FLOOD-REAPER

TABLE OF CONTENTS

Introduction

FloodReaper is a comprehensive and powerful tool designed to simulate a variety of flooding attacks against websites and domains. It offers an advanced command-line interface (CLI) and a suite of features tailored for stress testing and educational purposes.

Key Features

1. **Multi-Type Attacks:**
    - **SYN Flood:** Exploits the TCP handshake process to overwhelm the target with half-open connections, causing resource exhaustion.
    - **Slowloris:** Maintains many connections to the target web server, consuming its resources and making it unable to handle legitimate requests.
    - **IP Fragmentation (IP FRAG):** Sends fragmented packets to disrupt and overwhelm the target's ability to reassemble packets.
    - **DNS Amplification:** Uses DNS servers to amplify traffic towards the target, causing significant bandwidth consumption and potential downtime.

2. **CLI Interface:**
    - **User-Friendly:** Offers a straightforward CLI that allows users to configure and execute attacks efficiently.
    - **Command Options:** Provides various commands and options to customize attack parameters, such as target IP, port, attack duration, and more
    .
3. **Self-Checking and Setup:**
    - **System Check:** Automatically verifies system requirements and dependencies.
    - **Automated Installation:** Installs necessary tools and libraries, ensuring a smooth setup process for users.

4. **IP Hopping:**
    - **Tor Integration:** Leverages Tor for IP hopping, allowing the tool to rotate IP addresses to avoid detection and blockades.
    - **Tornet Integration:** Uses Tornet for additional IP rotation and anonymity, enhancing the tool's ability to bypass security measures.

5. **Multithreading:**
    - **Increased Performance:** Utilizes multithreading to send a high volume of requests simultaneously, amplifying the attack’s impact and effectiveness.
    - **Concurrency:** Allows multiple attacks to be executed in parallel, optimizing the tool’s performance and capability.

Installation

To install and get started with **FloodReaper**, follow these simple steps:

1. **Clone the Repository**

   First, clone the FloodReaper repository from GitHub to your local machine: https://github.com/TushN101/FloodReaper

```
git clone https://github.com/TushN101/FloodReaper.git
```

2. **Navigate to the Project Directory**

   Change to the directory where FloodReaper has been cloned:

```
cd FloodReaper
```

3. **Install scapy**

   Install the required library

```
pip install scapy
```

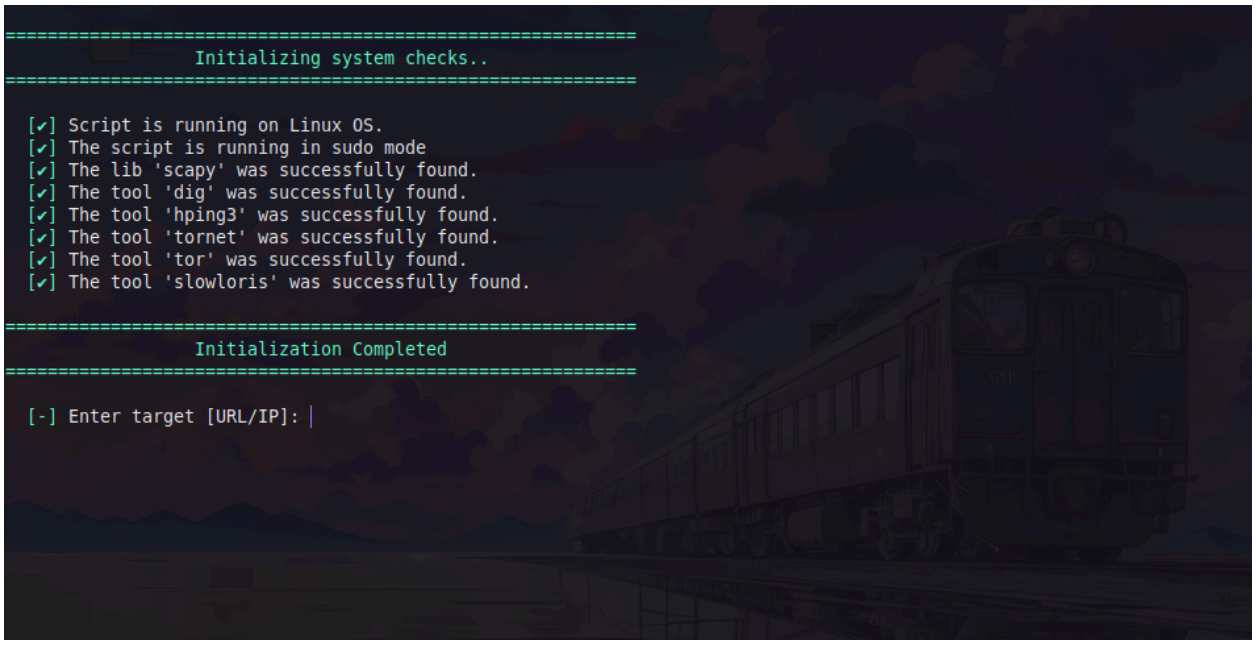
4. **Run the Tool**

   Execute the main script to start **FloodReaper**:

```
python main.py
```

This command will launch **FloodReaper**

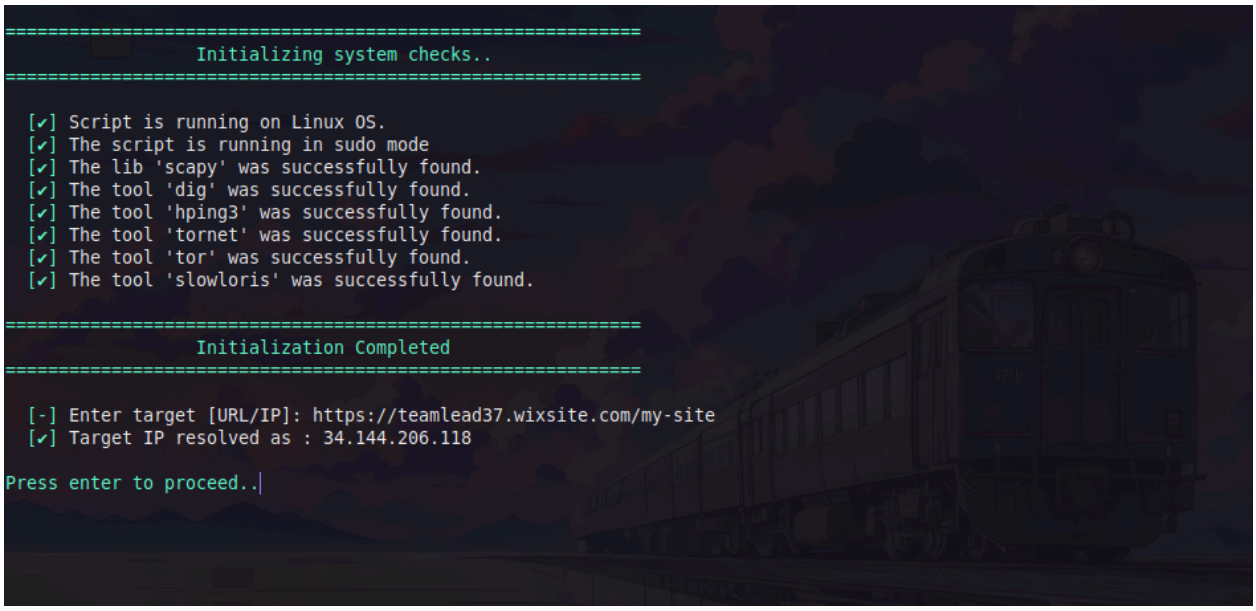
Initial Setup and Configuration



Before using FloodReaper, the script performs the following automatic checks:

1. **Operating System Check**: Confirms that **FloodReaper** is running on a Linux-based system, as it is designed for Linux environments.
2. **Sudo Privileges Check**: Verifies that the script is executed with root permissions, which are necessary for accessing network sockets and performing certain operations.
3. **Scapy Installation Verification**: Checks if the Scapy library is installed and accessible, as it is crucial for packet manipulation and network scanning.
4. **`dig` Tool Check**: Ensures that the `dig` tool is available for domain-to-IP conversion, which is needed for resolving domain names.
5. **`hping3` Tool Check**: Confirms that `hping3` is installed, which is required for executing TCP SYN flood attacks.
6. **Tor and Tornet Check**: Verifies that both Tor and Tornet tools are installed and configured for IP hopping capabilities.
7. **Slowloris Tool Check**: Checks if the Slowloris tool is present, as it is used for performing Slowloris attacks

Setting the Target



Before using FloodReaper, the script locks the target:

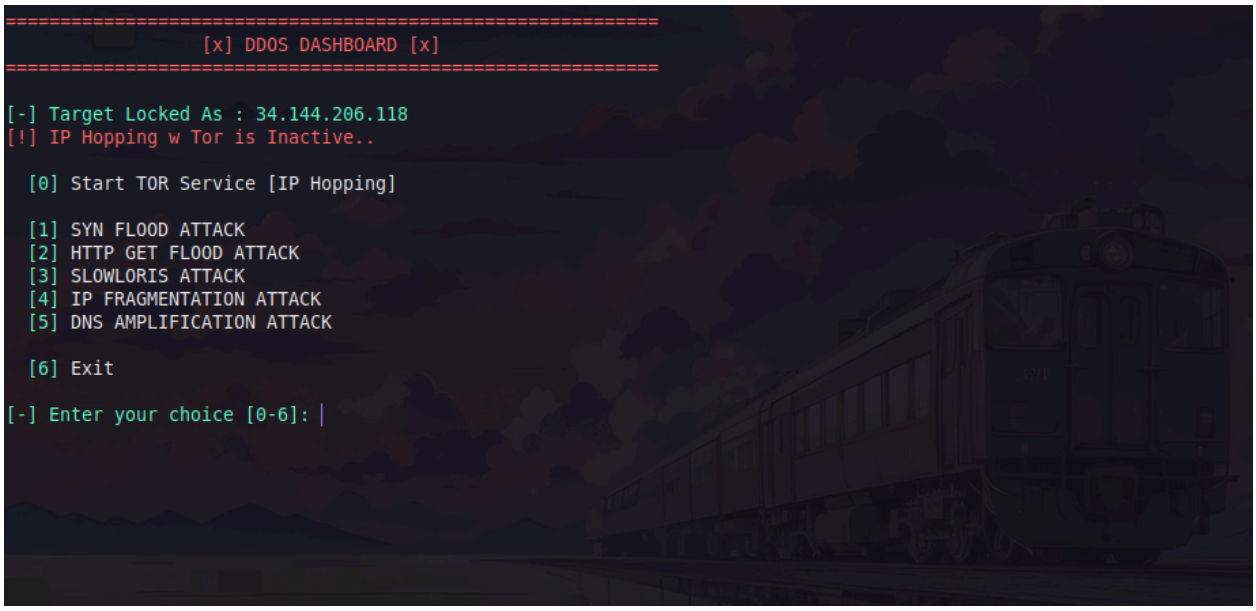
IP Address Input: If you provide a direct IP address, the script locks onto that IP immediately, allowing you to proceed with the attack.

Website URL Input: If a website URL is provided, the script uses a function to resolve the domain name. It then utilizes the `dig` tool to convert the domain into an IP address.

Error Handling: `dig` tool Issues : If the `dig` tool fails to resolve the domain or is not available, the script detects this and prompts you with an error message.In such cases, the script will ask you to manually input the IP address, ensuring that you can still proceed with the attack despite the tool's failure.

Overall Error Safety: The script includes robust error-handling mechanisms to ensure smooth operation. If any issues arise during target resolution, users are guided through troubleshooting steps to resolve them and continue with their tasks.

THE DASHBOARD:



Target Display: The panel shows the currently locked target, providing a clear view of which IP address or domain is being targeted.

Attack Options: You are provided with five different types of attacks:

1. **SYN Flood:** Floods the target with SYN packets to overwhelm it.
2. **HTTP GET Flood:** Sends a high volume of HTTP GET requests to the target.
3. **Slowloris:** Conducts a Slowloris attack to keep connections open and exhaust server resources.
4. **IP Fragmentation:** Sends fragmented IP packets to disrupt the target.
5. **DNS Amplification:** Uses DNS queries to amplify and flood the target with traffic.

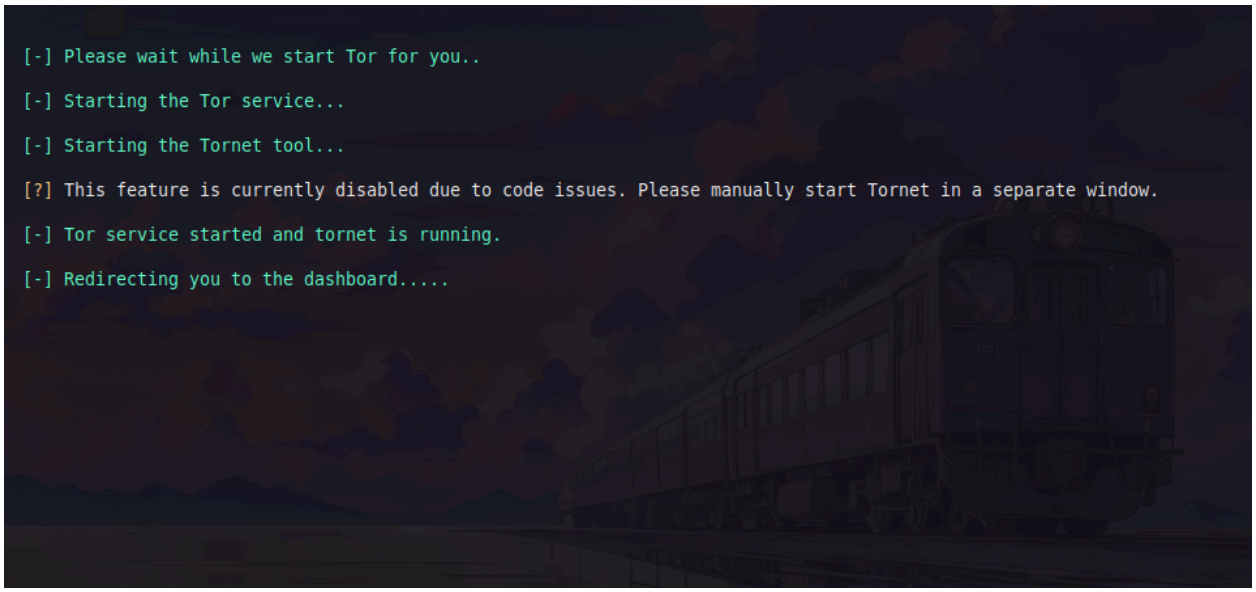
Tor Service Integration:

- **Enable Tor:** Option to activate Tor service for anonymizing traffic.
- **Tornet Tool:** Automatically starts the Tornet tool, enabling IP hopping with an interval of 5 seconds to enhance anonymity and evade detection.

Safe Exit:

- **Exit Option:** Provides a safe exit functionality that closes all Tor and Tornet services and exits the application. This ensures that the system can be used smoothly for future operations without leaving any services running.

Inbuilt IP Hopping Feature:



The inbuilt IP hopping feature of FloodReaper enhances anonymity and makes it difficult for targets to trace the origin of the attacks. This feature allows your IP address to change at regular intervals automatically.

Activating IP Hopping

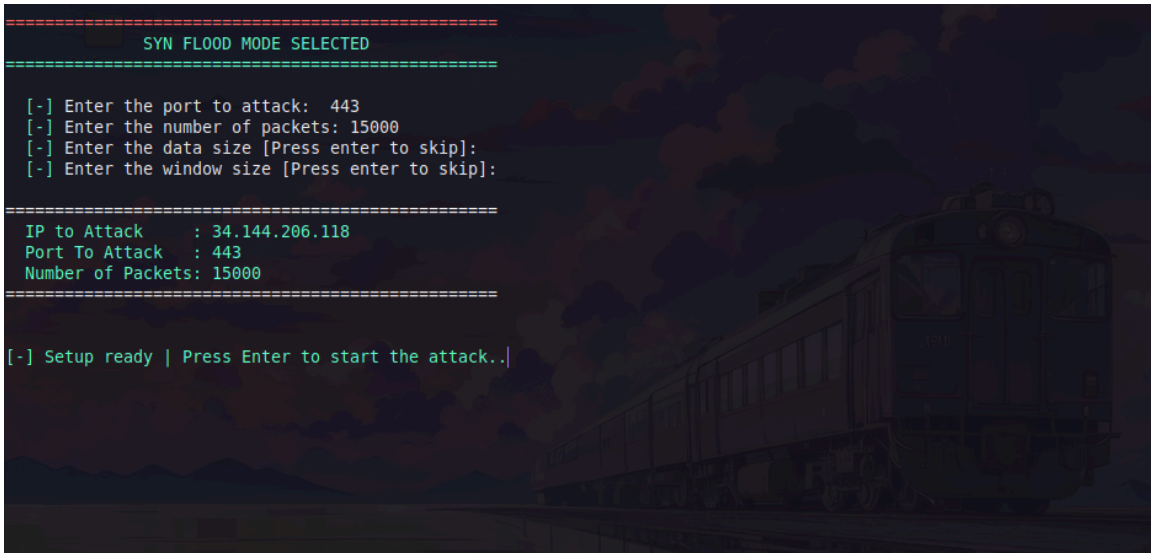
If you observe that IP hopping is inactive in your script, you can easily activate it by following these steps:

1. **Select IP Hopping Option:** From the dashboard panel, choose the **'Start Tor Service [IP HOPPING]'** option.
2. **Start Tor Service:** This option initiates the Tor service, which routes your traffic through multiple relays, anonymizing your connectio**n.**
3. **Launch Tornet Tool:** Alongside Tor, the Tornet tool will start with a configuration to change your IP address every 5 seconds. The interval is set to 5 seconds with a count of 0, ensuring continuous IP hopping without a predefined limit.

Troubleshooting

In case the IP hopping feature is disabled due to code issues, you can manually start the Tornet tool in a separate window to achieve the same effect. Follow the manual instructions provided in the tool’s documentation for detailed steps.

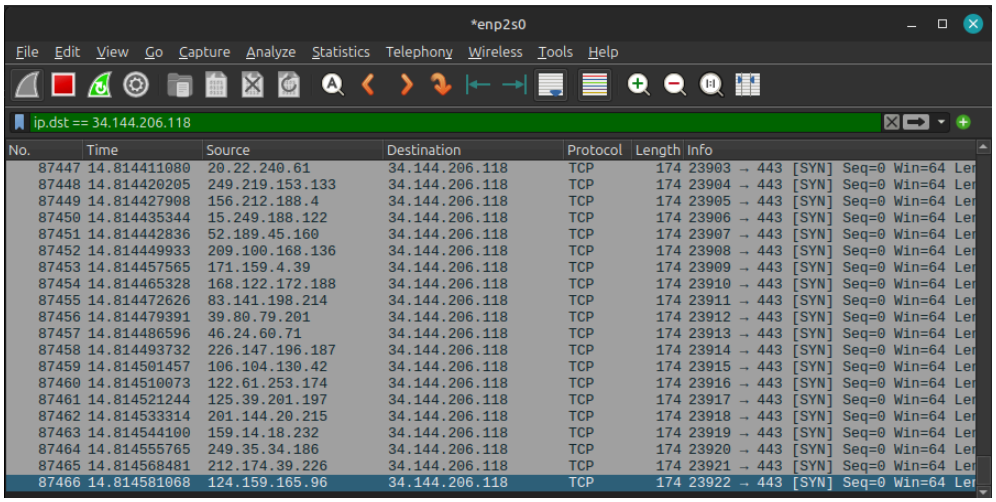
SYN FLOOD ATTACK:



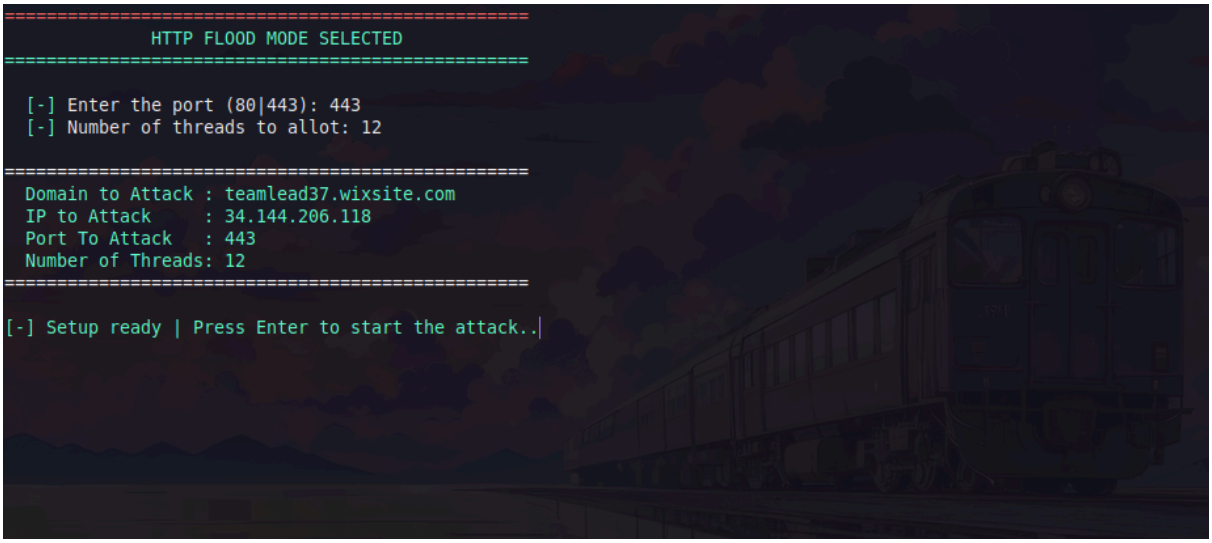
What is SYN FLOOD ATTACK?

A **SYN flood attack** is a type of Denial of Service (DoS) attack that overwhelms a target system by exploiting the TCP handshake process. The attacker sends numerous SYN (synchronize) packets to the target, but either uses spoofed IP addresses or doesn't complete the handshake with an ACK (acknowledge) packet. This causes the target server to allocate resources for half-open connections, eventually exhausting its capacity and preventing legitimate connections**.**

For this attack, our script utilizes the hping3 tool. It takes the user-specified parameters, passes them to hping3, and initiates the attack also with randomized source, sending around 10,000 packets per second. The effects can be observed using Wireshark



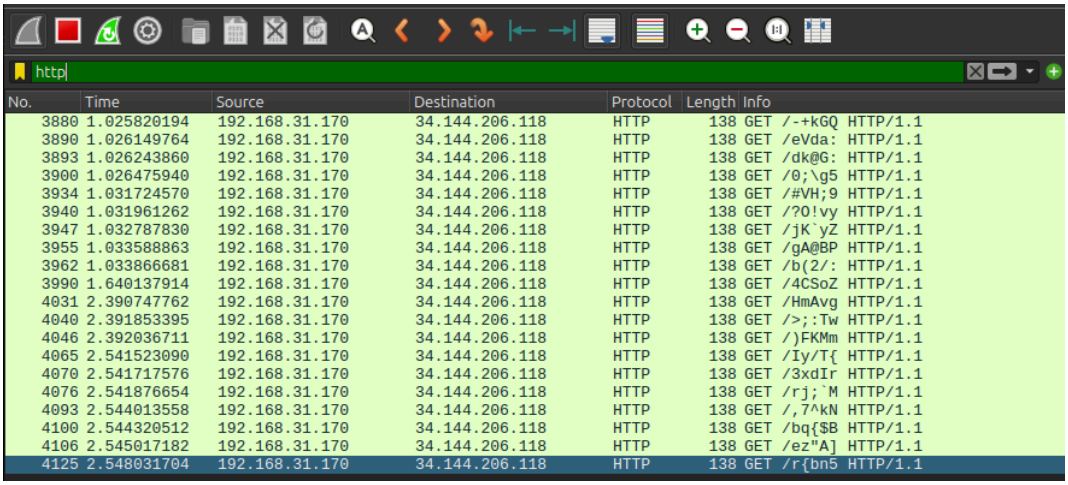
HTTP FLOOD ATTACK:



What is HTTP FLOOD ATTACK?

An **HTTP flood attack** is a type of Distributed Denial of Service (DDoS) attack that overwhelms a target server by sending a large number of HTTP GET or POST requests. The goal is to exhaust the server's resources, making it unable to respond to legitimate traffic.

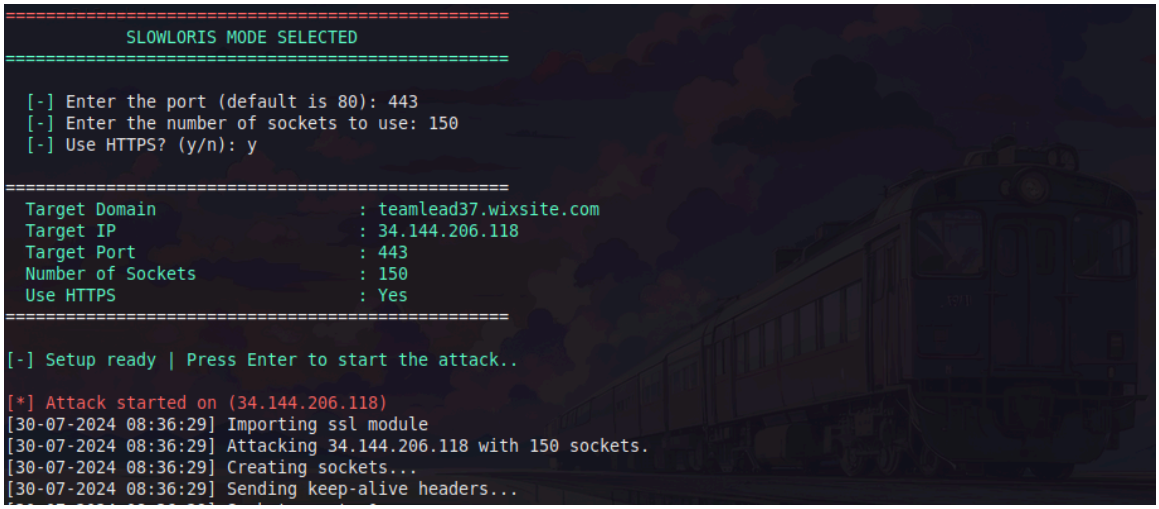
For this attack, our script uses a Python-based HTTP request library. It takes user-specified parameters, generates a high volume of HTTP GET requests, and sends them to the target, typically sending hundreds of requests per second depending on number of threads. The effects can be observed using Wireshark.



http

No.	Time	Source	Destination	Protocol	Length	Info
3880	1.025820194	192.168.31.170	34.144.206.118	HTTP	138	GET /-+kGQ HTTP/1.1
3890	1.026149764	192.168.31.170	34.144.206.118	HTTP	138	GET /eVda: HTTP/1.1
3893	1.026243860	192.168.31.170	34.144.206.118	HTTP	138	GET /dk@G: HTTP/1.1
3900	1.026475940	192.168.31.170	34.144.206.118	HTTP	138	GET /0;\g5 HTTP/1.1
3934	1.031724570	192.168.31.170	34.144.206.118	HTTP	138	GET /#VH;9 HTTP/1.1
3940	1.031961262	192.168.31.170	34.144.206.118	HTTP	138	GET /?O!vy HTTP/1.1
3947	1.032787830	192.168.31.170	34.144.206.118	HTTP	138	GET /jK`yZ HTTP/1.1
3955	1.033588863	192.168.31.170	34.144.206.118	HTTP	138	GET /gA@BP HTTP/1.1
3962	1.033866681	192.168.31.170	34.144.206.118	HTTP	138	GET /b(2/: HTTP/1.1
3990	1.640137914	192.168.31.170	34.144.206.118	HTTP	138	GET /4CSoZ HTTP/1.1
4031	2.390747762	192.168.31.170	34.144.206.118	HTTP	138	GET /HmAvg HTTP/1.1
4040	2.391853395	192.168.31.170	34.144.206.118	HTTP	138	GET />;:Tw HTTP/1.1
4046	2.392036711	192.168.31.170	34.144.206.118	HTTP	138	GET /}FKMm HTTP/1.1
4065	2.541523090	192.168.31.170	34.144.206.118	HTTP	138	GET /Iy/T{ HTTP/1.1
4070	2.541717576	192.168.31.170	34.144.206.118	HTTP	138	GET /3xdIr HTTP/1.1
4076	2.541876654	192.168.31.170	34.144.206.118	HTTP	138	GET /rj;`M HTTP/1.1
4093	2.544013558	192.168.31.170	34.144.206.118	HTTP	138	GET /,7^kN HTTP/1.1
4100	2.544320512	192.168.31.170	34.144.206.118	HTTP	138	GET /bq{$B HTTP/1.1
4106	2.545017182	192.168.31.170	34.144.206.118	HTTP	138	GET /ez"A] HTTP/1.1
4125	2.548031704	192.168.31.170	34.144.206.118	HTTP	138	GET /r{bn5 HTTP/1.1

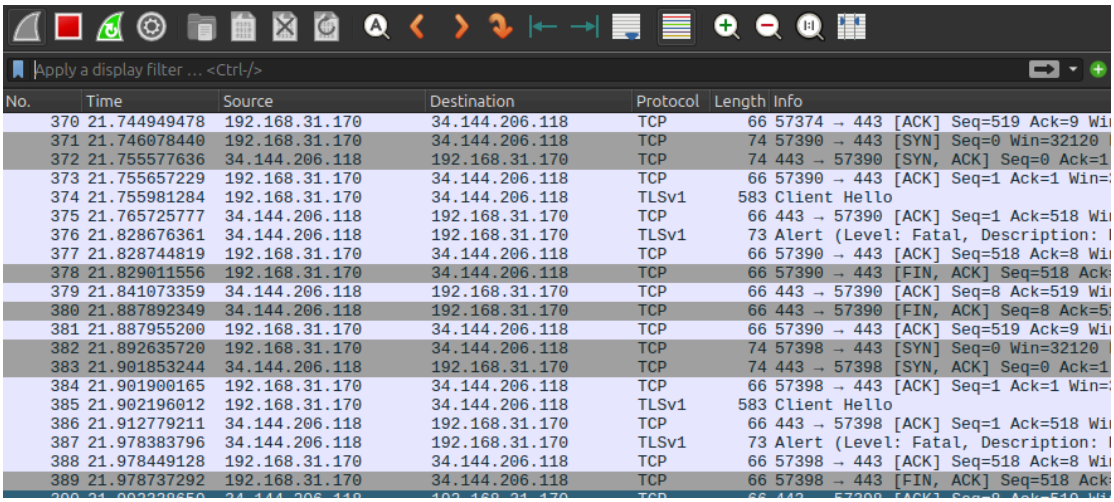
SLOWLORIS ATTACK:



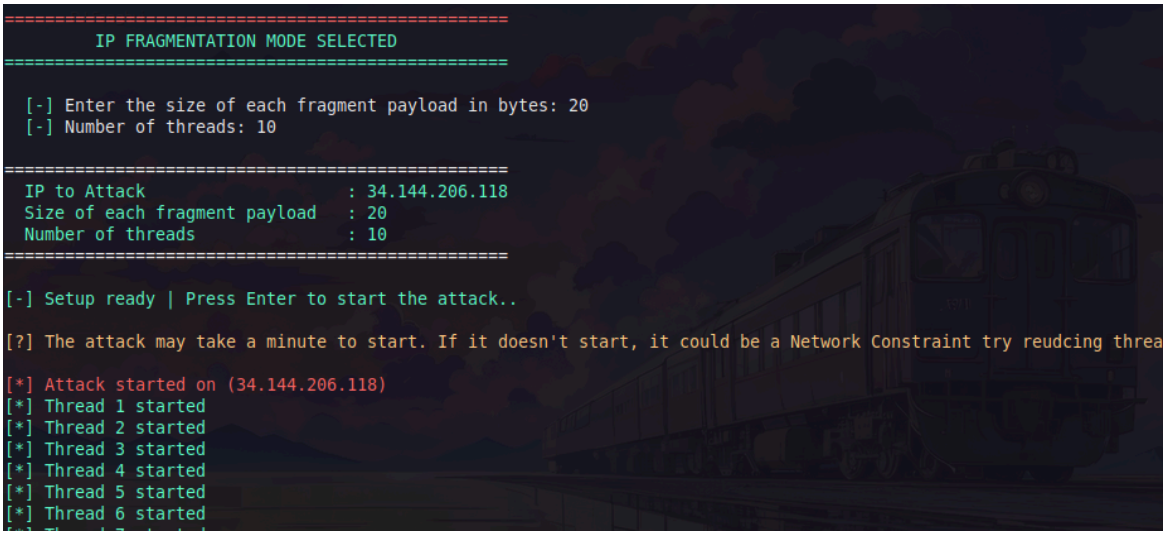
What is SLOWLORIS ATTACK?

A **Slowloris attack** is a type of Denial of Service (DoS) attack that aims to keep many connections to the target web server open and hold them open as long as possible. This exhausts the server's available connections, preventing legitimate users from accessing the server.

For this attack, our script uses the Slowloris tool. It takes user-specified parameters, opens multiple connections to the target server, and sends partial HTTP requests at regular intervals to keep these connections open. This can tie up the server's resources indefinitely. The effects can be observed using Wireshark.



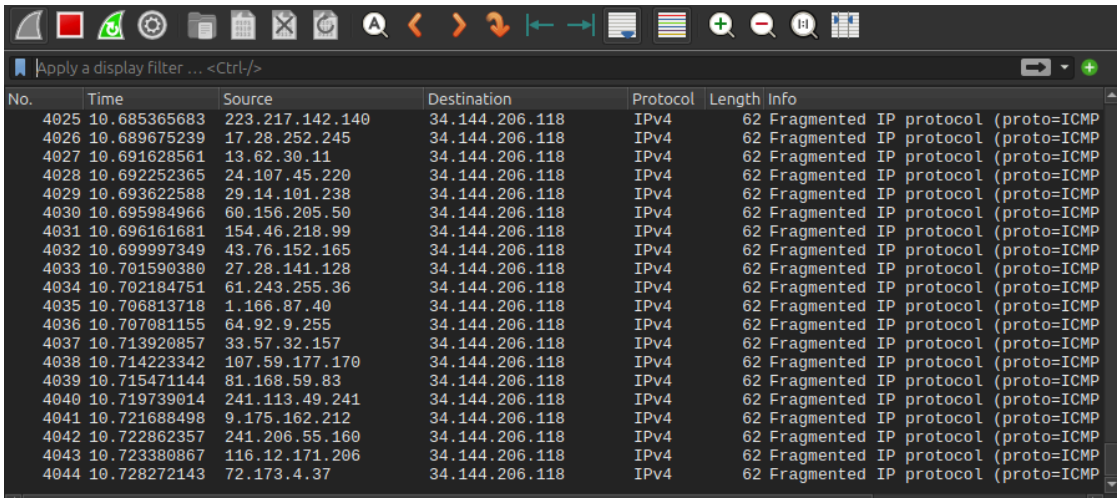
IP FRAGMENTATION ATTACK:



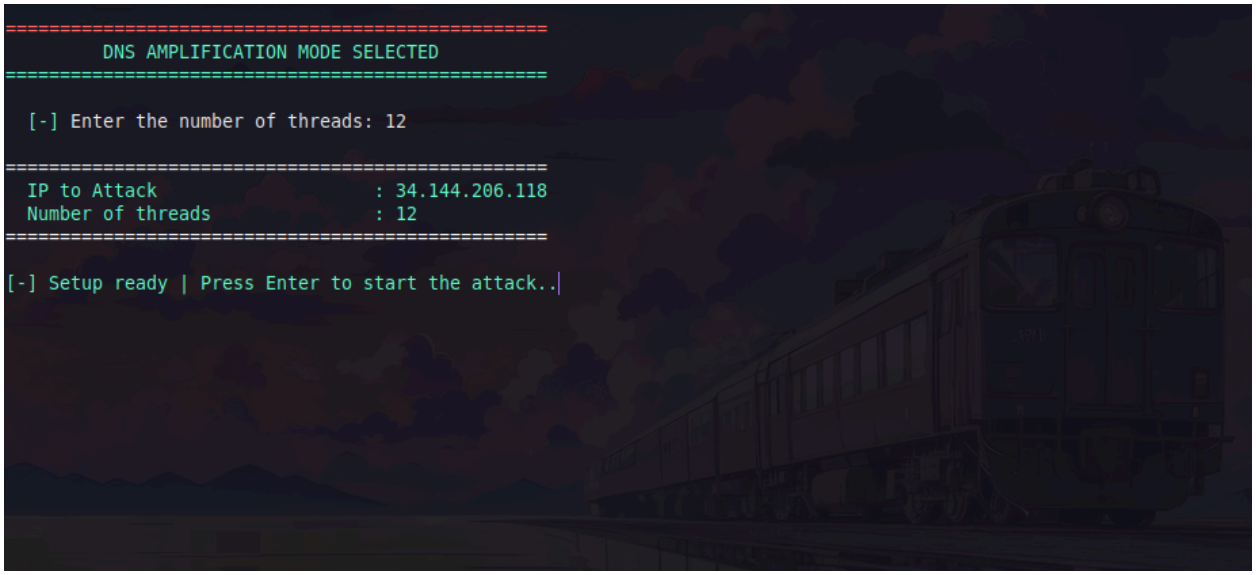
What is IP FRAGMENTATION ATTACK?

An **IP fragmentation attack** aims to overwhelm a target by sending fragmented IP packets, which can disrupt the target's ability to reassemble and process these packets.

For this attack, our script utilizes Python libraries to generate and send fragmented IP packets. It takes user-specified parameters to control the size and number of fragment also it generates random source ip address to mask the real identity . The high volume of fragmented traffic can be observed and analyzed using Wireshark.



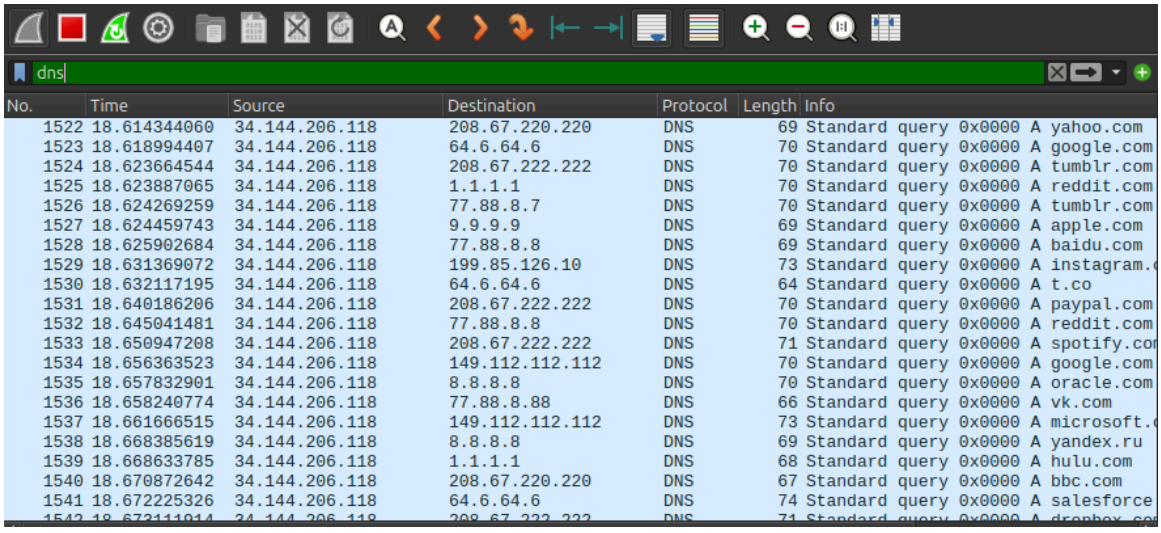
DNS AMPLIFICATION ATTACK:



What is DNS AMPLIFICATION ATTACK?

A **DNS amplification attack** leverages open DNS resolvers to flood a target with a large volume of traffic by amplifying DNS query responses.

For this attack, our script uses Python libraries to send randomized DNS queries to a set of open resolvers. Each query is crafted with the target IP address as the source, causing the DNS resolvers to send large response packets to the target. This results in a high volume of amplified traffic, which can be monitored and analyzed using Wireshark.



Understanding the code:

Importing Necessary Libraries:

```
import os , subprocess ,sys
import urllib.parse,ipaddress
import socket,string,time
from random import randint , choice ,random
import threading
from scapy.all import IP, ICMP, send ,conf,UDP, DNS, DNSQR
```

- **os, subprocess, sys:** These libraries are used for interacting with the operating system, executing system commands, and handling system-level operations.
- **urllib.parse, ipaddress:** These modules help with URL parsing and IP address manipulation.
- **socket, string, time:** Used for network communications, string operations, and time management.
- **random:** Provides methods for generating random numbers and selecting random elements.
- **threading:** Enables concurrent execution of functions using threads.
- **scapy.all:** A powerful library for network packet manipulation, including crafting and sending packets (IP, ICMP, UDP, DNS) and analyzing network traffic.

Setting global Variables

```
is_dig_avail = False
domain =  ''
target_ip = '0.0.0.0'
is_ip_hopping = False
conf.verb = 0
```

By declaring these variables as global, any function can access and modify them directly using the global keyword, which simplifies data sharing and coordination across different parts of the script.

Initialization System Check:

```
    if os.name == "posix"
```

`if os.name == "posix"`: Checks if the operating system is Unix-like (e.g., Linux or macOS), which is necessary because the script is designed for POSIX-compliant systems.

```
    if os.geteuid() == 0:
```

`if os.geteuid() == 0:`: Verifies that the script is running with root (superuser) privileges, which are required for certain network operations and system commands.

```
    subprocess.run(['dig', '-v']
```

`subprocess.run(['dig', '-v'])`: Runs the `dig` command to check its version, confirming if the DNS query tool is installed on the system.

```
    subprocess.run(['hping3', '--version'],
    subprocess.run(['sudo', 'apt', 'install', 'hping3', '-y']
```

`subprocess.run(['hping3', '--version'])`: Executes the `hping3` command to check its version, ensuring that the packet crafting tool for SYN flood attacks is available.

```
    subprocess.run(['pip', 'show', 'tornet']
    run(['pip','install', 'tornet','--root-user-action=ignore']
```

`subprocess.run(['pip', 'show', 'tornet'])`: Uses `pip` to check if the `tornet` Python package is installed, which is needed for IP hopping functionality.

```
    subprocess.run(['tor', '--version']
    subprocess.run(['apt','install', 'tor']
```

`subprocess.run(['tor', '--version'])`: Checks the version of the `tor` command to ensure the Tor service is installed and available for IP hopping.

```
    result = subprocess.run(['slowloris', '-h']
    subprocess.run(['sudo', 'apt', 'install', 'slowloris', '-y']
```

`result = subprocess.run(['slowloris', '-h'])`: Runs the `slowloris` command with the help flag to check if the Slowloris tool is installed and functional.

All this completes our Initialization Checks

Understanding Functions:

```
def resolve_target_dig()
```

- The **resolve_target_dig()** function determines the target based on the user's input, checking whether an IP address or URL is provided.
- If an IP address is given, it is directly set as the target and the function proceeds to the dashboard. For URLs, the function extracts the domain using urllib.parse and resolves it to an IP address using the dig +short tool.
- If the dig command fails, the function calls resolve_target_ip() to allow manual IP entry as a fallback solution.
- This approach ensures that the target is accurately set and allows for troubleshooting if automatic resolution fails.

```
def resolve_target_ip()
```

- The resolve_target_ip() function serves as a fallback when the resolve_target_dig() function fails to resolve the target using the dig tool.
- This function prompts the user to manually enter the IP address of the URL.
- It provides a straightforward method for users to specify the target IP directly if automated resolution does not work, ensuring that the tool can still proceed with the specified IP address.

```
def start_tor()
```

- The start_tor() function is responsible for initiating the Tor service to facilitate IP hopping and enhance anonymity.
- It first starts the Tor service and then waits for a few seconds to ensure that the service has fully initiated. Subsequently, it attempts to start the tornet tool, which is designed to manage IP hopping.
- However, this part has been commented out in the current implementation due to issues it was causing in the code. When operational, the function helps maintain anonymity by frequently changing IP addresses through the Tor network.

```
def exit()
```

- The exit() function, while seemingly simple, is crucial for proper tool operation.
- It handles the termination of both the Tor and tornet services, ensuring that these processes are cleanly shut down when you are finished using the tool.
- This function ensures that no lingering services remain, allowing you to smoothly transition away from the tool without any residual effects or conflicts.

Understanding Attacks:

```
def syn_flood()
```

- The syn_flood() function initiates a SYN flood attack using the hping3 tool. First, it retrieves the locked target IP address.
- The function then prompts the user for the port number, window size, and data size, providing default values if any of these parameters are omitted.
- It constructs the appropriate hping3 command with the specified parameters: --count for the number of packets, --data for the data size, --syn for sending SYN packets, --win for the window size, -p for the port, and --flood for continuous packet sending. The --rand-source option randomizes the source IP address to help obscure the origin of the attack.
- Finally it starts the attack sending over 10K packets per second

```
def http_flood()
```

- The http_flood() function performs an HTTP flood attack by sending a large volume of GET requests to the target.
- If the target is specified by IP address, the function prompts the user for a URL to extract the website domain; if a URL is already provided, this step is skipped.
- It then collects the port number and the number of threads to use for the attack.
- A socket is created using the target IP address and port to facilitate communication.
- The function includes a nested url_path function that generates random URLs to include in the GET requests.
- Finally, it sends GET requests in the format: f"GET /{url_path} HTTP/1.1\r\nHost: {domain}\r\nConnection: close\r\n\r\n", targeting the specified domain and port with randomized paths to simulate a flood of HTTP requests.

```
def ip_frag()
```

- The ip_frag() function executes an IP fragmentation attack by first prompting the user for the fragment size, number of fragments, and the number of threads.
- It generates a payload of repeated characters ('x') with the specified fragment size and creates a random source IP address.
- The function then constructs an IP packet with the target IP and random source IP, fragments the payload into smaller pieces, and sends these fragments to the target.
- This attack aims to disrupt the target by overwhelming it with fragmented packets, complicating the reassembly process.

```
def slowloris_attack()
```

- The slowloris() function initiates a Slowloris attack by first prompting the user for the target port, the number of sockets to use, and whether to use HTTPS.
- It then crafts the appropriate command for the Slowloris tool based on the provided inputs.
- Finally, the function executes this command to start the attack, which aims to exhaust the server's resources by keeping many connections open with partial HTTP headers, thus disrupting its ability to handle legitimate requests.

```
def dns_amp()
```

- The dns_amp() function performs a DNS amplification attack. It begins by prompting the user for the number of threads to use and then locks in the target IP and thread count.
- The function uses a predefined list of DNS servers (e.g., 8.8.8.8 and 1.1.1.1) and a set of known domains (e.g., google.com and netflix.com). It creates DNS query packets with the target IP as the source.
- The function sends these queries to the DNS servers, which then respond with amplified DNS responses, flooding the target IP with responses containing the IP addresses of the queried domains.
- This amplifies the traffic directed at the target, overwhelming it with excessive DNS response data.

FloodReaper is a powerful and versatile tool for stress testing and educational purposes, demonstrating various network attack techniques. With features like IP hopping, multithreading, and multiple attack methods, it offers a comprehensive platform for understanding offensive strategies in network security. Ensure that FloodReaper is used responsibly and within legal boundaries to maximize its effectiveness for research and learning.