**CSCE 5580: Computer Network**

**Bonus Project Report**

**Title**: Implementation and Analysis of HTTP Flood Attack

**Author**: Kishan Kumar Zalavadia

**1. Introduction**

This project implements an HTTP Flood attack. This project is implemented in Python programming language, which demonstrates the behaviors and impacts on the target server that we created. The objective is to learn and understand the HTTP attack and gain insights into the methodologies.

**2. Methodology**

In this project, we have created our own server, which is running on a 127.0.01 IP address with 65535 as the port number to ensure that the attack is not impacting the real servers.

**2.1 HTTP Flood Attack**

In this attack, a client sends a large number of HTTP GET requests with random paths, headers, and numbers. It implements multiple threads to simulate concurrent clients. Each thread generates HTTP GET requests with a 5-character URL path, header, and random port number. This project targets the server with IP address 127.0.01 and port number 65535. This server is created by us, so in order to execute the HTTP attack, we first need to run the server and then perform the attack.

We create a number of clients, and each client requests five http requests to the client. So, for example, if we have 100 clients, there will be 500 requests made to the client.

**3 Results**

**3.1 Experimental Setup**

**Server**: A basic Python socket server that we created to accept connections and log activities.

**Client**: We use multi-threading to execute clients concurrently.

**Wireshark**: We use Wireshark to monitor the network traffic during the attacks.

**3.2 Console Output**

A screenshot of a computer

Description automatically generated

Figure 3.2.1

Figure 3.2.1 is a console from the server. To perform the HTTP attack, we run our server on 127.0.0.1:65535. Once the server is running, it waits for the client to establish a connection. Figure 3.2.1 represents the server console after the attack. It shows that the connection to the client has started, and it received a get HTTP request having a random character path, and the connection is kept alive.

A screen shot of a computer screen

Description automatically generated

Figure 3.2.2

Once the server is up and running, we perform the http attack. Figure 3.2.2 represents the console on the client side after the attack. As you can see, the client is connected to our server, and it sends the HTTP request, which has a random character path.

**3.3 Wireshark Output**

A screenshot of a computer

Description automatically generated

Figure 3.3.1

A screenshot of a computer

Description automatically generated

Figure 3.3.2

Figure 3.3.1 and figure 3.3.2 shows the Wireshark screenshot after the attack has been triggered.

As seen in the images, there are multiple HTTP requests, and each request goes from a different port number. (Highlighted in yellow)

**4 Conclusion**

This project gives a hands-on experience with network HTTP flood attack that impacts on server performance. The following are the insights gained.

Understanding of HTTP attack: How the server is impacted by the HTTP attack.

Traffic analysis: Looking at Wireshark, we gain insights on how to use Wireshark to monitor the network.

**5 References**

https://www.python.org/

https://realpython.com/installing-python/

https://scapy.net/

https://www.imperva.com/learn/ddos/http-flood/