Assignment 1

Introduction

For this assignment, we chose an *ICMP Source Quench DoS* attack. In a setting where a server is sending data to a client, the client can send an ICMP Source Quench packet to the server requesting that it slows down the rate at which it's sending data (usually because the client cannot keep up with the server's rate).

From a malicious perspective, this can be abused by an attacker that spoofs the client's IP/MAC addresses and floods the server with ICMP Source Quench packets, causing a server-side denial of service by minimizing its data sending rate.

Tool Description

1. Attack (Houssein & Rami)

In our attack, we flood a client (or servers, depending on which version of the attack is being used) with ICMP Source Quench packets from random source IPs, MACs, and source/destination ports. Although for a Quench attack to be successful it would require that the source IP, MAC, and source/destination ports be those of a valid client (i.e., one that is actively communicating with the server),For the single target attack (Attack 1) we manually enter the IP of the target PC. In reality, we would need to sniff for a TCP connection between a client and server on the network, and extract the required header information which we did in the second (attack 2) but the runtime is slow if the set number of IP's to attack is high (we recommend to set the number of IPs to attack up to 5-10).

Note that although the attack is functional (Quench packets reach the server normally), it is NOT successful since (1) we're randomizing the header information (and thus not masquerading as a specific client) and (2) most operating systems nowadays implement updated versions of TCP that automatically ignore any incoming ICMP Source Quench packets to prevent DoS attacks (as recommended by RFC6633).

2. Detection (Adam & Toufic)

In our detection scheme, we relied on the assumption that ICMP Source Quench packets are deprecated and rarely sent (RFC6633 prevents a host from sending them). Therefore, our assumption is that they would mostly be sent by legacy systems, and their frequent occurrence (i.e., few consecutive Quench packets) may constitute an attack.

Thus, our detection scheme is as follows: Sniff and filter any incoming ICMP Source Quench packets, regardless of source information (i.e., IP/MAC). If their count exceeds a user-set threshold, an alert is displayed to the user informing them that a Quench attack is ongoing. If this count is below the threshold, the user is warned of a potential attack. Given that Quench packets sent by the attacker contain random header information, no effort was made to reveal the attacker's identity.

Running the Tools

attack_1.py

This is the first version of the attack. It targets a specific host, and floods it with a user-set number of ICMP Source Quench packets. To run the code, change the following in the main call: (1) the target IP address [`inputIp`] and (2) the number of Quench packets to send [`packetsToSend`].

attack_2.py

This is the second version of the attack. Instead of targeting a specific host, it floods all active hosts on the network with Quench packets. To run the code, modify the following in the main call: (1) number of packets to send to each active host [`packetsToSend`], (2) OPTIONAL (set to 0 by default): time delay between sent packets [`delay`], and (3) OPTIONAL (set to all by default): number of IPs to flood with packets [`LimitIps`].

quench_detection.py

To run this code, modify the following in the main call: (1) threshold, which when exceeded, constitutes an attack [`threshold`], and (2) OPTIONAL (automatically detected by default): the name of the interface to detect incoming Quench packets on [`interface`].

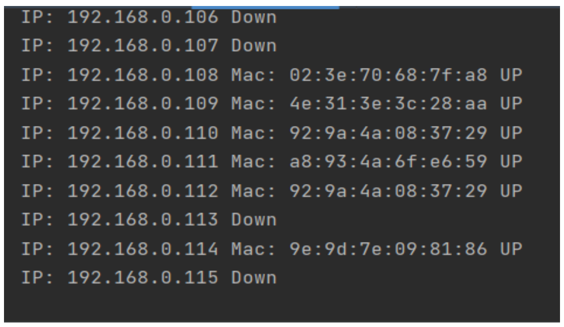
This code must be running before/during an attack, and requires ~30 seconds to initialize properly (i.e., before it starts detecting any Quench packets). Also note that, as we observed, the large majority of Quench packets are not usually received by the target (we tested that with Wireshark as well), so we made sure to make the number of sent packets large to detect their existence. We also disabled any firewalls to maximize the number of detected packets.

The following libraries were used in the code :

- **getmac**
- **scapy**
- **Python standard library**

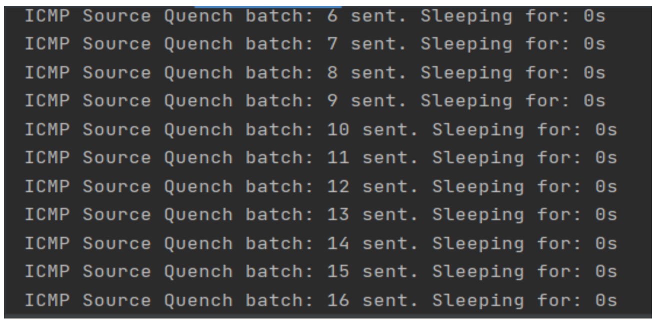
Results & Testing

Figure 1: Attacker scanning All IP's for a target



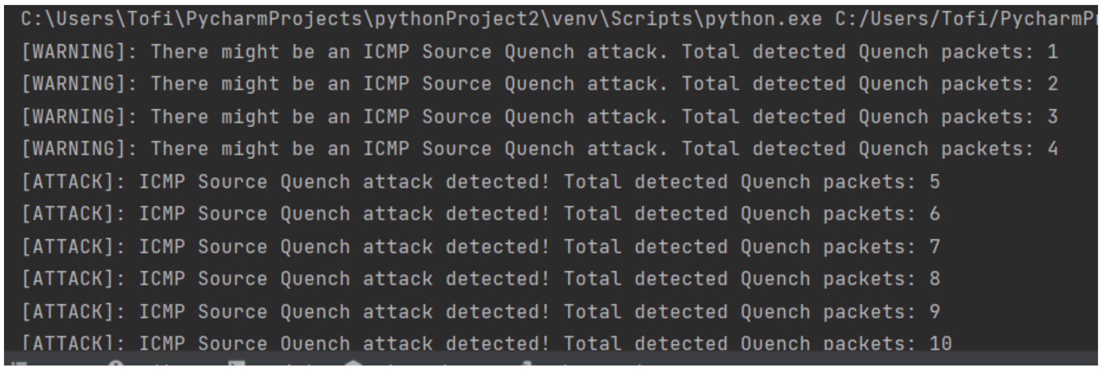
```
IP: 192.168.0.106 Down
IP: 192.168.0.107 Down
IP: 192.168.0.108 Mac: 02:3e:70:68:7f:a8 UP
IP: 192.168.0.109 Mac: 4e:31:3e:3c:28:aa UP
IP: 192.168.0.110 Mac: 92:9a:4a:08:37:29 UP
IP: 192.168.0.111 Mac: a8:93:4a:6f:e6:59 UP
IP: 192.168.0.112 Mac: 92:9a:4a:08:37:29 UP
IP: 192.168.0.113 Down
IP: 192.168.0.114 Mac: 9e:9d:7e:09:81:86 UP
IP: 192.168.0.115 Down
```

Figure 2: Attacker sending ICMP Quench Packets



```
ICMP Source Quench batch: 6 sent. Sleeping for: 0s
ICMP Source Quench batch: 7 sent. Sleeping for: 0s
ICMP Source Quench batch: 8 sent. Sleeping for: 0s
ICMP Source Quench batch: 9 sent. Sleeping for: 0s
ICMP Source Quench batch: 10 sent. Sleeping for: 0s
ICMP Source Quench batch: 11 sent. Sleeping for: 0s
ICMP Source Quench batch: 12 sent. Sleeping for: 0s
ICMP Source Quench batch: 13 sent. Sleeping for: 0s
ICMP Source Quench batch: 14 sent. Sleeping for: 0s
ICMP Source Quench batch: 15 sent. Sleeping for: 0s
ICMP Source Quench batch: 16 sent. Sleeping for: 0s
```

Figure 3: Defender detecting received Quench packets and notifying user



```
C:\Users\Tofi\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/Tofi/PycharmP
[WARNING]: There might be an ICMP Source Quench attack. Total detected Quench packets: 1
[WARNING]: There might be an ICMP Source Quench attack. Total detected Quench packets: 2
[WARNING]: There might be an ICMP Source Quench attack. Total detected Quench packets: 3
[WARNING]: There might be an ICMP Source Quench attack. Total detected Quench packets: 4
[ATTACK]: ICMP Source Quench attack detected! Total detected Quench packets: 5
[ATTACK]: ICMP Source Quench attack detected! Total detected Quench packets: 6
[ATTACK]: ICMP Source Quench attack detected! Total detected Quench packets: 7
[ATTACK]: ICMP Source Quench attack detected! Total detected Quench packets: 8
[ATTACK]: ICMP Source Quench attack detected! Total detected Quench packets: 9
[ATTACK]: ICMP Source Quench attack detected! Total detected Quench packets: 10
```

Contributions

- **Hussein (Team Manager):** Monitored team progress and reviewed the code .
- **Adam (Defense):** Filtered incoming ICMP Source Quench packets, implemented the threshold mechanism, and encapsulated all code within classes for user friendliness.
- **Toufic (Defense):** Wrote the ICMP packet detection function that filters out packets that are not ICMP packets.
- **Houssein (Attack):** Wrote the get all IP addresses and MAC addresses functions for the attack. Getting all IPs that are up by automatically getting PC's local IP and based on that scanning with broadcast ARP for other IPs.
- **Rami (Attack):** Added "send icmp blind packet" function which sends icmp quench packets to the targeted IP/victim(s) and "prepare" function which randomizes sender IP and uses Fake sender MAC address to make it less suspicious to the defender. Also added a feature to choose the number of quench requests to send.
- Quick note: the ICMP blind packet function was taken/inspired from the web.