

# SYN FLOOD ATTACK PREVENTION

Ramyasree Bakka

## Script for detection

**Step1:** open ubuntu terminal and start writing the given script in a nano file and save the file

```
from scapy.all import sniff, TCP
from collections import defaultdict

SYN_THRESHOLD = 100 # Number of SYN packets to trigger the alarm.
TIME_WINDOW = 60 # Time window in seconds to check for SYN_THRESHOLD
detected_ips = defaultdict(int) # Dictionary to hold IP addresses and their SYN counts

def packet_callback(packet):
    global detected_ips

    if TCP in packet and packet[TCP].flags == 'S':
        # If packet is TCP and has only the SYN flag set.
        source_ip = packet[IP].src
        detected_ips[source_ip] += 1

def syn_flood_detector():
    sniff(prn=packet_callback, filter="tcp", store=0)

if __name__ == "__main__":
    import threading
    import time

    # Start the detector in a separate thread
    detector_thread = threading.Thread(target=syn_flood_detector)
    detector_thread.start()

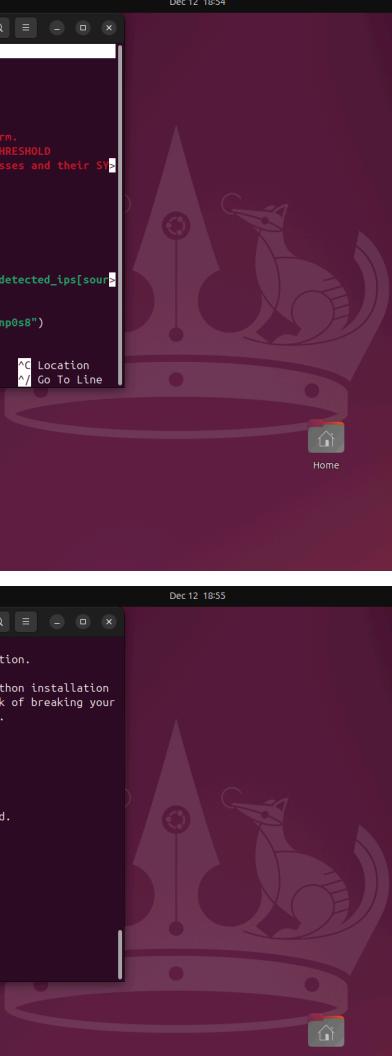
    try:
        while True:
            time.sleep(TIME_WINDOW)
```

```
# Check the counts and trigger alert for potential SYN
flood
    for ip, count in detected_ips.items():
        if count > SYN_THRESHOLD:
            print(f"[ALERT] Possible SYN flood attack from
{ip}. Detected {count} SYN packets in the last {TIME_WINDOW}
seconds.")

    # Reset the counts after checking
    detected_ips.clear()

except KeyboardInterrupt:
    print("Stopping the SYN flood detector.")
```

This script will continuously monitor for SYN flood attacks.



The image shows two screenshots of a Linux terminal window on an Ubuntu desktop. The desktop background features a large, faint watermark of a crown. In the top screenshot, a nano editor window is open showing Python code for a SYN flood detector. The code defines a `packet\_callback` function that increments a counter for each SYN packet received from a source IP. It also defines a `syn\_flood\_detector` function that starts a sniffer using the defined callback. The bottom screenshot shows the terminal window after running the script. It displays several lines of output, including a note about iptables being up-to-date, the command used to run the script, and a warning message indicating that the script may break the system if overridden. The terminal window has a dark theme with white text and a light gray background.

## Step2: start running the python script which is saved

### Step3: DEMO ATTACK

start the attack using hping3 from kali send syn packets

U can see the python script running terminal it is flooded with syn traffic and can see the alert

`hping3 -S -p 80 --flood 192.168.100.4`

- `192.168.100.4`: Replace this with the IP address of your Ubuntu machine.
- `-S`: Sends SYN packets.
- `-p 80`: Targets port 80.

--flood: Sends packets as fast as possible.

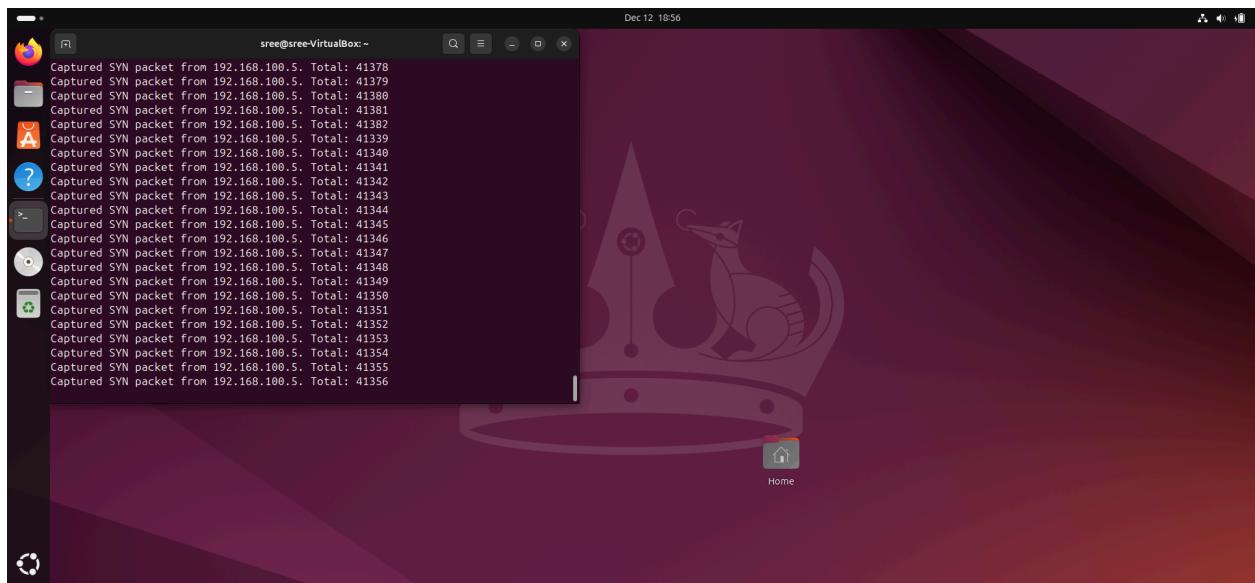
```
File Actions Edit View Help
[ramya@kali] ~]
$ hping3 -S -p 80 --flood 192.168.100.4
[open_sockraw socket(): Operation not permitted
[main() can't open raw socket
[ramya@kali] ~]
$ sudo hping3 -S -p 80 --flood 192.168.100.4
[sudo] password for ramya:
HPING 192.168.100.4 (eth0 192.168.100.4): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

#### **Step4: Mitigation Using Iptables**

## On Ubuntu Server (Target Machine)

**Block the Attacker IP (Reactive Mitigation):** After detecting the attacker's IP (e.g., 192.168.100.5), block it:cmd `sudo iptables -A INPUT -s 192.168.100.5 -j DROP`





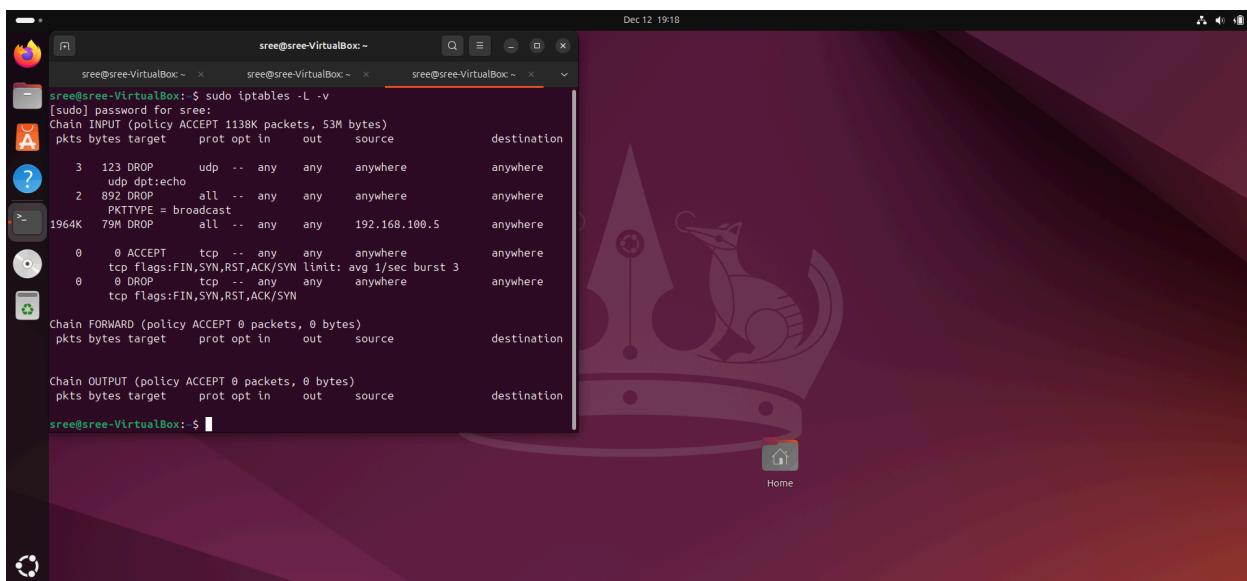
## Step 5: Test the Mitigation

On Kali Linux (Attacker Machine):

Run the SYN flood attack again:

```
hping3 -S -p 80 --flood 192.168.100.4
```

On Ubuntu Server (Target Machine): Verify that the attacker's packets are blocked. No new alerts should appear in the detection script after applying the `iptables` rules.



## **Step 7: Verify and Monitor**

**On Ubuntu Server (Target Machine):**

**Check Active `iptables` Rules:** `sudo iptables -L -v`