

Week 6 Task 6.2C

Question A

1. Include a screenshot of the tcpdump output running on the first Terminal (i.e., 172.17.0.2)

```
# tcpdump -n icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, Link-Type EN10MB (Ethernet), capture size 262144 bytes
13:44:44.4430916 IP 172.17.0.3 > 172.17.0.2: ICMP echo request, id 0, seq 0, length 8
13:44:44.464748 IP 172.17.0.2 > 172.17.0.3: ICMP echo reply, id 0, seq 0, length 8
13:46:57.180283 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 0, length 8
13:46:57.181157 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 0, length 8
13:46:58.181994 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 256, length 8
13:46:58.182064 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 256, length 8
13:46:59.182489 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 512, length 8
13:46:59.182552 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 512, length 8
13:47:00.183187 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 768, length 8
13:47:00.193862 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 768, length 8
13:47:01.194382 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 1024, length 8
13:47:01.194424 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 1024, length 8
13:47:02.195095 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 1280, length 8
13:47:02.195166 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 1280, length 8
13:47:03.195755 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 1536, length 8
13:47:03.196830 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 1536, length 8
13:47:04.196369 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 1792, length 8
13:47:04.196425 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 1792, length 8
13:47:05.196911 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 2048, length 8
13:47:05.196969 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 2048, length 8
13:47:06.197283 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 2304, length 8
13:47:06.197374 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 2304, length 8
13:47:07.198171 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 2560, length 8
13:47:07.198228 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 2560, length 8
13:47:08.198559 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 2816, length 8
13:47:08.198629 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 2816, length 8
13:47:09.199227 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 3072, length 8
13:47:09.199291 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 3072, length 8
13:47:10.199868 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 3328, length 8
13:47:10.199932 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 3328, length 8
13:47:11.200550 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 3584, length 8
13:47:11.200616 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 3584, length 8
13:47:12.201398 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 3840, length 8
13:47:12.201442 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 3840, length 8
13:47:13.201744 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 4096, length 8
13:47:13.201797 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 4096, length 8
13:47:14.202637 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 4352, length 8
13:47:14.202700 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 4352, length 8
13:47:15.203209 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 4608, length 8
13:47:15.203279 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 4608, length 8
13:47:16.203664 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 4864, length 8
13:47:16.203732 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 4864, length 8
13:47:17.204367 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 5120, length 8
13:47:17.204418 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 5120, length 8
13:47:18.205207 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 5376, length 8
13:47:18.205280 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 5376, length 8
13:47:19.205611 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 5632, length 8
13:47:19.205676 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 5632, length 8
13:47:20.206439 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 5888, length 8
13:47:20.206503 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 5888, length 8
13:47:21.206968 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 6144, length 8
13:47:21.207086 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 6144, length 8
13:47:22.207688 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 6400, length 8
13:47:22.207767 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 6400, length 8
13:47:23.208469 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 6656, length 8
13:47:23.208533 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 6656, length 8
13:47:24.209066 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 6912, length 8
13:47:24.209138 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 6912, length 8
13:47:25.209882 IP 172.17.0.4 > 172.17.0.2: ICMP echo request, id 2304, seq 7168, length 8
13:47:25.209966 IP 172.17.0.2 > 172.17.0.4: ICMP echo reply, id 2304, seq 7168, length 8
```

2. In your own words, explain what did just happen?
 - Hear, an ICMP echo request is sent to the first container by the second Docker container, but the packet is masqueraded to look as though it is from the third container. This results in the generation of a packet with a fabricated source IP address.

Question B

1. What is the password that you have cracked?
 - ‘no replies will be shown’

2. Why would an attacker spoof his IP when running an attack against a victim?
 - An attacker uses a fake IP address to conceal his location on to pretend to be another user.

3. Let's assume a Web Server was running on the host you targeted (i.e., 172.17.0.2). How could they have prevented their system from being targeted by your ICMP flooding attack?
 - A viable defense against an ICMP flooding assault is available
 - Make sure load balance and redundancy.
 - Setting up firewall rules to restrict or stop ICMP traffic.
 - Setting up ICMP rate limitation.

Question C

Based on the above and your own research about SYN Flood Attack, answer the following questions:

1. What Are the Signs of a SYN Flood DDoS Attack?
 - A SYN flood assault can be identified by keeping an eye on server performance and network traffic for particular signs.
 - One of the main indicators of a SYN flood attack is an abrupt and persistent raise in half-open TCP connections (connections in the SYN_RECEIVED state). Server logs or network monitoring software can be used to see it. Customers may see delays when trying to access services, which is a sign that the server is being overloaded with SYN packets. The sheer volume of pending connections may put a heavy burden on the server's CPU and memory. Crashing or reduced performance may result from this. With the aid of network traffic analysis tools, it may be possible to detect a discernible increase in incoming traffic that is mostly made up of SYN packets. Service outages could occur if legitimate users are unable or unwilling to create new connections to the server. When SYN packets arrive from fictitious or strange IP addresses, monitoring programs may identify them as potential signs of an attack.
2. How to Mitigate and Prevent a SYN Flood DDoS Attack?
 - Using a combination of hardware equipment, software, and network configuration is necessary to mitigate and avoid a SYN flood assault.
 - Get rid of harmful SYN packets by using intelligent firewalls and intrusion prevention systems. These tools have the ability to recognize patterns suggestive of an attack and stop malicious communication. Load balancers can be used to split up incoming traffic among several servers. This makes it less likely that the attack will overwhelm a single server. Determine which IP addresses are sending a lot of SYN messages and block them. This works very well in situations where there are direct attacks and no IP address spoofing. Shorten the timeout period for connections that are partially open. This guarantees that the server releases resources faster by preventing it from waiting too long acknowledgement. Keep a close eye on network traffic for any indications of strange activity. Rapid reaction and mitigation are made possible by early assault detection. Plan your server and network architecture with high availability and redundancy in mind. This makes it possible for other servers to step in and keep service uninterrupted even in the event that one is compromised.

Question D

1. Investigate what is “syn cookie” and explain this in your own words. (1 reference is expected, word count is up to 300 words).
 - Servers are shielded from SYN flood attacks, a kind of Denial-of-Service (DoS) attack, by means of SYN cookies. These exploits take use of the TCP handshake procedure to flood a server with partially open connections, preventing it from receiving valid traffic.
 - The SYN cookies solution is for the server to compute a unique sequence number (SYN cookie) using data from the SYN packet and a secret key, rather than devoting resources for each SYN request. SYN-ACK messages contain this sequence number. The sequence number is included in the ACK, if the client provides one. With the use of a reverse function, the server confirms this number. In the event that it is legitimate, it drops the connection and establishes the required resources.
 - A SYN request is sent by the client to establish a connection (TCP Handshake Review). With a SYN-ACK, the server answers. Customer completes the handshake by sending an ACK.
 - Attacker sends a lot of SYN request and does not respond to SYN-ACKs, which leaves connections partially open. This is known as a SYN flood Attack. Each half-open connection requires resource from the server, which fills its buffer and prevents new connections.
 - Benefits: The server guards against resource exhaustion by refusing to allocate resource for connections that are not validated. Service availability is ensured since legitimate connections remain unblocked even in the event severe SYN flooding.
 - Reference: GeeksforGeeks. (n.d.). How SYN cookies are used to preventing SYN Flood attack. Retrieved from GeeksforGeeks

Question E

1. Include a screenshot of the output in the first Terminal showing the huge amount of flooding packets after the attack is executed against the server.

```

15:37:52.793194 IP 10.0.0.1.17049 > 172.17.0.2.80: Flags [S], seq 1915026395, win 512, length 0
15:37:52.793218 IP 10.0.0.1.17050 > 172.17.0.2.80: Flags [S], seq 1028537527, win 512, length 0
15:37:52.793239 IP 10.0.0.1.17051 > 172.17.0.2.80: Flags [S], seq 2073307981, win 512, length 0
15:37:52.793260 IP 10.0.0.1.17052 > 172.17.0.2.80: Flags [S], seq 1690897655, win 512, length 0
15:37:52.793278 IP 10.0.0.1.17053 > 172.17.0.2.80: Flags [S], seq 330298879, win 512, length 0
15:37:52.793326 IP 10.0.0.1.17054 > 172.17.0.2.80: Flags [S], seq 1848326846, win 512, length 0
15:37:52.793383 IP 10.0.0.1.17055 > 172.17.0.2.80: Flags [S], seq 734036109, win 512, length 0
15:37:52.793407 IP 10.0.0.1.17056 > 172.17.0.2.80: Flags [S], seq 823873767, win 512, length 0
15:37:52.793427 IP 10.0.0.1.17057 > 172.17.0.2.80: Flags [S], seq 305996394, win 512, length 0
15:37:52.793448 IP 10.0.0.1.17058 > 172.17.0.2.80: Flags [S], seq 107844739, win 512, length 0
15:37:52.793470 IP 10.0.0.1.17059 > 172.17.0.2.80: Flags [S], seq 1417923824, win 512, length 0
15:37:52.793509 IP 10.0.0.1.17060 > 172.17.0.2.80: Flags [S], seq 982221595, win 512, length 0
15:37:52.793528 IP 10.0.0.1.17061 > 172.17.0.2.80: Flags [S], seq 1467297253, win 512, length 0
15:37:52.793545 IP 10.0.0.1.17062 > 172.17.0.2.80: Flags [S], seq 1921042252, win 512, length 0
15:37:52.793563 IP 10.0.0.1.17063 > 172.17.0.2.80: Flags [S], seq 515521750, win 512, length 0
15:37:52.793584 IP 10.0.0.1.17064 > 172.17.0.2.80: Flags [S], seq 459357443, win 512, length 0
15:37:52.793605 IP 10.0.0.1.17065 > 172.17.0.2.80: Flags [S], seq 711459478, win 512, length 0
15:37:52.793625 IP 10.0.0.1.17066 > 172.17.0.2.80: Flags [S], seq 164508627, win 512, length 0
15:37:52.793646 IP 10.0.0.1.17067 > 172.17.0.2.80: Flags [S], seq 2082969183, win 512, length 0
15:37:52.793717 IP 10.0.0.1.17068 > 172.17.0.2.80: Flags [S], seq 64727743, win 512, length 0
15:37:52.793739 IP 10.0.0.1.17069 > 172.17.0.2.80: Flags [S], seq 509242629, win 512, length 0
15:37:52.793761 IP 10.0.0.1.17070 > 172.17.0.2.80: Flags [S], seq 618384375, win 512, length 0
15:37:52.793811 IP 10.0.0.1.17071 > 172.17.0.2.80: Flags [S], seq 1603343878, win 512, length 0
15:37:52.793827 IP 10.0.0.1.17072 > 172.17.0.2.80: Flags [S], seq 717607976, win 512, length 0
15:37:52.793880 IP 10.0.0.1.17073 > 172.17.0.2.80: Flags [S], seq 109765515, win 512, length 0
15:37:52.793908 IP 10.0.0.1.17074 > 172.17.0.2.80: Flags [S], seq 1368365337, win 512, length 0
15:37:52.793965 IP 10.0.0.1.17075 > 172.17.0.2.80: Flags [S], seq 890058501, win 512, length 0
15:37:52.793988 IP 10.0.0.1.17076 > 172.17.0.2.80: Flags [S], seq 220893525, win 512, length 0
15:37:52.794009 IP 10.0.0.1.17077 > 172.17.0.2.80: Flags [S], seq 2084364823, win 512, length 0
15:37:52.794031 IP 10.0.0.1.17078 > 172.17.0.2.80: Flags [S], seq 1865685332, win 512, length 0
15:37:52.794046 IP 10.0.0.1.17079 > 172.17.0.2.80: Flags [S], seq 208884358, win 512, length 0
15:37:52.794065 IP 10.0.0.1.17080 > 172.17.0.2.80: Flags [S], seq 1962639850, win 512, length 0
15:37:52.794115 IP 10.0.0.1.17081 > 172.17.0.2.80: Flags [S], seq 110336669, win 512, length 0
15:37:52.794136 IP 10.0.0.1.17082 > 172.17.0.2.80: Flags [S], seq 673760364, win 512, length 0
15:37:52.794149 IP 10.0.0.1.17083 > 172.17.0.2.80: Flags [S], seq 995086314, win 512, length 0
15:37:52.794161 IP 10.0.0.1.17084 > 172.17.0.2.80: Flags [S], seq 605937291, win 512, length 0
15:37:52.794174 IP 10.0.0.1.17085 > 172.17.0.2.80: Flags [S], seq 1829151914, win 512, length 0
15:37:52.794185 IP 10.0.0.1.17086 > 172.17.0.2.80: Flags [S], seq 1390203097, win 512, length 0
15:37:52.794197 IP 10.0.0.1.17087 > 172.17.0.2.80: Flags [S], seq 341351444, win 512, length 0
15:37:52.794209 IP 10.0.0.1.17088 > 172.17.0.2.80: Flags [S], seq 209940074, win 512, length 0
15:37:52.794264 IP 10.0.0.1.17089 > 172.17.0.2.80: Flags [S], seq 1734308862, win 512, length 0
15:37:52.794293 IP 10.0.0.1.17090 > 172.17.0.2.80: Flags [S], seq 704180760, win 512, length 0
15:37:52.794313 IP 10.0.0.1.17091 > 172.17.0.2.80: Flags [S], seq 384816440, win 512, length 0
15:37:52.794333 IP 10.0.0.1.17092 > 172.17.0.2.80: Flags [S], seq 135430646, win 512, length 0
15:37:52.794353 IP 10.0.0.1.17093 > 172.17.0.2.80: Flags [S], seq 1783885583, win 512, length 0
15:37:52.794377 IP 10.0.0.1.17094 > 172.17.0.2.80: Flags [S], seq 165894620, win 512, length 0
15:37:52.794398 IP 10.0.0.1.17095 > 172.17.0.2.80: Flags [S], seq 966817439, win 512, length 0
15:37:52.794418 IP 10.0.0.1.17096 > 172.17.0.2.80: Flags [S], seq 1283842572, win 512, length 0
15:37:52.794439 IP 10.0.0.1.17097 > 172.17.0.2.80: Flags [S], seq 873483864, win 512, length 0
15:37:52.794458 IP 10.0.0.1.17098 > 172.17.0.2.80: Flags [S], seq 502459946, win 512, length 0

```

Question F

1. A common interview questions these days is about Mirai Botnet. Investigate this Botnet and in your own words explain how it worked and what was its impact. Is this Botnet still affecting IoT devices? Please ensure that you use references for your answer. The suggested word count is 400-500 words.

- In 2016, a highly disruptive malware that targeted Internet of Thing (IoT) device surfaced, known as the Mirai Botnet. It was first developed by paras Jha, Josiah white, and Dalton Norman. They used it to perform Distributed Denial of Service (DDoS) assaults against competing servers in the video game Minecraft, giving them an advantage. But the botnet soon became one of the most well-known in cybersecurity history as its capabilities grew.

- **The operation of Mirai**

The intrinsic flaws in LoT devices, like default usernames and passwords, were exploited by Mirai. Users frequently forget to modify the default passwords that come with a lot of internet of things devices, such as DVRs, routers and switches, and cameras. After searching the internet for these susceptible gadgets, Mirai used a list of frequently used default credentials to obtain access. Once a device was compromised, Mirai would download its software, thereby turning it into a slave within the botnet.

Mirai could use this enormous network of compromised devices to overload target systems with traffic after a signification proportion of them were hacked, disrupting services. Because LoT devices are more common than traditional computing devices and frequently have laxer security regulations, they are easy targets for botnet was particularly successful.

- **The effects of Mirai**

Considerable and extensive effects were caused by the Mirai Botnet. When a DDoS attack against major DNS provider Dyn was launched using Mirai in October 2016, it was one of the most noteworthy events. Popular websites and services including Twitter, Reddit, Netflix, and Airbnb were all negatively impacted by this attack, which also caused widespread internet disruptions.

Significant disruption could be possible, as evidenced by Mirai's capacity to enslave a large number of IoT devices into a botnet that could produce terabits of malicious traffic per second. In addition to bringing attention to IoT devices vulnerabilities, the attacks also made clear that the quickly expanding IoT industry needs better security procedures.

- **Present Condition of Mirai**

There is still a threat from variation of the Mirai Botnet, even after its developers were apprehended and found guilty in 2017. After Mirai's source code was made available to that public in 2016, additional cybercriminals were able to alter and customize it to suit their needs. Consequently, novel strains of Mirai persist in surfacing, focusing on an expanded array of IoT devices and employing increasingly intricate techniques to elude identification and countermeasure initiatives.

The ongoing existence of Mirai variations suggests that Internet of Things devices are still susceptible. The default password on a lot of Internet of Things devices are still in place, and users frequently forget to install security upgrades. Furthermore, botnets such as Mirai have an ever-growing attack surface due to the growing prevalence of IoT devices.

- To sum up, the Mirai botnet caused a great deal of disruption by taking advantage of poor security in IoT devices, which made it clear how urgently the IoT ecosystem needed stronger security measure. Though its original developers have been captured, Mirai's legacy lives on in the form of continuous variations that pose a threat to global internet services reliability and security
- Reference - Mimoso, M. (2017). Mirai Botnet Authors Avoid Jail Time, Will Assist FBI. Threatpost. Retrieved from Threatpost