Detection & Prevention of the Slow Loris attack utilizing Isolation Forest Algorithm & HTTP Port Blocking

A PROJECT REPORT

Submitted by

Shalem Raju Maddirala (20BCY10001)

Chinmay Chougule (20BCY10060)

Rohit Bhat (20BCY10056)

Niture Harshwardhan (20BCY10049)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY AND DIGITAL FORENSICS

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRIKALAN, SEHORE

MADHYA PRADESH - 466114

DECEMBER 2021

BONAFIDE CERTIFICATE

Certified that this project report titled **"DETECTION AND PREVENTION OF SLOW LORIS ATTACK UTILIZING ISOLATION FOREST ALGORITHM AND HTTP PORT BLOCKING"** is the Bonafide work of "**(20BCY10001) SHALEM RAJU MADDIRALA (20BCY10155) CHINMAY CHOUGULE (20BCY10060) ROHIT BHAT (20BCY10056) AND NITURE HARSHWARDHAN (20BCY10049)**" who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR

Dr. R. Rakesh, Assistant Professor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE

Dr. Ganeshan R, Assistant professor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on 17/12/2021

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. Shishir Kumar Shandilya, Head of the Department, School of Computer Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr. Ganeshan R for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF ABBREVIATIONS

ABBREVIATIONS	FULL FORM
IP	Internet Protocol
DDOS	Distributed Denial of Service
DOS	Denial of Service
HTTP	Hypertext Transfer Protocol
UFW	Uncomplicated Firewall

LIST OF FIGURES AND GRAPHS

Abstract

One of the difficulties we encountered in our research was the scarcity of trustworthy training datasets. In fact, each researcher in the subject encounters this issue. Despite the fact that there are several publications regarding DDoS or DoS attacks, there is little factual information accessible about how a Slowloris attack works and how to prevent or mitigate it. The Slowloris attack delivers HTTP legitimate requests in a leisurely manner, making it difficult for a firewall to recognize or detect.

In this report, the essential elements that have shown to be sound and successful in detecting and preventing the Slowloris attack are highlighted. We also suggested some new features, experimented with assigning new rules to certain well-known features, and updated some existing features using research articles as a reference.

TABLE OF CONTENTS

CHAPTER-1

PROJECT DESCRIPTION AND OUTLINE

1.1 Introduction:

Slow Loris is difficult to identify or detect by a firewall because it sends HTTP valid requests but in a slow way. We tried to detect Slow Loris attack on Apache web server which is mainly affected by slow Loris by python code to list out IPs. As of title we tried to utilize the Isolation Forest in our own way by making a code that drags out the odd IPs. In the mitigation process against Slow Loris, we are blocking IP extracted in the Detection phase.

1.2 Motivation for the work:

Comparatively, DoS attack is easy to implement but difficult to prevent which makes it more in quota and hence creates a problem for many web networks. Choosing this as a project is our first move to briefly understand it and find possibilities in the security domain, to create more advanced techniques to tackle this critical problem.

1.3 Introduction to the project including techniques:

Phase I: Performing the Slow Loris Attack

The procedure of implementing the things:

- Run the msfconsole in Kali Linux
- Search the Slow Loris Module
- Set the address of the website that to be attacked
- Set the socket counts
- Run the attack

Phase II: Detection of Slow Loris attack

The procedure of implementing the things:

- Extract network file from Wireshark
- Write a code for detection of Slow Loris attack
- Run the detection code over the PCAP file
- Store the output results in the result.csv file

Phase III: Prevention from Slow Loris attack

The procedure of implementing the things:

- Check if root privileges are permitted. If permitted proceed.
- Using subprocess start ufw. If not installed, install it.
- Check and see how many, and which, IP addresses are connecting to your server at once.
- If total connections are larger than defined. Deny the IP to connect.
- Add the blocked IP in the 'blockedIPs.txt' file.
- Now using the iptables add the bad IP in the blacklist.

1.4 Problem Statement:

- Fewer security techniques present.
- Complex methodologies available.
- Less information available about testing and prevention of Slow Loris DDoS attack.

1.5 Objective of the work:

Our goal is to create a simple methodology amongst very few existing techniques for prevention and detection of Slow Loris attack.

1.6 Organization of the project:

Our Project is organized by our team according to some split-ups of information and data. We gathered information about Slow Loris attack and some ways to identify and prevent the attack through some research papers. By using these methods, we created a Python code.

1.7 Summary:

A DoS attack is a malicious attempt to disrupt the normal traffic of a targeted server, service, or network. Detection and Prevention of DoS is really important. Slowloris is an application layer DoS attack which continuously sends alive HTTP requests to open connections between a single computer and a targeted Web server.

CHAPTER-2:
RELATED WORK INVESTIGATION

2.1 Introduction

In the following chapter, we will see about the work-related investigation we carried out to develop our project. References were taken from various research papers and other platforms to successfully complete the project work. This chapter will clearly explain the core area, methodology used and the pros and cons of our project.

2.2 Core area of the project

The core area of our project work is to detect and prevent the slowloris attack which is a type of denial-of-service attack tool which uses partial HTTP requests to open connections between a single computer and a targeted Web server, then keeping those connections open for as long as possible, thus overwhelming and slowing down the target. This type of DDoS attack requires minimal bandwidth to launch and only impacts the target web server, leaving other services and ports unaffected. Slowloris attacks can go on for an extended period of time if they remain undetected. It is very dangerous as a single computer can take down an entire web server and can also prevent log file creation, which would prevent red flags from appearing in log file entries, rendering the attack invisible. Due to its widespread threat, prevention of slowloris attack becomes very important. In this project, we have worked towards detection and prevention of Slowloris attack.

2.3 Existing Approaches/Methods

At first, we researched the existing methodologies to detect and prevent the slowloris attack. There are very few methods present which concentrate on detection and mitigation of the slowloris attack and all these are very complex. Tuning the Web server configuration is effective to an extent, although there is always a tradeoff between limiting slow HTTP attacks and dropping legitimately slow requests. Beyond configuring the web server, it's possible to implement other layers of protection like event-driven software load balancers, hardware load balancers to perform delayed binding, and intrusion detection/prevention systems to drop connections with suspicious patterns. Additional approaches for Slowloris protection include instituting reverse proxies, firewalls, load balancers or content switches.

2.4 Pros and cons of the stated Approaches/Methods

Pros: Our project is one of the simplest methodologies for detection and prevention among the few existing techniques over the internet. All the previously stated methodologies work partially

towards preventing the slowloris attack and are not fully efficient. It works fine for a minor level of attack.

Cons: This project is carried out using the local web server on Kali Linux and prevention could not be used until one has the access to web server. Because Slowloris sends partial packets, instead of corrupted ones, traditional intrusion detection systems are not particularly effective at detecting this type of attack.

2.5 Issues/observations from investigation

We have observed that all the existing detection and prevention methodologies against slowloris are either partially working or are complex. Also, there is very less information available about slowloris and not many people know about it in depth. Therefore, we decided to create an efficient and simple methodology which serves this purpose. In this project, our vision would be to create a detection code by analyzing the captured packets through Wireshark and then blocking the attacker's IP using HTTP port blocking algorithm.

2.6 Summary

In this chapter, we have mainly seen about the references and sources that we had to take before implementing our project. After going through various research papers and articles, we tried to find out the defects or flaws in them and tried to improve them so that the approach could be improvised and provide better efficiency. The existing methods and approaches were analyzed by their pros and cons in detail. Finally, useful observations were made which proved to be the starting step to our project.

CHAPTER-3:

REQUIREMENT ARTIFACTS

3.1 Introduction

We have previously discussed the main idea of the project. Here we can discuss how features need to be implemented in Python for creating the detection and prevention code.

3.2 Hardware and Software requirements

Hardware:

- Any modern CPU
- Windows OS - 64 bit
- Linux system (Kali Linux)
- Minimum 4 GB of free RAM
- Disk space - 2.5 GB and another 1 GB for storing cache

Software:

- Anaconda/Google Collab
- Python3
- Wireshark
- PyCharm/VS Code

Other essential requirements:

- PCAP files for the attack detection

3.3 Specific Project requirements

Some specific requirements needed for this project are good understanding of **Python Language**, basic **Linux Commands**, **Kali Linux Operating System**, and Project Algorithm Developing Skills, and, Google collab or terminal to run the codes.

3.4 Data requirement.

One should have the knowledge of how slow loris actually takes place to understand and create the possible methodologies for prevention. In our project, for understanding, one should learn how to use kali operating system to carry out the attack and should know how to extract data from wireshark. The data extracted from the Wireshark is network transmission and is mainly required for detection of the IP.

3.5 Summary

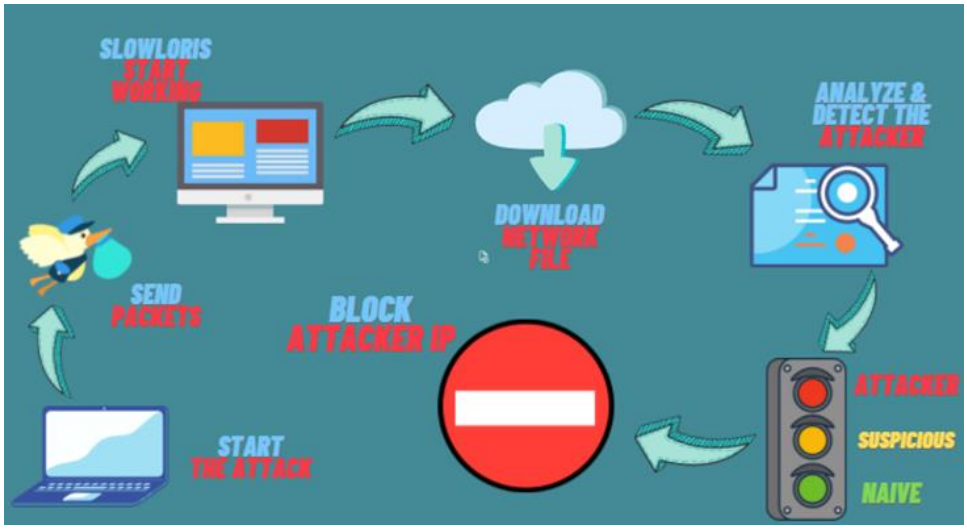
We can summarize as the requirement for the implementations of the code are **Python3** and **Kali Linux Operating System**. The code will analyze the PCAP file extracted from Wireshark and block the attacking IP.

CHAPTER-4

METHODOLOGY AND ITS NOVELTY

4.1 Methodology and goal:

Methodology:



Goal: Our goal is to mitigate the slow loris by the simple method possible to provide smooth service to users without any traffic, data loss or even monetary loss.

4.2 Tools and Modules used and their explanation

In this session we have mentioned all the modules that we have used to perform the whole thing. Including the attack, detection, and the prevention of the Slowloris Attack.

Tools and modules used for attack:

msfconsole

The msfconsole is a popular interface to the Metasploit Framework (MSF). It provides an "all-in-one" centralized console and allows you efficient access to virtually all of the options available in the MSF.

apache2

The Apache HTTP Server Project is an open-source HTTP server to provide a secure, efficient, and extensible server that provides HTTP services coordinated with the current HTTP standards.

slowloris

It is the module of msfconsole that is used to run the slowloris attack over the targeted website by sending over packets continuously.

Module: auxiliary/dos/http/slowloris Source code: modules/auxiliary/dos/http/slowloris.py

Wireshark

Wireshark is an open-source packet analyzer, which is used for education, analysis, software development, and network troubleshooting.

Tools and modules used for detection:

Pandas

Pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool

io

This module of python allows managing the file-related input and output operations.

files

Google Collab has its inbuilt files module, with which one can upload or download files effortlessly.

Tools and modules used for Prevention:

os

Python's OS Module works as an interface between the Python Programming Language and the Host Operating System.

time

Time module in Python provides various time-related functions. For example, getting the current time, pausing the Program from executing.

subprocess

Subprocess is the task of executing or running other programs in Python by creating a new process.

4.3 Subsystem services:

PyCharm/VS code and Google Collab is the subsystem service used in our project. It helps us to code in uncomplicated way effortlessly.

4.4 Summary:

In this chapter, we have very elaborately discussed about the methodology and goal of our project. Also, we described the functional modules and subsystem services using Python coding.

CHAPTER-5

TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 Outline:

In this chapter will look into the idea of writing the code, its success rate and the commands we used to attack, detect and prevent the slowloris attack.

5.2 Technical coding and code solutions:

We used python language to complete our project.

5.2.1 Slowloris Attack commands.

To run the Slowloris open terminal of Kali Linux and run the following commands:

$ sudo su.

sudo su will give you the root privileges to work with all the commands allowing you to perform all the operations by authenticating the user just once.

$ msfconsole.

This command will take you to the instance of Metasploit Framework. In which our Slowloris module is located that we want to use.

msf6 > search slowloris.

The command above will help you locate the Slowloris module in the msfconsole.

msf6 > use 0.

Now, the ‘use 0’ command will give you the access to the module and after that you can use the Slowloris module to initialize the attack. The module is located at 0‘th location hence we are using the ‘use 0’ command.

msf6 auxiliary(dos/http/slowloris) > set rhost <IP of target website>

Here, with this command we will set the IP address of the target website that we want to attack.

msf6 auxiliary(dos/http/slowloris) > set sockets <socket count>

By default, the script runs with 150 sockets unless you specify it so, instead of 150 we will try increasing the sockets counts to pull down the website. You can choose any number you wish to and which works perfectly for you. For example: ‘set sockets 1000’

msf6 auxiliary(dos/http/slowloris) > run.

Now the last step is to run the program. For running the program, you just need to type run in the shell and hit enter. This will initialize the Slowloris attack over the target website.

5.2.2 Detection Code :

Before using the detection code, one need to download csv file for the targeted web server for the analysis using Wireshark.

```
# -*- coding: utf-8 -*-
"""Detection of Slowloris
Automatically generated by Colaboratory.
Original file is located at
    https://colab.research.google.com/github/21goldy/DOS_Detection---Project-
Exhibition-1/blob/main/Detection_of_Slowloris.ipynb
"""

#@title Detection & Prevention of the Slow loris attack utilizing the
IsolationForest algorithm and HTTP port blocking - Group 16 { display-mode: "form"
}
# Detection & Prevention of the Slow loris attack utilizing the IsolationForest
algorithm and HTTP port blocking - Group 16

# importing the required modules of python

import io
```

```
import pandas as pd
from google.colab import files

upload_file = files.upload() # uploading the collected CSV from wireshark for DOS
analysis and attack detection

for fn in upload_file.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(upload_file[fn])))

data = pd.read_csv(io.StringIO(upload_file['detect_test3.csv'].decode('utf-8'))) #
reading the CSV file and storing it under the name "data"

data.shape # defining the total rows and columns present (Here total rows can also
be understood as the number of transmissions captured)

transmission_count = data.shape[0] # storing the rows/transmissions captured as
"transmission_count"

data.head() # representing the starting 5 data sets in "data"

data.tail() # representing the ending 5 data sets in "data"

data.describe() # description of the data present in the dataset "data"

data.Source # listing the Source IP's present in the captured file

source = data.Source
source.nunique() # finding the unique IP's present in the Source column

destination = data.Destination
destination.nunique() # finding the unique IP's present in the Destination column

new_source = source.drop_duplicates() # removing the all the duplicates from the
Source column
source_list = list(new_source) # creating list of all the unique IP's
source_list
print(len(source_list))

major_ips = [] # empty list for keeping the majorly participating IP
minor_ips = [] # empty list for keeping the less participating IP
for i in source_list:
  analyse_ip = data[data.Source == i]
```

```
  transmission_of_ip = analyse_ip.shape[0]
  set_criteria = (transmission_count * 35)/100
  # print(set_criteria)
  if transmission_of_ip > set_criteria:
    major_ips.append(i)
  else:
    minor_ips.append(i)

major_ips # list of actively participating ip

minor_ips # list of less active ip

#@title Analysis of the Major IPs { display-mode: "form" }

for i in major_ips:
  dest_list_of_major_ip = list(data[data.Source ==
i].Destination.drop_duplicates()) # transmitted destination list of major IP
  major_df = pd.DataFrame(
      [
       (
           i,
           dest_list_of_major_ip,
           f'{data[data.Source == i].Protocol.max()}',
           f'{data[data.Source == i].Length.sum()}',
           'Attack'
       )
      ],
      columns=(
          'Source', 'Destination', 'Max_Protocol', 'Total_Length', 'Result'
          )
                )
  major_df.to_csv (f'result_of_{i}.csv', index = False, header=True) # writing data
in the csv file "result_of_{ip}"
  print(major_df.head())
  print(major_df.Destination)

a = data[data.Source == '10.0.2.15']
b = a[a.Destination == '192.185.184.157']
b # transmission between the major IP "a" = '10.0.2.15' to "b" = '192.185.184.157'

b.to_csv (r'transmission_1.csv', index = False, header=True) # writing data in the
csv file "transmission_1"
```

```
c = data[data.Source == '192.185.184.157']
d = c[c.Destination == '10.0.2.15']
d # transmission between the major IP "c" = '192.185.184.157' to "d" = '10.0.2.15'

d.to_csv (r'transmission_2.csv', index = False, header=True) # writing data in the
csv file "transmission_2"

#@title Analysis of the Minor IPs { display-mode: "form" }

for j in minor_ips:
  dest_list_of_minor_ip = list(data[data.Source ==
j].Destination.drop_duplicates()) # transmitted destination list of minor IP
  if data[data.Source == j].Length.sum() <= 13000:
    naive_df = pd.DataFrame(
    [
     (j,
      dest_list_of_minor_ip,
      f'{data[data.Source == j].Protocol.max()}',
      f'{data[data.Source == j].Length.sum()}',
      'Naive'),
    ],
    columns=(
        'Source', 'Destination', 'Max_Protocol', 'Total_Length', 'Result'
        )
                )
    print(naive_df)
    # print('============================================================')

  else:
    suspicious_df = pd.DataFrame(
    [
     (j,
      dest_list_of_minor_ip,
      f'{data[data.Source == j].Protocol.max()}',
      f'{data[data.Source == j].Length.sum()}',
      'Suspicious'),
    ],
    columns=(
        'Source', 'Destination', 'Max_Protocol', 'Total_Length', 'Result'
        )
                )
    print(suspicious_df)
    # print('============================================================')
```

Detection code: [21goldy/DOS_Detection---Project-Exhibition-1 (github.com)]

5.2.3 Prevention Code :

```
Import os
import subprocess
import time

allowedConnections = 15
refreshRate = 3

blocked_ips = []
blank_list = []
ips = []

while True:
    if os.geteuid() != 0:
        print("Authentication Required!\n No root privileges.")
        break
    if os.path.isdir('/etc/ufw/'):
        Uncomplicated_firewall = subprocess.Popen(["ufw", "status"],
stdout=subprocess.PIPE)
        response = str(Uncomplicated_firewall.communicate())
        if 'inactive' in response:
            print('UFW Disabled. To enable, enter `sudo ufw enable` into your
terminal.')
            break
    else:
        print('UFW not installed. To install, enter `sudo apt-get install ufw` into
your terminal.')
        break

    file = open('blockedIPs.txt', 'a')

    # list of connected IPs to the server and their details
    connectionDetails = os.popen("netstat -ntu|awk '{print $5}'|cut -d: -f1 -
s|sort|uniq -c|sort -nk1 -r")
    readDetails = connectionDetails.read()
```

```
    print(readDetails)

    scrapedIPs = list(readDetails.split())
    for x in range(len(scrapedIPs)):
        if x % 2 == 0:
            blank_list.append(scrapedIPs[x])
        else:
            ips.append(scrapedIPs[x])
    for x, y in enumerate(blank_list):
        if int(y) > allowedConnections:
            if ips[x] != '127.0.0.1' and ips[x] not in blocked_ips:
                print('Blocking %s with %d connections' % (ips[x], int(y)))
                # os.system(str('ufw insert 2 deny from %s' % ips[x]))
                os.system(str('ufw reload'))
                blocked_ips.append(ips[x])
                file.write(ips[x] + '\n')

    file.close()
    time.sleep(refreshRate)

    # using iptables to block the ip from the system

    IPTABLES_LIST = os.popen("iptables -L")
    read_iptables_list = IPTABLES_LIST.read()
    # print(read_iptables_list)

    os.popen('iptables -I INPUT -s 10.0.2.15 -j DROP')

    # creating a blacklist of bad IPs [Run only once and comment the below line]
    # os.popen("ipset create blacklist hash:ip hashsize 4096")

    # Set up iptables rules. Match with blacklist and drop traffic [Run only once
and comment the below 2 lines]
    # os.popen("iptables -I INPUT -m set --match-set blacklist src -j DROP")
    # os.popen("iptables -I FORWARD -m set --match-set blacklist src -j DROP")

    # Add a specific IP address to your newly created blacklist
    os.popen("ipset add blacklist ")

    # show details of the blocked ip
    blocking_IPTABLES = os.popen("ipset list")
    ip_IPTABLES = blocking_IPTABLES.read()
    print(ip_IPTABLES)
```

```
            break
```

Prevention code: [21goldy/DOS_Prevention---Project-Exhibition-1 (github.com)]

5.3 Working Layout of Forms:

Slowloris tries to keep many connections to the target web server open and hold them open as long as possible. It accomplishes this by opening connections to the target web server and sending a partial request. Periodically, it will send subsequent HTTP headers, adding to-but never completing-the request. Affected servers will keep these connections open, filling their maximum concurrent connection pool, eventually denying additional connection attempts from clients.

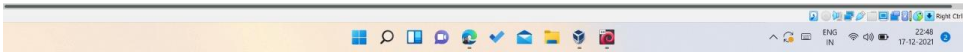
What we will be doing is, we will try to analyze all the IPs that are transmitting to the server. And look up something that is not normal. So, the working sequence will be like, first we will initialize a Slowloris over our local server only and will try to point out the odds from it and pick up the IP which is responsible for the major transmission.

We have first found several ways of detecting and preventing Slow Loris attack and then implemented it in our own way. Our working layout would look like:

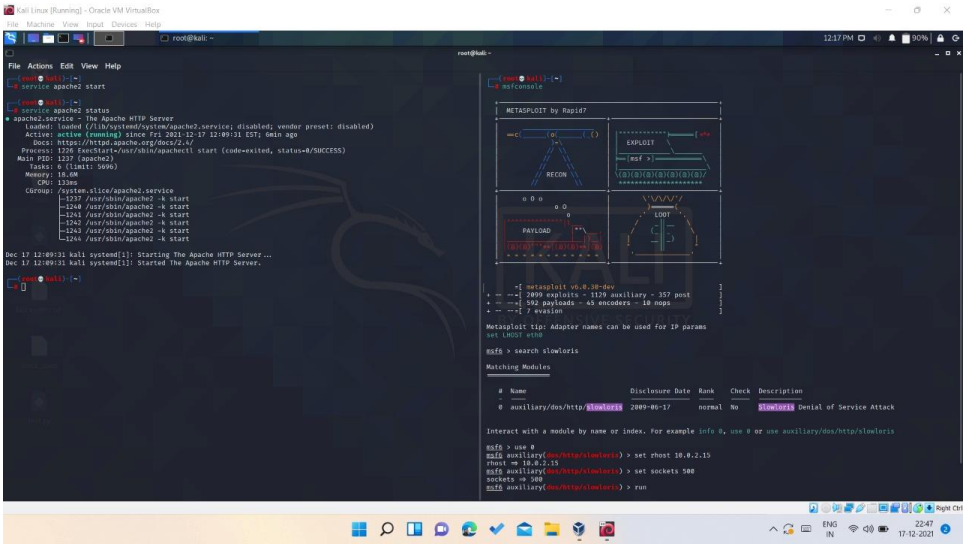
Attack the site using Kali Linux >> Detect the major IP using Wireshark and the code we developed >> Prevention by blocking the attacker IP

5.4 Test and validation:

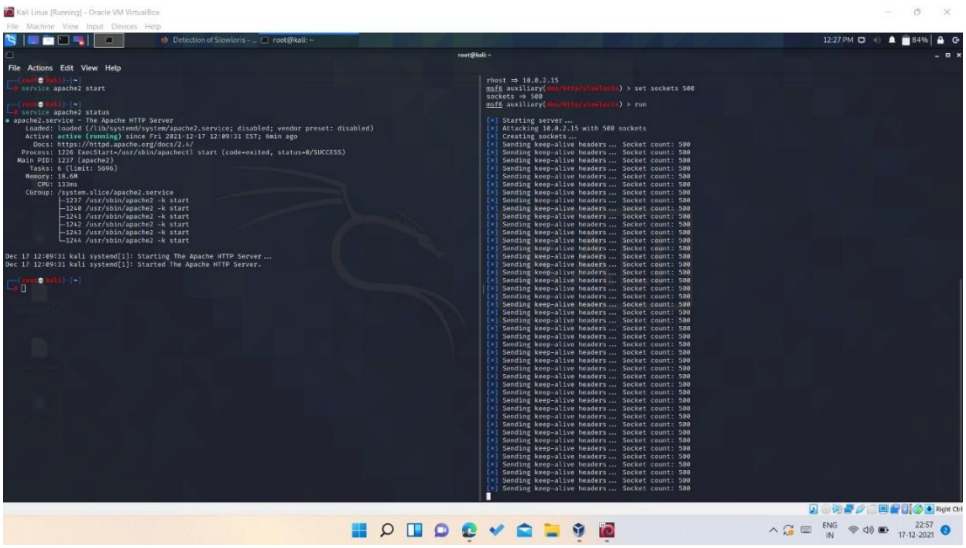
Our Apache web server:



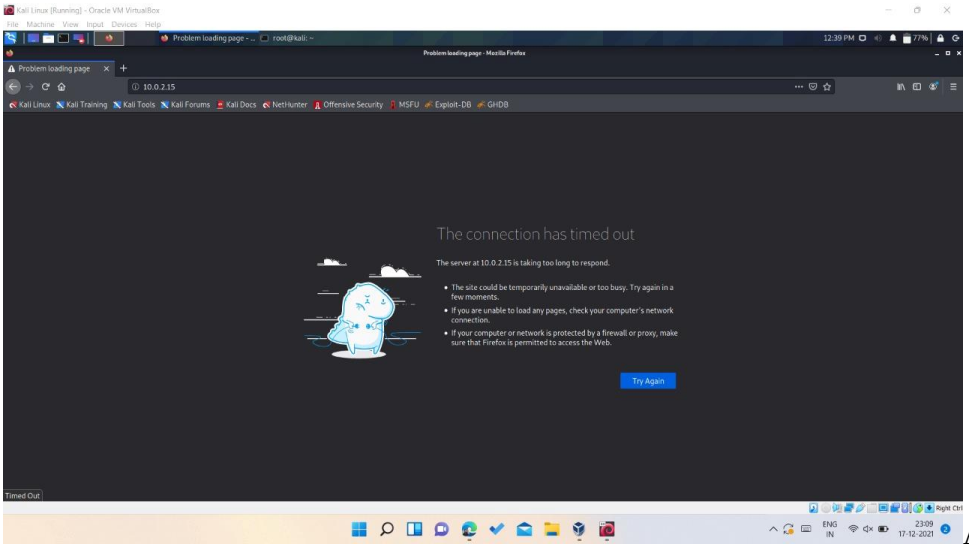
Attacking the Apache Web server:



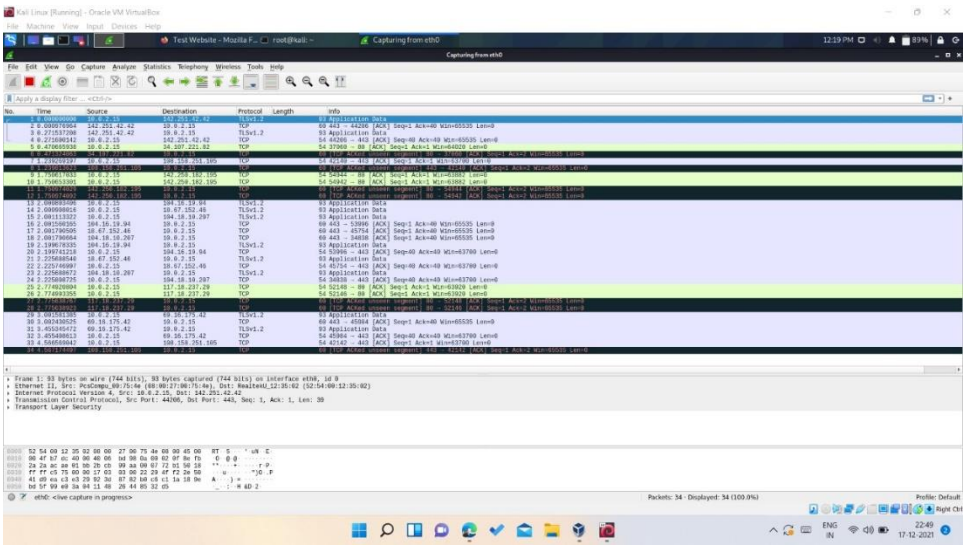
Sending the keep-alive headers with socket count: 500



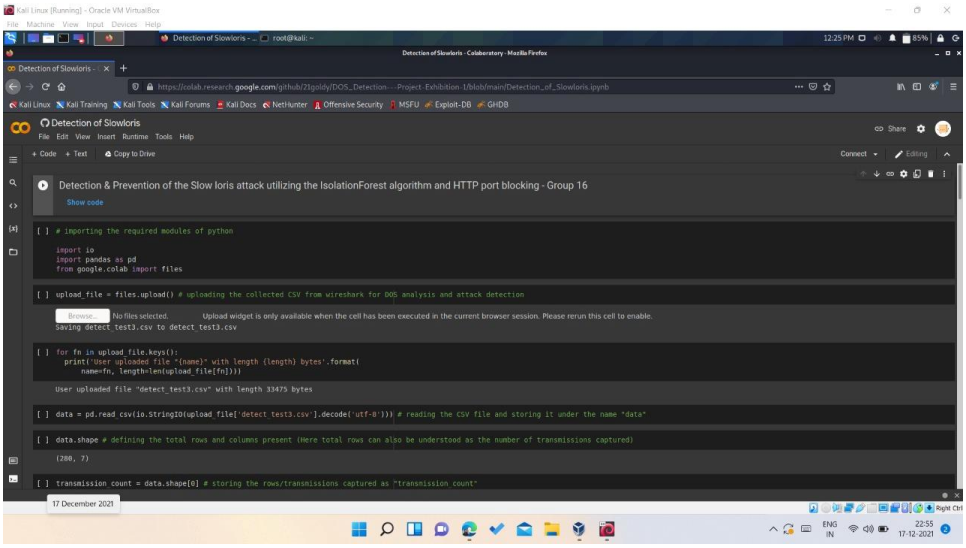
Site taken down after the attack:



Analyzing the packets in Wireshark:

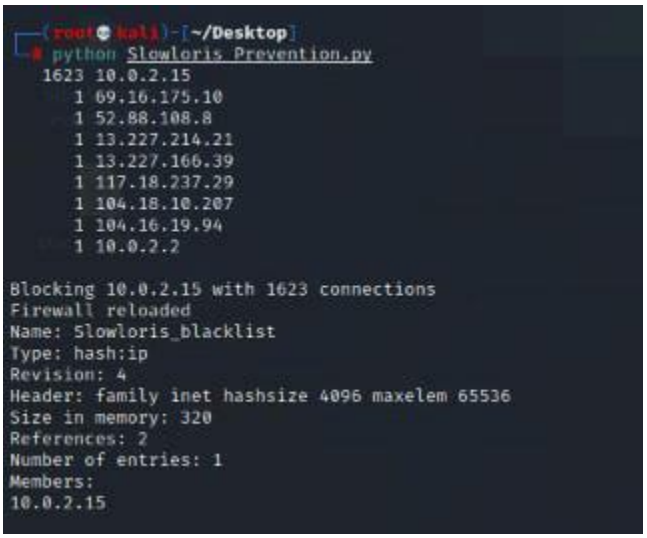


The Detection code using the analysis done through Wireshark for finding odd IP:



The Prevention code for blocking IP:

Slowloris Prevention:



5.5 Summary:

In this chapter we described technical coding, working layout of our project, then we have shown the test analysis and performance analysis of our project.

CHAPTER-6

PROJECT OUTCOME AND APPLICABILITY

6.1 Outline:

The project was successfully implemented by our team. It operates smoothly and finely. The result was exactly what we expected it correctly detects the IPs and bans them as a preventative measure.

6.2 Key implementations outline of the System:

The implementation of the code is the most important aspect of the project. We initially used Kali Linux and its commands to attack our own local web server, then used Python to discover the Odd IP using the Wireshark Detection technique. We banned IP addresses detected throughout the detection procedure using Python as a preventative measure.

6.3 Significant project outcomes:

The project was successfully implemented after the team's dedicated and challenging work, and the detection code we created detects IPs as attacking IP, suspicious IP, and naive IP based on data extracted from Wireshark, and thus our IP blocking code based on the input mitigates the attack by blocking the most communicating HTTP requester's IP, allowing the webserver to continue providing service.

6.4 Inference:

In this chapter, we explained how we developed and implemented project and tried to imagine how it may be utilized in the real world. It may function effectively as a stand-alone program.

CHAPTER-7

CONCLUSIONS AND RECOMMENDATION

7.1 Outline:

We can conclude that the Slow Loris is not an active attack and hence need to be taken into light and hence we created one of the simple methodologies to detect and mitigate the slow loris attack. Also using this report one can gain the knowledge about slow loris attack and hence create more advance method.

7.2 Limitation/Constraints of the System:

The limitation of this project is for implementation of the prevention codes one should have access of target web server to mitigate the attack.

7.3 Future Enhancements:

This project is carried out using the Kali Linux operating system's local web server. Some aspects of this project are done manually, such as obtaining the file from Wireshark and uploading it again for analysis, although this might be automated. This project could also be adopted and analyzed on massive web servers in order to advance the codes.

7.4 Inference:

As discussed, Slow Loris can be dangerous, so we must have prevention methodology against it. Most of times Slow Loris attack is used to take down a particular web server. So, we can infer from this that slowloris can be used for demanding ransom and it can cause damage to web server of the organization. So, any sort of prevention methodology should be used for nullifying the risk of the attack.

RELATED WORK INVESTIGATION

Thomas Lukaseder, Shreya Ghosh Et. Al. published a research paper titled "Mitigation of Flooding and Slow DDoS Attacks in a Software-Defined Network" which was published in the year 2018. According to them, Distributed denial of service (DDoS) attacks are a constant threat for services on the Internet. In the year 2018, i.e., when they published the following paper, the record for the largest DDoS attack ever observed was set at 1.7 Tbps. They found that slow attacks work differently. Instead of sending requests as fast as possible, they send as slow as possible. The slowloris or slow header attack sends HTTP get requests in as many packets as possible. They mentioned a defense strategy against DDoS attacks which requires many methods to be undertaken. For one, the attack has to be observed. Then, the attack has to be classified to find the matching defense mechanism. In a third step, the individual attackers need to be identified and in a last step, the attackers need to be excluded or blocked from the network to mitigate the attack.

REFERENCES

- Classifying DDoS attacks with Artificial Intelligence DDos Attack Classification | Classifying DDoS Attacks with AI (analyticsvidhya.com)
- How to Perform a Slowloris Attack on Metasploitable2 using Msfconsole & Prevention Techniques https://youtu.be/_m0sQFx5-2I
- How to Configure Apache Server in Kali Linux complete Tutorial for beginner How to Configure Apache Server in Kali Linux complete Tutorial for beginner (cyberpratibha.com)
- How to do a DoS attack https://github.com/moulik-source/ddos
- The Slowloris attack https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/
- Slowloris pentesting script in Python https://github.com/LukeBob-zz/Bob-loris
- Slowloris DoS attack and Mitigation on NGINX web server https://hexadix.com/slowloris-dos-attack-mitigation-nginx-web-server/
- How to Block an IP Address From Your Website https://youtu.be/DUMeRcfn9lY
- Block IP addresses using Django https://djangosnippets.org/snippets/744/
- Slowloris Attack https://www.netscout.com/what-is-ddos/slowloris-attacks