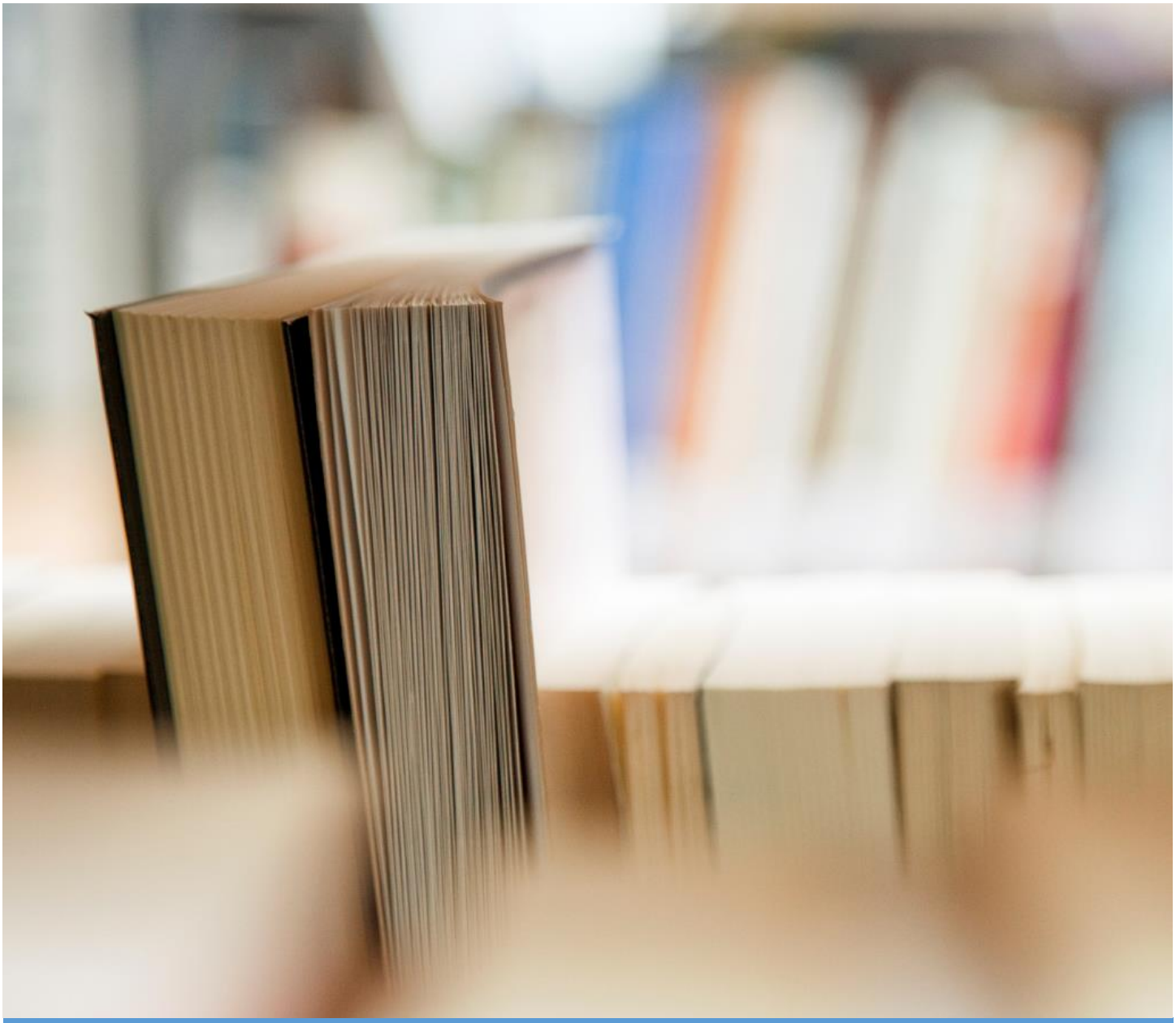

The Cyber Inquisitors

CDX Report

Team members:
Anuj Shah
Chakresh Singh
Naveen Reddy Aleti
Tarun Kumar Kurra

Contents

Abstract

The Cyber Defense Exercise (CDX) started at the midnight of 7th of December and ended by 11:59 AM of 10th December. A set of pre-assigned services were to be kept up and running on three virtual machines on OpenVPN at the end of defending teams (Blue teams) while the attackers (Red teams) were trying to bring down the services. The present report presents the setups that we made prior and during the cyber defense exercise. We have tried to include everything thing that we did and observed during the 60 hours' exercise.

Environmental setup:

Host machine setup:

We were given a .tar file which had configuration, certificate and key files We were given three sets, one each for the Virtual machines. Our team used only two sets of these files because one laptop was hosting two Virtual machines. Then we installed openvpn, easy-rsa. We placed the key and certificate files in /etc/openvpn/easy-rsa/keys folder. We copied the configuration file to /etc/openvpn folder. We made the following changes to the configuration file:

remote 10.52.10.253 1194 -> remote helios.ececs.uc.edu 1194

cert /etc/openvpn/easy-rsa/keys/blue-11.crt

key /etc/openvpn/easy-rsa/keys/blue-11.key

We didn't have to connect to CDX network making use of Socks proxy because we decided to stay in campus for the exercise. We navigated to the /etc/openvpn folder and we typed the following command in the shell on our host Ubuntu:

sudo openvpn client.conf

Now the tap0 connection have been made after typing the above command. To test this, we performed ifconfig in the host machine. The screenshot of the result looks like:

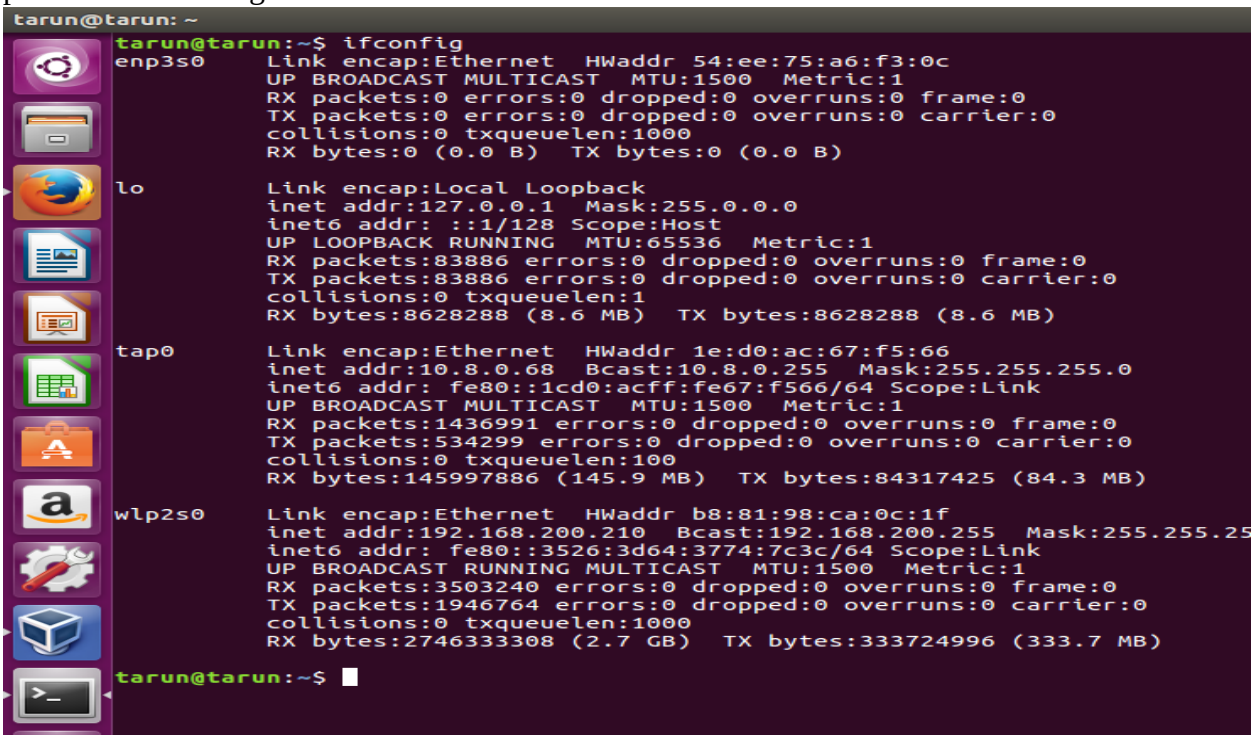

Figure 1: tap0 enabled

Since we can see the tap0 internet interface, now we can connect the Virtual machines to this interface.

Virtual machine setup:

The setting up of the Virtual machine was performed after this. We followed all the instructions provided to us in Blackboard by Prof. John Franco. Then we got our Virtual machines up and rolling. To check if the Virtual machines are connected to this interface or not, we should type the ifconfig command in a shell. If the result looks like below, we are connected:

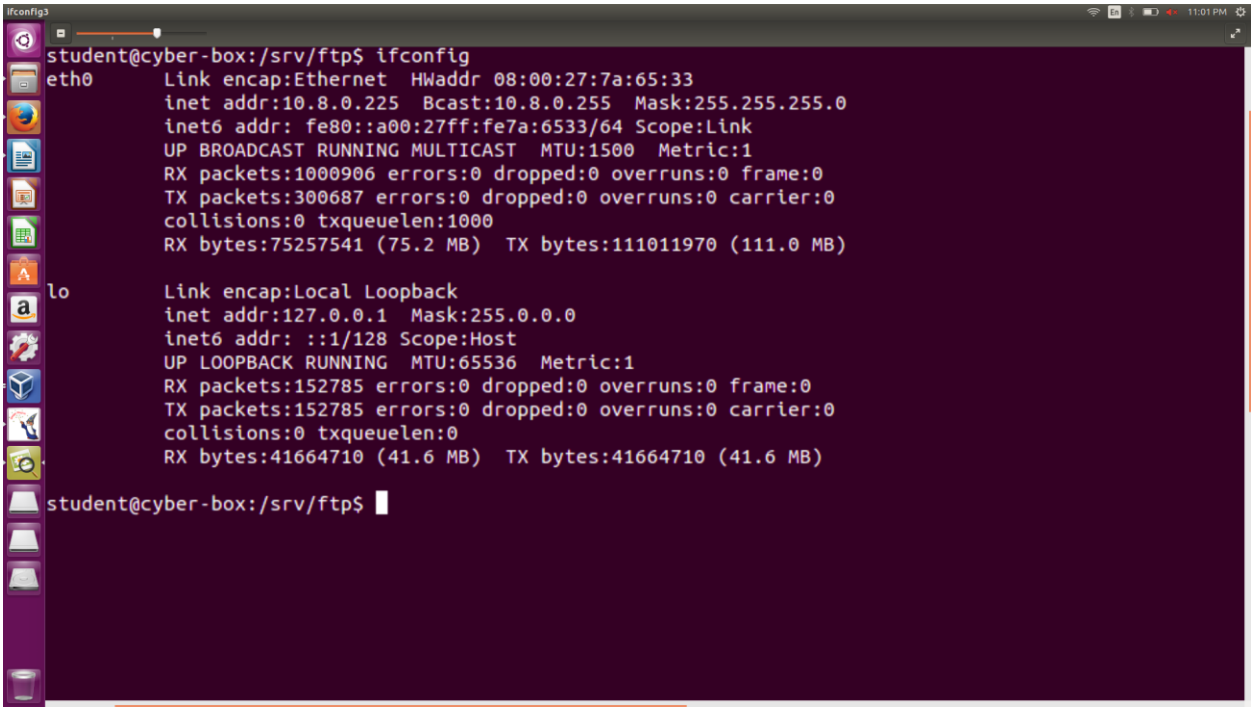

Figure 2: VM setup

Now since We were connected to the internet interface, we could ping from our IP address to any other IP address in the 10.8.0.x (x=0 - 255) range. If the ping is successful, then we are sure that we are connected in the network.

The IP addresses assigned to our host Ubuntu OS are:

10.8.0.53

10.8.0.68

The IP addresses assigned to our Virtual machines are:

10.8.0.224

10.8.0.225

10.8.0.226

We could ping amongst our Virtual machine IP addresses and we were also able to ping to the test machine which is 10.8.0.58. So, this ensured us that we are in the connection.

Pre-Competition:

Hardening the Machine:

Update the OS:

First, we updated the OS.

```
sudo apt-get update
```

Doing this, downloads the package lists from different repositories and "updates" them with the newest versions of packages and their dependencies available.

```
sudo apt-get dist-upgrade
```

This will do the same job which is done by apt-get upgrade. In addition, it will also intelligently handle the dependencies, so it might remove obsolete packages or add new ones.

Inspecting the services:

To know the kind of services running on the system and to have a quick look at the processes accepting connection(ports), we executed the following commands respectively.

```
ps ax
```

It lists all the running processes on the system.

netstat is a useful tool for checking your network configuration and activity.

```
netstat -lnp
```

```
netstat -a
```

Install and configure Firewall – ufw (Uncomplicated Firewall):

Firewall is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules. We were instructed to allow some services, which we have done by typing the following commands:

sudo apt-get install ufw	(For installing firewall security)
sudo ufw allow ssh	(For allowing the ssh service)
sudo ufw allow http	(For allowing the http service)
sudo ufw allow ftp	(For allowing the ftp service)
sudo ufw allow 786	(we have changed the default port of ssh connection from 22 to 786)

sudo ufw allow 3306	(The default port of Mysql which is 3306 is allowed)
sudo ufw status verbose	(To check the status of the firewall)

Secure the shared memory:

Shared memory can be used in an attack against running services. We modified /etc/fstab to make it more secure, we added the following lines, saved and rebooted the system for the settings to take effect. We edited the file by opening it in an editor. The following command is typed:

```
sudo vi  /etc/fstab
```

To the opened file, we added the following line:

```
tmpfs   /run/shm      tmpfs   defaults, noexec, nosuid      0       0
```

Passwords:

We changed the admin password by entering the following command:

```
sudo passwd
```

Then we were prompted to enter the current password of the user and enter the new password and confirm it.

We also changed the password of the user. The command used is:

```
sudo passwd <USER NAME>
```

We created a new user and gave it root privileges. We did this by following the steps below:

```
sudo adduser anuj          (This command adds a user anuj)
```

We will be prompted to enter the new password for the user anuj and we should confirm the password.

```
usermod -aG sudo anuj      (for adding the user to sudo list)
```

We made settings such that, the use of *sudo* would always prompt for password.

SSH Hardening:

1. To secure SSH we have changed the default port from 22 to port number 786 , the new port number (786) is chosen such that it is below 1024, as these are privileged ports that can only be opened by root or processes running as root.
2. As we are using password authentication for logging into services , we disabled ROOT login to secure SSH so that attackers cannot misuse SSH to gain root privileges. But before disabling the root login we created a new SSH user and made sure that the user belongs to the admin group.
3. The above changes are made using following commands:

sudo vi /etc/ssh/sshd_config (to openssh configuration file)

Added the following lines and saved the configuration file:

Port 786 (changes the default port of SSH)
Protocol 2
PermitRootLogin no (this will disable the root login into the os)

After editing the ssh configuration file we restarted the ssh service and opened the new port 786 and allowed it in ufw firewall.

sudo service ssh restart

sudo ufw allow 786

Apache SSL Hardening:

As the SSL v2/v3 protocol has been proven to be insecure, we disabled Apache support for the protocol and forced the use of newer protocols.

sudo vi /etc/apache2/mods-available/ssl.conf (opens the ssl configuration file of apache)

Change "SSLProtocol all" line to "SSLProtocol all -SSLv3" and save the file and close it.

sudo service apache2 restart (restarts the apache service)

Protect "su" by limiting access only to admin group:

To limit the use of **su** by admin users only we created an admin group, then added users to that group and limited the use of su to the admin group only. The commands are

sudo groupadd admin
sudo usermod -a -G admin anuj
sudo dpkg-statoverride --update --add root admin 4750 /bin/su

Harden Network with sysctl settings:

To prevent source routing of incoming packets and log malformed IP's we made few changes in the sysctl.conf configuration file.

sudo vi /etc/sysctl.conf

Added the following lines in the configuration file:

 # IP Spoofing protection
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

 # Ignore ICMP broadcast requests

```
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Disable source packet routing
net.ipv4.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0

# Ignore send redirects
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0

# Block SYN attacks
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 2048
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 5

# Log Martians
net.ipv4.conf.all.log_martians = 1
net.ipv4.icmp_ignore_bogus_error_responses = 1

# Ignore ICMP redirects
net.ipv4.conf.all.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0

# Ignore Directed pings
net.ipv4.icmp_echo_ignore_all = 1
```

sudo sysctl -p (To reload sysctl with the latest changes)

Disabled Open DNS Recursion and Remove Version Info – BIND DNS Server:

sudo vi /etc/bind/named.conf.options

We then added the following to the 'Options' section:

```
recursion no;
version "Not Disclosed";
```

sudo service bind9 restart (restarts the BIND DNS server)

Prevent IP Spoofing:

sudo nano /etc/host.conf (To open host.conf in a nano editor)

We added the following lines to the host configuration file and saved it:

order bind, hosts
nospoof on

Harden PHP for IP Security:

sudo nano /etc/php5/apache2/php.ini (To open and edit the php.ini file)

We added the following lines to the file)

disable_functions = exec, system, shell_exec, passthru
register_globals = Off
expose_php = Off
display_errors = Off
track_errors = Off
html_errors = Off
magic_quotes_gpc = Off
mail.add_x_header = Off
session.name = NEWSESSID

sudo service apache2 restart (Restarted the apache server)

Restrict Apache Information Leakage:

sudo vi /etc/apache2/conf-available/security.conf (to open the apache security configuration file in editor)

We added some lines and we edited some lines in the apache security configuration file:

ServerTokens Prod
ServerSignature Off
TraceEnable Off
Header unset ETag
Header always unset X-Powered-By
FileETag None

sudo service apache2 restart (Restart the service)

Web Application Firewall – ModSecurity & Protection from DDOS Attacks- ModEvasive:

Installed and configured Apache2 ModSecurity and mod_evasive modules on Ubuntu server. It is an open source web application firewall. This is managed by OWASP. ModSecurity is a toolkit for real-time web application monitoring, logging, and access control.

sudo apt-get install libxml2 libxml2-dev libxml2-utils

sudo apt-get install libaprutil1 libaprutil1-dev

We tried to install this programs but the installation was not successful so we skipped it because we already had a UFW firewall.

Fail2Ban – Scans Logs and Ban Suspicious Hosts:

Fail2Ban is a tool used for monitoring the logs of services like **SSH, Apache, Courier, FTP, and more**. It scans the log files and bans IPs that show the, malicious sign like too many password failures, seeking for exploits, etc. Fail2Ban is used to update firewall rules to reject the IP addresses for a specified amount of time, although any arbitrary other action could also be configured. We installed Fail2Ban in our Ubuntu machine and configured the files using the commands below:

sudo apt-get install fail2ban (installs Fail2Ban)

sudo vi /etc/fail2ban/jail.conf (opens the configuration file of Fail2Ban in vi editor)

The following lines are added to the filter rules of Fail2Ban in order to enable the SSH monitoring and banning jail:

enabled = **true**
port = 786
filter = sshd
logpath = /var/log/auth.log
maxretry = 3

(If the hosts are banned then Fail2Ban sends the mail alert to us through email. This function is enabled by adding the following lines in the configuration file)

destemail = anuj.shah.94@gmail.com

action = %(action_**mwl**)s

ignoreip = 127.0.0.1/8 10.8.0.224/24 10.8.0.225/24 10.8.0.226/24

sudo service fail2ban restart (restarted the service fail2ban)

Intrusion Detection - PSAD:

PSAD is a collection of three lightweight system daemons that run on Linux machines and analyze iptables log messages to detect port scans and other suspicious traffic. We installed PSAD in our machine and configured it using following commands.

sudo apt-get install psad (installs PSAD)

sudo vi /etc/psad/psad.conf (opens the PSAD configuration file for editing)

We changed the email address and enable_auto_ids to y and added the following Iptables policies:

```
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG
ip6tables -A INPUT -j LOG
ip6tables -A FORWARD -j LOG
```

After saving the configuration file ,to restart, update the signature file and reload PSAD to complete the installation we used the following commands:

```
sudo psad -R
sudo psad --sig-update
sudo psad -H
```

Check for Rootkits – RKHunter and CHKRootkit:

Both the tools check the system for rootkits. To install the above software tools we used the following commands:

sudo apt-get install rkhunter chkrootkit

sudo chkrootkit (To run CHKRootKit)

sudo rkhunter –update (To run and update RKHunter)

sudo rkhunter –propupd
sudo rkhunter –check

Scan Open Ports - Nmap:

Network Mapper is a free and open source utility for network discovery and security auditing. Nmap is installed using below commands:

sudo apt-get install nmap (to install nmap scan)

nmap -v -sT localhost (Scanning the system for all open ports)

sudo nmap -v -sS localhost (For SYN Scanning)

Analyze system LOG files - LOGWATCH:

LogWatch is a customizable log analysis system. Logwatch parses through our system's logs and creates a report analyzing areas that we specify. Logwatch is easy to use and will work right out of the package on most systems. To install LogWatch on our system and to generate output we used the following commands:

sudo apt-get install logwatch libdate-manip-perl (To view LogWatch output)

sudo logwatch | less (To get email of logwatch report for past 7 days)

sudo logwatch –mailto anuj.shah.94@gmail.com --output mail --format html --range 'between -7 days and today'

Apparmor - Application Armor:

AppArmor is a Linux Security Module implementation of name-based mandatory access controls. AppArmor confines individual programs to a set of listed files and POSIX 1003.1e draft capabilities.

AppArmor is installed and loaded by default. It uses profiles of an application to determine what files and permissions the application requires. Some packages will install their own profiles, and additional profiles can be found in the apparmor-profiles package.

We installed the apparmor-profiles packages from a terminal prompt by following commands:

sudo apt-get install apparmor apparmor-profiles

sudo apparmor_status (To check if the things are running)

Audited our system security - Tiger and Tripwire:

Tripwire is a popular host-based intrusion detection system on Linux. This software can keep track of many different filesystem data points to detect whether unauthorized changes have occurred. It works by collecting details about your computer's filesystem and configuration. It then stores this information to reference and validate the current state of the system. If changes are found between the known-good state and the current state, it could be a sign that your security has been compromised. To install tripwire

sudo apt-get install tripwire

To create a policy file:

sudo twadmin --create-pofile /etc/tripwire/twpol.txt

Then enter the passphrase (Anuj shah) we configured earlier. This creates an encrypted policy file from the plain text one that we specified in the /etc/tripwire/directory. This encrypted file is what tripwire reads when running its checks.

We can now initialize the database that tripwire will use to validate our system. This uses the policy file that we just initiated and checks the points that are specified within. Because this file has not been tailored for our system yet, we will have a lot of warnings, false positives, and errors. We will use these as a reference to fine-tune our configuration file in a moment.

```
sudo tripwire --init                    (The basic way to initialize the database)
```

Tiger:

It is a security tool that can be use both as a security audit and intrusion detection system. Installation

```
sudo apt-get install tiger

sudo tiger                              (To run tiger)

sudo less /var/log/tiger/security.report    (To view the tiger security reports)
```

Prevent Denial of Service (DoS) Attacks:

To prevent the Dos attacks we have limited the number of tcp packets on port 80. That is we have allowed only 25 packets per minute and also 100 packets only in single burst. This is implemented using the following command:

```
sudo iptables -A INPUT -p tcp –dport 80 -m limit 25/minute --limit-burst 100 -j ACCEPT
```

Added Delay to All Requests:

We added one second delay to all network traffic to deter intruders.

```
sudo tc qdisc add dev eth0 root netem delay 1000ms
```

Removed Unnecessary Dependencies, services, programs:s

During inspection of the processes running in the system we found many unwanted services running which are not required for the competition. So in order to save the resources and also to eliminate the risk of security breaches through those systems we have removed all unwanted stuff.

```
sudo apt-get autoremove                 (To remove the unnecessary  dependencies)

sudo apt-get remove mutt

sudo apt-get speech-dispatcher

sudo apt-get modemmanager
```

Removed SSH Access for User libuuid:

As the user libuuid has unrestricted SSH access, there is a chance of attacker gaining libuuid privileges in case there is a vulnerability. So, to restrict the attackers, we have removed this user from SSH access. This is done by following commands:

```
sudo usermod -s /usr/sbin/nologin libuuid
```

Removing DVWA (DAMN VULNERABLE WEB APPLICATION):

We have found a web application with many vulnerabilities in it. As we felt that is not required for keeping up any of the services that are required for the competition we removed that by executing following command:

```
sudo rm -r /var/www/html/dvwa
```

Changing the Default Passwords:

Changed the MYSQL default password (forementioned process was not successful).

```
mysql -u root
update mysql.user set password = “new password” where user = “root” ;
flush privileges;
```

Changing Wordpress passwords:

```
mysql >  use wordpress;
         Show tables;
         Select *from wp_users;
         Update wp_users set user_pass = “md5(new password)” where ID=1;
```

Go to system settings > security & privacy > files & Applications

Turn record files and application usage off. (for more security).

Ensuring that all the Services are up:

We should make sure that all the services which are mentioned to be up and running while the exercise is being performed are in good shape. We should also make sure that the services are made public. We can do this by making the following changes:

Making services public:

Sendmail - nano /etc/mail/sendmail.mc

Replace “127.0.0.1” with “0.0.0.0” (in two places)

Save it

m4 /etc/mail/mail/sendmail.mc > /etc/mail/mail/sendmail.cf

Service sendmail restart

MYSQL - nano /etc/mysql/my.cnf

Set bind-address = 0.0.0.0

Save

Service mysql restart

CUPS - nano /etc/cups/cupsd.conf

Change “Listen localhost:631” to “Listen 0.0.0.0:631”

Save

Service cups restart

Adding DNS server:

We should install a Domain Name Server. The most widely used DNS software on the internet now is Bind9. Security issues that are discovered in BIND 9 are patched and publicly disclosed in keeping with common principles of open source software. A complete list of security defects that have been discovered and disclosed in BIND9 is maintained by Internet Systems Consortium, the current authors of the software.

apt-get install bind9 (install Bind9 software)

Edited named.conf.local, db.myzone and db.reverse.

Dumped all the three files into /etc/bind, overwriting existing named.conf.local.

/etc/init.d/bind9 restart (to restart the Bind9 software after the changes are made)

Penetration testing with Metasploit:

After we hardened the operating system given to us, it was time for us to see if we missed out on any important vulnerability. So, we performed penetration testing with Kali Linux. We used Armitage tool and performed Nmap scan, Hail Mary and we checked for any exploits. We also checked if there were any backdoor entries which can be used by the attackers. The result of all these test is “the target is not vulnerable”. This says that we are in good shape.

After we performed penetration testing on this Operating system we were more confident that the OS that we hardened is strong enough. We were sure that our OS cannot be hacked by any software. We were quite confident of what we had done. We found many vulnerabilities when we performed Hail Mary before starting the hardening process. But once we were done with the hardening we could not find even one exploitable target.

Cloning the hardened OS:

After we hardened the CDX Virtual machine, we cloned the VM two times so that we can use these for the other two IP addresses. We didn't want to repeat the whole hardening process again on the VM's. We may unknowingly skip some steps while hardening the VM, So we cloned the other two VM's from the hardened VM. This saved us a lot of time and energy. It gave us ample time to experiment and play around with the VM. We cloned the VM two times and assigned different IP addresses to them. Now, after all these settings were done we connected to the openvpn and we pinged from one VM to the other and it worked. This says that we are in good shape.

Monitoring:

Shell script for continuous monitoring:(Automation)

For continuously monitoring our services and restart them if they are down we wrote the following shell script. We put that service into cronjob which will run every minute and AUTOMATICALLY TURNS ON the services and ALERTS US in case if they are TURNED OFF by the ATTACKERS.

Code:

```
#!/bin/bash

service1=datetime

service2=time

service3=mysql

service4=apache2

service5=bind

service6=ssh

service7=proftpd

if (( $(ps -ef | grep -v grep | grep $service1| wc -l ) > 0 ))

then

echo "$service1 is running"

else

echo "$service1 is stopped"

/etc/init.d/$service1 start

fi

if (( $(ps -ef | grep -v grep | grep $service2| wc -l ) > 0 ))

then

echo "$service2 is running"

else

echo "$service2 is stoped"

/etc/init.d/$service2 start
```

```
fi
if (( $(ps -ef | grep -v grep | grep $service3| wc -l ) > 0 ))
theni
echo "$service3 is running"
else
echo "$service3 is stoped"
/etc/init.d/$service3 start
fi
if (( $(ps -ef | grep -v grep | grep $service4| wc -l ) > 0 ))
then
echo "$service4 is running"
else
echo "$service4 is stoped"
/etc/init.d/$service4 start
fi
if (( $(ps -ef | grep -v grep | grep $service5| wc -l ) > 0 ))
then
echo "$service5 is running"
else
echo "$service5 is stoped"
/etc/init.d/$service5 start
fi
if (( $(ps -ef | grep -v grep | grep $service6| wc -l ) > 0 ))
then
echo "$service6 is running"
else
echo "$service6 is stoped"
/etc/init.d/$service6 start
```

```
fi
if (( $(ps -ef | grep -v grep | grep $service7| wc -l ) > 0 ))
then
echo "$service7 is running"
else
echo "$service7 is stoped"
/etc/init.d/$service7 start
fi
url=http://localhost/wordpress
if [[ $(wget $url -O-) ]] 2>/dev/null
then
echo "wordpress is running"
else
echo "Opps not running"
fi
```

Save this file at some location and then add an executable. This will help us a lot in automatizing the monitoring process. There is no need for us to continuously keep staring at the monitor and see if the services are being put down or not. If we put this script in use the services will get restarted once they are down, so there is no need for us to worry. While the other teams were continuously monitoring their services, we were analyzing the pcap files. This helped us a lot in understanding their attack patterns.

Steps for running file in cronjob:

chmod +x <file path> (This will make our sh file executable)

sudo crontab -e (If this command is being run for the first time, this will ask us to choose an editor of our choice)

*/1 * * * * <file path> > /tmp/log.log (Put this line in the crontab file)

This command will run our file every minute and store output in log.log file which is created in /tmp folder.

To monitor log look into the file: Dg5kuQT:\C\zY<+ and Qazwsxedcrfv

Installation process of MONIT:

(A software tool has enables us to monitor services from GRAPHICAL USER INTERFACE)

After the competition started, we made up our mind to install and make use of Monit. In Monit we can continuously keep check of the services we need from the host machine without logging into the Virtual machine. We thought this will be a good idea and we started the installation process. We later realized that we did not have internet connection, So we could not install the package from internet. Then we started looking for alternatives and we came up with a solution. We decided that we will use Filezilla and transfer the .tar file of Monit to the CDX Virtual machine from our host.

We made the file transfer from host to the Virtual machine and we untarred it in the virtual machine. Then we had to make a directory(mkdir) and we had to change the mode of that directory(chmod). We had to create a new file and name it monitrc. In this file we need to include a script. In the script we need to mention all the services that need to be viewed and we need to give a condition to mention when to give us an alert.

Monitrc Script:

```
#Monitoring Interval in Seconds

set daemon  60

#Enable Web Access

set httpd port 2812

   allow admin:TarunKumar@7

#Event Queue for 5000 events

set eventqueue basedir /var/monit slots 5000

#MySQL Monitoring

check process mysqld with pidfile "/var/run/mysqld/mysqld.pid"

     if cpu > 80% for 2 cycles then alert

#proftp

check process proftp  with pidfile "/var/run/proftpd.pid "

     if cpu > 80% for 2 cycles then alert

#ssh

check process sshd  with pidfile "/var/run/sshd.pid"

     if cpu > 80% for 2 cycles then alert
```

#sendmail

check process sendmail with pidfile "/var/run/sendmail/mta/sendmail.pid"

if cpu > 80% for 2 cycles then alert

#apache2

check process apache2 with pidfile "/var/run/apache2/apache2.pid"

if cpu > 80% for 2 cycles then alert

#ftp

check filesystem ftp with path /srv/ftp

if space usage > 80% then alert

set mailserver localhost

set alert you@example.com

##mmonit (optional)

#set mmonit http://localhost:8123/collector

Monit can be checked from the host OS also. The URL for monit will be:

http://<IP address>:<httpd port>

Example:

http://10.8.0.224:2812/ (URL for checking monit status for the IP address 10.80.224)

http://10.8.0.225:2812/ (URL for checking monit status for the IP address 10.80.225)

http://10.8.0.226:2812/ (URL for checking monit status for the IP address 10.80.226)

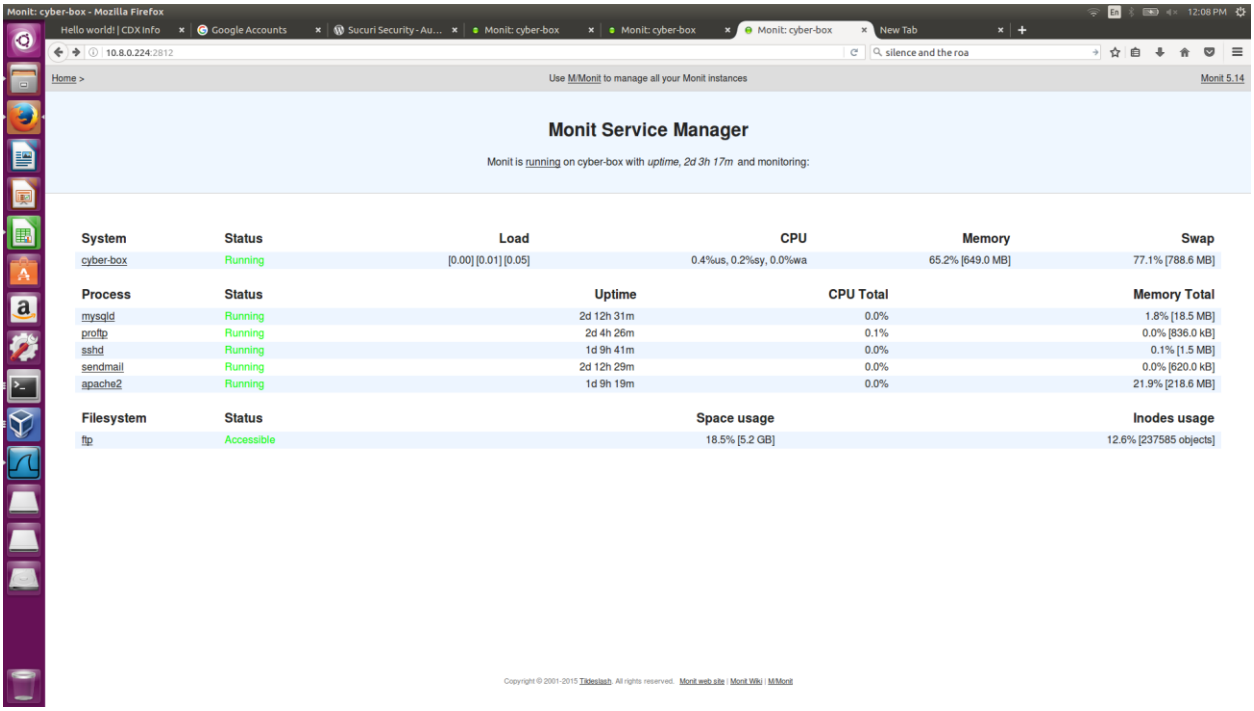

Figure 3 Monit frontend

This is how the screenshot of Monit looks like. You can see that it clearly shows the status of the services and it also alerts if the service goes down. This software made our lives much easier.

Mid-Competition:

We were all setup and ready to get attacked. We were confident about our setup but were also wary of some unforeseen attack vectors. We did not underestimate our opponents. Once the competition started we were witnessing a flood of packets. We captured the flow of packets using *Wireshark* in the host Operating System. We were saving the pcap files from time to time. We did not want to miss any attack. We were only focusing on the traffic which was targeted towards our IP addresses. A screenshot of wireshark is given below:

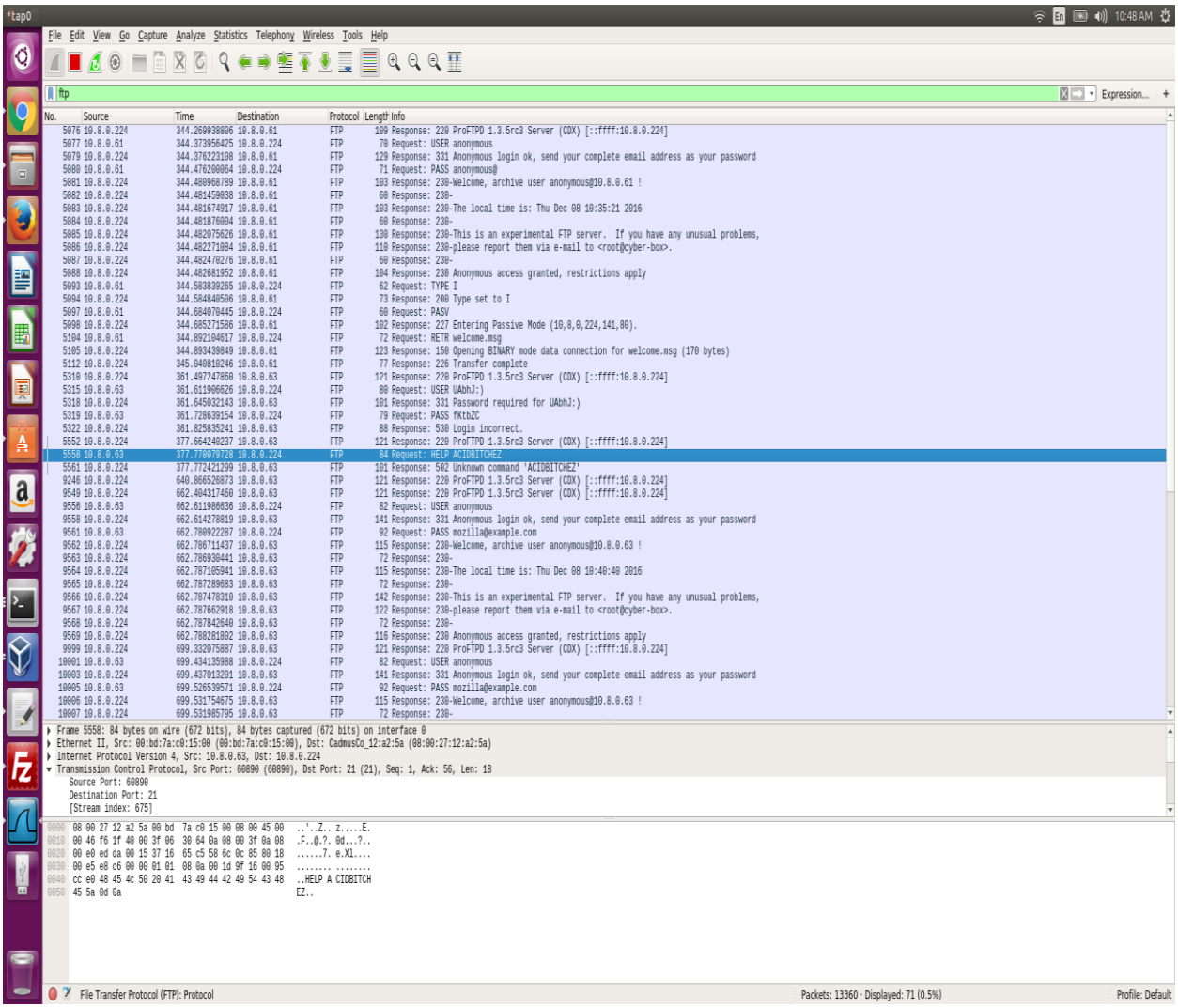

Figure 4: Wireshark screenshot

The following is an image which includes the IP address of users who tried to use login credentials to access the services on our machine. These IP addresses can be of attackers or scorers who evaluate us:

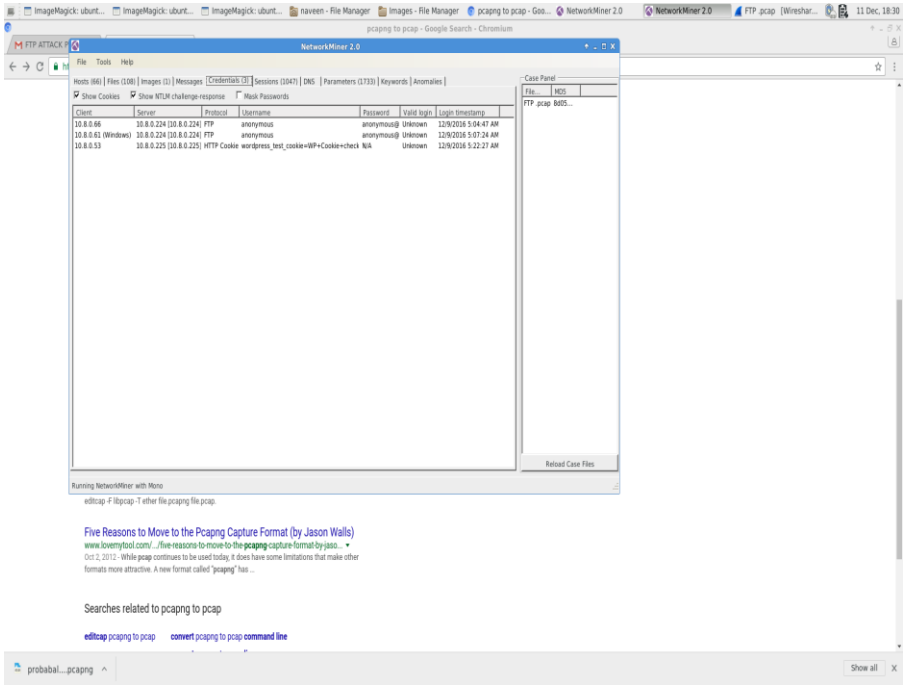

Figure 5: IPs of users who tried to login to our VMs

The following image shows all the IP addresses of the red teams as well as others who tried to access machine 10.8.0.224:

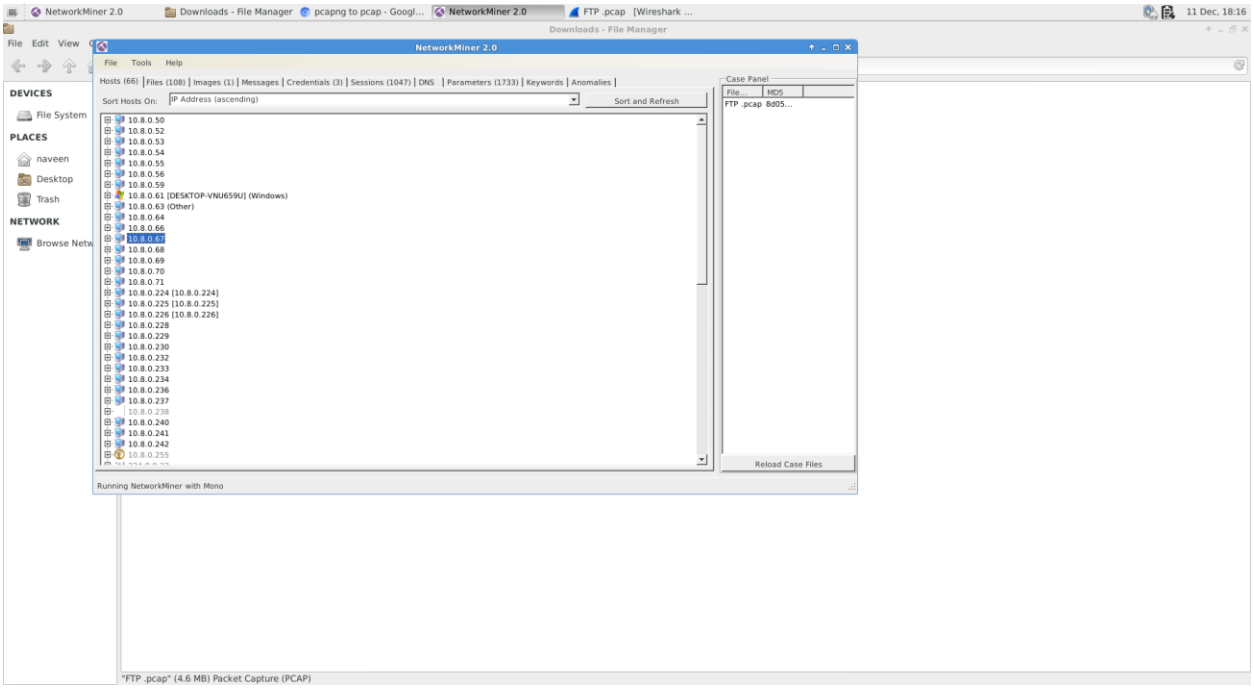

Figure 6: IPs of Red team users who tried to access our machine at 10.8.0.224

The next image, shows what all are the services the Red team tried to access through port 80 on the machine 10.8.0.224:

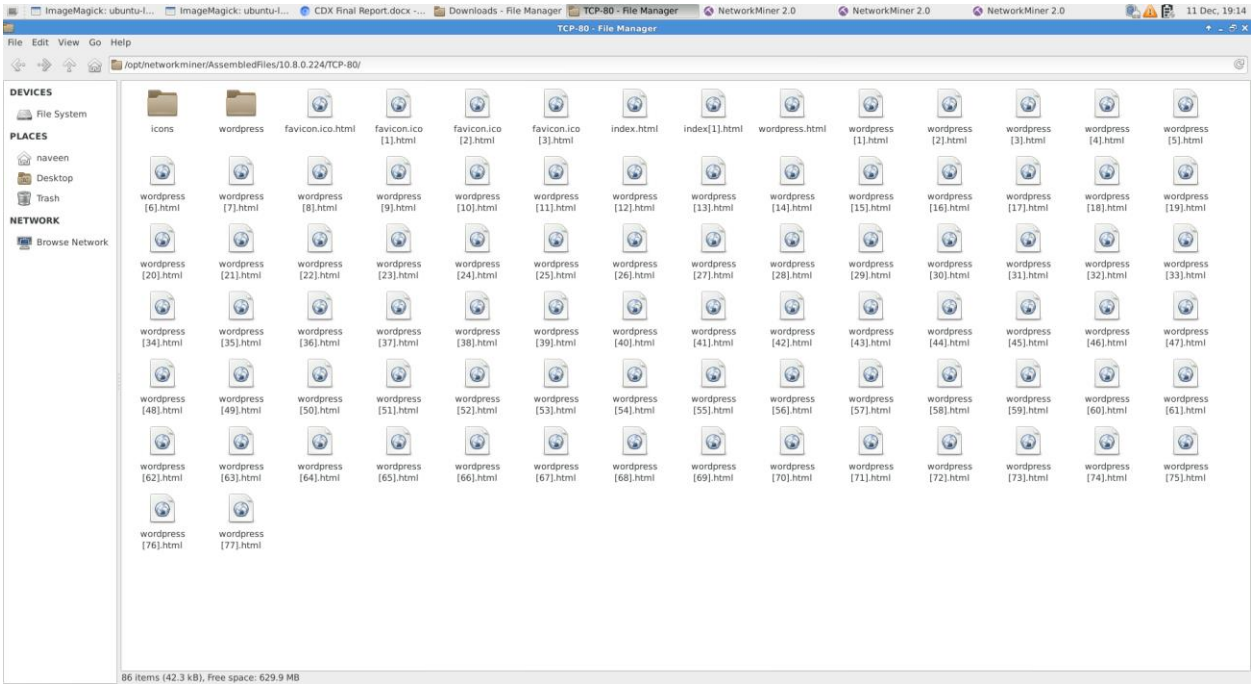

Figure 7: Red teams attempts to access port 80

During the competition, we were receiving a lot of traffic from 3 specific IP addresses. They were:

10.8.0.60

10.8.0.63

10.8.0.66

We knew that amongst these three there should be an attacker. So, we started searching for the attacker. The attackers could only manage to make changes to the welcome.msg file. The we considered the FTP logs and we searched for the IP address of the person who changed the contents in the file. The changed welcome.msg file looks like:

Figure 8: The welcome.msg file

The attacker's name might be "Brian". He could not bring our services down so he thought he could panic us by changing the welcome.msg file. We gave him permission to send us FTP requests and make changes to the file. Later we realized that we should only give them permission to download the file and we need not give them permission to overwrite the file. Then we changed the access permissions(chmod) and gave them access only to download the files and not overwrite the files.

We wanted to figure out the IP address of “Brian”. So, we looked into the FTP logs and we found this:

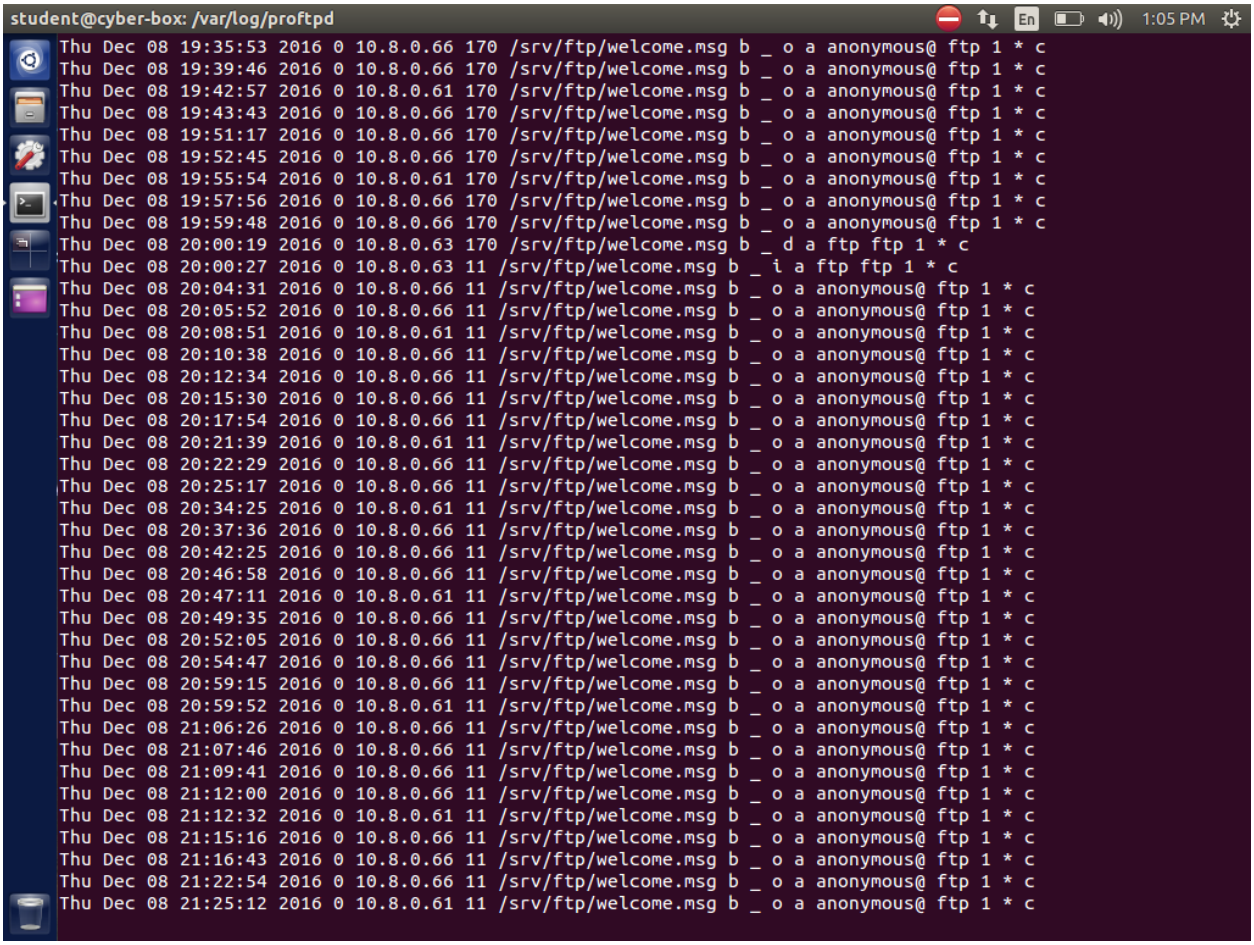

Figure 8: FTP log file

From the above figure, it was evident that the attacker’s IP address is 10.8.0.63. This is the only IP that deleted the welcome.msg file and replaced it with its own welcome.msg file.

Side Notes:

- We tried installation of snort (for intrusion detection) on host machine before the competition. We faced several dependency related errors during installation and setup. We could successfully run snort and include the “community rules” but later we did not use it and kept monitoring using “monit”.

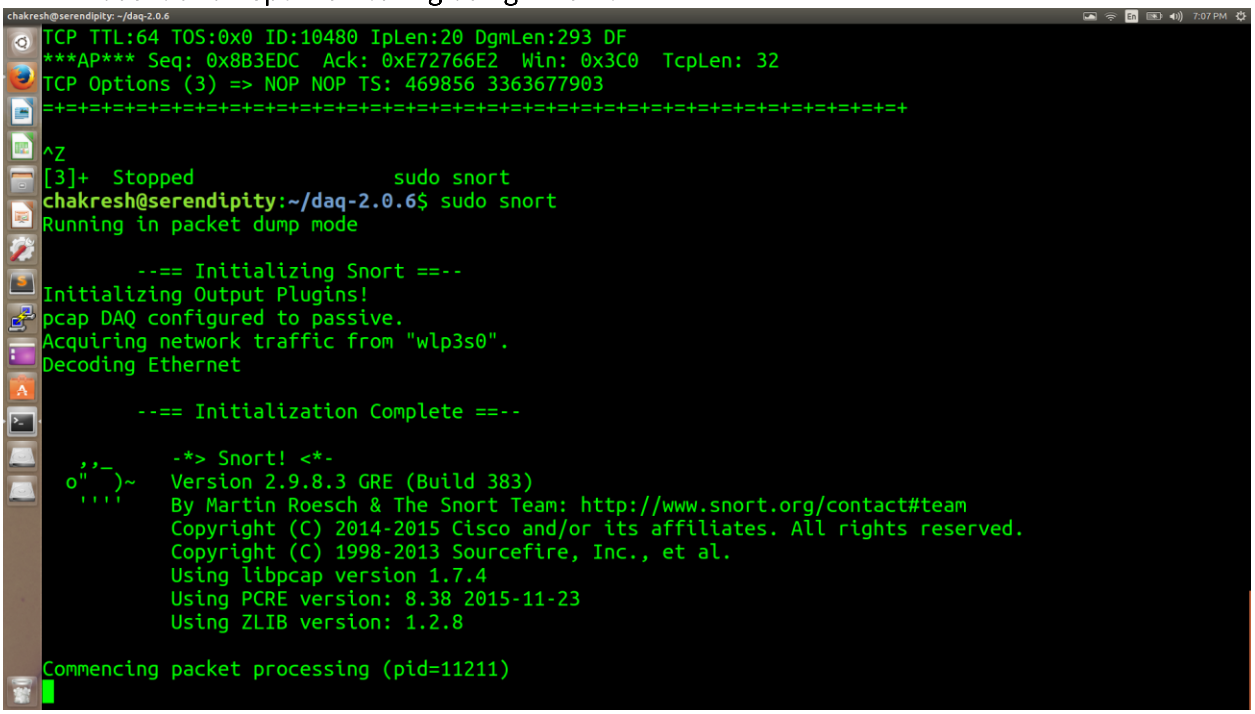

Figure 10: Successful Snort implementation on host machine

- We also came to know about some honeypot software tools for trapping the attackers in a fake environment and study the attack vectors. But to avoid adding unforeseen vulnerabilities to our systems through such installations and setups, we chose not to use them.
- Before updating the .ova file given to us, we checked the SHA256 hash of the file with that given in the Blackboard along with the file.
- We checked the command history to see what all was done with the OS image before it was given to us. We noticed that the history was deleted a number of times and mutt email client was used. We also observed few ftp requests. We checked the Internet for vulnerabilities in the mutt email client. And also, we made it compulsory for each team member to delete the command history after every step using the command:
    - history -c && history -w

- On the second day we saw some changes in the welcome.msg file in the ftp.

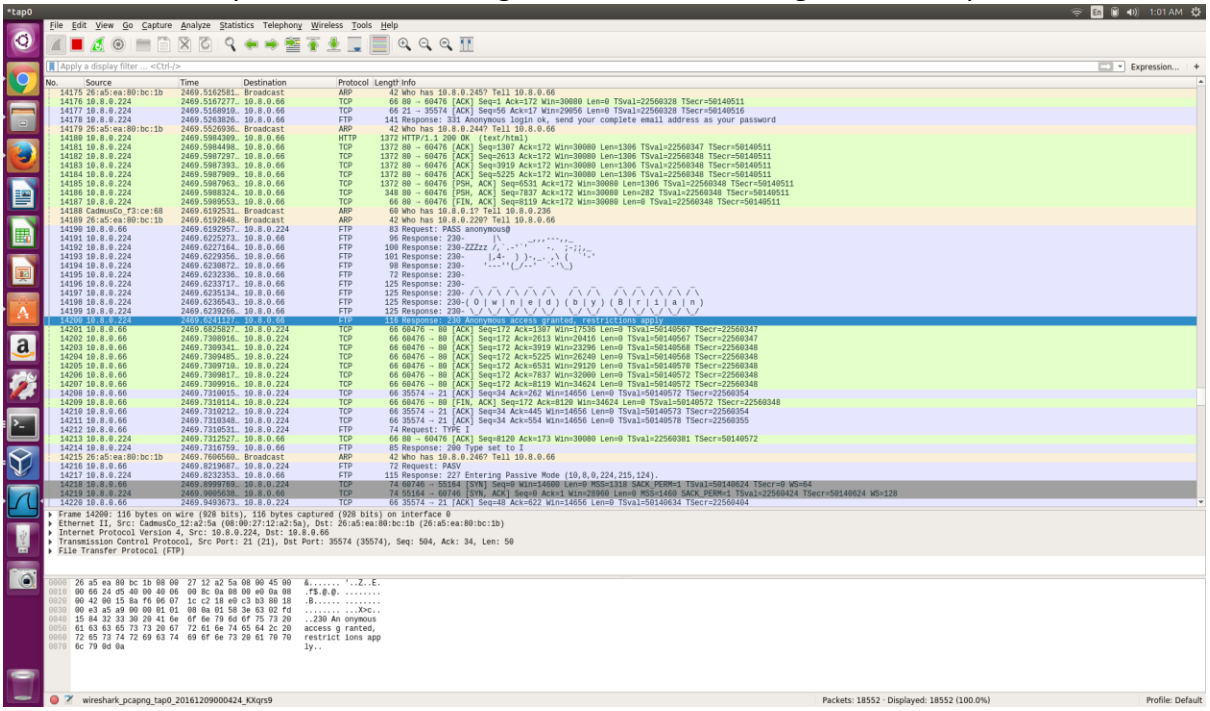

Figure 11: Wireshark showing change in ftp welcome.msg file

- Cyber kill chain:
  We were witnessing the first stage of the cyber kill chain – The Reconnaissance stage. In the reconnaissance stage the attackers just research, identify and select the targets. The next stage is the weaponization stage, in this stage the Intruder creates remote access malware weapon, such as a virus or worm, tailored to one or more vulnerabilities. We were not infected with any virus. They tried to infect us with a virus by inserting some shell script files and some php files, but we were intelligent enough not to get infected by that virus. We did not open these files and we deleted them. We retrieved the information in one of the files. The script in the file wanted to add itself to get permission to login as root. If we would have opened the file by mistake then we would have been infected by the virus. The attacker would come into the system and become root and he would have brought down all our services. The script written in one of these files is:

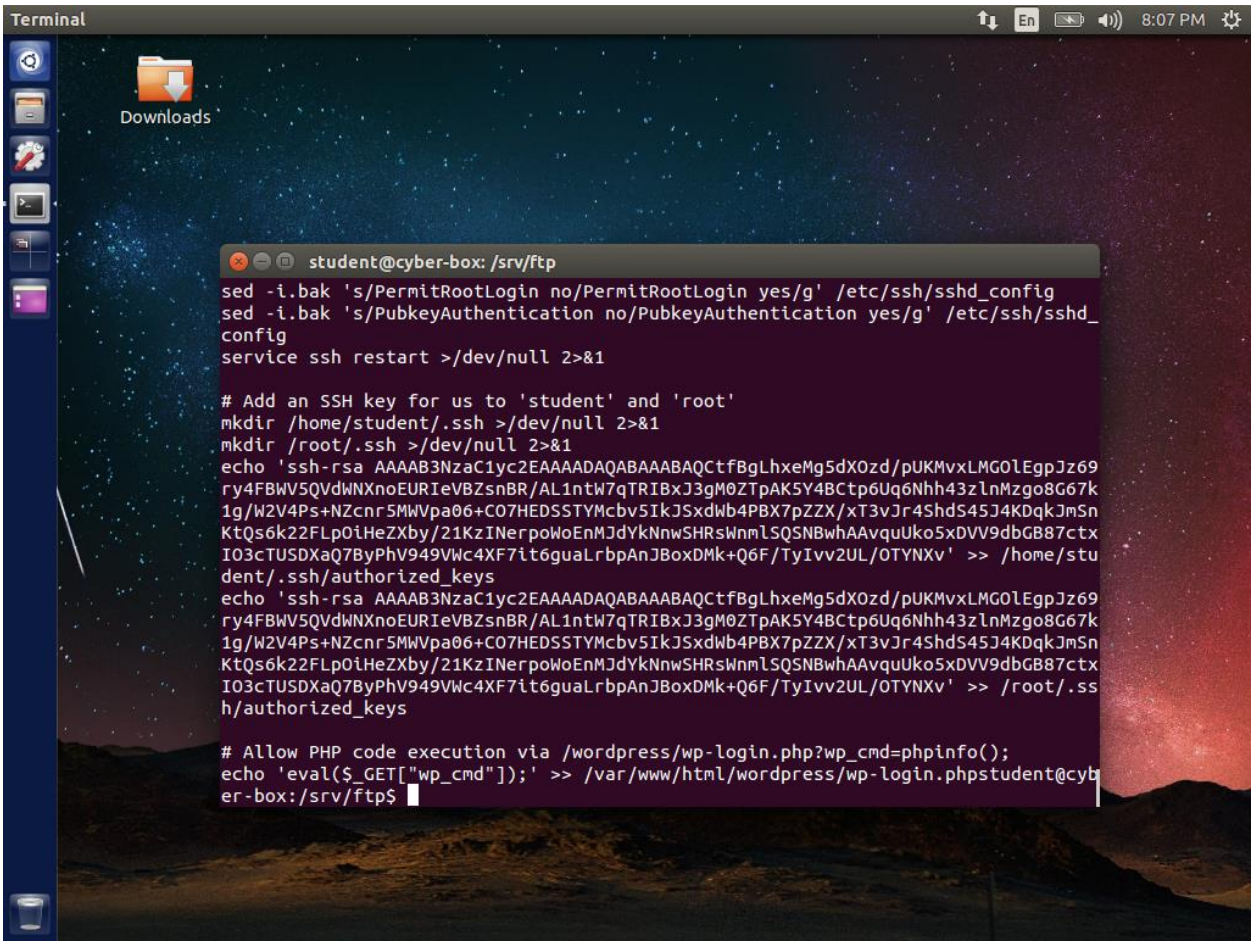

Figure 12: fun.sh

We were not infected any further. The attackers could not succeed in getting us, we did not take the bait.

- We removed some unwanted services from Linux to avoid breaches in the system. We uninstalled Mozilla Firefox and mutt email client.

- We changed the privilege of wp-admin folder (wordpress admin)
    - chmod 700 wp-admin

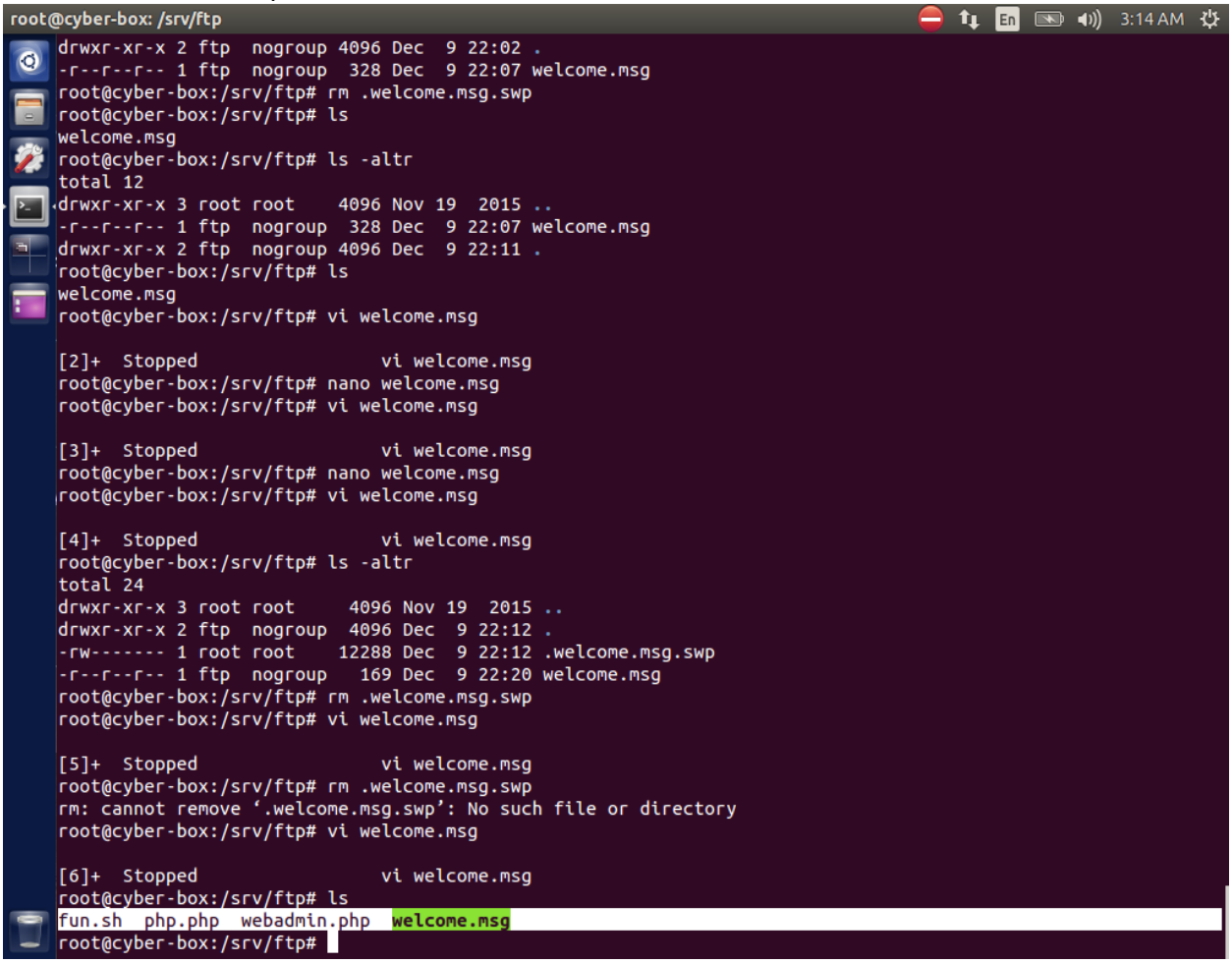

Figure 9Changing privileges

Learnings:

- We learnt a lot about various kinds of configuration settings that can be done to make any system immune to cyber attacks.
- We also learnt about how to use various tools such as uncomplicated firewall (ufw), fail2ban, monit, snort, chkrootkit, rootkit hunter, iptables, wireshark etc. Also, we became more familiar to the Ubuntu Operating System.
- We learnt something about shell scripting and logging.
- We could have made our monitoring even better is we could have used *snort* properly.
- Instead of just relying on strong passwords, we could have also used password encryption.

References:

After downloading the CDX Ubuntu .ova file, we started looking for the online resources to aide us in finding out the common vulnerabilities in the services that must be hosted. We also looked for other changes that can be been done in the configuration files and kernel modules. The online resources that we made use of are:

https://www.thefanclub.co.za/how-to/how-secure-ubuntu-1604-lts-server-part-1-basics

http://www.tecmint.com/remove-unwanted-services-from-linux/

http://www.thegeekstuff.com/2010/08/snort-tutorial