**Report**

**On**

**Implementing HTTP Flood Attack and Socket Programming using Python**

*by Uday Bhaskar Valapadasu – 11696364*

**Abstract**

This report presents the implementation and analysis of a network programming project focusing on multi-threading, socket programming, and HTTP flood attack simulation.

**1. Introduction**

The project implements several key networking concepts:

* Multi-threading programming in Python
* Socket programming (client-server communication)
* TCP SYN flood attack simulation
* HTTP flood attack implementation
* Network packet manipulation using Scapy

**2. Implementation Details**

**2.1 Multi-threading Implementation**

The project utilizes Python's threading module to create multiple concurrent threads. This is demonstrated through:

* Single thread execution
* Multiple thread management
* Thread synchronization
* Resource sharing between threads

**2.2 Socket Programming**

The socket programming implementation includes:

* TCP server implementation
* Client connection handling
* Data exchange protocols
* Error handling and connection management

**2.3 HTTP Flood Attack Implementation**

The HTTP flood attack simulation includes several key features:

1. Multi-threaded attack simulation
2. Random URL generation
3. IP address and port randomization
4. Packet creation and manipulation
5. Progress monitoring and reporting

Key components of the HTTP flood implementation and the scripts which simulates an **HTTP flood attack** by sending multiple HTTP requests to a target server:

1. **generate\_random\_url()**: Creates random URLs with optional query parameterA screenshot of a computer program

   Description automatically generated
2. **create\_http\_request(url)**: Forms an HTTP request (GET, POST, or HEAD) with the generated URL and headers.A screenshot of a computer program

   Description automatically generated
3. **http\_flood\_worker()**: A worker thread that sends a specified number of HTTP requests (packets) to the target server. It uses random source IPs and ports to make the attack appear distributed.

A screen shot of a computer

Description automatically generated

1. **start\_http\_flood\_attack()**: Initializes and starts multiple threads (**each sending a set number of packets),** effectively launching a flood of HTTP requests to the target IP and port.A screen shot of a computer

   Description automatically generated

The attack overloads the target by sending a large volume of HTTP requests.

**3. Testing and Results**

**3.1 Wireshark Analysis**

Packet capture analysis showed:

* Successful packet generation and multi-threads.

Console Output:

A screen shot of a computer

Description automatically generated

Wireshark Output:

A screenshot of a computer

Description automatically generated

* Proper HTTP request formatting

A screen shot of a computer

Description automatically generated

* Random IP address distribution

A screenshot of a computer

Description automatically generated

* Varied port number allocation

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**5. Conclusions**

The project successfully demonstrates:

1. Effective multi-threading implementation
2. Proper socket programming practices
3. Functional HTTP flood attack simulation
4. Network packet manipulation capabilities

**References**

1. Python Threading Documentation
2. Scapy Documentation
3. HTTP/1.1 RFC 2616
4. Network Security Principles and Practices