



SWINBURNE  
UNIVERSITY OF  
TECHNOLOGY

**Swinburne University of Technology**  
*Faculty of Science, Engineering and Technology*

**ASSIGNMENT AND PROJECT COVER SHEET**

Unit Code: COS3005 Unit Title: IT Security

Assignment number and title: Practical Project (Assignment 1) Due date: 05/09

Lab group: \_\_\_\_\_ Tutor: Yasas Akurudda Liyanage Don Lecturer: \_\_\_\_\_

Family name: Minh Hoang Identity no: 104487115

Other names: Duong

**To be completed if this is an INDIVIDUAL ASSIGNMENT**

I declare that this assignment is my individual work. I have not worked collaboratively, nor have I copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for me by another person.

Signature: \_\_\_\_\_

**To be completed if this is a GROUP ASSIGNMENT**

We declare that this is a group assignment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

ID Number	Name	Signature
_____	_____	_____
_____	_____	_____

Marker's comments:

Total Mark: \_\_\_\_\_

**Extension certification:**

This assignment has been given an extension and is now due on \_\_\_\_\_

Signature of Convener: \_\_\_\_\_ Date: \_\_\_\_\_ / 2024

# COS3005 – IT Security

*Practical Assignment (Assignment 1)*

**Topic:** *Denial of Service (DoS)*

**Student:** *Minh Hoang Duong (104487115)*

**Student Name:** *Minh Hoang Duong*

**Student ID:** *104487115*

**Word Count:** *(excluding reference, coversheet, title page, and multiple titles)*

**Due Date:** *05/09/2024*

**Submission Date:** *05/09/2024*

# I. Criteria 1: Planning and Justification

## 1. Overview:

**Denial-of-Service (DoS)**<sup>[1]</sup> is a kind of malicious cyber threat in which the threat actor's main objective is to render computers, devices, network infrastructure, or websites unavailable to legitimate users. Taking advantage of the limited capacities that apply to any type of network, or computing infrastructures, the attackers flood the infrastructure resources with traffic or requests, making the system unable to process legitimate traffic or simply crashing due to running out of computation resources. As a result, the targeted service became unresponsive or significantly slower, causing disruptions in user experience and multiple damages to the targeted organization and owners.

**Distributed Denial-of-Service (DDoS)**<sup>[1]</sup> is a type of DoS intrusion that originates from many distributed sources. The attackers often operate multiple botnets (compromised devices) to conduct a big-scale DoS attack on the target.

## 2. Type of DoS attacks:

While every type of DoS attack has the same objective is to disrupt the target's service, each type of DoS attack has its unique characteristics and methods. In terms of categorizing, there are three types of DoS attacks <sup>[2]</sup>:

- **Volume-based attack** <sup>[2]</sup>: a kind of DoS attack that includes flooding one server's bandwidth with requests or traffic. For instance, ICMP/ping flood <sup>[2]</sup>.
- **Protocol attack** <sup>[2]</sup>: a kind of DoS attack that takes advantage of the nature of the set of rules defined in internet protocol, often works at layer 2 or 3 of the OSI model. For example, SYN flood, SYN-ACK flood <sup>[2]</sup>.
- **Application layer attack** <sup>[2]</sup>: a type of DoS attack that focuses on the application layer (layer 7) and its protocol. For instance, HTTP flood <sup>[2]</sup>.

Even though multiple types of DoS attacks could be classified into multiple categories, the threat actors could combine multiple DoS methods, and create multiple attacks, which renders many difficulties for the defender.

## 3. Threat Justification, Impacts, and Case Study:

The impacts of the DoS attack could range from temporary service disruption to severe physical damage (high heat in computational devices) leading to financial losses, affecting user experience and the owner's, organization's reputation. On one hand, a DoS attack can also act as a distraction while other types of malicious activities are being carried out, leading to further damage <sup>[6]</sup>.

Due to the fact that this type of malicious cyber-attack exploits the natural weakness that is applied to any network, or computational infrastructure, and is arguably easy to execute as there are many tools that support DoS and stress-testing features (LOIC, HOIC, T50, h3ping,...), this type of malicious intrusion is highly disruptive and popular in the field, ranging from highly skilled malicious attackers, penetration testers to hacktivist, cyber-criminal and script kiddies, reflecting through the following famous cases in the past.

**The 2018 GitHub Attack** <sup>[9]</sup>: On Wednesday, February 28<sup>th</sup>, 2018 the GitHub.com website, one of the largest source code management tools was rendered unavailable from 5:21 pm to 5:26 pm. The attackers took advantage of the Memcached instances, which are accessible through the public internet and support UDP communications, flooding them requests with spoofed IP addresses. By spoofing the IP address, the attacker could allow memcached's responses to be diverted to the Github.com IP address and send more data toward that server, amplifying up to 51000 times<sup>[9]</sup>.

**The 2016 Dyn Attack** <sup>[11]</sup>: On October 21<sup>st</sup>, 2016, a botnet DDoS attack was launched toward Dyn (now Oracle), a company that serves the majority of DNS hosting infrastructure, resulting in a severe outage of many of largest services including Paypal<sup>[12]</sup>, Netflix<sup>[12]</sup>, Reddit<sup>[12]</sup>, Twitter<sup>[12]</sup>,... The threat actors utilized the "Mirai Botnet", which infected up to 100,000 IoT devices<sup>[11]</sup> and sent multiple requests.

**The 2007 Estonia Attack** <sup>[13]</sup>: In April 2007, a series of cyber-attacks were launched on websites of Estonian organizations and services including the Estonian Government, Banks, and Ministries. The intrusion is a multiple vector attack ranging from ping floods to botnet and DDoS attacks.

In addition, there has been an increase in DoS in 2024 as Cloudflare reported mitigating 4.5 million cases in Q1 and 4 million in Q2 <sup>[4]</sup>, compared with 14 million DDoS attacks for the whole of 2023 <sup>[4]</sup>, which is considered a 20% year-over-year increase <sup>[4]</sup>. According to Netscout analysis, there have been 7,035,170 attacks at the time of writing (7:38 pm, 29/08/2024) <sup>[5]</sup> with multiple attacking vectors such as SYN flooding, ACK flooding, UDP flooding, RIPv1 AMP <sup>[5]</sup>.

Therefore, even though DoS is an arguably simple type of malicious attack, it still plays a crucial role in the open-vast cyber security world. As a result, it has been chosen to be the topic of this practical report.

#### 4. Tools Evaluation and Justification

##### Offensive tool

Tool/Activities	Ease of Install	Amount of documentation	Community Activity	Available Features
Slowloris <sup>[7]</sup>	<ul style="list-style-type: none"> <li>Simple installation.</li> <li>Required pre-installed python</li> </ul>	<ul style="list-style-type: none"> <li>Fairly documented.</li> </ul>	<ul style="list-style-type: none"> <li>Open-source.</li> </ul>	<ul style="list-style-type: none"> <li>Opening and maintaining many simultaneous HTTP connections.</li> </ul>
T50 <sup>[8]</sup>	<ul style="list-style-type: none"> <li>Simple installation.</li> <li>Pre-installed in Kali Linux.</li> <li>Available in APT and YUM for Linux systems.</li> </ul>	<ul style="list-style-type: none"> <li>Well documented</li> <li>Fairly large official Documentation base.</li> </ul>	<ul style="list-style-type: none"> <li>Open-source.</li> <li>Barely active community.</li> </ul>	<ul style="list-style-type: none"> <li>Multiple-protocol packet injector.</li> <li>Supporting TCP, UDP, ICMP, IGMPv2,...<sup>[8]</sup></li> <li></li> </ul>
Low Orbit Ion Cannon (LOIC) <sup>[9]</sup>	<ul style="list-style-type: none"> <li>Simple installation.</li> <li>Cross-platform.</li> </ul>	<ul style="list-style-type: none"> <li>Well documented.</li> <li>Many user-based guides.</li> </ul>	<ul style="list-style-type: none"> <li>Open-source</li> <li>Barely active community.</li> </ul>	<ul style="list-style-type: none"> <li>User Interface.</li> <li>TCP, UDP, HTTP floods.</li> </ul>

Due to ease of use, simple installation compared to LOIC, and capabilities with the attacking method the chosen tool for this testing scenario is **Slowloris**.

## Defensive tool

Tool/Activities	Ease of Install	Amount of documentation	Community Activity	Available Features
Snort <sup>[14]</sup>	<ul style="list-style-type: none"> <li>Required installation.</li> </ul>	<ul style="list-style-type: none"> <li>Well documented.</li> <li>Public official documentation based.</li> <li>Many user-based guides</li> </ul>	<ul style="list-style-type: none"> <li>Open-source.</li> <li>Highly active community.</li> <li>Frequent update.</li> </ul>	<ul style="list-style-type: none"> <li>Packet sniffing.</li> <li>Snort rules.</li> <li>Network intrusion prevention.</li> </ul>
iptables <sup>[15]</sup>	<ul style="list-style-type: none"> <li>Part of the Linux system.</li> <li>Based on Firewall.</li> </ul>	<ul style="list-style-type: none"> <li>Well documented.</li> <li>Large public official documentation based.</li> <li>Many user-based guides</li> </ul>	<ul style="list-style-type: none"> <li>Open-source.</li> <li>Fairly active community.</li> </ul>	<ul style="list-style-type: none"> <li>Managing incoming, and outgoing packets.</li> <li>Blocking, allowing traffic.</li> </ul>
Wireshark <sup>[16]</sup>	<ul style="list-style-type: none"> <li>Pre-installed in Kali Linux</li> <li>Simple installation</li> </ul>	<ul style="list-style-type: none"> <li>Well documented.</li> <li>Large public documentation</li> <li>Many user-based guides</li> </ul>	<ul style="list-style-type: none"> <li>Open-source.</li> <li>Highly active community.</li> </ul>	<ul style="list-style-type: none"> <li>Packet sniffing, tracking, analyzing.</li> <li>User-interface.</li> </ul>

In the scenario, **Iptables** is also used for mitigation due to the ease of usage and availability. In addition, **Wireshark** is used for monitoring, detection, and protocol, packet tracking due to its availability, ease of installation, and ease of usage on the defender sides.

## 5. Scenario proposal

The scenario is based on a type of Layer 7 DoS attack known as **Slowloris Attack**.

**Slowloris Attack**<sup>[17]</sup> is a type of layer 7 DoS attack in which the attackers exploit the behavior of HTTP communication. After establishing a reliable connection (TCP) between the host and the user, the user's machine will send a request host's server, and the host's server will open a thread for each incoming request. Exploiting this process, the attacker will send a partial request without ending it, keeping the thread open and maintaining a simultaneous connection between the host and the attacker. To prevent the server from timing out, the attacker will periodically send an HTTP request header to keep the server up and prevent the target from handling legitimate requests. The attacks were originally developed by Robert Hansen (RSnake) in 2009<sup>[20]</sup>, and demonstrated by Sam Bowne at DEFCON 17<sup>[18]</sup>.

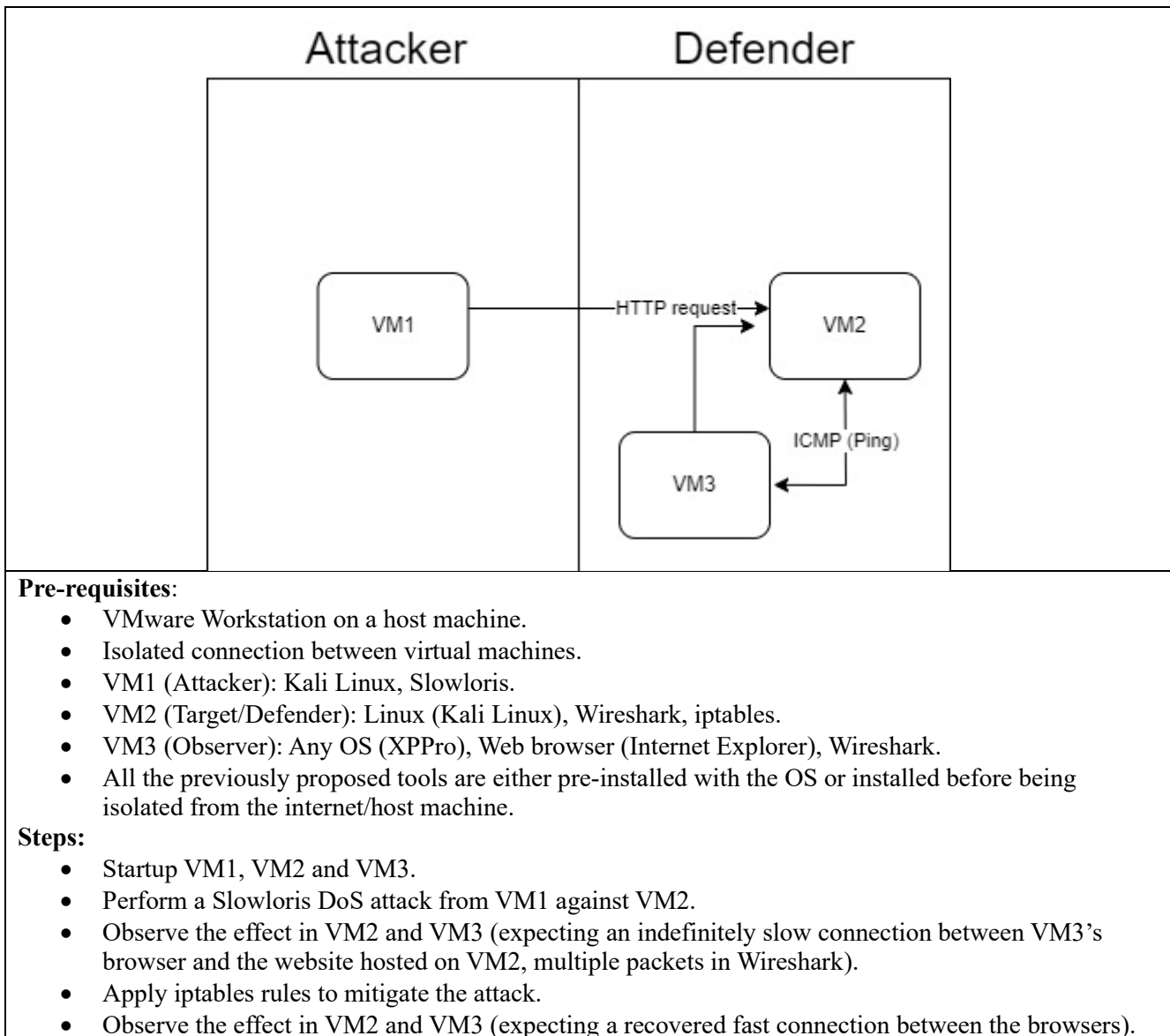
Based on the definition explained above, the attacker and defender aims are categorized as below:

- **The attacker (VM1):** Establishing HTTP communication with the host (VM2), keeping the connection up, and preventing the host (VM2) from processing requests from the observer (VM3).
- **The defender (VM2):** Restoring the connection from the observer (VM3).

The proposed scenario is carried out in an isolated environment which is created with VMware Workstation Pro, assuming each other has known their IP address and there will be no IP address spoofing.

VM1: 192.168.100.100.200/24

VM2: 192.168.100.183/24, VM3: 192.168.100.129/24



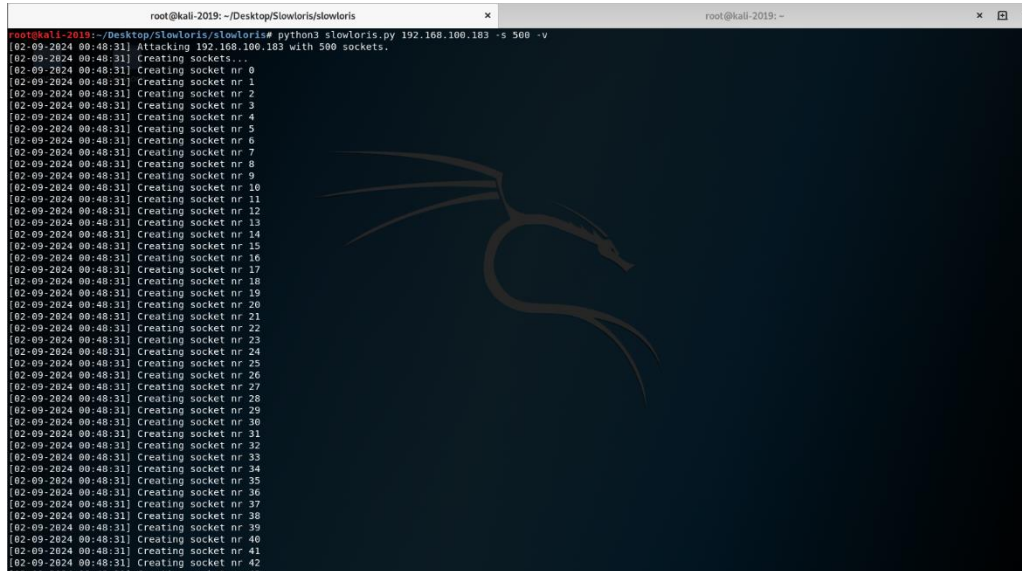
## II. Criteria 2: Application and Documentation

### 1. Attacker perspective (VM1)

#### - Initiating the attack process:

- The chosen attacking tool implements the original Slowloris attack developed by Robert Hansen (RSnake) in 2009 <sup>[20]</sup>. The tool is written by Gokberk Yaltirakli in 2015 <sup>[10]</sup>.
- By running “python3 slowloris.py <target IPv4>”, the attacking machine will execute the script in Python and start initiating the attack toward the target.
- The tool also features optional flags while running the command which can be easily found in the official GitHub repository <sup>[10]</sup>. In this scenario, there will be 2 additional flags used.

- “-s <number of sockets>” is the flag to specify the number of sockets used in the test. The intended sockets to be used in the test is 500.
- “-v” is the flag to enable “verbose mode” or to display more information about the attack.

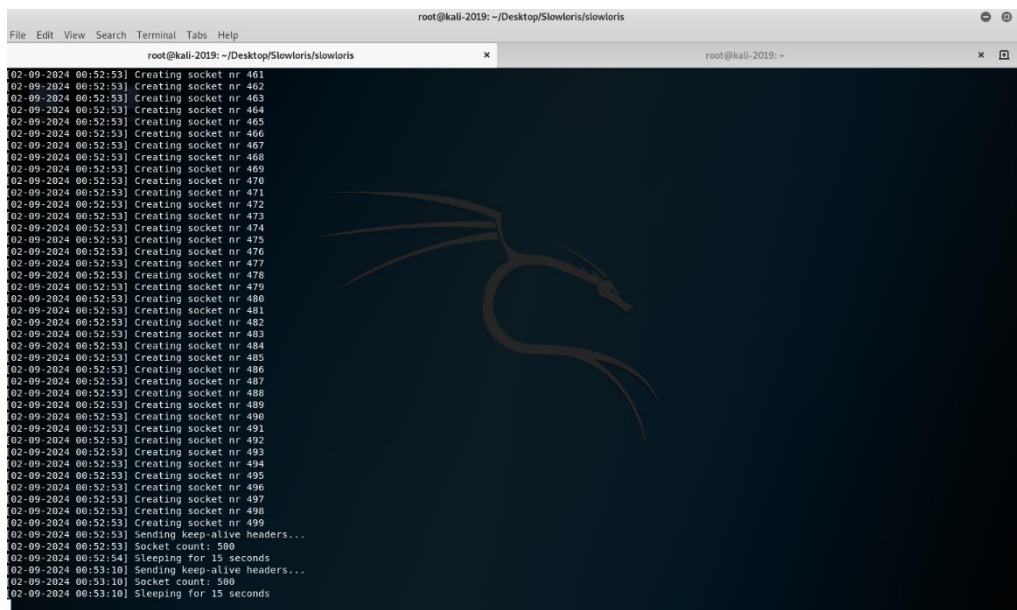


```

root@kali-2019: ~/Desktop/Slowloris/slowloris
root@kali-2019: ~/Desktop/Slowloris/slowloris# python3 slowloris.py 192.168.100.103 -s 500 -v
[02-09-2024 00:48:31] Attacking 192.168.100.103 with 500 sockets.
[02-09-2024 00:48:31] Creating sockets...
[02-09-2024 00:48:31] Creating socket nr 0
[02-09-2024 00:48:31] Creating socket nr 1
[02-09-2024 00:48:31] Creating socket nr 2
[02-09-2024 00:48:31] Creating socket nr 3
[02-09-2024 00:48:31] Creating socket nr 4
[02-09-2024 00:48:31] Creating socket nr 5
[02-09-2024 00:48:31] Creating socket nr 6
[02-09-2024 00:48:31] Creating socket nr 7
[02-09-2024 00:48:31] Creating socket nr 8
[02-09-2024 00:48:31] Creating socket nr 9
[02-09-2024 00:48:31] Creating socket nr 10
[02-09-2024 00:48:31] Creating socket nr 11
[02-09-2024 00:48:31] Creating socket nr 12
[02-09-2024 00:48:31] Creating socket nr 13
[02-09-2024 00:48:31] Creating socket nr 14
[02-09-2024 00:48:31] Creating socket nr 15
[02-09-2024 00:48:31] Creating socket nr 16
[02-09-2024 00:48:31] Creating socket nr 17
[02-09-2024 00:48:31] Creating socket nr 18
[02-09-2024 00:48:31] Creating socket nr 19
[02-09-2024 00:48:31] Creating socket nr 20
[02-09-2024 00:48:31] Creating socket nr 21
[02-09-2024 00:48:31] Creating socket nr 22
[02-09-2024 00:48:31] Creating socket nr 23
[02-09-2024 00:48:31] Creating socket nr 24
[02-09-2024 00:48:31] Creating socket nr 25
[02-09-2024 00:48:31] Creating socket nr 26
[02-09-2024 00:48:31] Creating socket nr 27
[02-09-2024 00:48:31] Creating socket nr 28
[02-09-2024 00:48:31] Creating socket nr 29
[02-09-2024 00:48:31] Creating socket nr 30
[02-09-2024 00:48:31] Creating socket nr 31
[02-09-2024 00:48:31] Creating socket nr 32
[02-09-2024 00:48:31] Creating socket nr 33
[02-09-2024 00:48:31] Creating socket nr 34
[02-09-2024 00:48:31] Creating socket nr 35
[02-09-2024 00:48:31] Creating socket nr 36
[02-09-2024 00:48:31] Creating socket nr 37
[02-09-2024 00:48:31] Creating socket nr 38
[02-09-2024 00:48:31] Creating socket nr 39
[02-09-2024 00:48:31] Creating socket nr 40
[02-09-2024 00:48:31] Creating socket nr 41
[02-09-2024 00:48:31] Creating socket nr 42

```

Figure 2.1.1: Executing “python3 slowloris.py <target IPv4> -s <number of socket> -v”



```

root@kali-2019: ~/Desktop/Slowloris/slowloris
root@kali-2019: ~/Desktop/Slowloris/slowloris# python3 slowloris.py 192.168.100.103 -s 500 -v
[02-09-2024 00:52:53] Creating socket nr 461
[02-09-2024 00:52:53] Creating socket nr 462
[02-09-2024 00:52:53] Creating socket nr 463
[02-09-2024 00:52:53] Creating socket nr 464
[02-09-2024 00:52:53] Creating socket nr 465
[02-09-2024 00:52:53] Creating socket nr 466
[02-09-2024 00:52:53] Creating socket nr 467
[02-09-2024 00:52:53] Creating socket nr 468
[02-09-2024 00:52:53] Creating socket nr 469
[02-09-2024 00:52:53] Creating socket nr 470
[02-09-2024 00:52:53] Creating socket nr 471
[02-09-2024 00:52:53] Creating socket nr 472
[02-09-2024 00:52:53] Creating socket nr 473
[02-09-2024 00:52:53] Creating socket nr 474
[02-09-2024 00:52:53] Creating socket nr 475
[02-09-2024 00:52:53] Creating socket nr 476
[02-09-2024 00:52:53] Creating socket nr 477
[02-09-2024 00:52:53] Creating socket nr 478
[02-09-2024 00:52:53] Creating socket nr 479
[02-09-2024 00:52:53] Creating socket nr 480
[02-09-2024 00:52:53] Creating socket nr 481
[02-09-2024 00:52:53] Creating socket nr 482
[02-09-2024 00:52:53] Creating socket nr 483
[02-09-2024 00:52:53] Creating socket nr 484
[02-09-2024 00:52:53] Creating socket nr 485
[02-09-2024 00:52:53] Creating socket nr 486
[02-09-2024 00:52:53] Creating socket nr 487
[02-09-2024 00:52:53] Creating socket nr 488
[02-09-2024 00:52:53] Creating socket nr 489
[02-09-2024 00:52:53] Creating socket nr 490
[02-09-2024 00:52:53] Creating socket nr 491
[02-09-2024 00:52:53] Creating socket nr 492
[02-09-2024 00:52:53] Creating socket nr 493
[02-09-2024 00:52:53] Creating socket nr 494
[02-09-2024 00:52:53] Creating socket nr 495
[02-09-2024 00:52:53] Creating socket nr 496
[02-09-2024 00:52:53] Creating socket nr 497
[02-09-2024 00:52:53] Creating socket nr 498
[02-09-2024 00:52:53] Creating socket nr 499
[02-09-2024 00:52:53] Sending keep-alive headers...
[02-09-2024 00:52:53] Socket count: 500
[02-09-2024 00:52:54] Sleeping for 15 seconds
[02-09-2024 00:53:10] Sending keep-alive headers...
[02-09-2024 00:53:10] Socket count: 500
[02-09-2024 00:53:10] Sleeping for 15 seconds

```

Figure 2.1.2: There were 500 sockets (0-499) being used in the attack.

- After establishing the sockets, the program will send the first “keeping alive” HTTP header before entering a 15s “sleeping” period as the explained tactic in the scenario proposal parts.
- While running, after every 15 seconds, the attacking script will display a report about the number of remaining sockets and send out another pack of “keeping alive” HTTP headers.

## - The attacking process after applying the defense mechanism

- After applying the defense mechanism from the defender's perspective, the attacking script started sending out “fail to create new socket: timed out” messages, and eventually, the sockets count number dropped to 20 (the connection limit specified below) as the program tried to create another 480 sockets for connection in the total of 500 sockets.

```

root@kali-2019: ~/Desktop/Slowloris/slowloris
[02-09-2024 23:52:26] Sleeping for 15 seconds
[02-09-2024 23:52:41] Sending keep-alive headers...
[02-09-2024 23:52:41] Socket count: 500
[02-09-2024 23:52:41] Creating 132 new sockets...
[02-09-2024 23:52:41] Sleeping for 15 seconds
[02-09-2024 23:52:56] Sending keep-alive headers...
[02-09-2024 23:52:56] Socket count: 500
[02-09-2024 23:52:56] Creating 24 new sockets...
[02-09-2024 23:53:11] Sleeping for 15 seconds
[02-09-2024 23:53:11] Sending keep-alive headers...
[02-09-2024 23:53:11] Socket count: 500
[02-09-2024 23:53:11] Creating 144 new sockets...
[02-09-2024 23:53:15] Failed to create new socket: timed out
[02-09-2024 23:53:15] Sleeping for 15 seconds
[02-09-2024 23:53:30] Sending keep-alive headers...
[02-09-2024 23:53:30] Socket count: 376
[02-09-2024 23:53:30] Creating 256 new sockets...
[02-09-2024 23:53:34] Failed to create new socket: timed out
[02-09-2024 23:53:34] Sleeping for 15 seconds
[02-09-2024 23:53:49] Sending keep-alive headers...
[02-09-2024 23:53:49] Socket count: 244
[02-09-2024 23:53:49] Creating 310 new sockets...
[02-09-2024 23:53:53] Failed to create new socket: timed out
[02-09-2024 23:53:53] Sleeping for 15 seconds
[02-09-2024 23:54:08] Sending keep-alive headers...
[02-09-2024 23:54:08] Socket count: 190
[02-09-2024 23:54:08] Creating 430 new sockets...
[02-09-2024 23:54:12] Failed to create new socket: timed out
[02-09-2024 23:54:12] Sleeping for 15 seconds
[02-09-2024 23:54:27] Sending keep-alive headers...
[02-09-2024 23:54:27] Socket count: 90
[02-09-2024 23:54:27] Creating 400 new sockets...
[02-09-2024 23:54:31] Failed to create new socket: timed out
[02-09-2024 23:54:31] Sleeping for 15 seconds
[02-09-2024 23:54:46] Sending keep-alive headers...
[02-09-2024 23:54:46] Socket count: 20
[02-09-2024 23:54:46] Creating 480 new sockets...
[02-09-2024 23:54:50] Failed to create new socket: timed out
[02-09-2024 23:54:50] Sleeping for 15 seconds
[02-09-2024 23:55:05] Sending keep-alive headers...
[02-09-2024 23:55:05] Socket count: 40
[02-09-2024 23:55:05] Creating 480 new sockets...
[02-09-2024 23:55:09] Failed to create new socket: timed out
[02-09-2024 23:55:09] Sleeping for 15 seconds
[02-09-2024 23:55:24] Sending keep-alive headers...
[02-09-2024 23:55:24] Socket count: 20
[02-09-2024 23:55:24] Creating 480 new sockets...

```

Figure 2.1.2: Connection timing out, creating 480 (500 – 20 = 480) new sockets message from the attacking script.

## 2. Defender perspective (VM2)

### - Wireshark observation before the attacks.

- The traffic remained normal with the TCP connects establishment and HTTP request from the observer (VM3), determined by the source and destination IPv4 address.

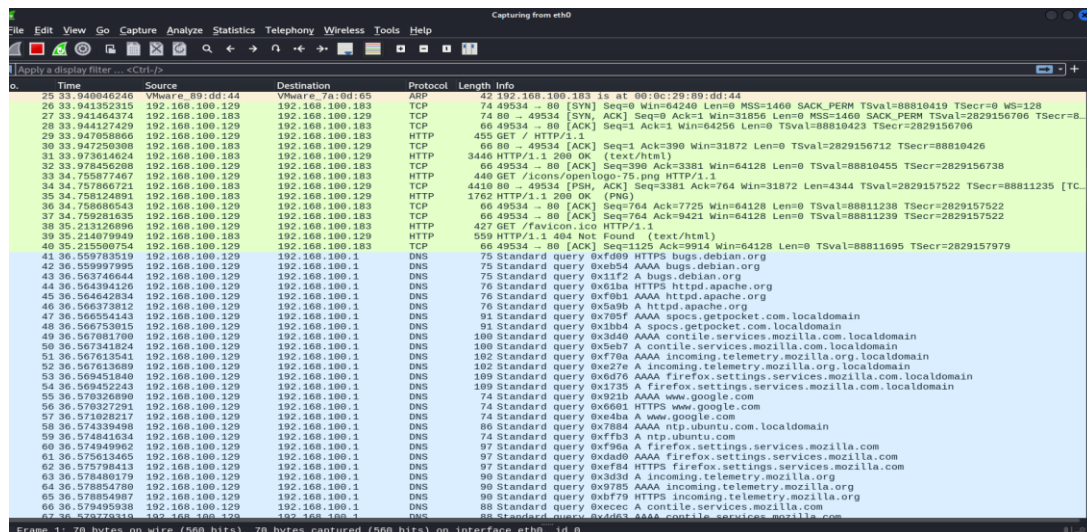


Figure 2.2.1: Traffic between the observer (VM3) and the defender (VM2).



- **Wireshark observation during the attacks, before applying defense mitigation.**
  - o During the first stage of the attack, Wireshark displayed numerous TCP establishment packets between the host (VM2) and the attacker (VM1).

No.	Time	Source	Destination	Protocol	Length	Info
3684	4.459943272	192.168.100.183	192.168.100.200	TCP	60	60 → 57788 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629983 TSecr=1321216800
3685	4.459692613	192.168.100.183	192.168.100.200	TCP	60	60 → 57790 [ACK] Seq=1 Ack=199 Win=31872 Len=0 TSval=2679629983 TSecr=1321216800
3686	4.459776665	192.168.100.183	192.168.100.200	TCP	60	60 → 57792 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629983 TSecr=1321216800
3687	4.459917346	192.168.100.183	192.168.100.200	TCP	60	60 → 57794 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629983 TSecr=1321216800
3688	4.459952906	192.168.100.183	192.168.100.200	TCP	60	60 → 57796 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629983 TSecr=1321216800
3689	4.460137683	192.168.100.183	192.168.100.200	TCP	60	60 → 57798 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629983 TSecr=1321216800
3690	4.460518327	192.168.100.200	192.168.100.183	TCP	77	57800 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216800 TSecr=2679628974
3691	4.460811729	192.168.100.200	192.168.100.183	TCP	77	57802 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628978
3692	4.460518859	192.168.100.200	192.168.100.183	TCP	77	57804 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628978
3693	4.460518966	192.168.100.200	192.168.100.183	TCP	77	57806 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628980
3694	4.460519105	192.168.100.200	192.168.100.183	TCP	77	57808 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628982
3695	4.460519385	192.168.100.200	192.168.100.183	TCP	77	57810 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628983
3696	4.460519860	192.168.100.200	192.168.100.183	TCP	77	57812 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628985
3697	4.460543139	192.168.100.183	192.168.100.200	TCP	60	80 → 57800 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629984 TSecr=1321216800
3698	4.460688013	192.168.100.183	192.168.100.200	TCP	60	80 → 57802 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629984 TSecr=1321216801
3699	4.460765736	192.168.100.183	192.168.100.200	TCP	60	80 → 57804 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629984 TSecr=1321216801
3700	4.460910252	192.168.100.183	192.168.100.200	TCP	60	80 → 57806 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629984 TSecr=1321216801
3701	4.460959355	192.168.100.183	192.168.100.200	TCP	60	80 → 57808 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629984 TSecr=1321216801
3702	REF	192.168.100.183	192.168.100.200	TCP	60	80 → 57810 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629984 TSecr=1321216801
3703	8.000076138	192.168.100.183	192.168.100.200	TCP	60	80 → 57812 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629984 TSecr=1321216801
3704	8.000491397	192.168.100.200	192.168.100.183	TCP	77	57814 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216801 TSecr=2679628988
3705	8.000491710	192.168.100.200	192.168.100.183	TCP	77	57816 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216802 TSecr=2679628989
3706	8.000491819	192.168.100.200	192.168.100.183	TCP	77	57818 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216802 TSecr=2679628990
3707	8.000492111	192.168.100.200	192.168.100.183	TCP	77	57820 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=10 TSval=1321216802 TSecr=2679628991
3708	8.000492244	192.168.100.200	192.168.100.183	TCP	77	57822 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216802 TSecr=2679628993
3709	8.000492379	192.168.100.200	192.168.100.183	TCP	77	57824 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216802 TSecr=2679628995
3710	8.000492505	192.168.100.200	192.168.100.183	TCP	77	57826 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216802 TSecr=2679628996
3711	8.000517101	192.168.100.183	192.168.100.200	TCP	60	80 → 57814 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629985 TSecr=1321216801
3712	8.000587870	192.168.100.183	192.168.100.200	TCP	60	80 → 57816 [ACK] Seq=1 Ack=199 Win=31872 Len=0 TSval=2679629985 TSecr=1321216802
3713	8.000714216	192.168.100.183	192.168.100.200	TCP	60	80 → 57818 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629985 TSecr=1321216802
3714	8.000789958	192.168.100.183	192.168.100.200	TCP	60	80 → 57820 [ACK] Seq=1 Ack=199 Win=31872 Len=0 TSval=2679629985 TSecr=1321216802
3715	8.000930123	192.168.100.183	192.168.100.200	TCP	60	80 → 57822 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629985 TSecr=1321216802
3716	8.001004976	192.168.100.183	192.168.100.200	TCP	60	80 → 57824 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629985 TSecr=1321216802
3717	8.001136280	192.168.100.183	192.168.100.200	TCP	60	80 → 57826 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629985 TSecr=1321216802
3718	8.001497690	192.168.100.200	192.168.100.183	TCP	76	57828 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=10 TSval=1321216803 TSecr=2679629900
3719	8.001498200	192.168.100.200	192.168.100.183	TCP	77	57830 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216803 TSecr=2679629903
3720	8.001498346	192.168.100.200	192.168.100.183	TCP	77	57832 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216803 TSecr=2679629905
3721	8.001498454	192.168.100.200	192.168.100.183	TCP	77	57834 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216803 TSecr=2679629906
3722	8.001498596	192.168.100.200	192.168.100.183	TCP	77	57836 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216803 TSecr=2679629910
3723	8.001498735	192.168.100.200	192.168.100.183	TCP	77	57838 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216803 TSecr=2679629912
3724	8.001498876	192.168.100.200	192.168.100.183	TCP	77	57840 → 80 [PSH, ACK] Seq=189 Ack=1 Win=29312 Len=11 TSval=1321216803 TSecr=2679629915
3725	8.001523156	192.168.100.183	192.168.100.200	TCP	60	80 → 57828 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=2679629986 TSecr=1321216803
3726	8.001675139	192.168.100.183	192.168.100.200	TCP	60	80 → 57830 [ACK] Seq=1 Ack=201 Win=31872 Len=0 TSval=2679629986 TSecr=1321216803

Figure 2.2.2: TCP establishment between each of the attacker (VM1: 192.168.100.200) sockets and the host (VM2: 192.168.100.183)

- o After that is HTTP timeout requests which is an effort of the defender to disconnect from the attacker as explained in the scenario proposal.
- o Wireshark also captured the HTTP GET method which is mentioned in the “keep alive” headers, as an effort of the attacker to keep the connection up against network timing out.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

eth0

Current filter: http

No.	Time	Source	Destination	Protocol	Length	Info
6092	63.835192772	192.168.100.200	192.168.100.183	TCP	66	59934 → 80 [ACK] Seq=211 Ack=484 Win=30336 Len=0 TSval=1322122776 TSecr=2680535912
6093	63.835988865	192.168.100.200	192.168.100.183	TCP	66	59938 → 80 [ACK] Seq=211 Ack=484 Win=30336 Len=0 TSval=1322122781 TSecr=2680535918
6094	63.835990248	192.168.100.200	192.168.100.183	TCP	66	59942 → 80 [ACK] Seq=210 Ack=484 Win=30336 Len=0 TSval=1322122781 TSecr=2680535917
6095	63.835990589	192.168.100.200	192.168.100.183	TCP	66	59946 → 80 [ACK] Seq=211 Ack=484 Win=30336 Len=0 TSval=1322122782 TSecr=2680535917
6096	63.843305546	192.168.100.200	192.168.100.183	TCP	66	59944 → 80 [ACK] Seq=209 Ack=484 Win=30336 Len=0 TSval=1322122785 TSecr=2680535922
6097	63.843774388	192.168.100.200	192.168.100.183	TCP	66	59940 → 80 [ACK] Seq=212 Ack=484 Win=30336 Len=0 TSval=1322122786 TSecr=2680535921
6098	64.814559295	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6099	64.814732548	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6100	64.814738452	192.168.100.183	192.168.100.200	TCP	60	80 → 59948 [FIN, ACK] Seq=483 Ack=209 Win=31872 Len=0 TSval=2680536939 TSecr=1322111425
6101	64.815026992	192.168.100.183	192.168.100.200	TCP	60	80 → 59946 [FIN, ACK] Seq=483 Ack=212 Win=31872 Len=0 TSval=2680536940 TSecr=1322111424
6102	64.816363206	192.168.100.183	192.168.100.183	TCP	66	59946 → 80 [ACK] Seq=212 Ack=483 Win=30336 Len=0 TSval=1322123758 TSecr=2680536939
6103	64.816363732	192.168.100.200	192.168.100.183	TCP	66	59948 → 80 [ACK] Seq=209 Ack=483 Win=30336 Len=0 TSval=1322123758 TSecr=2680536939
6104	64.820493750	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6105	64.820633838	192.168.100.183	192.168.100.200	TCP	60	80 → 59950 [FIN, ACK] Seq=483 Ack=209 Win=31872 Len=0 TSval=2680536945 TSecr=1322111425
6106	64.820836776	192.168.100.200	192.168.100.183	TCP	66	59950 → 80 [ACK] Seq=209 Ack=483 Win=30336 Len=0 TSval=1322123763 TSecr=2680536945
6107	64.823182374	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6108	64.823484390	192.168.100.183	192.168.100.200	TCP	60	80 → 59952 [FIN, ACK] Seq=483 Ack=212 Win=31872 Len=0 TSval=2680536948 TSecr=1322111425
6109	64.824410075	192.168.100.200	192.168.100.183	TCP	66	59952 → 80 [ACK] Seq=212 Ack=483 Win=30336 Len=0 TSval=1322123766 TSecr=2680536948
6110	64.841810899	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6111	64.842167086	192.168.100.183	192.168.100.200	TCP	60	80 → 59956 [FIN, ACK] Seq=483 Ack=212 Win=31872 Len=0 TSval=2680536967 TSecr=1322111425
6112	64.843767021	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6113	64.843767055	192.168.100.183	192.168.100.200	TCP	60	80 → 59958 [FIN, ACK] Seq=483 Ack=209 Win=31872 Len=0 TSval=2680536967 TSecr=1322111425
6114	64.844215122	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6115	64.844406277	192.168.100.183	192.168.100.200	TCP	60	80 → 59954 [FIN, ACK] Seq=483 Ack=211 Win=31872 Len=0 TSval=2680536969 TSecr=1322111425
6116	64.844932545	192.168.100.200	192.168.100.183	TCP	66	59958 → 80 [ACK] Seq=212 Ack=483 Win=30336 Len=0 TSval=1322123786 TSecr=2680536966
6117	64.844933092	192.168.100.200	192.168.100.183	TCP	66	59958 → 80 [ACK] Seq=209 Ack=483 Win=30336 Len=0 TSval=1322123787 TSecr=2680536967
6118	64.845258302	192.168.100.200	192.168.100.183	TCP	66	59954 → 80 [ACK] Seq=211 Ack=483 Win=30336 Len=0 TSval=1322123787 TSecr=2680536969
6119	64.851259743	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6120	64.851529767	192.168.100.200	192.168.100.183	TCP	66	59962 → 80 [ACK] Seq=210 Ack=483 Win=30336 Len=0 TSval=1322123795 TSecr=2680536977
6121	64.852032206	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6122	64.852343416	192.168.100.200	192.168.100.183	TCP	60	80 → 59960 [FIN, ACK] Seq=483 Ack=210 Win=31872 Len=0 TSval=2680536976 TSecr=1322111426
6123	64.852545106	192.168.100.183	192.168.100.200	TCP	66	59962 → 80 [ACK] Seq=210 Ack=483 Win=30336 Len=0 TSval=1322123795 TSecr=2680536977
6124	64.852929352	192.168.100.200	192.168.100.183	TCP	66	59962 → 80 [ACK] Seq=210 Ack=483 Win=30336 Len=0 TSval=1322123795 TSecr=2680536977
6125	64.861180426	192.168.100.200	192.168.100.183	TCP	66	59946 → 80 [ACK] Seq=212 Ack=484 Win=30336 Len=0 TSval=1322123803 TSecr=2680536948
6126	64.861181008	192.168.100.200	192.168.100.183	TCP	66	59948 → 80 [ACK] Seq=209 Ack=484 Win=30336 Len=0 TSval=1322123803 TSecr=2680536939
6127	64.864893723	192.168.100.200	192.168.100.183	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6128	64.865273437	192.168.100.183	192.168.100.200	TCP	60	80 → 59964 [FIN, ACK] Seq=483 Ack=211 Win=31872 Len=0 TSval=2680536990 TSecr=1322111426
6129	64.865745954	192.168.100.200	192.168.100.183	TCP	66	59950 → 80 [ACK] Seq=209 Ack=484 Win=30336 Len=0 TSval=1322123807 TSecr=2680536945
6130	64.866802190	192.168.100.183	192.168.100.183	TCP	66	59952 → 80 [ACK] Seq=211 Ack=483 Win=30336 Len=0 TSval=1322123809 TSecr=2680536990
6131	64.871576191	192.168.100.200	192.168.100.183	TCP	66	59952 → 80 [ACK] Seq=212 Ack=484 Win=30336 Len=0 TSval=1322123811 TSecr=2680536948
6132	64.873224420	192.168.100.183	192.168.100.200	HTTP	548	HTTP/1.1 408 Request Timeout (text/html)
6133	64.873744512	192.168.100.183	192.168.100.200	TCP	60	80 → 59966 [FIN, ACK] Seq=483 Ack=212 Win=31872 Len=0 TSval=2680536998 TSecr=1322111427
6134	64.874274453	192.168.100.200	192.168.100.183	TCP	66	59966 → 80 [ACK] Seq=211 Ack=483 Win=30336 Len=0 TSval=1322123819 TSecr=2680536998

Frame 6112: 548 bytes on wire (4384 bits). 548 bytes captured (4384 bits) on interface eth0. Id 0

No.	Time	Source	Destination	Protocol	Length	Info
10487	82.624023623	192.168.100.200	192.168.100.183	TCP	77	GET /756 HTTP/1.1 [TCP segment of a reassembled PDU]
10488	82.624023757	192.168.100.200	192.168.100.183	TCP	77	GET /7206 HTTP/1.1 [TCP segment of a reassembled PDU]
10489	82.624023910	192.168.100.200	192.168.100.183	TCP	76	GET /72160 HTTP/1.1 [TCP segment of a reassembled PDU]
10490	82.624040409	192.168.100.183	192.168.100.200	TCP	54	80 -> 42744 [RST] Seq=484 Win=0 Len=0
10491	82.624164216	192.168.100.183	192.168.100.200	TCP	54	80 -> 42746 [RST] Seq=484 Win=0 Len=0
10492	82.624295964	192.168.100.183	192.168.100.200	TCP	54	80 -> 42748 [RST] Seq=484 Win=0 Len=0
10493	82.624419115	192.168.100.183	192.168.100.200	TCP	54	80 -> 42750 [RST] Seq=484 Win=0 Len=0
10494	82.624540268	192.168.100.183	192.168.100.200	TCP	54	80 -> 42752 [RST] Seq=484 Win=0 Len=0
10495	82.624662474	192.168.100.183	192.168.100.200	TCP	54	80 -> 42754 [RST] Seq=484 Win=0 Len=0
10496	82.624786491	192.168.100.183	192.168.100.200	TCP	54	80 -> 42756 [RST] Seq=484 Win=0 Len=0
10497	82.625184516	192.168.100.200	192.168.100.183	TCP	77	GET /7959 HTTP/1.1 [TCP segment of a reassembled PDU]
10498	82.625184912	192.168.100.200	192.168.100.183	TCP	77	GET /71672 HTTP/1.1 [TCP segment of a reassembled PDU]
10499	82.625185043	192.168.100.200	192.168.100.183	TCP	77	GET /71931 HTTP/1.1 [TCP segment of a reassembled PDU]
10500	82.625185229	192.168.100.200	192.168.100.183	TCP	76	GET /71730 HTTP/1.1 [TCP segment of a reassembled PDU]
10501	82.625185315	192.168.100.200	192.168.100.183	TCP	77	GET /72007 HTTP/1.1 [TCP segment of a reassembled PDU]
10502	82.625185489	192.168.100.200	192.168.100.183	TCP	77	GET /7836 HTTP/1.1 [TCP segment of a reassembled PDU]
10503	82.625185641	192.168.100.200	192.168.100.183	TCP	77	GET /71837 HTTP/1.1 [TCP segment of a reassembled PDU]
10504	82.625200585	192.168.100.183	192.168.100.200	TCP	54	80 -> 42760 [RST] Seq=484 Win=0 Len=0
10505	82.625336264	192.168.100.183	192.168.100.200	TCP	54	80 -> 42760 [RST] Seq=484 Win=0 Len=0
10506	82.625459152	192.168.100.183	192.168.100.200	TCP	54	80 -> 42762 [RST] Seq=484 Win=0 Len=0
10507	82.625578196	192.168.100.183	192.168.100.200	TCP	54	80 -> 42764 [RST] Seq=484 Win=0 Len=0
10508	82.625705189	192.168.100.183	192.168.100.200	TCP	54	80 -> 42766 [RST] Seq=484 Win=0 Len=0
10509	82.625826473	192.168.100.183	192.168.100.200	TCP	54	80 -> 42768 [RST] Seq=484 Win=0 Len=0
10510	82.625957941	192.168.100.183	192.168.100.200	TCP	54	80 -> 42770 [RST] Seq=484 Win=0 Len=0
10511	82.626357115	192.168.100.200	192.168.100.183	TCP	77	GET /7626 HTTP/1.1 [TCP segment of a reassembled PDU]
10512	82.626357512	192.168.100.200	192.168.100.183	TCP	77	GET /71416 HTTP/1.1 [TCP segment of a reassembled PDU]
10513	82.626357641	192.168.100.200	192.168.100.183	TCP	76	GET /7127 HTTP/1.1 [TCP segment of a reassembled PDU]
10514	82.626357774	192.168.100.200	192.168.100.183	TCP	77	GET /7990 HTTP/1.1 [TCP segment of a reassembled PDU]
10515	82.626357908	192.168.100.200	192.168.100.183	TCP	77	GET /7790 HTTP/1.1 [TCP segment of a reassembled PDU]
10516	82.626358042	192.168.100.200	192.168.100.183	TCP	76	GET /7923 HTTP/1.1 [TCP segment of a reassembled PDU]
10517	82.626358194	192.168.100.200	192.168.100.183	TCP	77	GET /7295 HTTP/1.1 [TCP segment of a reassembled PDU]
10518	82.626383741	192.168.100.183	192.168.100.200	TCP	54	80 -> 42772 [RST] Seq=484 Win=0 Len=0
10519	82.626511576	192.168.100.183	192.168.100.200	TCP	54	80 -> 42774 [RST] Seq=484 Win=0 Len=0
10520	82.626635397	192.168.100.183	192.168.100.200	TCP	54	80 -> 42776 [RST] Seq=484 Win=0 Len=0
10521	82.626794554	192.168.100.183	192.168.100.200	TCP	54	80 -> 42778 [RST] Seq=484 Win=0 Len=0
10522	82.626860817	192.168.100.183	192.168.100.200	TCP	54	80 -> 42780 [RST] Seq=484 Win=0 Len=0
10523	82.627025699	192.168.100.183	192.168.100.200	TCP	54	80 -> 42782 [RST] Seq=484 Win=0 Len=0
10524	82.627104626	192.168.100.183	192.168.100.200	TCP	54	80 -> 42784 [RST] Seq=484 Win=0 Len=0
10525	82.627320151	192.168.100.200	192.168.100.183	TCP	77	GET /71859 HTTP/1.1 [TCP segment of a reassembled PDU]
10526	82.627528914	192.168.100.200	192.168.100.183	TCP	77	GET /71410 HTTP/1.1 [TCP segment of a reassembled PDU]
10527	82.627529056	192.168.100.200	192.168.100.183	TCP	77	GET /71346 HTTP/1.1 [TCP segment of a reassembled PDU]
10528	82.627529198	192.168.100.200	192.168.100.183	TCP	77	GET /71731 HTTP/1.1 [TCP segment of a reassembled PDU]
10529	82.627529435	192.168.100.200	192.168.100.183	TCP	76	GET /7247 HTTP/1.1 [TCP segment of a reassembled PDU]

Figure 2.2.4: HTTP GET packet detected by Wireshark being sent from the attacker (VM1)

## - Applying defense mechanism

- o Iptables is a firewall-based tool that is pre-installed in many Linux distributions. It allows for determining firewall rules that will control the traffic incoming, outgoing, and forwarding traffic, from and to that machine.
- o Since this required modifying the firewall, the command must be run with the highest privilege (root).
- o The mitigation tactic is limiting the connection establishment packets between each IP address and dropping all packets with invalid TCP Flags, focusing mostly on the TCP establishment stages of the attacks. The default setting, which applies during the first attack, allows all inbound, outbound, and forwarding traffic to be processed. This could potentially be harmful to the machine if a cyber-attack occurs.

```
(root@kali)-[/home/kali]
# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

(root@kali)-[/home/kali]
#
```

Figure 2.2.5: Default Iptables' rules, allowing all INPUT, OUTPUT, and FORWARD traffic to be processed.

- o The mitigation technique is applied through the following command:
  - iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 20 -j DROP
  - iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

- The first command is to limit all incoming connections. The connection will be dropped if an IP address establishes more than 20 connections (20 SYN packets) to port 80 simultaneously.
- The second command is to drop all the TCP packets with no flags (bogus TCP packets).

```
(root@kali)-[/home/kali]
# iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above
20 -j DROP

(root@kali)-[/home/kali]
# iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP

(root@kali)-[/home/kali]
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination            tcp dpt:http fl
DROP      tcp  --  anywhere              anywhere               tcp dpt:http fl
ags:FIN,SYN,RST,ACK/SYN #conn src/32 > 20
DROP      tcp  --  anywhere              anywhere               tcp flags:FIN,S
YN,RST,PSH,ACK,URG/NONE

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

(root@kali)-[/home/kali]
#
```

Figure 2.2.6: Applying Defense Mechanism to Iptables.

### 3. Observer perspective (VM3)

#### - Establishing connection between VM2 and VM3

- Using the Internet explorer browser, an HTTP connection between VM3 and VM2 was established with the address of “http://192.168.100.183”. Before the attack, the connection remained stable, displaying the default page of the Apache2 web-hosting system.

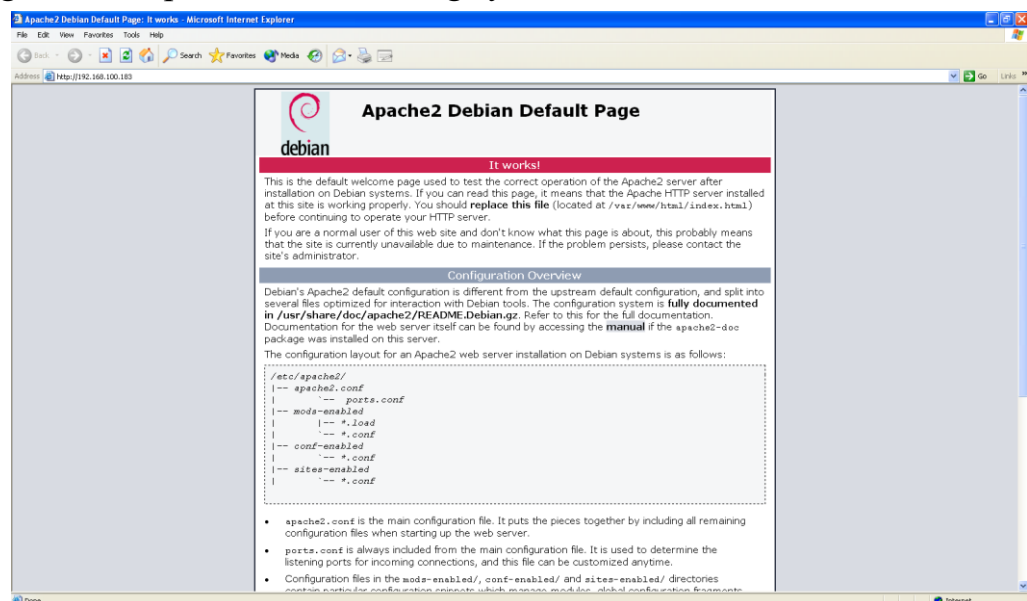


Figure 2.3.1: The default Apache2 webpage.

## - Establishing connection during the attack

- When establishing a connection while the server is under attack, the browser remained loading for an undoubtedly long period of time, displaying the message “Website found, waiting for reply”, proving that the server is under Slowloris attack and the attacker had successfully kept the server “busy” from replying to the legitimate request.

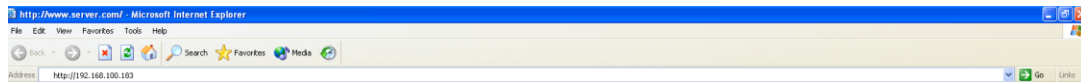


Figure 2.3.2: “Website found, waiting for reply” message at the bottom left.

- On the other hand, if the observer (VM3) tried ping (ICMP) the host (VM2) while the host was under attack and the webpage was waiting for a reply, the host machine still replied to the ping message, proving this type of attack only affect on the application layer (7).

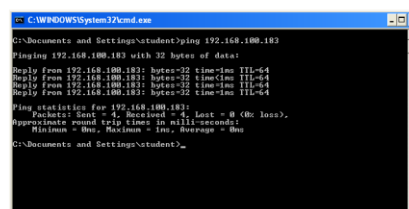
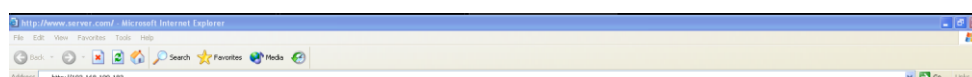


Figure 2.3.3: ICMP ping works while the webpage is “waiting for reply”



- **Establishing connection after VM2 applied defense mechanism**
  - The connection is recovered.

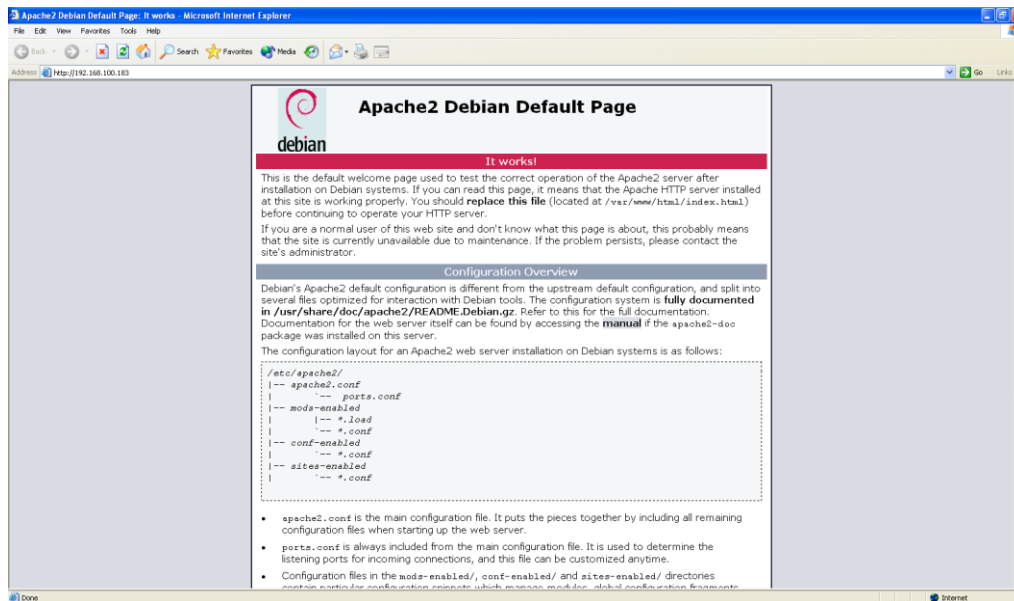


Figure 2.3.4: The website is recovered.

### III. Criteria 3: Analysis.

#### 1. Scenario Analysis

This documentation demonstrates one of many attack techniques that could be used in a cyber intrusion. During the first stage of the attack before applying the defense mechanism, Wireshark (from the defender's perspective) recorded a massive number of TCP-establishing messages, which is the first step of this type of attack described in scenario proposal, Criteria 1. After that Wireshark (from the defender's perspective) also captured some HTTP request time-out packets and HTTP GET methods in the HTTP header which respectively the host's effort to disconnect from the attack and the attack's effort to keep the connection up. Therefore, before the defender applies any defense mechanism, the attacker successfully achieves the aim mentioned in the Scenario proposal. Realistically, these are considered legitimate traffic, compared to SYN flood attacks in which the attacker sends many bogus TCP packets and could be detected through Wireshark, this type of attack cannot be detected by Wireshark observation.

On the other side, from the observer (VM3) viewpoint, the connection has been considerably slowed down, displaying "website found, waiting for a reply", means the observer has successfully connected the host but it had to wait for the host to reply, carrying the website's content. In addition, if the observer tries to ping the host, there are replies, that indicate the attack only affects the application layer (layer 7).

After applying the IP table rules to limit the number of connections to the host, the server starts rejecting massive amounts of simultaneous connections, which is reflected through the indication on the terminal from the running attack script, giving "space" for the host to process legitimate requests. The connection from the observer is also recovered. Therefore, the defender has successfully mitigated the attack, achieving the defender's aim previously mentioned in the scenario proposal.

## IV. Criteria 4: Evaluation

### 1. Attacking Evaluation

In this scenario, The Slowloris has been successfully demonstrated as an effective method against HTTP web servers. This attack works by establishing many complete TCP connections between the attacker and the defender's port 80 (HTTP) of the defender. After that, the attacker will start sending partial HTTP requests to keep the connection up for an extended period, preventing it from responding to legitimate traffic.

The attacking method is highly effective in causing serious delays in response time, is low in bandwidth<sup>[17]</sup>, and is hard to detect (as explained above in Criteria 3). Even though the attacking script used in the demonstration is not the original script written by Robert Hansen (RSnake), the tool is relatively more accessible to many cyber enthusiasts given that it was written Python.

On the other hand, the effectiveness of the attack still depends on several factors including the server's configuration and technical specification. For example, a server that is configured to limit the number of simultaneous connections and connection time is less vulnerable. In addition, the attack is considered less effective against modern, well-configured servers that are designed to handle large amounts of requests.

### 2. MITRE TTPs

According to MITRE ATT&CK Matrix for enterprise, DoS and DDoS attacks are classified as the tactic of Impact (TA0040)<sup>[3]</sup> followed by many types of techniques including Endpoint Denial of Service (T1499)<sup>[3]</sup> and Network Denial of Service(T1498)<sup>[3]</sup>

The Slowloris attack could be classified as **Endpoint Denial of Service: Application Exhaustion Flood (T1499.003)**<sup>[3]</sup>.

#### Tactic

##### Impact (TA0040)

- **Objective:** The primary goal of the attack is to disrupt the availability and deny operation of the service.
- **Context:** By maintaining multiple connections with the target, the attacker can keep the host server from processing legitimate or any other type of request.

#### Technique

##### Endpoint Denial of Service: Application Exhaustion Flood (T1499.003)

- **Description:** Established complete TCP connection. Sending partial legitimate HTTP request to keep the connection up, but the HTTP request is never finished.
- **Context:** Each thread-based web server hosting software such as Apache can only handle a certain amount of requests and use a timeout when they wait for an incomplete HTTP request, but it is set to 300 seconds by default. Therefore, establishing a large amount of connection and keeping the connection up will prevent it from processing other legitimate requests, causing a Denial of Service.

### 3. Defending/Mitigation Evaluation

The defense strategy implemented in this scenario is considered effective as limiting and restricting the number of concurrent access and requests is considered a realistic approach for DoS attack mitigation. In addition, as the mitigation method is applied through iptables into the firewall, the set-out rule is applied nearly immediately, reducing damage for the defender, therefore, even though the defender is overwhelmed before applying the security mechanism, **it is a win for the defender** as the attacker's target has been blocked.

On the other hand, although iptables has successfully mitigated the impact of the slowloris attack, the application may not be optimal for all environments. Realistically, many defense methods could be used against the Slowloris attack including using well-known legitimate cloud cyber security providers like Cloudflare or implementing event-based web server hosting software like nginx which is considered immune to this type of cyber intrusion <sup>[21]</sup>.

## V. Reference

1. "Understanding Denial-of-Service Attacks | CISA," *Cybersecurity and Infrastructure Security Agency CISA*, Feb. 01, 2021. <https://www.cisa.gov/news-events/news/understanding-denial-service-attacks>
2. Sharadin, G. (2023, December 20). DDOS attack types & mitigation Methods | Imperva. Learning Center. <https://www.imperva.com/learn/ddos/ddos-attacks/>
3. MITRE Corporation, "MITRE ATT&CK," <https://attack.mitre.org> (accessed Sep. 5, 2024).
4. "DDoS threat report for 2024 Q2," *The Cloudflare Blog*, Aug. 27, 2024. <https://blog.cloudflare.com/ddos-threat-report-for-2024-q2/>
5. Netscout, "Global DDOS Threat Intelligence Reports | NETSCOUT," *Latest Cyber Threat Intelligence Report*, Apr. 12, 2024. <https://www.netscout.com/threatreport/global-highlights/>
6. C. S. E. Canada, "Protecting your organization against denial of service attacks - ITSAP.80.100 - Canadian Centre for Cyber Security," *Canadian Centre for Cyber Security*, Jul. 29, 2022. <https://www.cyber.gc.ca/en/guidance/protecting-your-organization-against-denial-service-attacks-itsap80100>
7. GeeksforGeeks, "Slowloris DDOS attack tool in Kali Linux," *GeeksforGeeks*, Nov. 25, 2022. [https://www.geeksforgeeks.org/slowloris-ddos-attack-tool-in-kali-linux/?ref=oin\\_asr1](https://www.geeksforgeeks.org/slowloris-ddos-attack-tool-in-kali-linux/?ref=oin_asr1)
8. "t50 | Kali Linux Tools," *Kali Linux*. <https://www.kali.org/tools/t50/>
9. S. Kottler, "February 28th DDOS incident report - the GitHub blog," *The GitHub Blog*, Mar. 01, 2018. <https://github.blog/news-insights/company-news/ddos-incident-report/>
10. G. Yaltirakli [gkbrk], "GitHub - gkbrk/slowloris: Low bandwidth DOS tool. Slowloris rewrite in Python.," *GitHub*, 2015. <https://github.com/gkbrk/slowloris>
11. "The 2016 DYN attack and its lessons for IoT security," *MS&E 238 Blog, Stanford Management Science and Engineering*, Jul. 30, 2017. <https://mse238blog.stanford.edu/2018/07/clairem/the-2016-dyn-attack-and-its-lessons-for-iot-security/> (accessed Aug. 30, 2024).
12. W. Turton, "This is why half the internet shut down today," *Gizmodo*, Oct. 21, 2016. <https://gizmodo.com/this-is-probably-why-half-the-internet-shut-down-today-1788062835>
13. "StrATCOM | NATO Strategic Communications Centre of Excellence Riga, Latvia." <https://stratcomcoe.org/publications/hybrid-threats-2007-cyber-attacks-on-estonia/86>
14. CrowdStrike and Lenaerts-Bergmans Bart, "Snort explained: Understanding Snort rules and use cases," *crowdstrike.com*, Apr. 24, 2023. <https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/snort-rules/> (accessed Aug. 30, 2024).
15. Khess, "Sysadmin tools: How to use iptables," *Enable Sysadmin*, Jan. 12, 2023. <https://www.redhat.com/sysadmin/iptables>
16. R. Sharpe, E. Warnicke, and U. Lamping, "Wireshark User's Guide," *wireshark.org*. [https://www.wireshark.org/docs/wsug\\_html\\_chunked/](https://www.wireshark.org/docs/wsug_html_chunked/)

17. V. Markova, "The Slowloris Attack: How it works and how to protect your website - CloudDNS blog," *CloudDNS Blog*, Jan. 24, 4AD. <https://www.cloudns.net/blog/the-slowloris-attack-how-it-works-and-how-to-protect-your-website/>
18. Bowne, S., 2009. *Hijacking Web 2.0 Sites with SSLstrip and SlowLoris -- Sam Bowne and RSnake (Robert Hansen) at Defcon 17*. [video] Vimeo. Available at: <https://vimeo.com/7618090> [Accessed 4 September 2024].
19. R. Hansen [RSnake], "GitHub - XCHADXFAQ77X/SLOWLORIS: Slowloris HTTP DoS RSnake," *GitHub*. <https://github.com/XCHADXFAQ77X/SLOWLORIS?tab=readme-ov-file>
20. Ivicti. (2023, May 29). *Slowloris attack*. <https://www.invicti.com/learn/slowloris-attack>
21. I. Muscat, "Mitigate slow HTTP GET/POST vulnerabilities in the Apache HTTP server | Acunetix," *Acunetix*, Jun. 06, 2019. <https://www.acunetix.com/blog/articles/slow-http-dos-attacks-mitigate-apache-http-server/>