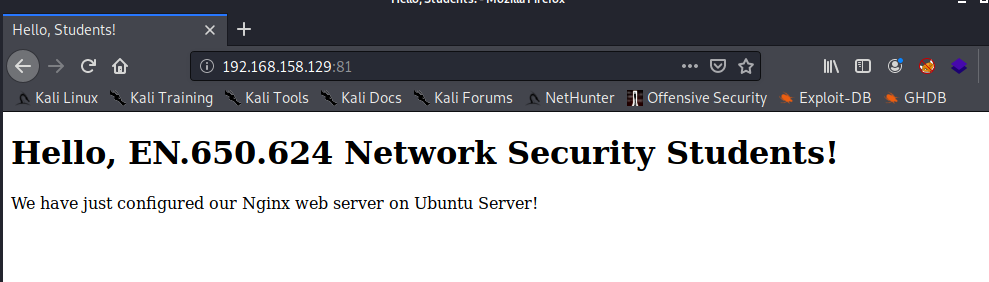
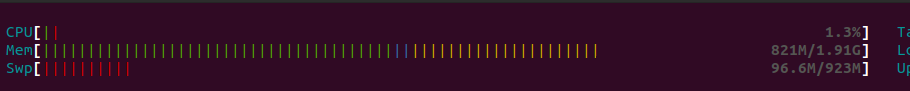
**Part 1**

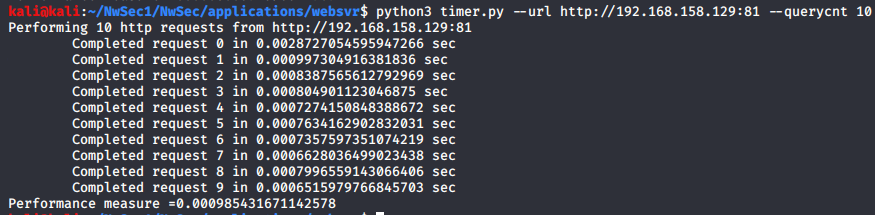
1. **Describe the sequence used.**

In this assignment, we are launching a DOS attack against a Nginx web server. The goal here is to make sure the normal/legitimate users of the website are unable to view/load the website.



Under normal scenario when the system is not under DOS attack, we can see that the system resources on the victim machine are not being depleted and the average time take to complete 10 HTTP request queries is 0.0009 seconds.





Now in this assignment, we will be using hping3 tool to launch a Denial-of-Service attack against the webserver. The command used in this scenario is: -

sudo hping3 --rand-source -i u1 -S -p 81 192.168.158.129

Please find the breakdown of the command used below: -

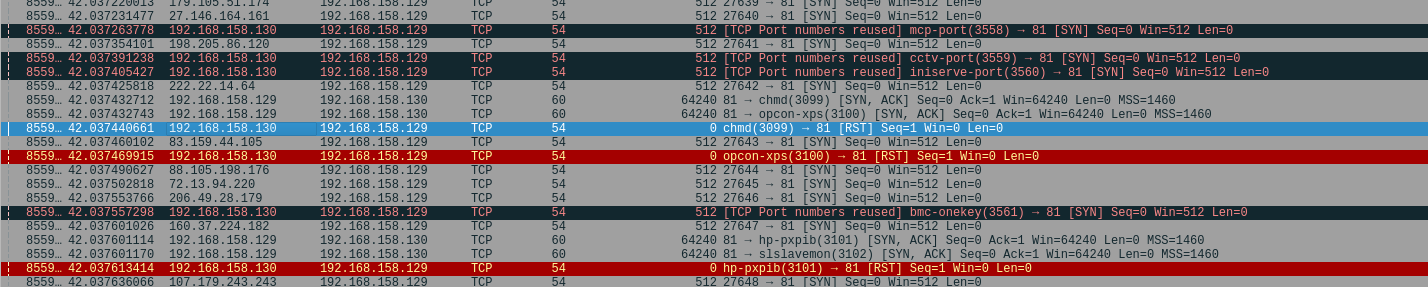
**--rand-source** -> activates the random source mode (hping3 will be crafting packets with random source addresses)

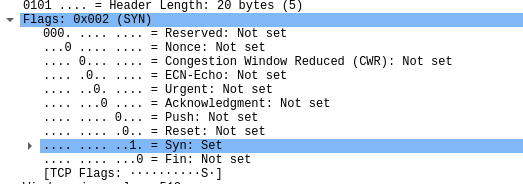
**S** –> indicates SYN flag

**p 81** -> Target port 81

**I u1** -> Interval duration between each packet is 1 microsecond

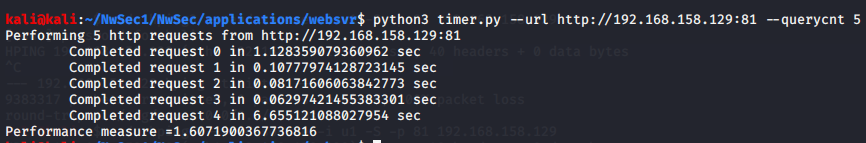
During the attack we can see that the victim machine is flooded with TCP packets with SYN packets set.





And the victim machine is unable to respond to these packets and as a consequence the website goes down after sometime.

Also, during the attack, the average time taken to complete 5 HTTP Request queries was 1.607~ seconds.



As we can clearly see, the Denial-of-Service attacks depletes and degrades the system performance. Consequently, the webserver goes down and site goes offline. Thereby, depriving legitimate users from accessing the website.

1. **Provide a python script that executes your attack.**

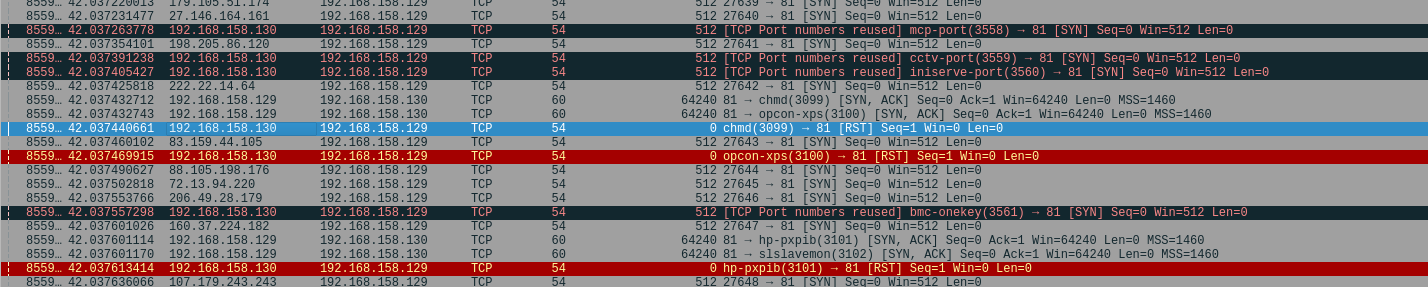
Provided in the zip folder.

1. **Provide \*.pcap network capture file from traffic between attack host and target running the application server.**

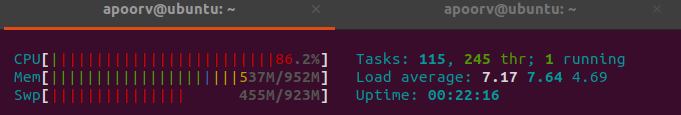
Provided in the zip folder.

1. **Describe the results of the attack.**

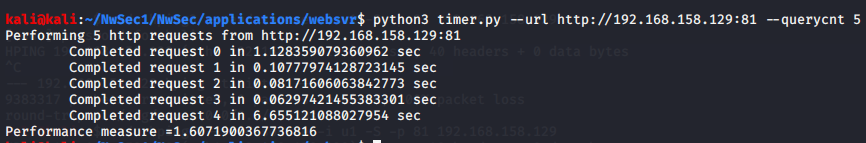
After launching the attack, we can see on the wireshark that the target is flooded with the packets.



And on the victim machine, we can clearly see the system resources being depleted. And after sometime the web server crashes.



Also, during the initial phase of the attack, we see that the average time taken to complete 5 HTTP request queries is 1.607 seconds (significantly higher than the time taken to complete same number of queries under normal scenario).



**Part 2**

1. **Provide the details for updated Nginx settings.**

Note:- a) Nginx.conf file is in the “part2” of this folder.

b) I updated/configured Nginx.conf and Iptable rules to improve defenses against DOS attack in this scenario.

Highlighted in **bold-black** are the updated Nginx settings

user www-data;

worker\_processes auto;

pid /run/nginx.pid;

include /etc/nginx/modules-enabled/\*.conf;

events {

worker\_connections **50**;

# multi\_accept on;

}

http {

##

# Basic Settings

##

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

**client\_body\_buffer\_size 200K;**

**client\_header\_buffer\_size 2k;**

**client\_max\_body\_size 200k;**

**large\_client\_header\_buffers 3 1k;**

# server\_tokens off;

# server\_names\_hash\_bucket\_size 64;

# server\_name\_in\_redirect off;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

##

# SSL Settings

##

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE

ssl\_prefer\_server\_ciphers on;

##

# Logging Settings

##

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

##

# Gzip Settings

##

gzip on;

# gzip\_vary on;

# gzip\_proxied any;

# gzip\_comp\_level 6;

# gzip\_buffers 16 8k;

# gzip\_http\_version 1.1;

# gzip\_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

##

# Virtual Host Configs

##

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

**# 3 requesters per minute**

**limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=3r/m;**

**server {**

**listen localhost:81;**

**location / {**

**limit\_req zone=one;**

**}**

**}**

**limit\_conn\_zone $binary\_remote\_addr zone=addr:5m;**

**server {**

**listen localhost:81;**

**location / {**

**limit\_conn addr 5;**

**}**

**}**

**server {**

**listen localhost:81;**

**client\_body\_timeout 5s;**

**client\_header\_timeout** 5s;

**}**

}

Additionally, below are the IP Table Rules which I configured :-

For blocking Invalid packets -

iptables -t mangle -A PREROUTING -m conntrack --ctstate INVALID -j DROP

For Blocking New Packets That Are Not SYN –

iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW -j DROP

For blocking Uncommon MSS Values:

iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss 536:65535 -j DROP

### For blocking Packets With Bogus TCP Flags:

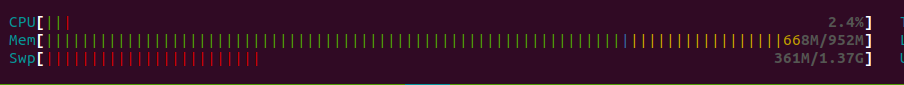
### iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j DROP iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j DROP iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j DROP iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP

1. **Provide \*.pcap network capture file from traffic between attack host and target running the application server**

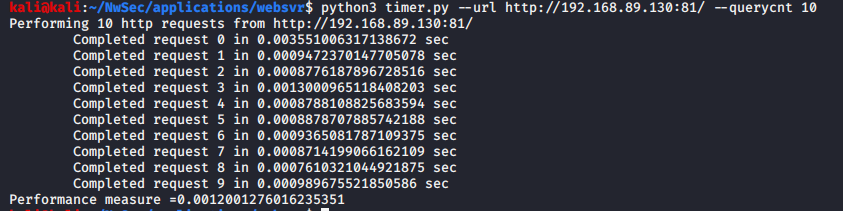
Provided in the zip folder.

1. **Describe the results of the attack**

On the victim machine, we can clearly see that this time DoS attacks via hping3 tool were ineffective with CPU Usage being around 2.4% consistently and consequently system resources not being depleted.



Also, during the initial phase of the attack, we see that the average time taken to complete 10 HTTP request queries is 0.0012 seconds (almost the same average time taken to complete when the machine is not under attack).

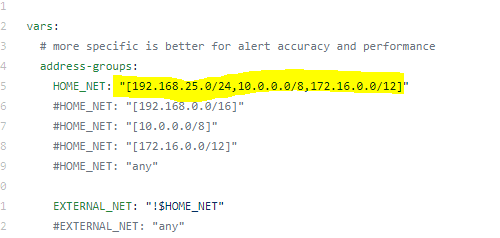


**Part 3**

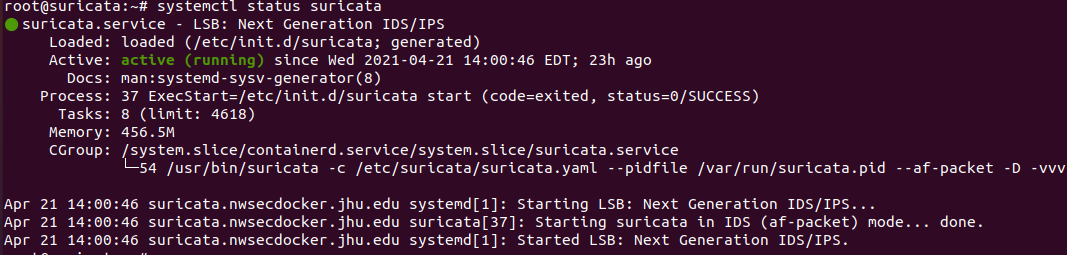
1. **Provide the details used to install and configure Suricata so that it could be replicated by someone else.**

To install and configure the Suricata, I followed the below details :-

1. Downloaded the Docker Container from the github link :- <https://github.com/jhu-information-securityinstitute/NwSec/tree/master/applications/suricata/UbuntuServerX86-64>
2. Checked the configurations in Suricata.yaml file before building the container to make sure that suricata indeed is monitoring my home network.
3. In Suricata.yaml, I also found the information about the files storing the logging and alert events.



1. After making sure everything is correct, I proceeded to build the container via docker build -t tsuricata.
2. I ran the container using - docker run -d --name suricata --hostname suricata.nwsecdocker.jhu.edu --add-host suricata.nwsecdocker.jhu.edu:127.0.1.1 --dns 192.168.25.10 --dns-search nwsecdocker.jhu.edu --privileged -v /sys/fs/cgroup:/sys/fs/cgroup:ro --network host --cpus="2" tsuricata:latest
3. To get interactive shell we fire off – sudo docker exec -it suricata bash
4. To check the status of suricata we use “ systemctl status suricata”



f) To detect the attacks in real time we can use “tail -f /var/log/suricata/fast.log”

1. **Provide reporting details from Suricata that show it detected the attack.**

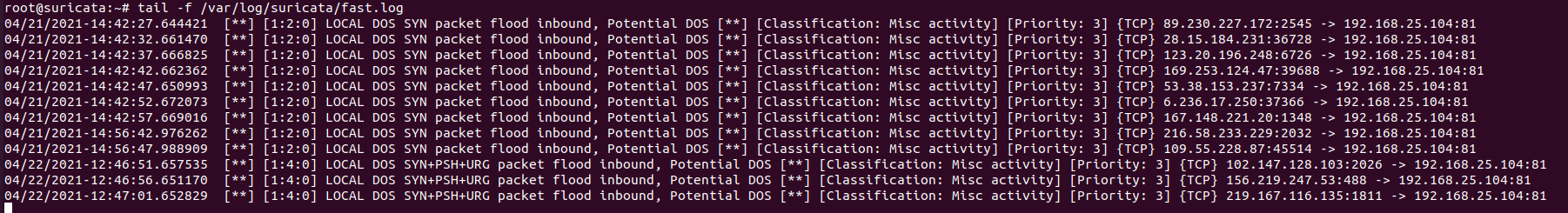
Suricata maintains the log details and related statistics of events under /var/log/suricata/. These logs can be used for monitoring alerts in real time and analysing related statistics. Logs related to Administrative purposes can be found here as well.



To monitor Logs in real time, we can run the “tail” command with “-f” parameters to monitor the latest entries in the fast.log.

**Command**: -tail -f /var/log/suricata/fast.log

As you can see in below, the Suricata is generating alerts in real time when under Denial-of-Service attack.



**Part 4**

1. **Discuss what else could be performed to defend against these attacks**

To defend against Denial-of-Service attacks, we can take many additional measures at various levels such as: -

1. Developing a response plan.
2. At the Network level, we can deploy firewalls, load balancers, Intrusion detection & Prevention Systems, leverage Cloud technologies etc.
3. At the System level, we can tweak the Kernel settings (/etc/sysctl.conf) to maximize the performance of our server as well as effectiveness of iptables rules under DOS attack
4. We can further configure the IP Tables to block ICMP packets, make use of SYNPROXY

Some of the sysctl.conf and additional Iptables rules are given below:-

We can also configure IPTables as well as increase the system resources etc.

### Blocking spoofing packets from private subnets via –

### /sbin/iptables -t mangle -A PREROUTING -s 192.168.25.0/16 -j DROP

### Blocking ICMP Packet and making use of SYNPROXY-

### iptables -t mangle -A PREROUTING -p icmp -j DROP

### iptables -t raw -A PREROUTING -p tcp -m tcp --syn -j CT –notrack

### iptables -A INPUT -p tcp -m tcp -m conntrack --ctstate INVALID, UNTRACKED -j SYNPROXY --sack-perm --timestamp --wscale 7 --mss 1460

### iptables -A INPUT -m conntrack --ctstate INVALID -j DROP

### Few highlighted sysctl based protection: -

### net.ipv4.conf.all.rp\_filter = 1 (protection against IP spoofing)

### net.ipv4.tcp\_syncookies = 1 (allows TCP SYN cookie protection)