IP SPOFFING PREVENTION

Bakka Ramyasree

1. Set Up the Environment

On Kali Machine (Attacker):
Install necessary tools:

```
sudo apt update
sudo apt install hping3 wireshark python3-pip
pip3 install scapy
```

On Ubuntu Machine (Target/Firewall):
Install required tools:

```
sudo apt update
sudo apt install iptables wireshark
```

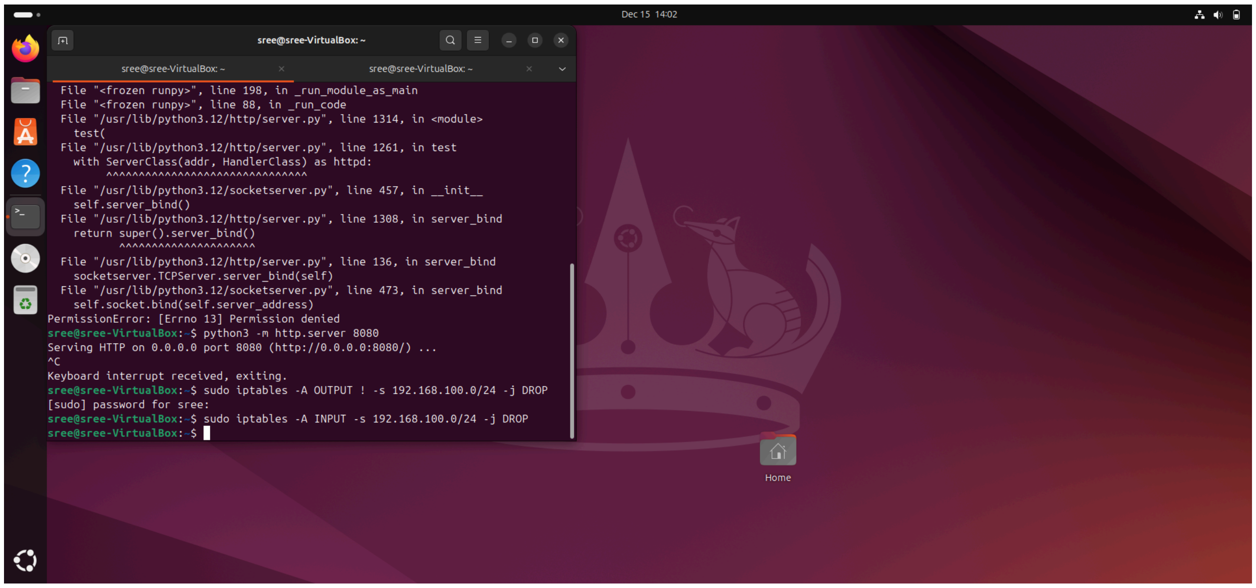
---

2. Configure Egress and Ingress Filtering on Ubuntu

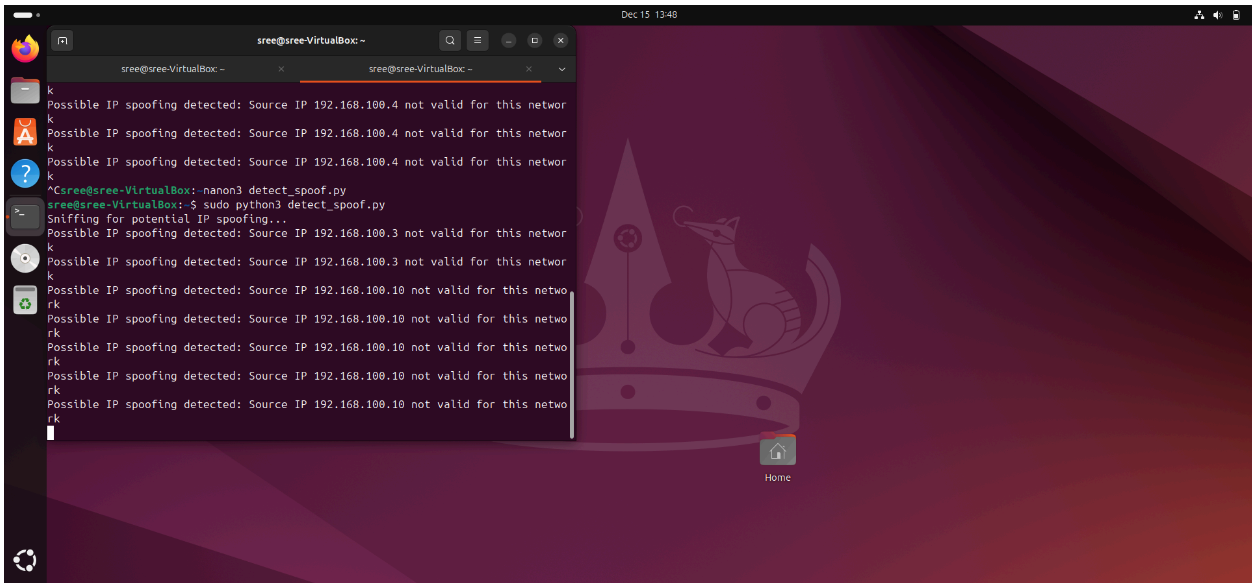
Egress Filtering (Outgoing Traffic):
Deny any outgoing packets with source IPs that are not from your network:sudo iptables -A OUTPUT ! -s 192.168.100.0/24 -j DROP

Ingress Filtering (Incoming Traffic):
Deny any incoming packets with source IPs from your internal network:

sudo iptables -A INPUT -s 192.168.100.0/24 -j DROP



```
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "/usr/lib/python3.12/http/server.py", line 1314, in <module>
    test(
  File "/usr/lib/python3.12/http/server.py", line 1261, in test
    with ServerClass(addr, HandlerClass) as httpd:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/socketserver.py", line 457, in __init__
    self.server_bind()
  File "/usr/lib/python3.12/http/server.py", line 1308, in server_bind
    return super().server_bind()
           ^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/http/server.py", line 136, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "/usr/lib/python3.12/socketserver.py", line 473, in server_bind
    self.socket.bind(self.server_address)
PermissionError: [Errno 13] Permission denied
sree@sree-VirtualBox:~$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
^C
Keyboard interrupt received, exiting.
sree@sree-VirtualBox:~$ sudo iptables -A OUTPUT ! -s 192.168.100.0/24 -j DROP
[sudo] password for sree:
sree@sree-VirtualBox:~$ sudo iptables -A INPUT -s 192.168.100.0/24 -j DROP
sree@sree-VirtualBox:~$ 
```



```
k
Possible IP spoofing detected: Source IP 192.168.100.4 not valid for this networ
k
Possible IP spoofing detected: Source IP 192.168.100.4 not valid for this networ
k
Possible IP spoofing detected: Source IP 192.168.100.4 not valid for this networ
k
^Csree@sree-VirtualBox:~nanon3 detect_spoof.py
sree@sree-VirtualBox:~$ sudo python3 detect_spoof.py
Sniffing for potential IP spoofing...
Possible IP spoofing detected: Source IP 192.168.100.3 not valid for this networ
k
Possible IP spoofing detected: Source IP 192.168.100.3 not valid for this networ
k
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
```

---

3. Set Up IP Spoof Detection Using Scapy on Ubuntu

Create a Scapy script to detect spoofed IP packets.

Steps:
On Ubuntu, create a Python script

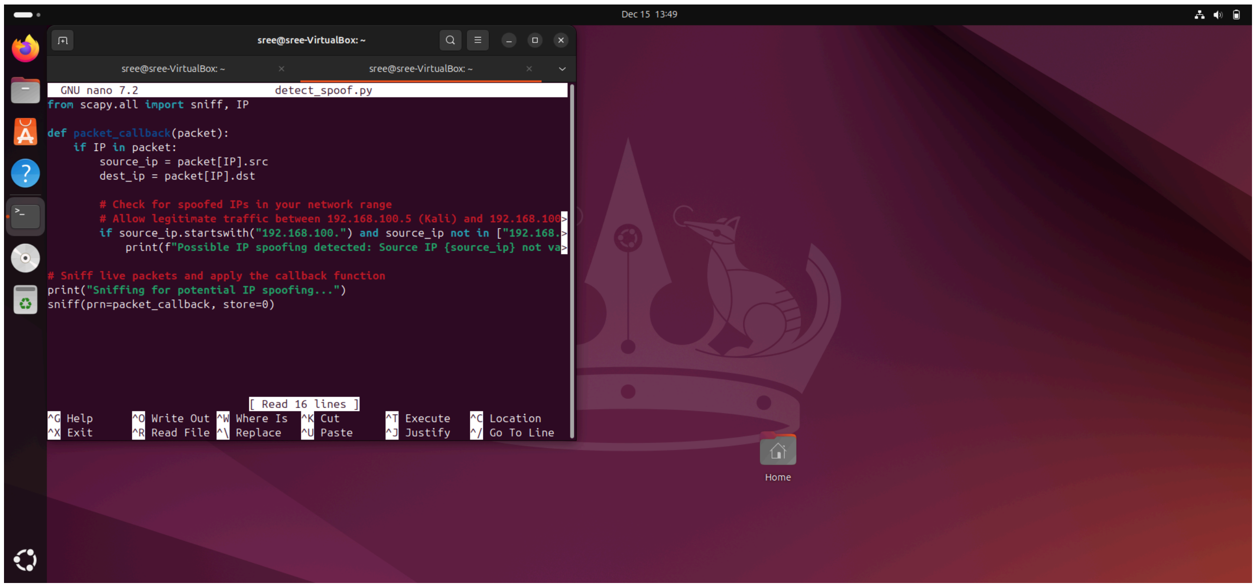
Python from scapy.all import sniff, IP

```
def packet_callback(packet):
    if IP in packet:
        source_ip = packet[IP].src
        dest_ip = packet[IP].dst

        # Check for spoofed IPs in your network range
        # Allow legitimate traffic between 192.168.100.5 (Kali) and 192.168.100.4 (Ubuntu)
        if source_ip.startswith("192.168.100.") and source_ip not in ["192.168.100.4", "192.168.100.5"]:
            print(f"Possible IP spoofing detected: Source IP {source_ip} not valid for this network")

# Sniff live packets and apply the callback function
print("Sniffing for potential IP spoofing...")
sniff(prn=packet_callback, store=0)
```

Run the script on Ubuntu with root privileges:
bash
Copy code sudo python3 detect_spoof.py



```
  GNU nano 7.2                     detect_spoof.py
from scapy.all import sniff, IP

def packet_callback(packet):
    if IP in packet:
        source_ip = packet[IP].src
        dest_ip = packet[IP].dst

        # Check for spoofed IPs in your network range
        # Allow legitimate traffic between 192.168.100.5 (Kali) and 192.168.100>
        if source_ip.startswith("192.168.100.") and source_ip not in ["192.168.>
            print(f"Possible IP spoofing detected: Source IP {source_ip} not va>

# Sniff live packets and apply the callback function
print("Sniffing for potential IP spoofing...")
sniff(prn=packet_callback, store=0)

                              [ Read 16 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

---

4. Simulate IP Spoofing from Kali

Use hping3 on the Kali machine to simulate IP spoofing.

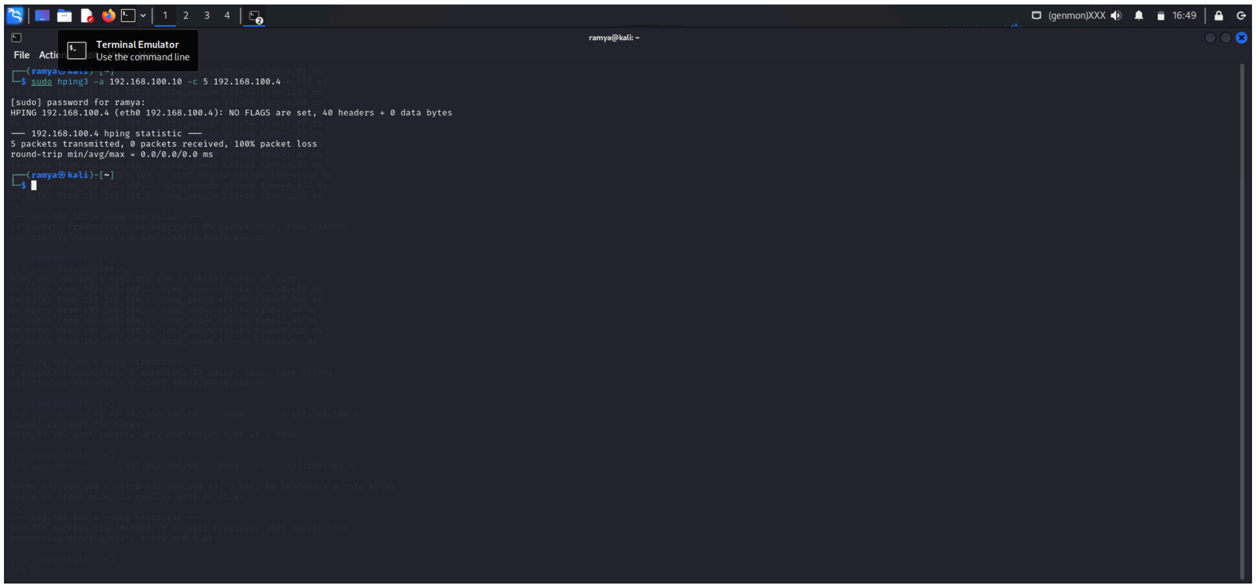
Send packets with a spoofed source IP:
bash
Copy code

```
sudo hping3 -a 192.168.100.10 -c 5 192.168.100.4
```

- **`-a 192.168.`**`100.10`: Spoofed source IP address.
- `-c 5`: Number of packets to send.
- `192.168.100.4`: Target IP (Ubuntu machine).

Monitor the output on the Ubuntu machine where the Scapy script is running. You should see a message like:

```
Possible IP spoofing detected: Source IP 192.168.100.10 not
valid for this network
```



```
┌──(ramya㉿kali)-[~]
└─$ sudo hping3 -a 192.168.100.10 -c 5 192.168.100.4

[sudo] password for ramya:
HPING 192.168.100.4 (eth0 192.168.100.4): NO FLAGS are set, 40 headers + 0 data bytes

─── 192.168.100.4 hping statistic ───
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(ramya㉿kali)-[~]
└─$ █
```

5. Monitor Traffic Using Wireshark

1. On Ubuntu:

Open Wireshark:

```
sudo wireshark
```

- Start a capture on the network interface (e.g., `eth0` or `wlan0`).
- Filter for spoofed packets (e.g., `ip.src == 192.168.100.10`)

---

6. Verify iptables Filtering

On Ubuntu, check the iptables rules:

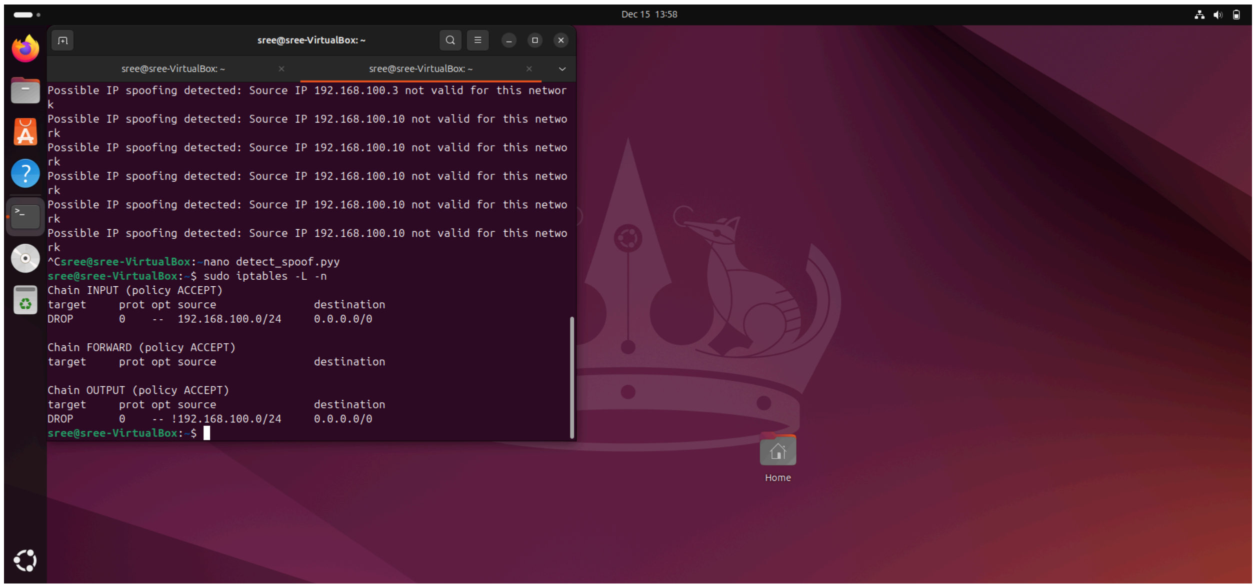
```
sudo iptables -L -v -n
```

You should see the `INPUT` and `OUTPUT` rules in place. Test the following:

1. Incoming packets (Ingress):

Use Kali to send spoofed packets:

```
sudo hping3 -a 192.168.100.10 -c 5 192.168.100.4
```



```
Possible IP spoofing detected: Source IP 192.168.100.3 not valid for this networ
k
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
Possible IP spoofing detected: Source IP 192.168.100.10 not valid for this netwo
rk
^Csree@sree-VirtualBox:~nano detect_spoof.pyy
sree@sree-VirtualBox:~$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       0    --  192.168.100.0/24     0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       0    -- !192.168.100.0/24     0.0.0.0/0
sree@sree-VirtualBox:~$
```

---

7. Cleanup

To reset iptables rules:

```
sudo iptables -F
```

The output confirms that the script is working correctly. It has successfully detected IP spoofing attempts originating from IP addresses:

- **`192.168.100.3`**
- **`192.168.100.10`**

Why This Output Matters:

1. `192.168.100.3` and `192.168.100.10`: These IPs are not part of your designated machines (`192.168.100.4` for Ubuntu and `192.168.100.5` for Kali).
2. The script flags these packets because they pretend to originate from within your network, which matches the behavior of spoofed packets.