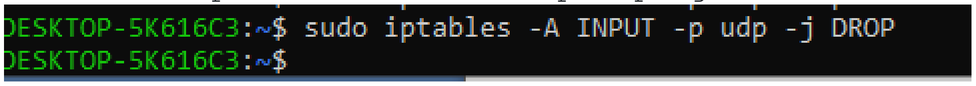


```
TOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
TOP-5K616C3:~$ _
```

4.)Block UDP traffic:

- Block all UDP traffic.

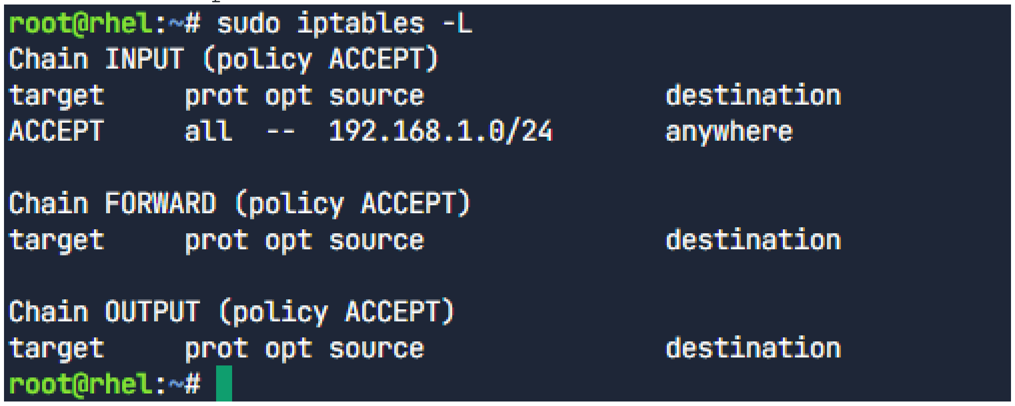
```
sudo iptables -A INPUT -p udp -j DROP
```



```
DESKTOP-5K616C3:~$ sudo iptables -A INPUT -p udp -j DROP
DESKTOP-5K616C3:~$
```

5.)Verify the changes: List the rules again to ensure all protocols and ports are configured as needed.

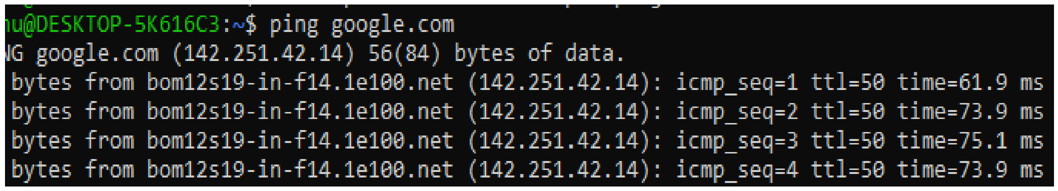
```
sudo iptables -L
```



```
root@rhel:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  192.168.1.0/24       anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@rhel:~#
```

6.)Test the configurations:

- Test ICMP by pinging the server.



```
ıu@DESKTOP-5K616C3:~$ ping google.com
IG google.com (142.251.42.14) 56(84) bytes of data.
 bytes from bom12s19-in-f14.1e100.net (142.251.42.14): icmp_seq=1 ttl=50 time=61.9 ms
 bytes from bom12s19-in-f14.1e100.net (142.251.42.14): icmp_seq=2 ttl=50 time=73.9 ms
 bytes from bom12s19-in-f14.1e100.net (142.251.42.14): icmp_seq=3 ttl=50 time=75.1 ms
 bytes from bom12s19-in-f14.1e100.net (142.251.42.14): icmp_seq=4 ttl=50 time=73.9 ms
```

- Test UDP and TCP services using tools like `nc`, `ping`, or `curl` to ensure proper functionality.

Using Curl

bash

```bash
curl -v http://<hostname_or_IP>:<port>
```

- Example:

bash

```bash
curl -v http://192.168.1.1:80
```

This tests if an HTTP service is running on port 80.

Using Ping

For basic reachability, use:

bash

```bash
ping <hostname_or_IP>
```

Example:

bash

```bash
ping 192.168.1.1
```

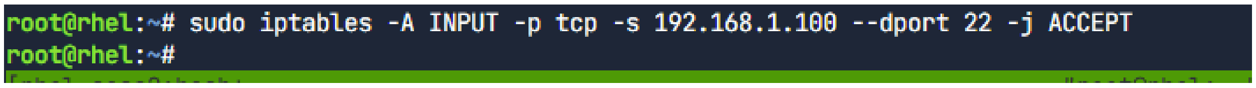
However, `ping` only tests ICMP and not TCP/UDP service availability.

6. Allow Specific IP and Port Combinations

You can also allow specific combinations of IP and port.

1. **Allow traffic from `192.168.1.100` to port 22 (SSH):**

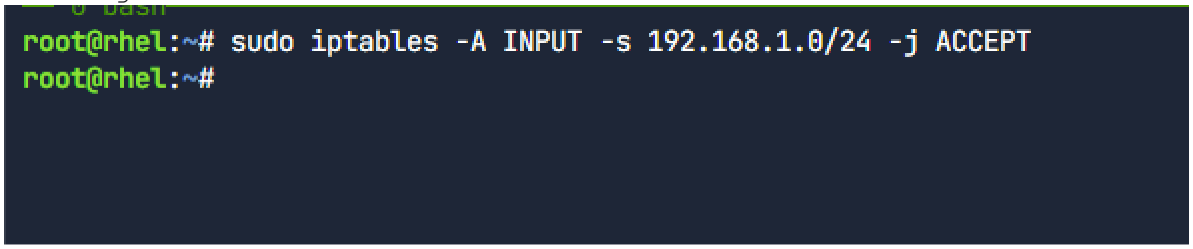
```
sudo iptables -A INPUT -p tcp -s 192.168.1.100 --dport 22 -j ACCEPT
```



```
root@rhel:~# sudo iptables -A INPUT -p tcp -s 192.168.1.100 --dport 22 -j ACCEPT
root@rhel:~#
```

2.Allow traffic from `192.168.1.0/24` to port 80 (HTTP):

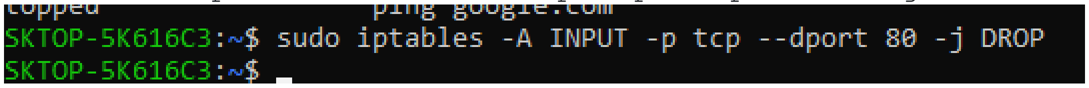
```
sudo iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 80 -j ACCEPT
```



```
root@rhel:~# sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
root@rhel:~#
```

3.)Block all other IP addresses from accessing port 80:

```
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
```



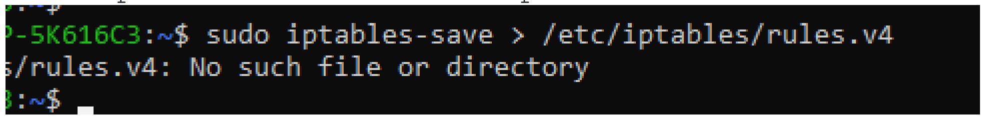
```
SKTOP-5K616C3:~$ sudo iptables -A INPUT -p tcp --dport 80 -j DROP
SKTOP-5K616C3:~$
```

7. Save and Make `iptables` Rules Persistent

Once you have configured your firewall rules, make them persistent across reboots.

1. **On Debian/Ubuntu:** Save the rules to a file to persist them across reboots.

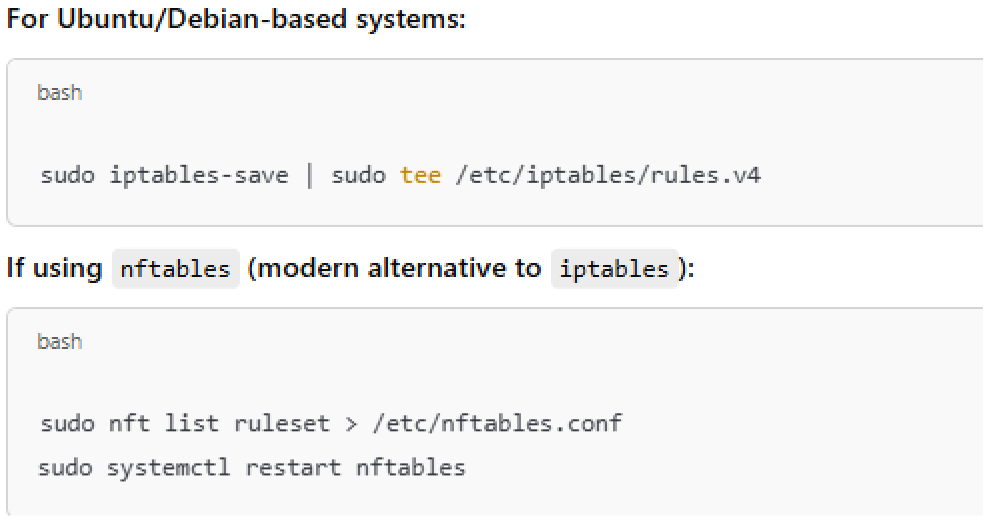
```
sudo iptables-save > /etc/iptables/rules.v4
```



```
-5K616C3:~$ sudo iptables-save > /etc/iptables/rules.v4
/rules.v4: No such file or directory
:~$
```

2.)On CentOS/RHEL:

```
sudo service iptables save
```



For Ubuntu/Debian-based systems:

```bash
sudo iptables-save | sudo tee /etc/iptables/rules.v4
```

If using `nftables` (modern alternative to `iptables`):

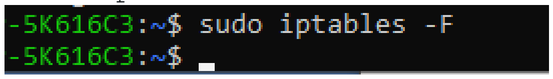
```bash
sudo nft list ruleset > /etc/nftables.conf
sudo systemctl restart nftables
```

8. Flush All Rules and Reset Firewall

If you want to reset the firewall to a clean state, you can flush all existing rules.

1. **Flush all `iptables` rules:**

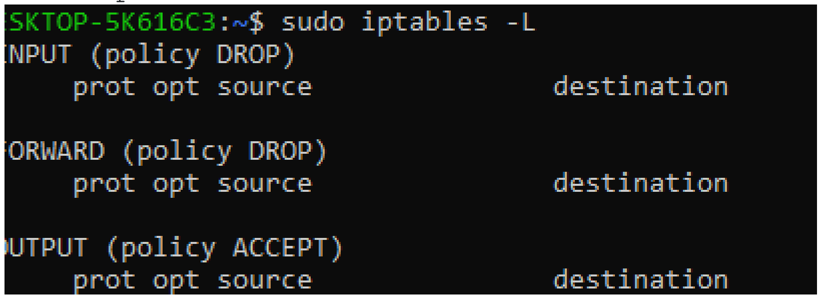
   ```
   sudo iptables -F
   ```

   ```
   -5K616C3:~$ sudo iptables -F
   -5K616C3:~$
   ```

2. **Verify that all rules are removed:**

   ```
   sudo iptables -L
   ```

   ```
   SKTOP-5K616C3:~$ sudo iptables -L
   NPUT (policy DROP)
       prot opt source               destination

   ORWARD (policy DROP)
       prot opt source               destination

   UTPUT (policy ACCEPT)
       prot opt source               destination
   ```