James Madison University

Integrated Science & Technology (ISAT)

ISAT 460 Networking & Cyber-Security II

Lab 1 Linux, Virtualization & Tools

Author(s):
Joey Arbogast
Josie Salcedo

Submitted to:
Dr. Emil Salib

February 27, 2017

Honor Pledge:I have neither given nor received help on his lab that violates the spirit of the JMU Honor Code.

Joey Arbogast	Josie Salcedo
Signature	*Signature*
Date	*Date*

What is new & Best Practices

- to do list: provides inline highlighted todo tasks and a summary list. check it out! To disable to do list and highlighting, see mystyle.sty
- to comment out text using % sign or begincomment ...endcomment
- highlight text using the package called soul. See example

Contents

List of Figures

List of Tables

Todo list

1 Introduction

The purpose of this lab is to understand botnets and how there structure works. The lab does not focus on exploits and infecting computers with botnet malware. Instead this lab focuses on how small single board computers could be used by either a disgruntled employee or a third party who is given access to a company's network infrastructure. Once planted these small devices could be used to gather information on the company's network, cause malicious actions or provide a base of operation to launch DDoS attacks from.

In this lab we focus on how to set up the botnet web application PlugBot, created by Jeremiah Talamantes at Red Team Security. We setup the system and demonstrate some simple DoS and DDoS attacks using the PlugBot web application. In our final analysis we discovered that DDWRT as a virtual machine, is incapable of handling the bandwidth and number of connections generated by an Hping3 tcp flood attack. Only one bot is really needed to bring down the network infrastructure. Continuing research into whether professional routing equipment can handle the attacks, and how many PlugBots would it take to crash a network.

Research Sources

- Red Team Security - Hardware Botnet Research [1]
- Red Team Security -PlugBot-Plug Issues [2]
- Red Team Security - PlugBot-C2C Issues [3]
- Mindi McDowell - Understanding Denial-of-Service Attacks [4]
- VMWare - LAN Segment Requirements [5]

2 Lab Network Diagrams & Tables

2.1 Part 1 Exercise 1 - Network Diagrams & Tables

Figure 1 shows the network configurations constructed and exercised in Part 1 Exercise 1 for the Raspberry PI setup and physical DDWRT AP.

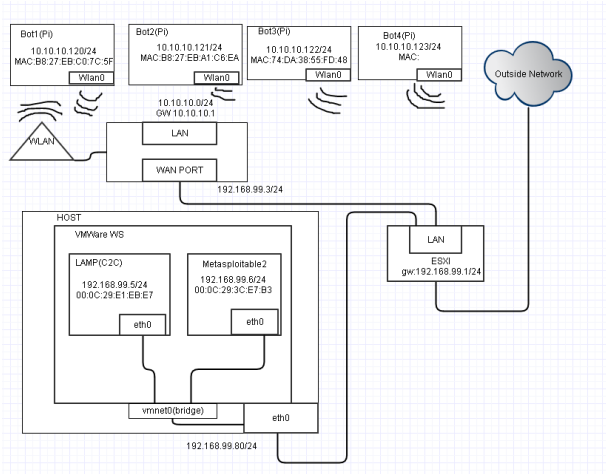

Figure 1: Network Topology For Part 1 Exercise 1-Raspberry Pi and Physical DDWRT AP

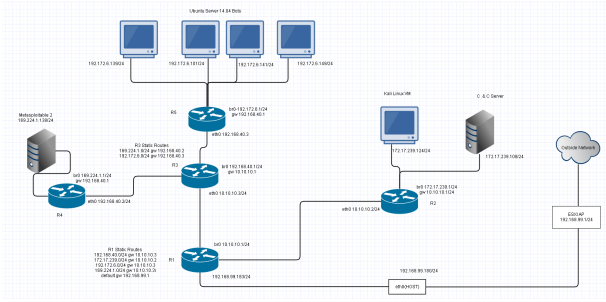

Figure 2: Network Topology For Part 1 Exercise 1-All Virutal Machine Setup

Figure 2 shows the network configurations constructed and exercised in Part 1 Exercise 1 for the alternate setup with all virtual machines.

Table 2 provides the network interfaces information for the hosts, switches in Figure **??**.

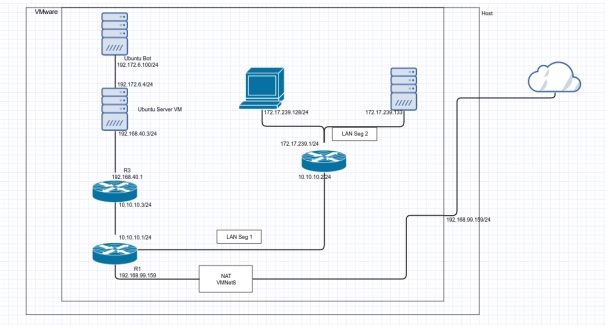

Figure 3: Network Topology For Part 2 Exercise 3 adn 4

2.2 Part 2 Exercise 3 and 4 - Network Diagrams & Tables

Figure 4 shows the network configurations constructed and exercised in Part 2 exercise 3 and 4.

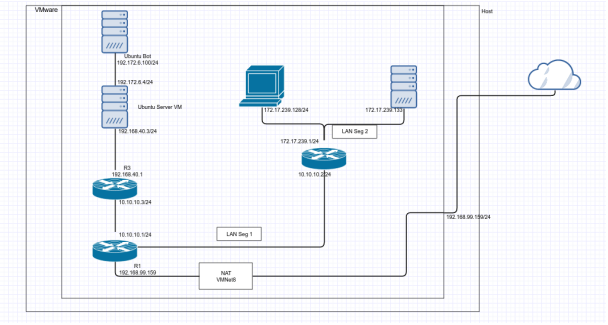

Figure 4: Network Topology For Part 2 Exercise 3 adn 4

2.3 Part 1 Exercise 2 - Network Diagrams & Tables

2.4 Part 2 Exercise 1 - Network Diagrams & Tables

2.5 Part 2 Exercise 2 - Network Diagrams & Tables

3 Lab Exercises : Results & Analysis

3.1 Part 1 Exercise 1: PlugBot Command and Control and Raspberry Pi Bot Configuration(OR VM BOT) (Joey Arbogast)

3.1.1 Analysis & Evidence

3.1.1.1 Step 0: Preparation Raspberry Pi OS, VM downloads

See Figure 1, 2 and Table ?? and ?? in Section 2

For the Virtual Machine Environment, we downloaded and ISO image of Ubuntu Server 14.04. Then we downloaded a Metasploitable 2 VM, Turnkey Linux LAMP Stack VM image, dd-wrt VM and Kali Linux 2016.2 VM.

For the dd-wrt VM setup, we opened the network configuration in VMware for the first dd-wrt VM. Next, we set the first network adapter to the Bridged(Automatic) network. Then we selected the second network adapter and then click the LAN Segments and added a new LAN segment 1 and LAN Segment 2, 3 and 4, shown in Figure 5. We clicked OK after creating the LAN segments.

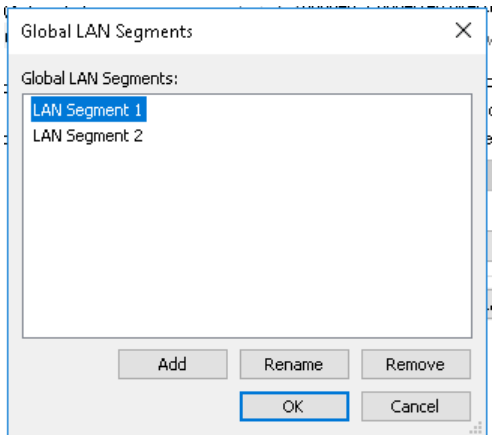

Figure 5: Created LAN Segments 1 and 2

We highlighted network adpater 2 and selected LAN Segment 1 for network adapter 2. Next, we booted up the VM and login with user:root passwd: admin. We used the command `ifconfig br0 10.10.10.1 netmask 255.255.255.0` to assign our br0 interface to the subnet 10.10.10.0/24.

Next, we assigned our Ubuntu Desktop VM network adapter to LAN Segment 1, and started the VM. Then we gave the Ubuntu VM a static IP on the same network as our br0 interface on the router. From a browser we logged into the DDWRT interface at 10.10.10.1 and clicked the setup tab to login.

Under the Local IP Address section under Network Setup we set the IP to 10.10.10.1, subnet mask 255.255.255.0 and made sure DHCP server is enabled. Then we clicked apply settings. Shown in Figure 6

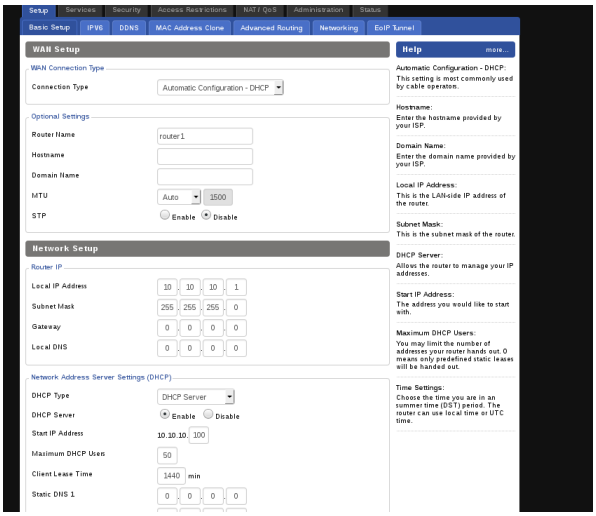

Figure 6: Shows the basic WAN and Network Setup

Then we clicked on the Advanced Routing Tab and created the static route by, Set the Operating Mode to Gateway. Under the static routing section, gave the route a name(toRouter2). Set the Destination LAN Net to 172.17.239.0, mask 255.255.255.0, Gateway 10.10.10.2, Interface LAN & WLAN. Then we applied the changes and saved.

On the second DD-WRT VM, set the first network adapter to LAN Segment 1, and the second Network Adapter to LAN Segment 2 and start the VM. We logged in to the dd-wrt VM and set br0 to 172.17.239.1, using the same command on the other dd-wrt. Next, we switch the network adapter on the Ubuntu Desktop VM to LAN Segment 2 and used ifdown ens33 and ifup ens33 to obtain a new IP from the subnet.

After logging in to the web interface, we changed the WAN setup to Static IP, and set the IP address to 10.10.10.2/24, GW 10.10.10.1. Then we set the Static DNS 1 to our Host's networks gateway(192.168.1.1). We found this was the only way to get an outside connection on this router, otherwise we couldn't ping google.com.

Then we set the Local IP address under Network Setup to 172.17.239.1/24 Apply and Saved. Then we clicked on the Advanced Routing Tab, set the operating mode to Gateway. We gave the route the name (toRouter1) and set the Desti-

nation LAN Net: 10.10.10.0, mask 24, GW 10.10.10.1, Interface ANY. Applied and Saved. With the Ubuntu Desktop VM still assigned to LAN Segment2, we started a Ubuntu Server 14.04 VM and connected the network adapter to LAN Segment 1.

First we pinged 10.10.10.1 from the Ubuntu Desktop VM. Then we pinged 10.10.10.124(Ubuntu Server VM). Then we pinged google.com and was successful on all ping test. Then from the Ubuntu Server VM we pinged 172.17.239.146 (Ubuntu Desktop VM) and google.com with success. Figure 7 shows that we can ping multiple subnets from 172.17.239.1/24 subnet.

Figure 7: Shows we can ping the subnets 10.10.10.0/24, 192.168.40.1/24, 192.168.1.1/24 and google.com

Q1: What is the purpose the br0 interface on the DDWRT router VM?
A1: It creates a bridge between the WAN(eth0) and the LAN port (eth1) on the DDWRT router allowing the machines inside the different subnets we created to reach the outside network on the Host machine. This allows us to communicate with say google.com through the host machine, because LAN segment 1 is bridged to the host network.

Q2: Explain what LAN segments are in VMWare.
A2: LAN segments in VMWare are like host-only vmnet interfaces, except there is no provided DHCP server. You can either create static IPs for all the VMs on the LAN segment or provide your own DHCP server. Which is what we did with the DDWRT Router VMs. They provide the DHCP and DNS servers for each corresponding LAN segment they are assigned to and

also allow us to route to other LAN segments. [5]

3.1.1.2 Step 1: Configuring the Command and Control Server for the Botnet

After downloading the Linux Turnkey LAMP stack VM, we set the network adapter to LAN Segment 2. When it booted up we gave the root user the password Checkout1. Then we entered the advanced menu and pressed enter on Networking shown in Figure 8. Using the arrow keys we went to StaticIP option. Then we entered the information shown in Figure **??**. After clicking enter we selected to go back to the main menu and then selected Quit. We answered yes to our we sure and then logged in with our root credentials.

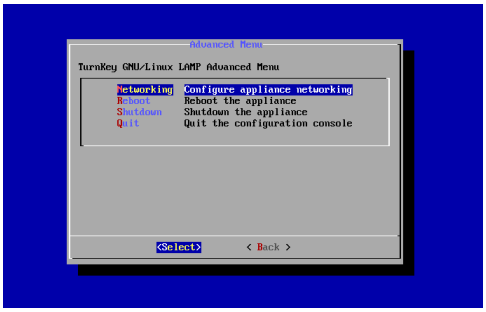

Figure 8: The main screen when booting up the Linux Turnkey LAMP Stack VM

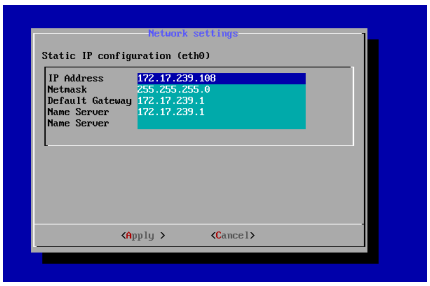

Figure 9: Shows the Static IP information we assigned to the VM

Next, next we installed the following packages with apt-get install: mysql-client, mysql-server, php5, libapach2-mod-php5, php5-mysql, php5-curl, php5-mcrypt, curl, wget, wput, zip, git vsftpd. After the packages finished installing we changed our directory to /var/www. Then we deleted everything in the www directory. When asked about a password for mysql we set it to Checkout1. Next, we created the file .htaccess in the www directory with nano and saved it

with the contents shown in Figure 10 which was taken from Appendix C of the Lab Instructions.

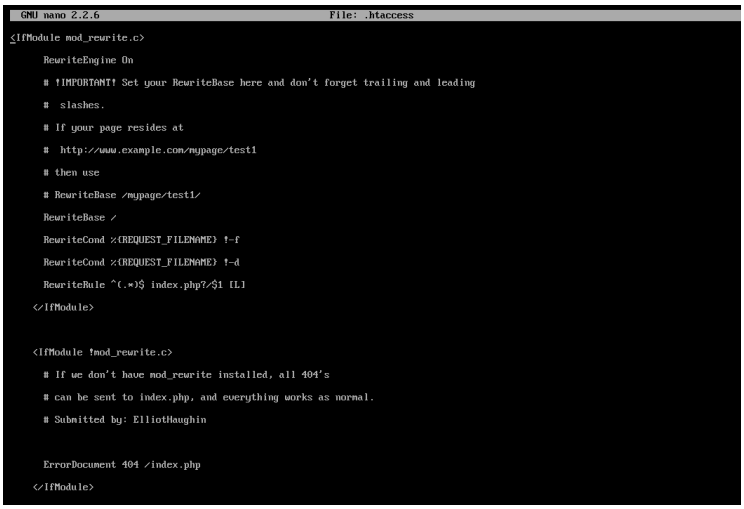

Figure 10: Shows the of the .htaccess file

Next when enabled the apache2 rewrite mod using the command `a2enmod rewrite` and then restarted apache with `service apache2 restart`. We opened the apache2.conf file and used `ctrl+w` to search for the line with `<Directory /var/www`. We changed the line AllowOverride from None, to All shown in Figure **??**. We restart apache after editing the config file.

Figure 11: Shows the line we changed in the apache2.conf file

Next, from the var/www directory, we used the command `git clone https://github.com/redteamsecurity/` to clone the PlugBot C2C files from the git repository. Then we used the command `cp -R PlugBot-C2C/* .` to copy all the files in the directory to the www directory. After verifying all the files had copied to /var/www we deleted the PlugBot-C2C directory. Figure 12 shows the contents of the /var/www directory now.

```
root@lamp /var/www# ls
LICENSE.md  README.md  application  css  db_plugbot.sql  db_plugbot.sql.zip  img  index.php  js  system
root@lamp /var/www#
```

Figure 12: Shows the directories in our /var/www directory

Next, we used the command `unzip db_plugbot.sql.zip` to unzip the plugbot SQL script. Then we used the command `mysql -u root -p` and entered the password toor when prompted. First, we created the database for the plugbot script with the command, `create database db_plugbot;`. Then we used the command `use db_plugbot` to select the database, followed by `source db_plugbot.sql` to run the SQL script. After the script ran, we exited mysql using the command exit. Next, we enter the /var/www/applications/config directory and opened the database.php file with nano. We changed the password line for our sql database to our password which was Checkout1. Figure 13 shows the line we changed.

```
 GNU nano 2.2.6                              File: application/config/database.php

|       ['pconnect'] TRUE/FALSE - Whether to use a persistent connection
|       ['db_debug'] TRUE/FALSE - Whether database errors should be displayed.
|       ['cache_on'] TRUE/FALSE - Enables/disables query caching
|       ['cachedir'] The path to the folder where cache files should be stored
|       ['char_set'] The character set used in communicating with the database
|       ['dbcollat'] The character collation used in communicating with the database
|                                 NOTE: For MySQL and MySQLi databases, this setting is only used
|                                 as a backup if your server is running PHP < 5.2.3 or MySQL < 5.0.
|                                 (and in table creation queries made with DB Forge).
|                                 There is an incompatibility in PHP with mysql_real_escape_string(
|                                 can make your site vulnerable to SQL injection if you are using a
|                                 multi-byte character set and are running versions lower than thes
|                                 Sites using Latin-1 or UTF-8 database character set and collation
|       ['swap_pre'] A default table prefix that should be swapped with the dbprefix
|       ['autoinit'] Whether or not to automatically initialize the database.
|       ['stricton'] TRUE/FALSE - forces 'Strict Mode' connections
|                                                       - good for ensuring strict SQL while devel
|
| The $active_group variable lets you choose which connection group to
| make active.  By default there is only one group (the 'default' group).
|
| The $active_record variables lets you determine whether or not to load
| the active record class
*/

$active_group = 'default';
$active_record = TRUE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = 'Checkout1';
$db['default']['database'] = 'db_plugbot';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

Figure 13: Shows the edited password line in database.php file

We then opened the php.ini file with nano located in /etc/php5/apache2/. We used `ctrl+w` to search for the line `short_open_tag =`. Then we changed it from Off to On, shown in Figure 14. We saved the file and exited, and restarted apache2. Next, from our Kali VM on LAN Segment 2, we opened a web browser and typed our C2C servers IP in the address bar, 172.17.239.108. Figure 15 shows the login screen and that we successfully set up the PlugBot C2C server. Figure 16 shows that we successfully logged into the administrator page.

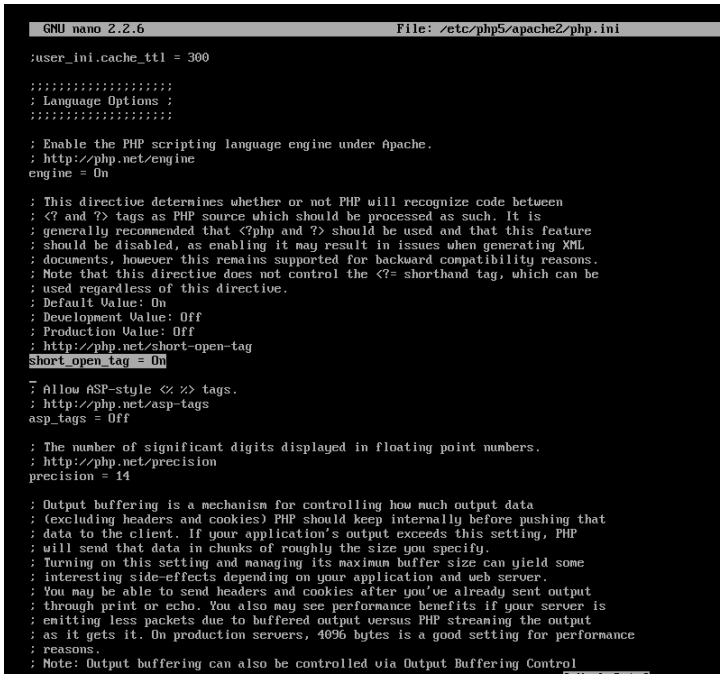

Figure 14: Shows the edited line in the php.ini file

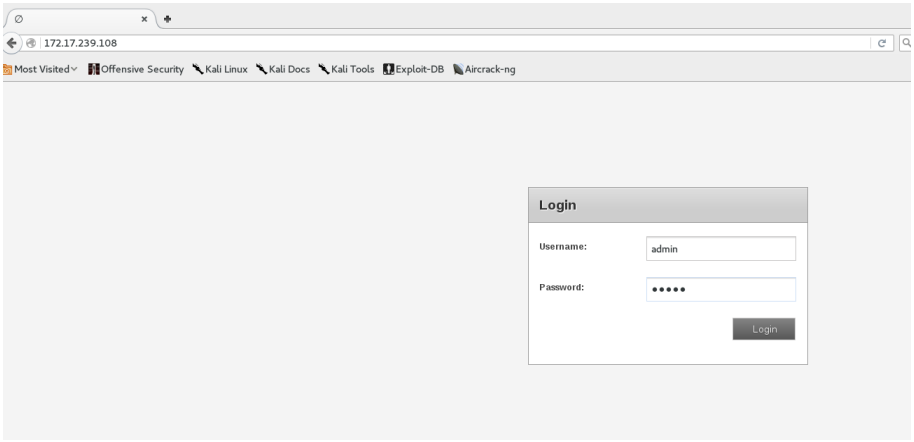

Figure 15: Shows the PlugBot admin login screen

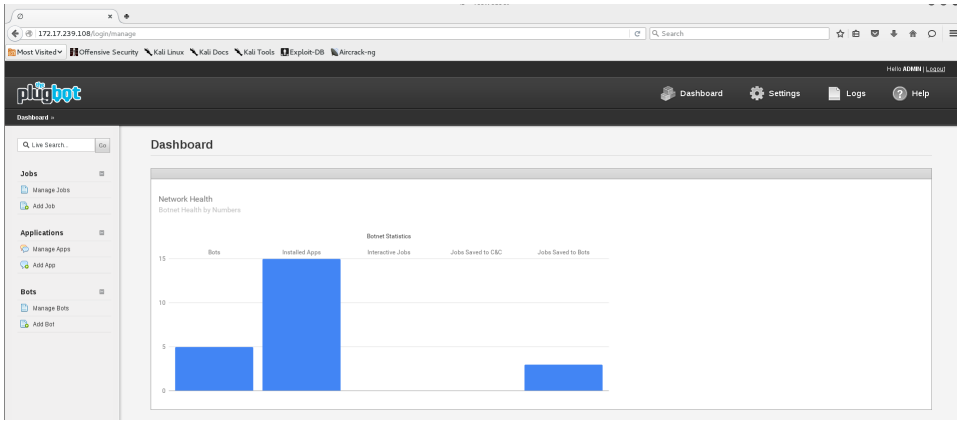

Figure 16: Shows the PlugBot main admin page

Next, we set up the FTP server on the C2C server to put our shell scripts for the bots to download and execute. We already installed vsftpd in the beginning of the step, so we changed our directory to /etc and renamed the the vsftpd.conf tp vsftpd.conf.bak with the command `mv vsftpd.conf vsftpd.conf.bak`. Then using nano we created a new file with the name vsftpd.conf and copied and pasted the text from Appendix D into it and saved the file and closed. Figure **??** shows the contents of our config file.

Figure 17: Shows the contents of our vsftpd.conf file

Next, we created the directory in our home folder ftp, and then created the directory opendir inside of it. We changed the permissions of the opendir folder using the command `chmod 777 opendir/`. We restarted the vsftpd service using the command `service vsftpd restart`. Next, we used scp to move the zip files with the shell scripts we created into the /home/ftp/opendir directory, Figure 18 shows the contents of our directory after moving the files to it.

Figure 18: Shows the contents of our opendir folder after copying the zip files to it

To verify that the ftp service was working correctly, from our Kali VM on LAN Segment 2, we entered ftp://172.17.239.108/opendir into a web browser shown in Figure 19

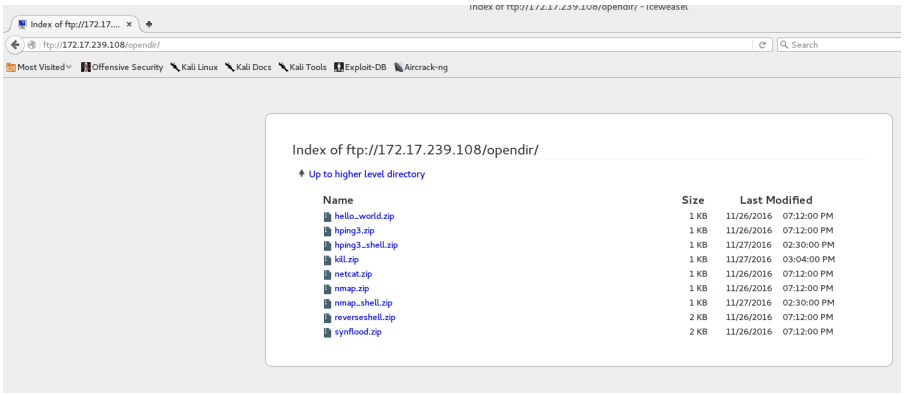

Figure 19: Shows that we can now access the ftp server from another machine

3.1.1.3 Step 2: Configuring The PlugBots(Raspberry Pi or VMs)

After, experimenting with the Raspberry Piś and successfully getting the project to run on them, we decided to strictly use virtual machines for easier portability. In this step we will be setting up a Ubuntu Server 14.04 virtual machine, but the same steps apply to the Raspberry Piś.

To start off, we created a fresh Ubuntu Server 14.04 VM, with user: checkout passwd: checkout.

First, we updated our apt packages, with apt-get update. Then we installed all the packages we installed on the C2C server and also added apache2, flip, and cron. We did not install vsftpd. Figure 20 shows the list of packages we installed on the Ubuntu Server VM. We entered the password toor for our mysql root user.

```
root@testbot:/home/checkout# apt-get install apache2 mysql-server mysql-client libapache2-mod-php5 p
hp5-curl php5-mysql php5-mcrypt flip cron zip curl wget wput php5_
```

Figure 20: Shows the packages installed on the bots

Next, removed the html directory in /var/www and added the .htaccess file like we did with the C2C server. We used SCP to copy the file from the C2C server shown in Figure 21.

Figure 21: Shows we copied the .htaccess file from the C2C server

Next, we edited the apache2.conf and php.ini files in the same way we did for the C2C server. We also had to change our root directory for our webserver from /var/www/html to /var/www. We did this by editing. We did this by editing the file 000-default.conf located in /etc/apache2/sites-available. Figure 22 shows that we edited the DocumentRoot directory to /var/www.

Figure 22: Shows the edited DocumentRoot path for 000-default.conf file

Next we enabled the rewrite mod for apache and restarted apache2 service shown in Figure 23. We changed our directory back to /var/www and used the command `git clone https://github.com/redteamsecurity/PlugBot-Plug`, shown in Figure 24, to clone the github repository for the Bot portion of PlugBot.

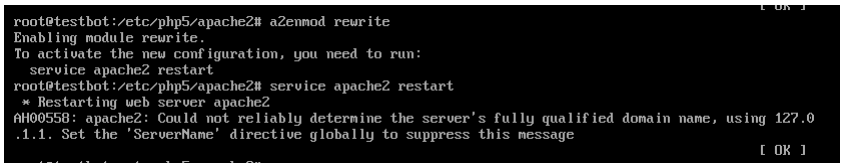

Figure 23: Shows that we enabled the rewrite mod for apache

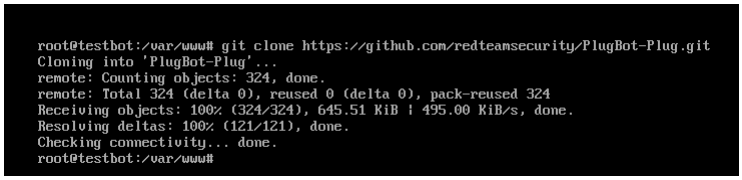

Figure 24: Shows that we cloned the PlugBot-Plug repository from github

Next we copied the contents of the PlugBot-Plug folder to www, by using the command `cp -R PlugBot-Plug/* .`, shown in Figure 25.

Figure 25: Shows that we copied the contents of PlugBot-Plug to /var/www

Then we deleted the PlugBot-Plug folder and using the command `nano application/config/database.php` opened the database.php file. We edited the database password line to toor, shown in Figure 26.

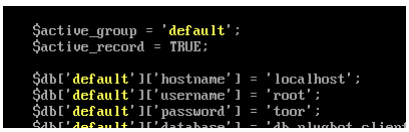

Figure 26: Shows password line edited in database.php

Next we unzipped the db_plugbot_client.sql.zip file and logged into mysql with `mysql -u root -p` and entered toor. Then we created the database with the command `create database db_plugbot_client;`, then used the command `use`

`db_plugbot_client` to use the database. Then we ran the SQL script with the command `source db_plugbot_client.sql`. All of this is shown in Figure 27.

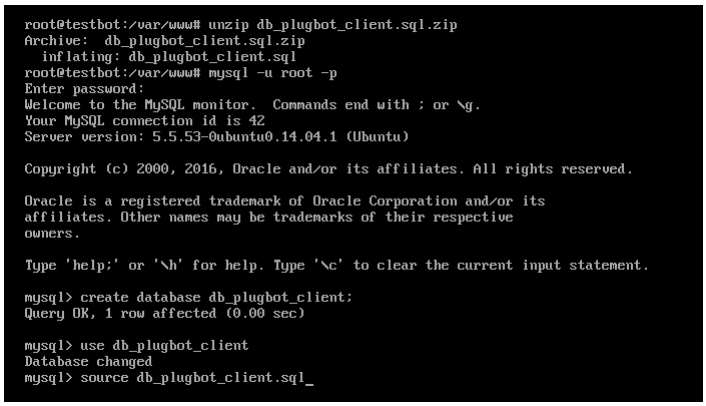

Figure 27: Shows the full sequence of events for mysql and to populate the database with the script

Next, we tried to go to the bots web gui at address: 192.168.40.116/index.php/home/setup. We received a mcrypt error, so we had to edit some things for mcrypt to continue. First, we cd into /etc/php5/mods-available. Then ran the command `updatedb`. Then we used the command `locate mcrypt.so` to find the path of the mcrypt.so file, shown in Figure 28

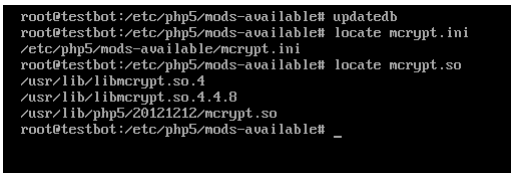

Figure 28: Shows the commands to update db and locate the mcrypt.so file

Next we used nano to open the mcrypt.ini file, and changed the `exension=` to the path of the mcrypt.so file shown in Figure 29, which was /usr/lib/php5/20121212/mcrypt.so. Next, we ran the command `php5enmod mcrypt`. To verify that the ini files we linked we used the commands `ls -al /etc/php5/cli/conf.d/20-mcrypt.ini` and `ls -al /etc/php5/apache2/conf.d/20-mcrypt.ini`, shown in Figure 30 To complete the mcrypt setup we restarted apache.

Figure 29: Shows path for extension changed to our mscrypt.so location in mcrypt.ini

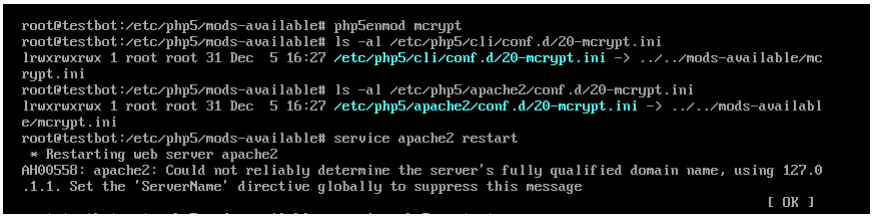

Figure 30: Verification that we are linked to our mcrypt.so location

Now we can access the bot config login page shown in Figure 31 and Figure 32 shows that we can login in using the credentials admin admin.

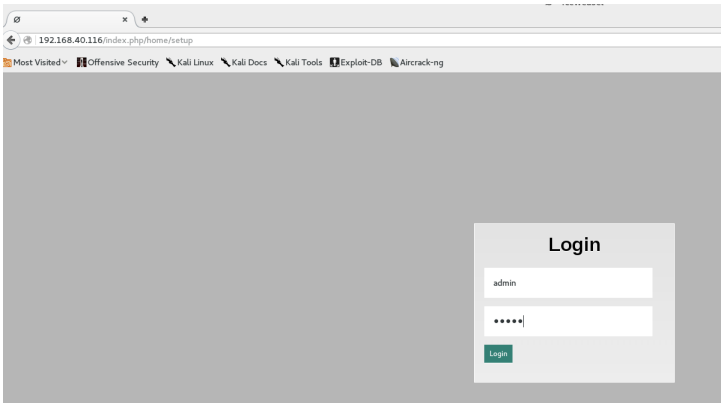

Figure 31: Shows we can now access the Bot Web application

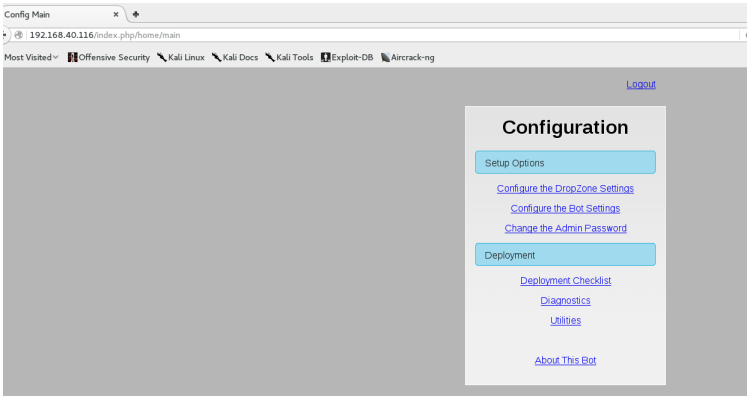

Figure 32: Shows we can access the configuration page for the bot

Next, we had to edit some permissions in order for the www-data user to download and execute jobs from the C & C server. First we ran the command `chgrp www-data /var/www/apps` followed by `chmod -R g+rwxs /var/www/apps`, shown in Figure 33. This gives the group www-data recursive read write and execute priviledges on the apps folder where it downloads apps and writes output to. Finally, we executed the command `crontab -u www-data /var/www/cron/cron.txt`.

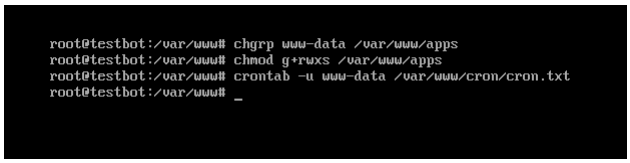

Figure 33: Shows commands to give the www-data group permissions on apps directory

Q3: Explain why we need to change the group permissions on the apps folder.
A3: The way that the PlugBot system works is that the Command and Control server adds apps to a bot. The bot downloads them, unzips the file. Www-data user is the user that runs the apache2 web server by default. The user needs to have permissions on the apps folder to download the scripts and unzip them into the apps folder. This is why we need to give recursive read, write and execute permissions to this folder for the www-data user.

Q4: What is the crontab.txt file used for and what does the command crontab -u www-data ... do?
A4: The crontab.txt file is a list of shell commands that are run periodically by the daemon cron. Cron is used to schedule periodic task to run in the background. The crontab -u command tells the cron daemon the

user whose crontab is to be used or modified and the path to the crontab file which contains the commands to be run periodically in the background. The crontab.txt file is how the bots conduct periodic check-ins with the command and control server.

3.1.1.4 Step 3: Bot Check In With C & C Server

To begin this step we went to our Kali Linux VM on LAN Segment 2, and entered the IP address of the C2C server and the went to 192.168.40.116/index.php/home/setup (our bot) and logged in to both pages. Next, from the main Dashboard page of the C2C server's web application we clicked the add bot button on the left column.
We copied the unique keys from the top and pasted them in to the Bot key field and Bot Private Key field, gave the bot the name TestBot and then clicked add shown in Figure **??**. Then we clicked on the Manage Bots button to view the keys for the bot.

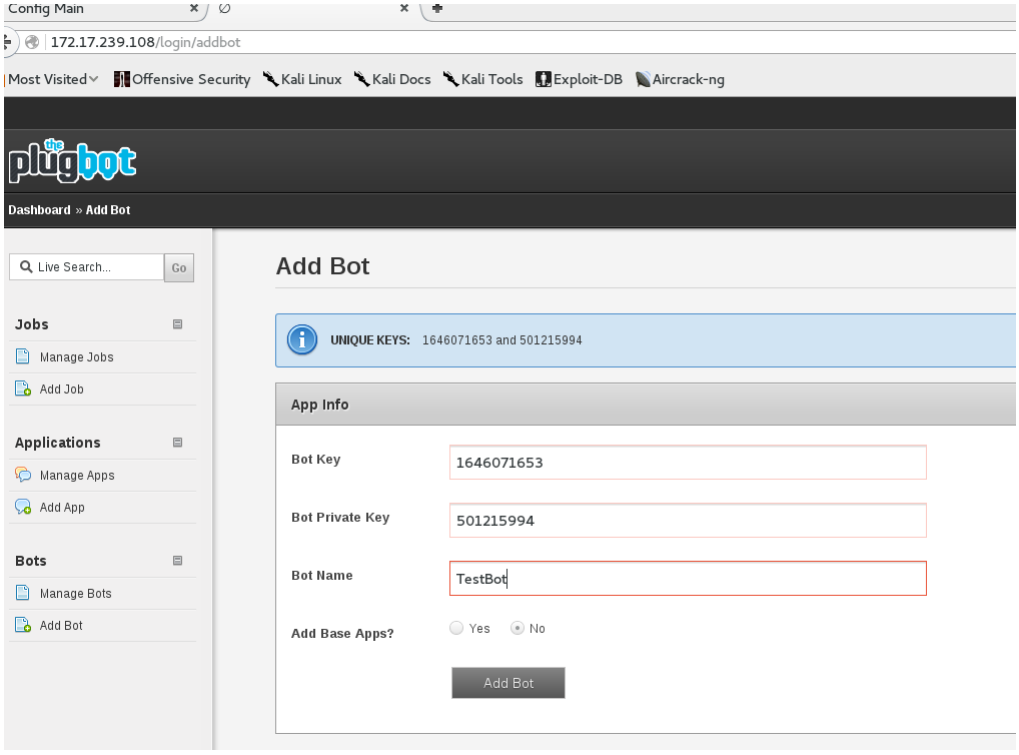

Figure 34: Shows add bot page and our unique keys

Next, we copied the Bot Key from the Manage Bots page, and then went to the confiuration page on the bots web application.

First, we clicked the Configure the DropZone Settings. Then we changed the dropzone url to our C2C server's(http://172.17.239.108) and saved, shown in Figure 35. We clicked the home button to go back.

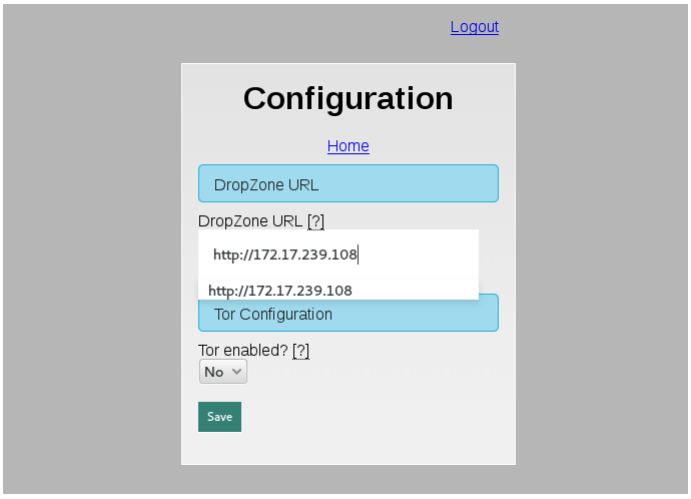

Figure 35: Added the url to our Command and Control Server

Then we clicked the Configure the Bot Settings button. We changed the Bot Name field to the bot's name we created(TestBot), then we pasted the Bot key we copied earlier from the Manage Bots page, and then we copied the private key for our bot from the C2C Manage bots page and pasted that followed by Save, shown in Figure 36.

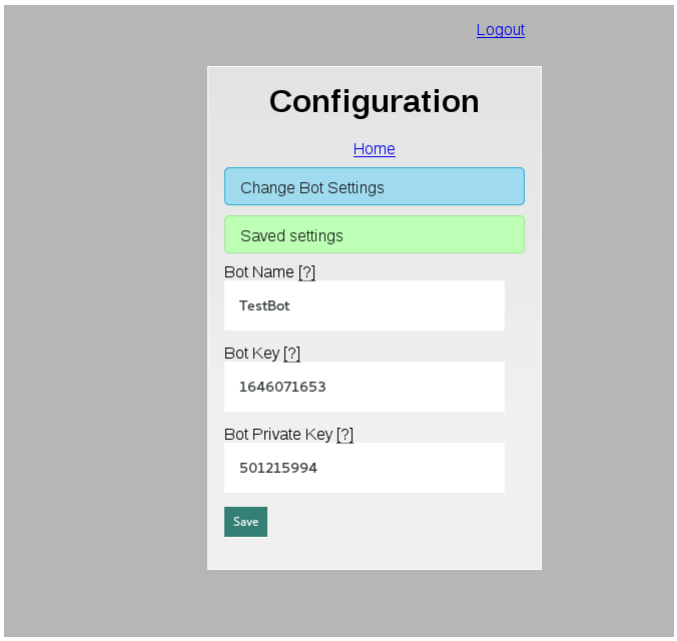

Figure 36: Shows the fields that we filled in, using the keys generated from the C & C server

Then we clicked the home button to go back to the main page on the bot. Then we clicked the Diagnostics link. We clicked the Start the Scheduler Link and then Test Bot Check-In link. We received a successfull check in shown in Figure 37. To verify that the C & C server saw the bot, we went back to the Manage Bots page and refreshed it. Figure 38 shows the IP address the TestBot checked in with.

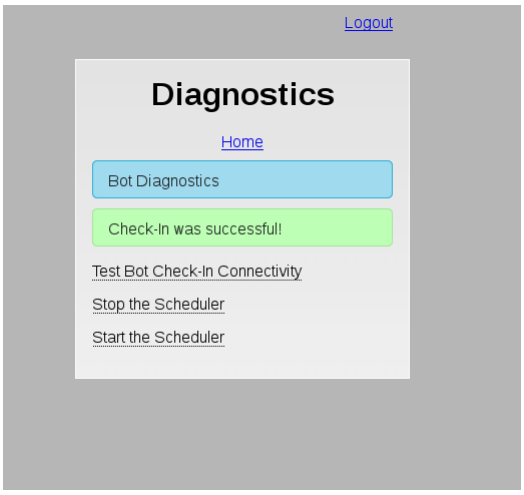

Figure 37: Check-in with C & C was successful

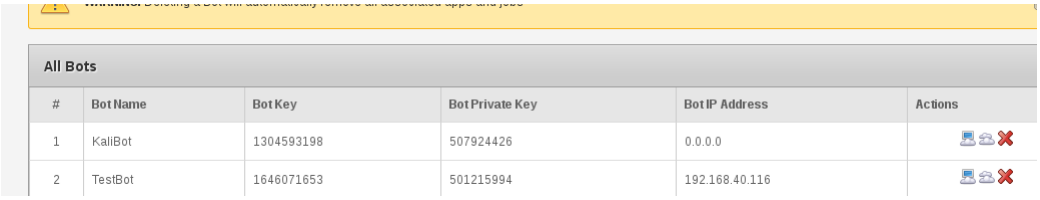

Figure 38: Verification on the C2C servers Manage Bots page, that the bot checked in with it's IP address

To finish things up we completed Steps 2 and 3 on three more Ubuntu Server 14.04 VMs.

3.1.2 Key Learning/Takeaways:

The key take aways from this exercise is that it required a lot of research to get the application running on the Bots. There was a lot of little things that were not mentioned in the developers Read Me file that required a lot of research to find out what was causing them.

Another key take away was understanding how bot nets are structured. We originally had everything setup on the same network when we were using the Pi's as bots. I realized how unreal this was, so we brought a DDWRT physical access point in to the mix. It was a really great learning experience for understanding the features on the DDWRT and how to set it up on a different subnet, while still maintaining access to the internet for downloading packages.

On a second point about the DDWRT router setup, when we switched to the DDWRT VM images the real fun began. This was probably the best learning experience I've ever had with routing and subnetting. At first it was really hard to comprehend what was going on with the bridge interface on the AP. Once I figured that out it got really fun. I Eventually we ended up with 4 routers and 4 subnets all talking to each other. There is a lot more I would like to experiment with in the future with this setup, particularly how to run two WAN's on a single AP. This was great experience with routing and I believe something similar to this should be used in the future of 460 for Lab 2. It gave me a deeper understanding of gateways, bridges and static routes to subnets. I also learned a great deal about VMWare and setting it up for this setup. Originally we were trying to use our own host only vmnet's, but couldn't get anything to work. We figure out through some research and dumb luck that we needed to use the LAN segments feature in VMWare. All in all the routing setup in step 0 we the biggest learning and takeway from this lab, besides the basics of how bot nets are structured.

3.2 Part 1 Exercise 2 - Issuing Commands From C & C and Dos and DDoS Attacks (Joey Arbogast)

3.2.1 Analysis & Evidence:

3.2.1.1 Step 1: Installing Apps on the Bots

See Figure 1, 2 and Table ?? and ?? in Section 2

After completing Exercise 1 Step 2 and 3 on 3 other bots we prepared to install Apps on the bots. To begin we installed Hping3 and Nmap on all 4 bots, using `apt-get install hping3 nmap`. We already copied our shell script zip files over to the C & C server in Exercise 1, so we continued with the exercise.
From the C & C Web application, we selected Add App from the left column. From the drop down we selected the first bot to install the app on (TestBot). Then we named the first app hping3_shell and gave it a description Next for the the app download link we entered ftp:// with our IP for our C & C server followed by the public ftp directory we created on the C2C. For download file name we entered hping3_shell.zip, which is the name of the file on the ftp directory. Then for the App Executable we entered hping3_shell.sh which is inside the hping3_shell.zip file. Figure 39 shows the information we filled. To finish we clicked add application. Then we did the same for the kill.zip and nmap_shell.zip files on our ftp server and then installed all three applications on the other 3 bots.

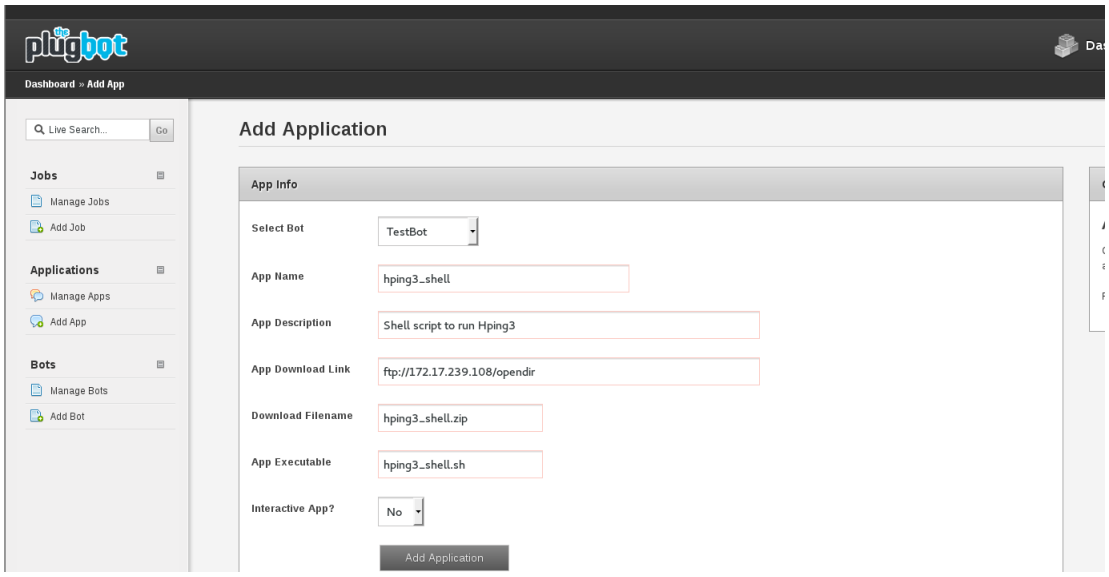

Figure 39: Adding the hping3_shell script to the bots

We clicked the Manage Apps button to view the status of our application install. Figure 40 shows that all the apps have successfully installed on the bots. Now

we had to edit some privileges so that the www-data user could execute the apps as root with sudo and not have to enter a password since we have no interaction with the bots after we issue the job commands.

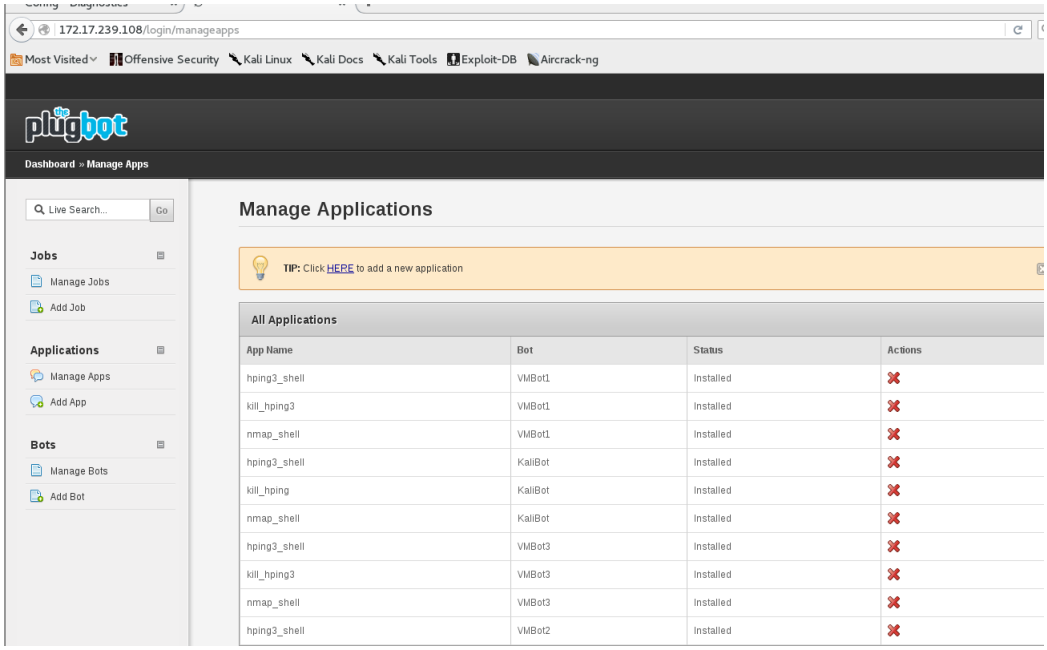

Figure 40: Shows all the apps we have installed and on which bots

First on all 4 bots we opened the sudoers file with nano, using `nano /etc/sudoers`. We added the line after root, `www-data ALL=(ALL) NOPASSWD: ALL`. Instead of setting the NOPASSWD option for ALL, we could have specified the specific path to the executable for specific files such as the hping3 executable, but it was easier just to do everything. After saving and exiting the sudoers file, run the command `/etc/init.d/sudo restart` to restart the service for the changes to take effect, shown in Figure 41

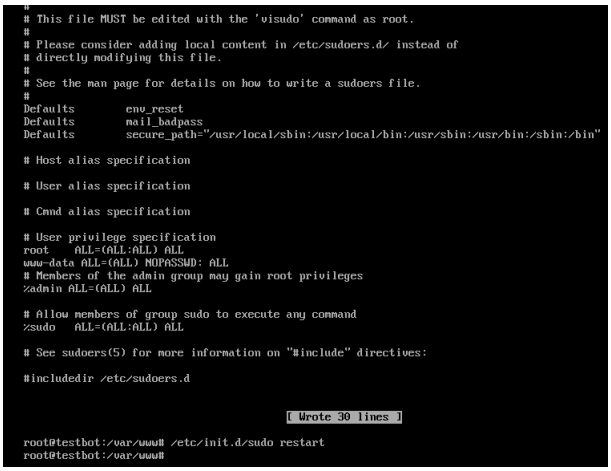

Figure 41: Shows modifications to the sudoers file and restarting the sudo service

3.2.1.2 Step 2: Dos(single bot) and DDoS(multiple bots) Attacks with Hping3

Q1: What is a DOS attack?
A1: A denial of service attack is when an attacker floods a network with data. When someone elsetries to access a web server on the victim's network a connection usually cannot be made and times out. The attack typically comes from a single computer and single IP address, though it can be spoofed to look like many different IP address sources. [4]

To begin this step we booted up our Metasploitable2 VM(Network Adapter set to LAN Segment 4) and assigned the static IP 169.224.1.138. In preparation for the attack on our Kali machine we pulled up each routers web page by going to there Ip addresses and then clicked the status tab, and then the bandwidth tab shown in Figure 42.

Figure 42: Shows bandwidth tab on DDWRT router so we can monitor the amount of traffic

Next, we went to the C & C web application, logged in and selected the Add Jobs button. We gave the job the name DoS, Selected the TestBot from the drop down. Then we selected hping3_shell for the application, shown in Figure 43.

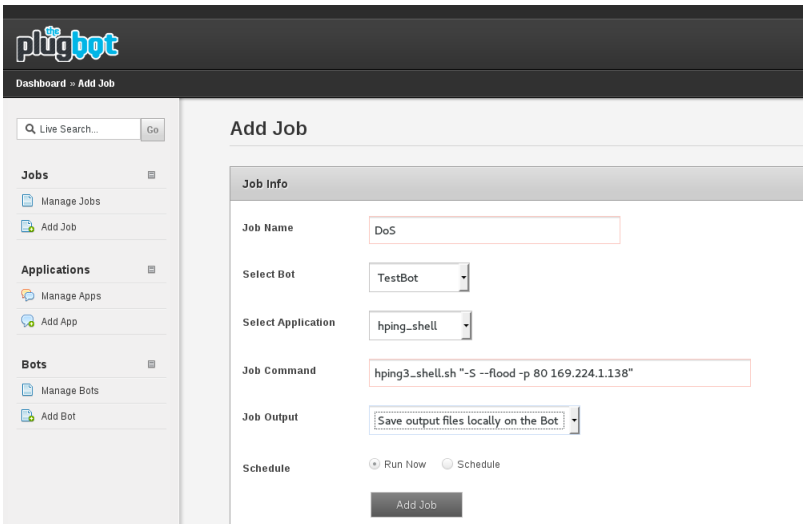

Figure 43: Shows job commands for the hping3_shell.sh script

For the Job Command we entered, `hping3_shell.sh "-S --flood -p 80 169.224.1.138"`, and selected Save Output files Locally on the Bot from the drop down below. Then we clicked add job.
We selected Manage Jobs tab to view the apps status. Next, we went to the TestBot and used the command `ps aux | grep hping3` to monitor when the hping3 utility began executing, shown in Figure 44.

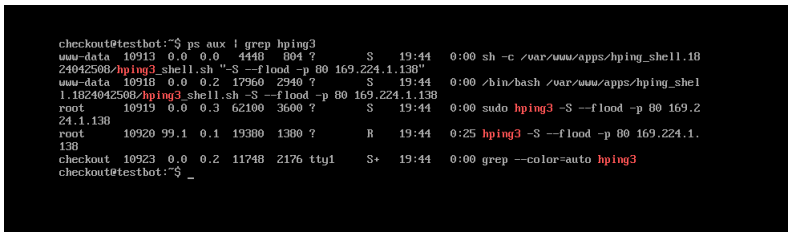

Figure 44: Shows that the hping3 proccess is running on the bot

Next, we attempted to connect to the Metasploitable2 website from a browser on Kali Linux VM on the subnet 172.17.239.0/24, shown in Figure 45.

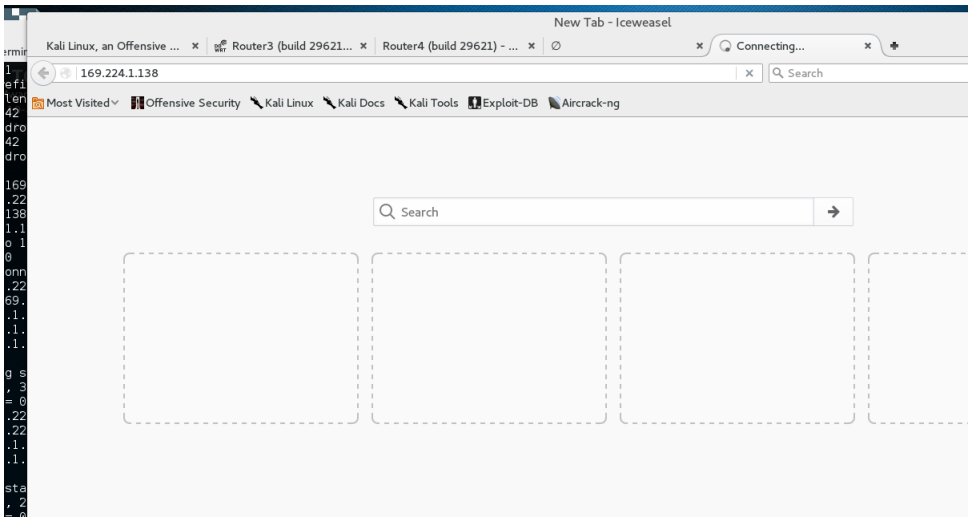

Figure 45: Shows that we are attempting to connect to the metasploitable 2 webserver

Next, we used the curl utility to try to connect to the Metasploitable Web server, using the command `curl -v 169.224.1.138` <replace IP with your Metasploitableś >. As you can see in Figure 46, the connection attempt says trying. It eventually timed out and failed.

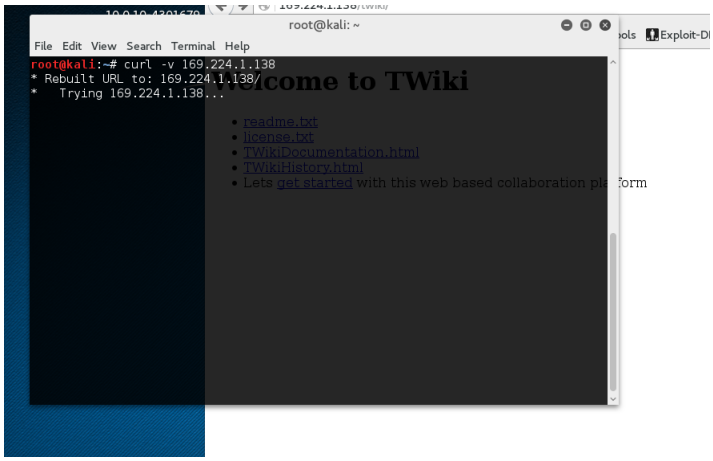

Figure 46: Shows we are attempting to connect with curl which fails eventually

We went back to the browser and tried to go directly to a link for the twiki page. `<IP of Metasploitable >/twiki/readme.txt`. As you can see in Figure 47, after some time of waiting for the page to load it eventually was successful. This proves our point that one bot is not necessarily enough to take down the site, but will extremely slow the connection and make the connection unreliable.

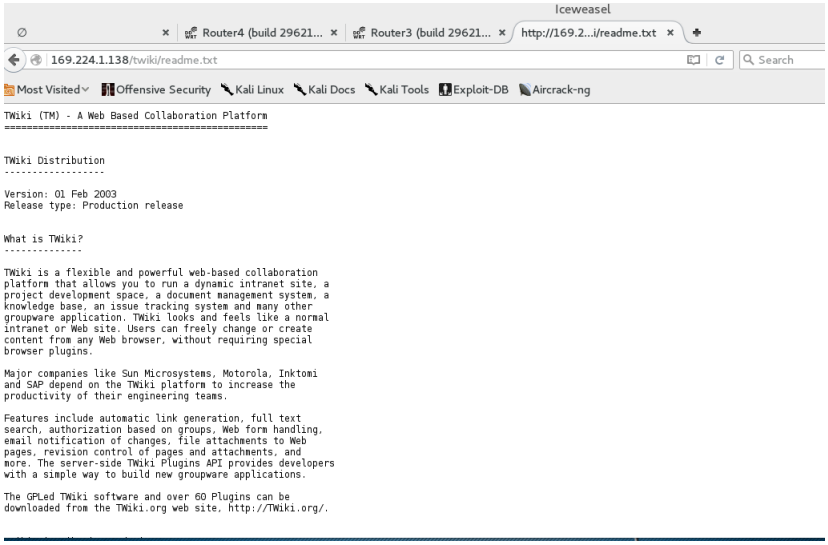

Figure 47: Shows eventually we were able to get a page to load

Next, we pulled up some of the real-time bandwidth charts that the DDWRT router provides. The only ones of importance are router 3, router 4 and router 5.

Figure 48 shows the bandwidth chart for Router 3. Figure 49, shows the bandwidth chart for router 4, which was sometimes unreliable, periodically flashing on the charts that it could not receive data about the interface. Figure 50 shows the bandwidth chart for Router 5, which was even more constant with the Cannot get data about the interface message.

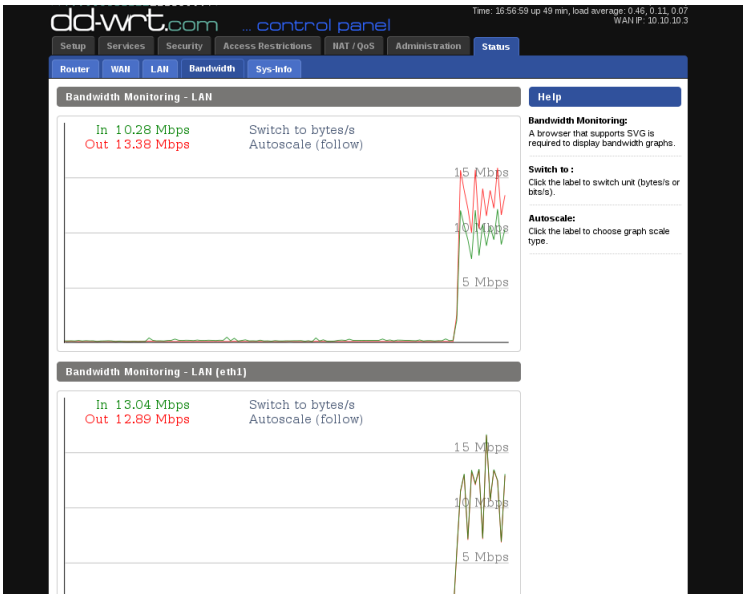

Figure 48: Shows bandwidth chart for data flowing over the router 3 LAN segment

Figure 49: Shows bandwidth chart for data flowing over the router 4 LAN segment

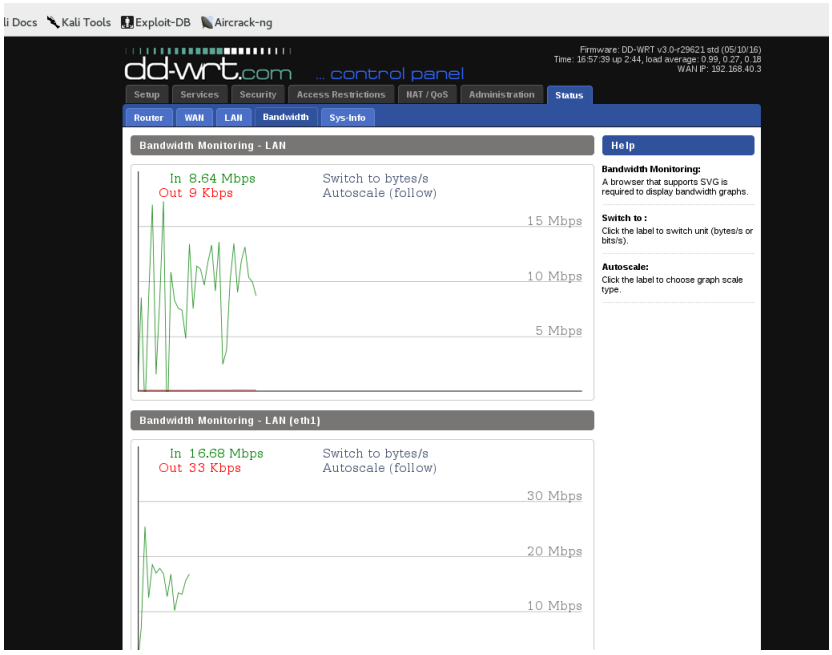

Figure 50: Shows bandwidth chart for data flowing over the router 5 LAN segment

Next we conducted another test. We pinged the metasploitable 2 VM at 169.224.1.138. Figure 51 shows that the packet loss is extremely high. We also pinged the gateways on router 3, router 4, and router 5 to see how they responded. Router 4 which is the gateway for the Metasploitable 2 VM showed about 0% packet loss, where as the Metasploitable VM shows around 57 % . Figure 52 shows that a second attempt at a ping shows 0 packet loss. We conducted another test to see if any other services were affected. Figure 52 also shows that when we try to SSH into the Metasploitable VM the connection times out.

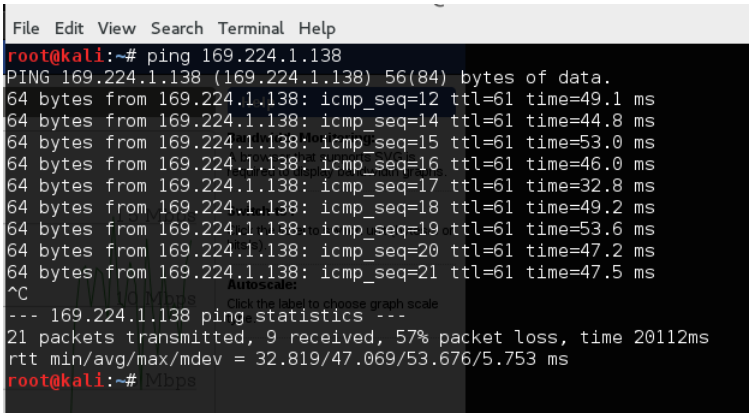

Figure 51: Shows the unreliability of the network 57 % packet loss

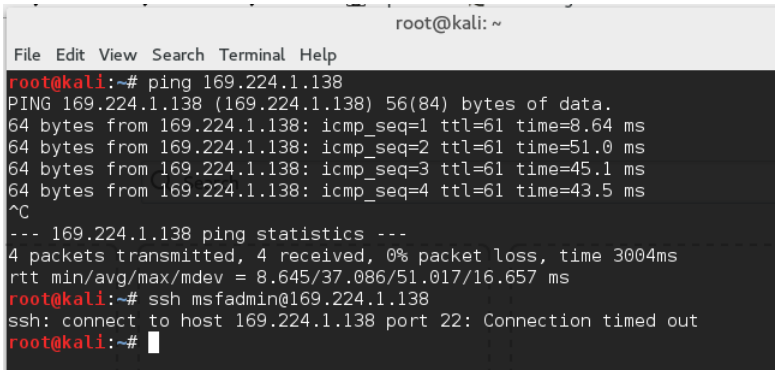

Figure 52: A second ICMP test shows there is 0 packet loss, but trying to ssh to the metasploitable vm fails.

To finish things, we created a shell script that can kill the hping3 process on the bots, but it takes a while for it to download to the bots and execute, so we simpily went to the bot and used the command `killall -v -9 hping3` to kill the hping3 process quickly.

Q2: Take a look at the real time bandwidth graphs on each of the routers. Provide screen shots as evidence.

A2: Figure 42 shows the amount of bandwidth on Router 4 before the DoS attack takes place. Figure 48 shows the amount of bandwidth travelling across Router 3. Figure 49 shows the traffic amount of bandwidth coming into the WAN port of Router 4, and Figure 50 shows the bandwidth being consumed on router 5.

Q3: Notice that router 4 may not be accessible, why is this?
A3: Router 4 is periodically unresponsive, because the WAN port is being flooded with data tying up all the bandwidth the router can handle. We can't get through to 169.225.1.1, because the WAN port is being flooded with so much traffic.

DDoS Attack

Q4: What is a DDOS attack and what is the difference between DOS attack?
A4: A DDoS attack is a distributed denial of service attack. DDoS typically involves multiple computers from different IP locations(typically infected by some sort of malware), which differs from a DoS attack with a single computer and IP source address. DDoS attack are typically harder to stop, because the traffic looks like legitimate web traffic since it is coming from multiple IP source addresses.[4]

From the Kali VM on LAN Segment 2, we went to the web application for the C & C server at 172.17.239.108. We went through the same process as the DoS attack, but used the job command `hping3.sh "-S -d 1500 --rand-source --flood -p 80 169.224.1.138"`, shown in Figure 53. We copy and pasted this command in the other bots and Figure **??** shows the jobs have been processed.

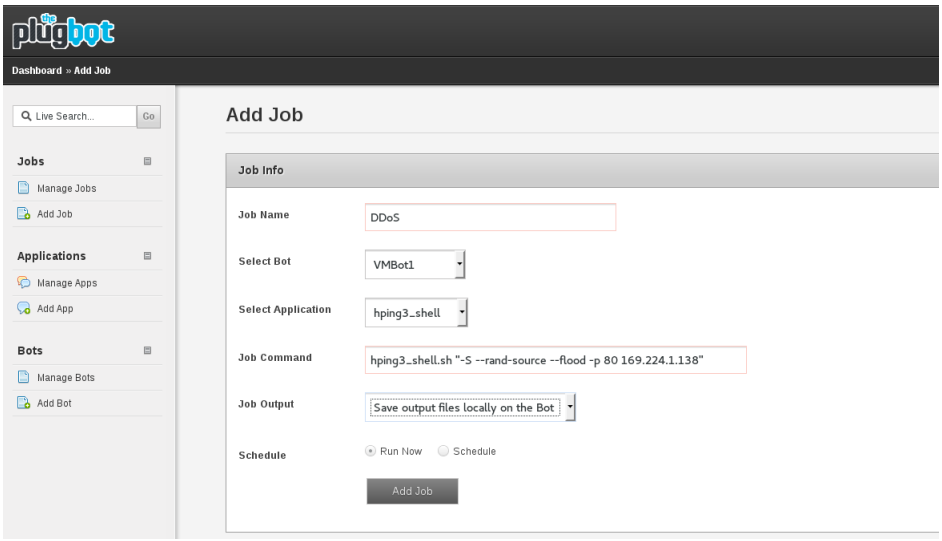

Figure 53: Shows the job command for the DDoS attack

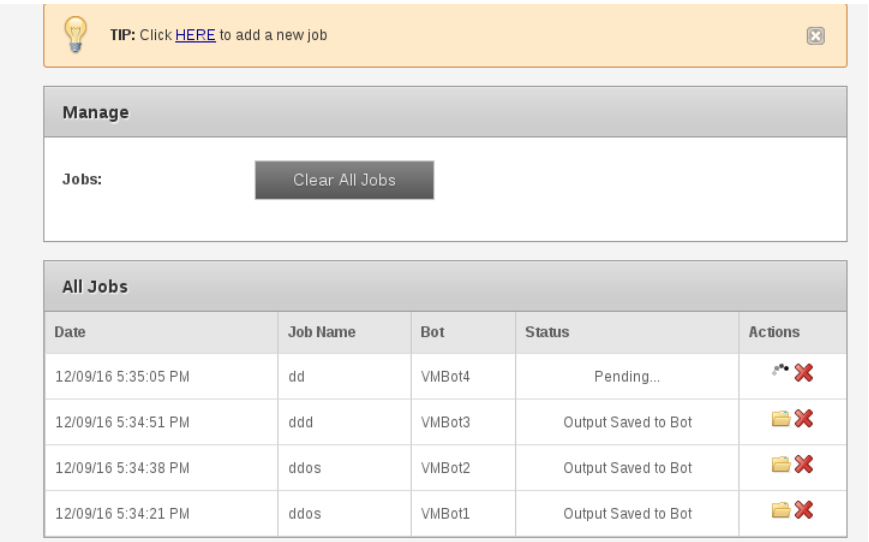

Figure 54: Shows that the hping3 job has been sent to all 4 bots.

After waiting for about 4 minutes, all the jobs began executing on the bots. We attempted to go to the Metasploitable 2 VM's IP address in a web browser on the Kali Linux VM on LAN segment 2. As you can see in Figure 55 the connection eventually timed out on us. Next, we attempted to connect to the server with the curl utility like we did with the DoS attack. Figure 56 shows that the connection timed out as well. We also conducted a ping test to the Metasploitable VM and Router 4. Figure 56 also shows that the packet loss is extremely high for the ping test between the Metasploitable 2 VM.

Q5: Was the ping of the Metasploitable 2 VM successful? Show evidence and explain why or why not this is.What else do you notice?

A5: Figure 56 Shows that we can ping the target VM, but there is significant packet loss. This is, because there is a bottle netck at the WAN port on router 4. Some of the ICMP request get through, but some are lost.

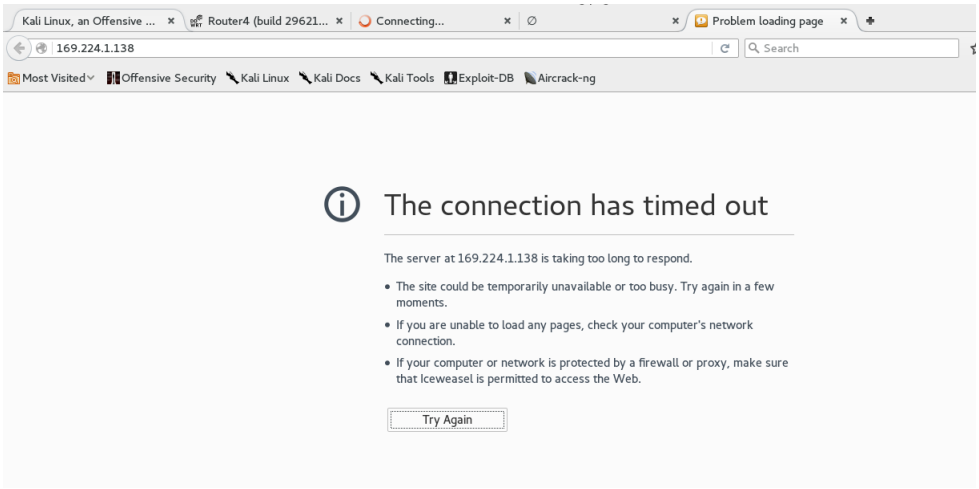

Figure 55: Shows the connection has timed out while trying to connect to Metasploitable

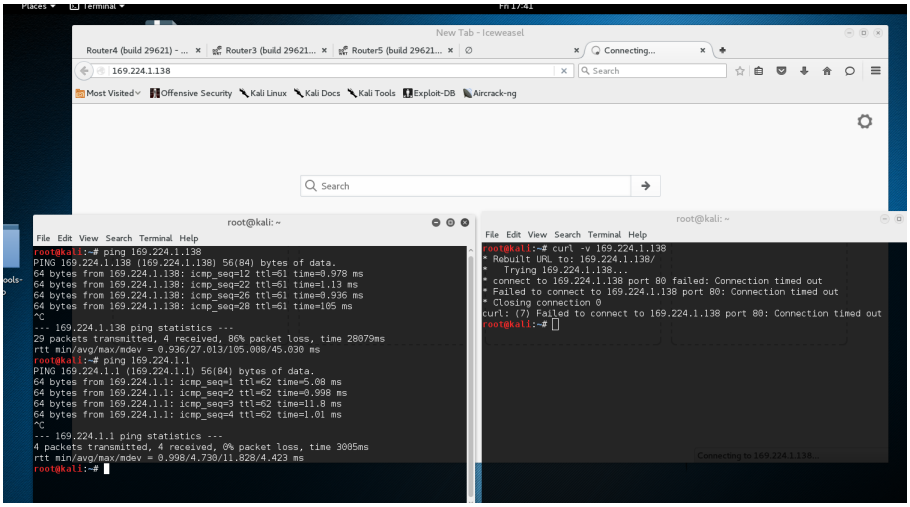

Figure 56: Shows curl connection times out and there is significant packet loss when pinging the Metasploitable 2 VM

Next We analyzed the bandwidth charts on the DDWRT Routers again, taking note that the bandwidth is much higher this time. Figure 57 shows the bandwidth of Router 3, Figure 58 shows the bandwidth consumption on router 4 and Figure 59 shows the bandwidth on router 5's subnet.

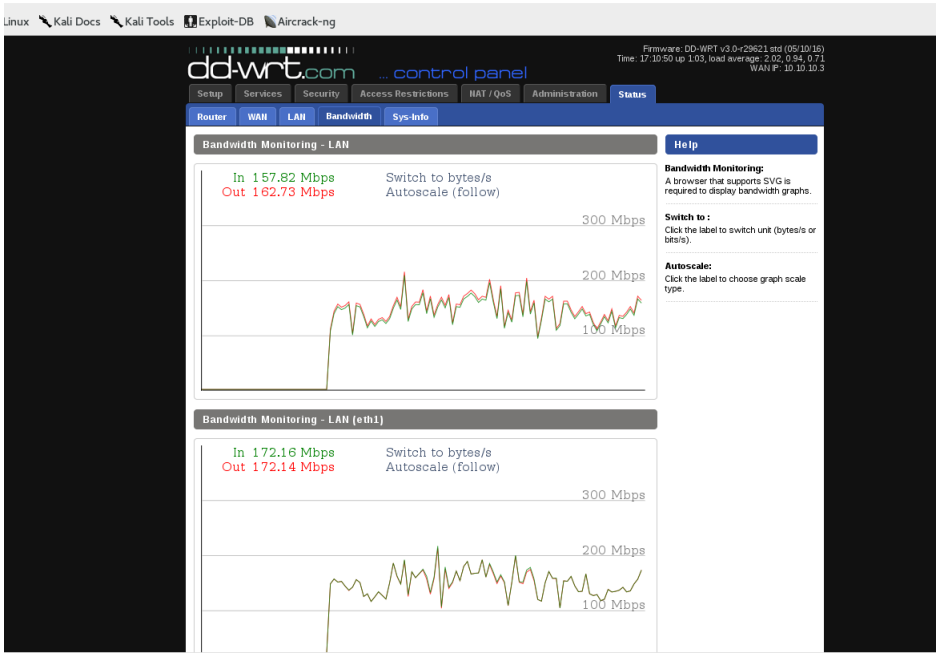

Figure 57: Shows the bandwidth consumption on router 3

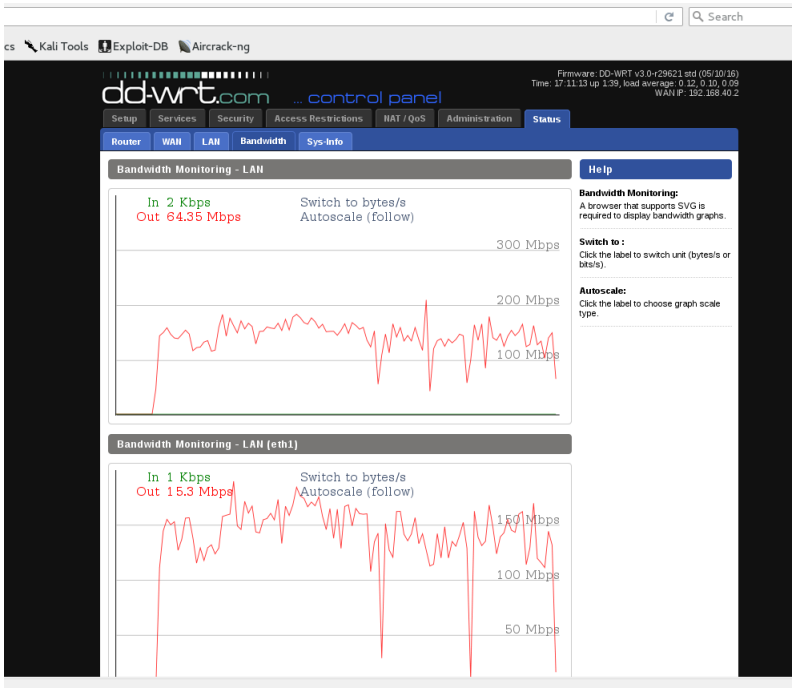

Figure 58: Shows the bandwidth consumption on router 4

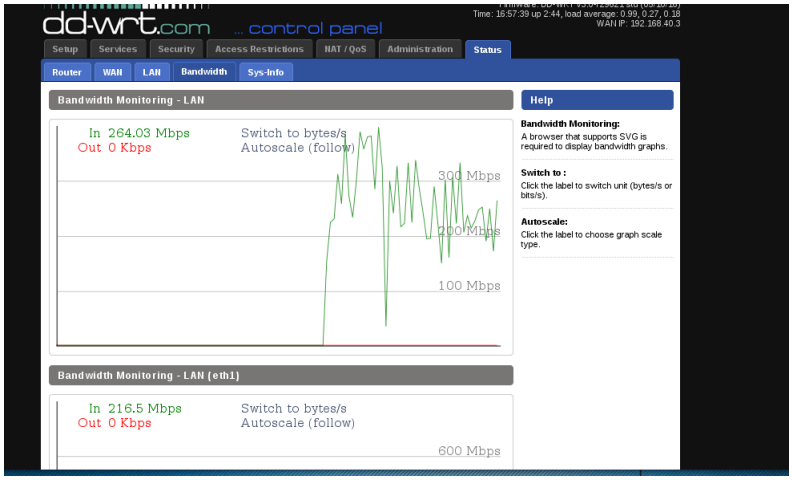

Figure 59: Shows the bandwidth consumption on router 5

Next, we took a look on the metasploitable 2 VM to view the attempted connections on the server. Using the command `netstat -nt`, you can see a bunch of SYN Flags received from lots of random IP address in Figure **??**, but no actual

connections made.

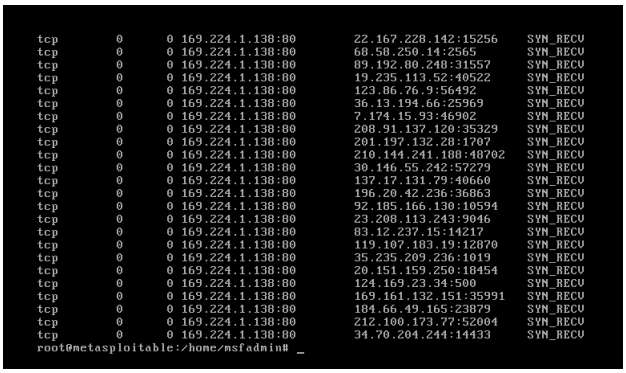

Figure 60: Shows SYN flags received by the Metasploitable 2 server

Next we took note of a few things taking place on Router 3. In the DDWRT administrator web page, we clicked on the Status tab. When the tab first opened up we scrolled to the bottom of the page where it says network. Figure 61 shows that there is an IP filter maximum port number of 4096. Notice we have almost reached the maximum limit of active IP connections of 4047. Also, going to the command line of Router 3 you can see in Figure 62 that there are a lot of dropping packet alerts.

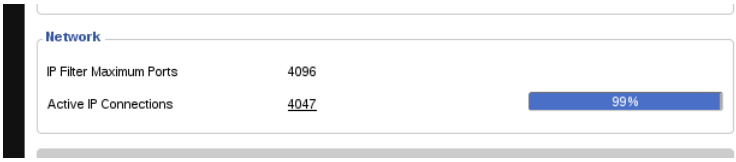

Figure 61: Shows the maximum number of connections the ddwrt router can handle and we have reached 99 percent of them

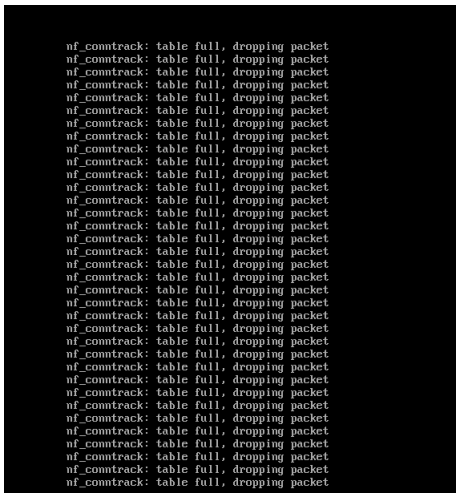

Figure 62: Shows packets are being dropped on Router 3

To complete this step we ran the killall command on all of the bots like we did in the DoS step in the interest of time.

3.2.1.3 Step 3: Preventing the DoS/DDoS Attack Using Iptables

On router 5 (The bots subnet) we add the IPTables rules, `iptables -A OUTPUT -p tcp --dport 80 -d 169.224.1.0/24 -j REJECT` and `iptables -A FORWARD -p tcp --dport 80 -d 169.224.1.0/24 -j REJECT` shown in Figure 63.

```
root@Router5:~# iptables -A OUTPUT -p tcp --dport 80 -d 169.224.1.138/24 -j REJECT
root@Router5:~# iptables -A FORWARD -p tcp --dport 80 -d 169.224.1.138/24 -j REJECT
```

Figure 63: Shows simple IPtables rules we added to router 5 which is the botnet subnet.

After we added the iptables rules we did a few simple test before redoing the DDoS attack in Step 2. First we pinged the 169.224.1.0/24 network from the bot's subnet 192.172.6.0/24. Figure 64 shows that it was successful.

Q6: Was the ping of the Metasploitable 2 VM successful? Show evidence and explain why or why not this is.

A6: Figure 64 shows the ping is successful and there is no packet loss either like in Step 2. This is because the iptable rules we added only specified anything destined for port 80 on the target network. ICMP is not destined for port 80 therefore it is not rejected by the rules.

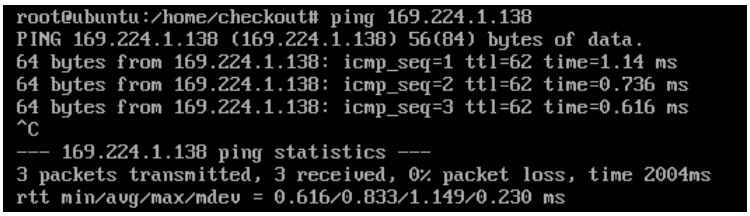

Figure 64: Shows that we can still ping Metasploitable 2 VM from the bots.

Q7: Was it successful? Why or Why not?
A7: Yes. We never specified any iptables rules about communicating with any other networks other than the target's network 169.224.1.0/24, therefore it is expected that we would be able to communicate with LAN Segment 2.

Next from our Kali Linux VM on subnet 2 172.17.239.0/24 we ping one of the bots at 192.172.6.149 shown in Figure 65

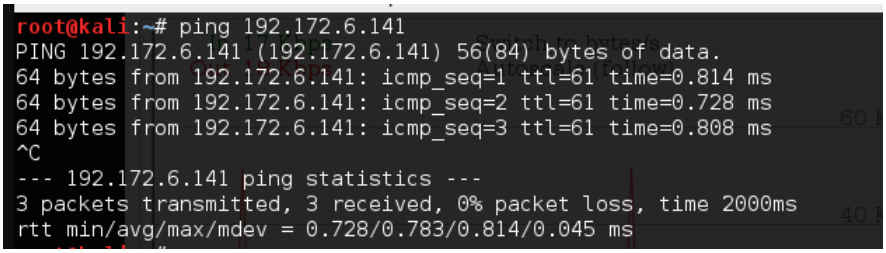

Figure 65: Shows that we can still ping the bots from other subnets.

Q8: Was it successful? Why or Why not?
A8: The ping from LAN segment 2 to LAN segment 5 was successful as shown in Figure 65. There was some lag as you can see that the time took about 2000ms, but there was no packet loss. This is because we never specified any INPUT rules for router 5 and all of the traffic that is being transmitted is trapped inside of the LAN network, therefore there is no bottle neck at the WAN port.

We launched the same DDoS attack from Step 2. Then we pulled up the bandwidth charts for router 3, router 4 and router 5. Figure 66 shows that there is not as much traffic on the LAN side of router 3. Figure 67 shows that there is not traffic coming in to the WAN port on router 4. Finally Figure 68 shows that there is lots of traffic on the LAN side of router 5, but nothing leaving the WAN port destined for 169.224.1.0/24 (router 4).

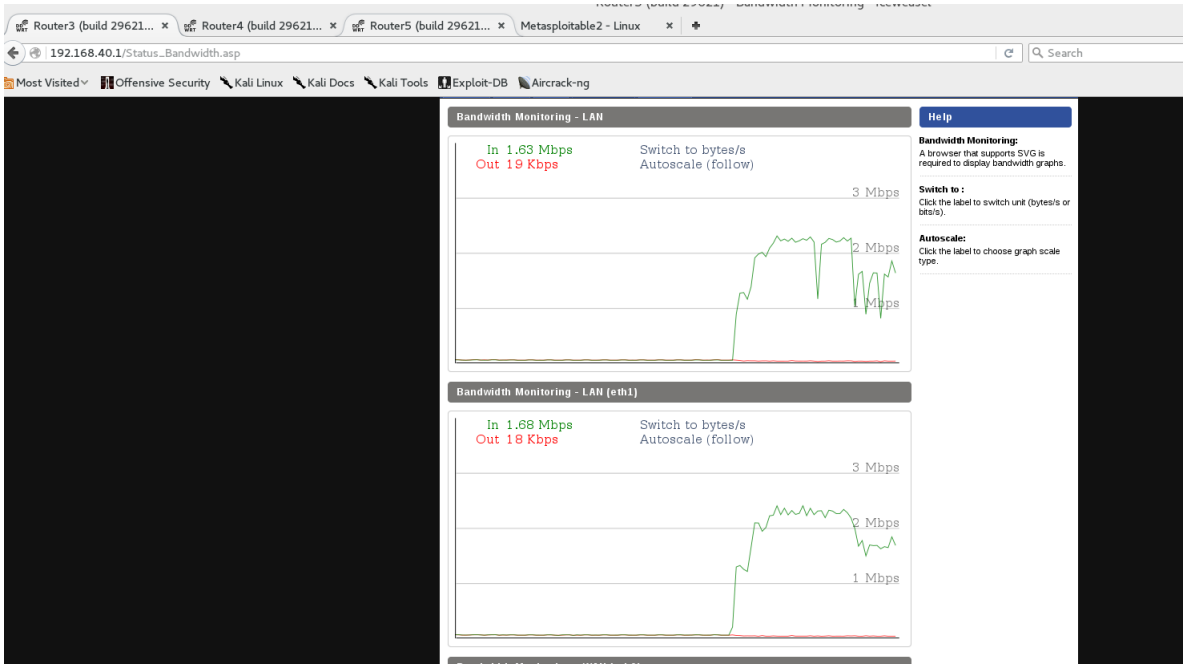

Figure 66: Shows bandwidth chart for the LAN side of router 3 with the iptables rules in place

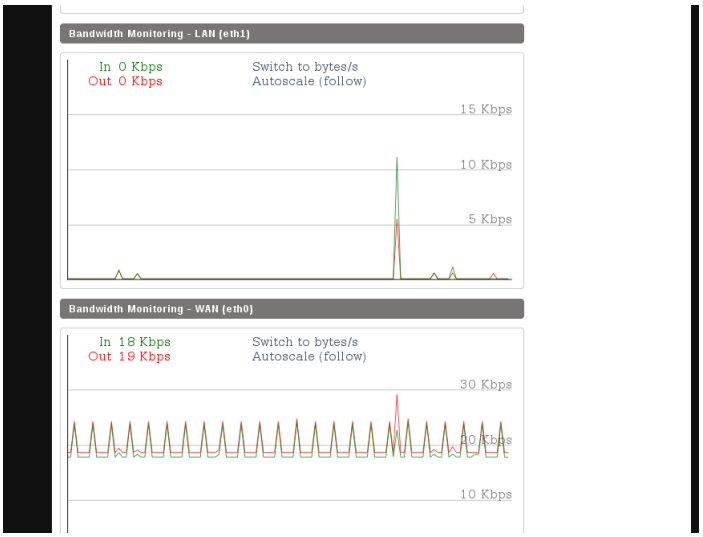

Figure 67: Shows bandwidth chart for the WAN port on router 4 with the iptables rules in place

Figure 68: Shows bandwidth chart for the LAN and WAN ports on router 5 with the iptables rules in place

Now we attempted to go the the Metasploitable 2 web page. Figure 69 shows that we can still connect to the server while the attack is being conducted. We killed all the hping3 processes on the bots to finish the exercise.

Q9: Was the attack successful? Why or Why not?

A9: As Figure 69 shows we can still access the target website with no issues or lag. This is because no traffic is leaving router 5 and bottlenecking the WAN port or slowing down the LAN side of router 3.

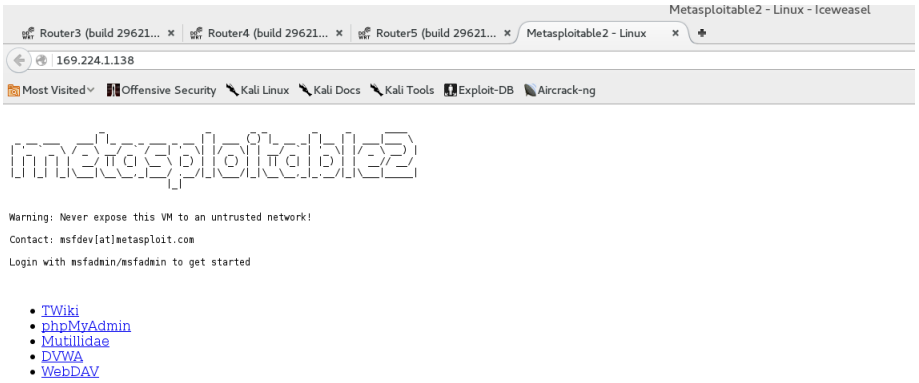

Figure 69: Shows that we can still connect to the Metasploitable server with the attack taking place

3.2.2 Key Learning/Takeaways:

The key takeaways and things that we learned from this exercise, is that routers are also a point of weakness in a DDoS attack. While our attack was very small scale compared to real life attacks and typically routers do load balancing to redirect traffic when the router gets overwhelmed is was a valuable lesson. Going in to this exercise, we originally were using only two routers. We had the C & C server and the Metasploitable 2 VM on the same subnet, and the bots on a different subnet. We noticed that when the attack was taking place, our Kali VM on the subnet with the C & C server could not get out to the internet, even though it should not have been directly affected by the DDoS attack. This prompted some investigation of the routers. When we discovered the real time bandwidth charts that the DDWRT provides for the interfaces, I realized not only was the target server getting overwhelmed, each router was getting knocked out and unresponsive along the path. We redesigned the exercise once again with 4 routers, but this still was not effective when we were experimenting with how to block the attack. We still had one critical router where all traffic had to pass through to get to the target web server. We then added a 5th router and put the bots on a 5th subnet. Then we were able to create some simple iptables rules to block any outgoing tcp traffic with destination port 80 and destined for the target network. This allowed us to prevent traffic coming from the bot's subnet from hitting the critical router point and knocking down the main path to the target web server.

We also found that there is an IP filter maximum port connection limit on the DDWRT routers. We could only have a maximum of 4096 active connection and since the –flood command for hping3 sends a massive amount of packets, we easily reached the limit, which I suspect has a great deal to do with the router getting overloaded. The field says that it can be set with a value of 256 - 65535, but we attempted to change this to the maximum value and it would not accept the change when saving. It would be interesting to see if physical routers running ddwrt have this limitation or if it's strictly because we are running it in a virtual environment. All in all this was a very valuable exercise and understanding that DDoS attacks can also overwhelm routers if powerful enough was an important takeaway that was unrecognized when developing this exercise.
We created a Youtube video for a real time look at the DDoS attack at https://youtu.be/h-1tkBWqyi4.

3.3 Part 2 Exercise 1 (Josie Salcedo)

3.3.1 Analysis & Evidence

3.3.1.1 Step 1:Capture Traffic of Bot Initialization In this part of the exercise we wanted to prevent the bots from receiving a command from the C&C server. In order to understand how to do this we had to better understand how that command is sent. So we began with a wireshark analysis.

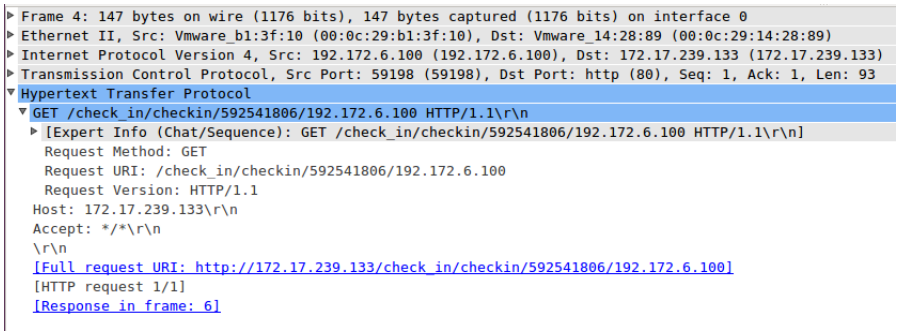

Figure 70

Figure 71

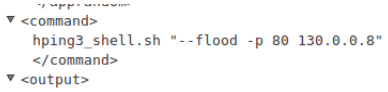

Figure 72

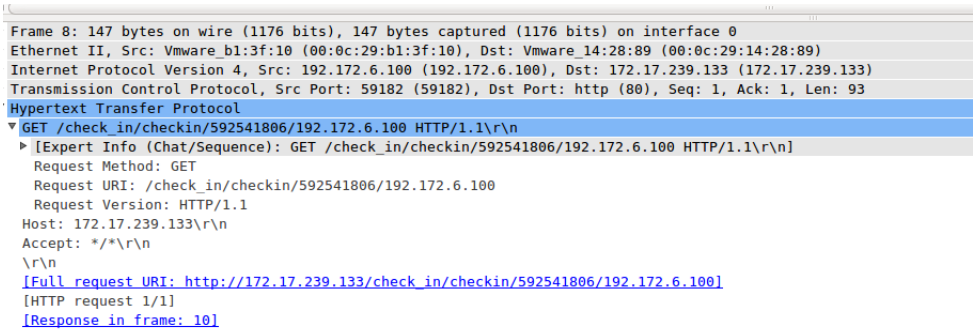

Figure 73

Figure 74

Figure 75

Q1: What protocols are found in both of the wireshark captures?
A1: HTTP, TCP and HTTP/XML which is eXtensible Markup Language, are both found in the wireshark capture of the hping3 and kill commands sent from the C&C server to the Bot being used for this exercise. This means that different commands have the same format regardless of what the command does.

Q2: What do you notice about the checkin packets of the capture? Hint: take note of the number after /check_in/checkin/XXX and the ip address.

A2: The number following /check_checkin/ in the "GET" part of the message corresponds to the Bot Private Key assigned during the addition of the Bot to the C&C server. This frame also show the host address in plain text of the payload.

Q3: What protocols are response packets? Export this packet from each capture.
A3: The Response packets to the job/get packet are in frame 30 of the kill capture and frame 36 of the hping3 capture.

Q4: What is notable about the expanded XML section? Hint: What command did you send the bot.
A4: By expanding the eXtensible Markup Language to show the 'job' header and details we find the commands that were send in plain text under the header 'command'. This is significant because, if we can filter by the payload verses the headers, then we can prevent the activation of a Bot army.

```
The following NEW packages will be installed:
  libdumbnet-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 55.9 kB of archives.
After this operation, 229 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libdumbnet-dev amd64 1.12-7 [55.9 kB]
Fetched 55.9 kB in 0s (82.9 kB/s)
Selecting previously unselected package libdumbnet-dev.
(Reading database ... 129413 files and directories currently installed.)
Preparing to unpack .../libdumbnet-dev_1.12-7_amd64.deb ...
Unpacking libdumbnet-dev (1.12-7) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up libdumbnet-dev (1.12-7) ...
checkout@ubuntu:/$ mkdir ~/snort_src
checkout@ubuntu:/$ cd ~/snort_src
checkout@ubuntu:~/snort_src$ _
```

Figure 76

```
daq-2.0.6/m4/libtool.m4
daq-2.0.6/m4/ltversion.m4
daq-2.0.6/m4/lt~obsolete.m4
daq-2.0.6/m4/ltoptions.m4
daq-2.0.6/configure.ac
root@ubuntu:/home/checkout/snort_src# cd daq-2.0.6
root@ubuntu:/home/checkout/snort_src/daq-2.0.6# _
```

Figure 77

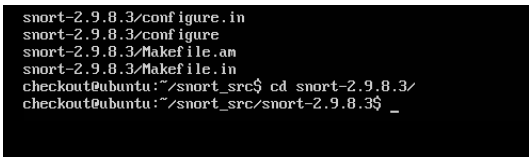

Figure 78

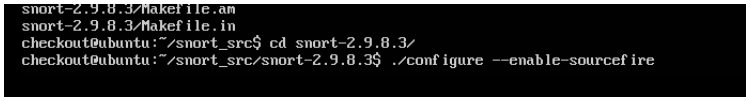

Figure 79

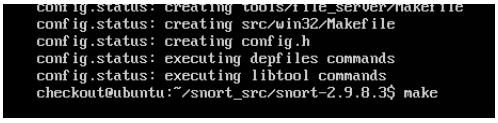

Figure 80

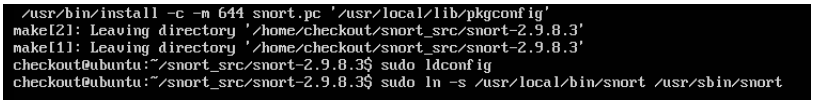

Figure 81

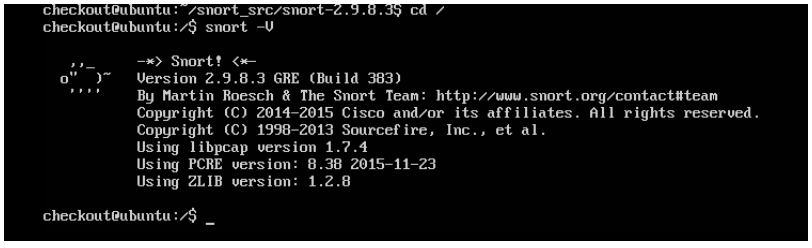

Figure 82

```
checkout@ubuntu:/$ sudo groupadd snort
checkout@ubuntu:/$ sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
checkout@ubuntu:/$ sudo mkdir /etc/snort
checkout@ubuntu:/$ sudo mkdir /etc/snort/rules
checkout@ubuntu:/$ sudo mkdir /etc/snort/rules/iplists
checkout@ubuntu:/$ sudo mkdir /etc/snort/lib/snort_dynamicrules
mkdir: cannot create directory '/etc/snort/lib/snort_dynamicrules': No such file or directory
checkout@ubuntu:/$ sudo mkdir /usr/local/lib/snort_dynamicrules
checkout@ubuntu:/$ sudo mkdir /etc/snort/so_rules
checkout@ubuntu:/$ sudo touch /etc/snort/rules/iplists/black_lists.rules
checkout@ubuntu:/$ sudo touch /etc/snort/rules/iplists/while_lists.rules
checkout@ubuntu:/$ sudo touch /etc/snort/rules/local.rules
checkout@ubuntu:/$ sudo touch /etc/snort/sid-msg.map
checkout@ubuntu:/$ sudo mkdir /var/log/snort
checkout@ubuntu:/$ sudo mkdir /var/log/snort/archived_logs
checkout@ubuntu:/$ sudo chown -R snort:snort /etc/snort
checkout@ubuntu:/$ sudo chown -R snort:snort /var/log/snort
checkout@ubuntu:/$ sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
checkout@ubuntu:/$ cd ~/snort_src/snort-2.9.8.3/etc
checkout@ubuntu:~/snort_src/snort-2.9.8.3/etc$ sudo cp *.conf* /etc/snort
checkout@ubuntu:~/snort_src/snort-2.9.8.3/etc$ sudo cp *.map /etc/snort
checkout@ubuntu:~/snort_src/snort-2.9.8.3/etc$ sudo cp *.dtd /etc/snort
checkout@ubuntu:~/snort_src/snort-2.9.8.3/etc$ cd ~/snort_src/snort-2.9.8.3/src/d
detection-plugins/     dynamic-output/        dynamic-preprocessors/
dynamic-examples/      dynamic-plugins/
checkout@ubuntu:~/snort_src/snort-2.9.8.3/etc$ cd ~/snort_src/snort-2.9.8.3/src/dynamic-preprocessor
s/build/usr/local/lib/snort_dynamicpreprocessor/
checkout@ubuntu:~/snort_src/snort-2.9.8.3/src/dynamic-preprocessors/build/usr/local/lib/snort_dynami
cpreprocessor$ sudo cp * /usr/local/lib/snort_dynamicpreprocessor/
checkout@ubuntu:~/snort_src/snort-2.9.8.3/src/dynamic-preprocessors/build/usr/local/lib/snort_dynami
cpreprocessor$
```

Figure 83

```
checkout@ubuntu:~$ tree /etc/snort
/etc/snort
├── attribute_table.dtd
├── classification.config
├── file_magic.conf
├── gen-msg.map
├── reference.config
├── rules
│   ├── iplists
│   │   ├── black_lists.rules
│   │   └── while_lists.rules
│   └── local.rules
├── sid-msg.map
├── snort.conf
├── so_rules
├── threshold.conf
└── unicode.map

3 directories, 12 files
checkout@ubuntu:~$ _
```

Figure 84

```
3 directories, 12 files
checkout@ubuntu:~$ sudo sed -i "s/include \$RULE\_PATH/#include \$RULE\_PATH/" /etc/snort/snort.conf
checkout@ubuntu:~$ _
```

Figure 85

Figure 86

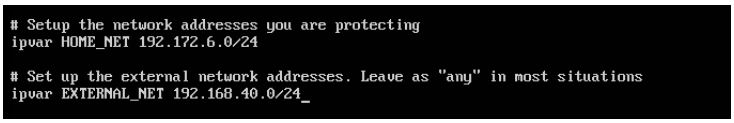

Figure 87

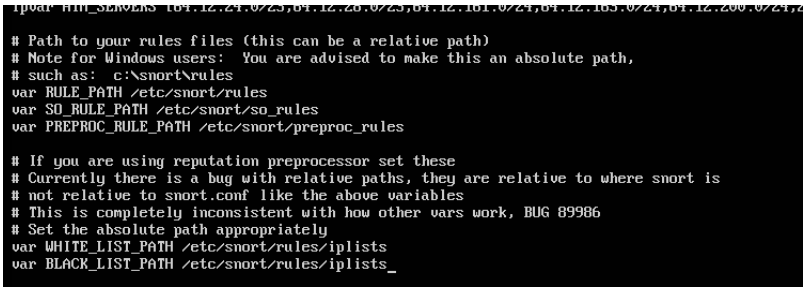

Figure 88

Figure 89

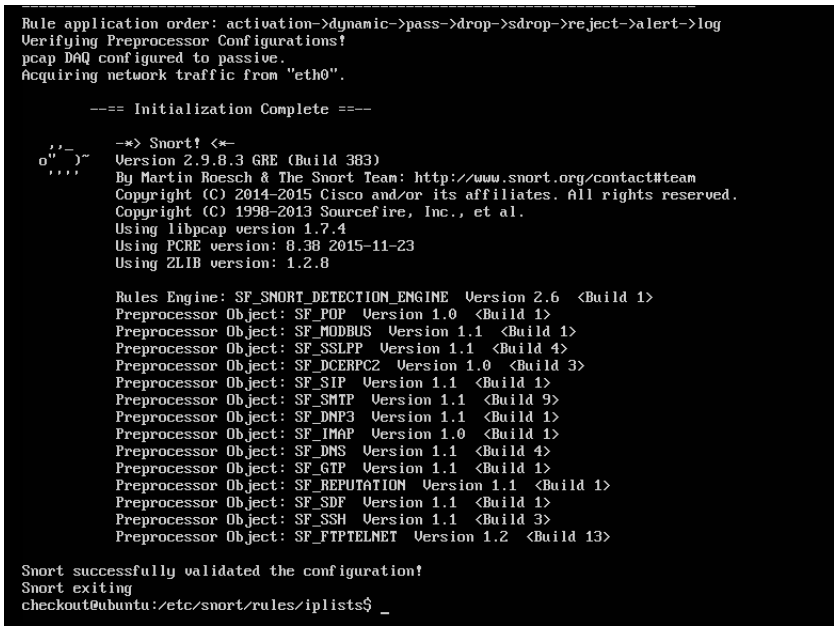

Figure 90

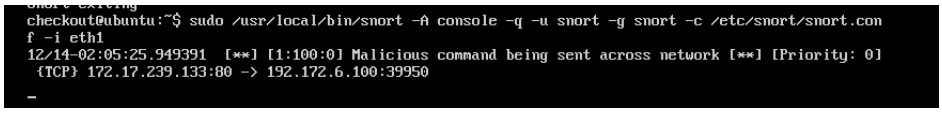

Figure 91

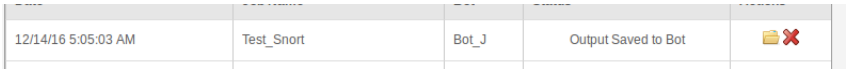

Figure 92

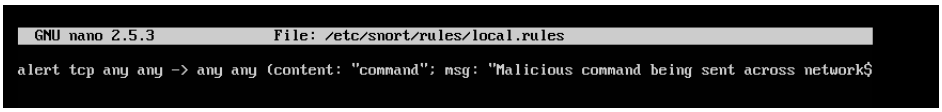

Figure 93

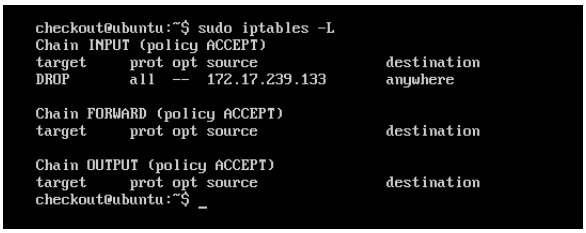

Figure 94

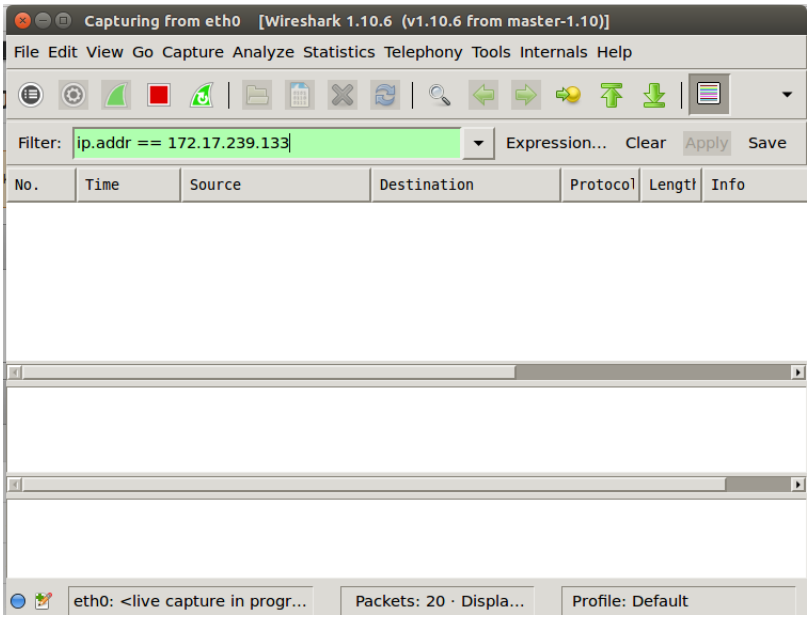

Figure 95

3.3.1.2 Step 2:Prevention

Q5: What does this command mean?
A5: This command alerts the snort admin if there are tcp packets from any ip address and to any ip address that contain the key word "command" in its payload. Th word command was used because by analysing the structure of both wireshark captures, they both have the header 'command' before the actual command that was sent.

Q6: What appears on the screen?
A6: The alert that we set in snort. It also shows what ip address it was sent from which could help us immediately block malicious addresses.

Q7: What command do you need to block the ip address of the C&C server?
A7: `iptables -A INPUT -s 172.17.239.133`

3.3.1.3 Step 3:Deep Packet Inspection

3.3.2 Key Learning/Takeaways:

The key learning of this exercise were preventative measures taken against commands being sent to bots. The ideal way to prevent this would be a Deep Packet Inspection and whie we did not successfully st this up, we simulated a similar experience through snort. We used snort to Scan the payload of packets sent into the network for the key word "command" since we would not want any commands sent to our machines without our permission. If a DPI were handling this situation, once the key word is detected, it would drop that packet based on the payload. We simply dropped all packets coming from the C&C server since we were unable to dro based on payload at this time.

3.4 Part 2 Exercise 2 (Josie Salcedo)

3.4.1 Analysis & Evidence

3.4.1.1 Step 1: Network

We first began by reconfiguring our network by taking away the metasploitable VM and replacing R3 with an Ubuntu 16.04 server and configured it so that its ip address was 192.168.40.4. We then ensured we had Internet connectivity by pinging google.com. We then ensured all the DDWRT VMs had unlimited bandwidth. We updated and upgraded the Server then installed apache2 on the server as seen below.

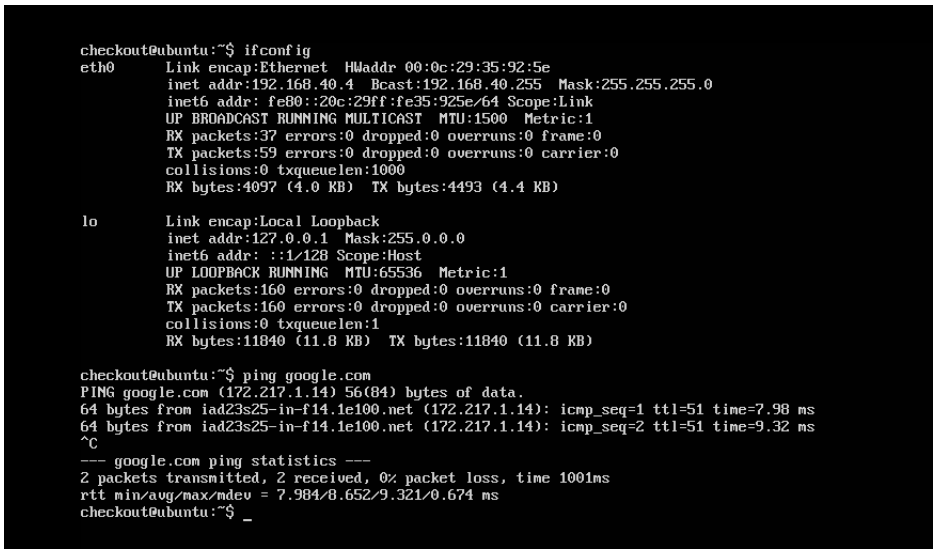

Figure 96: Network interfaces configured and connection to the internet established.

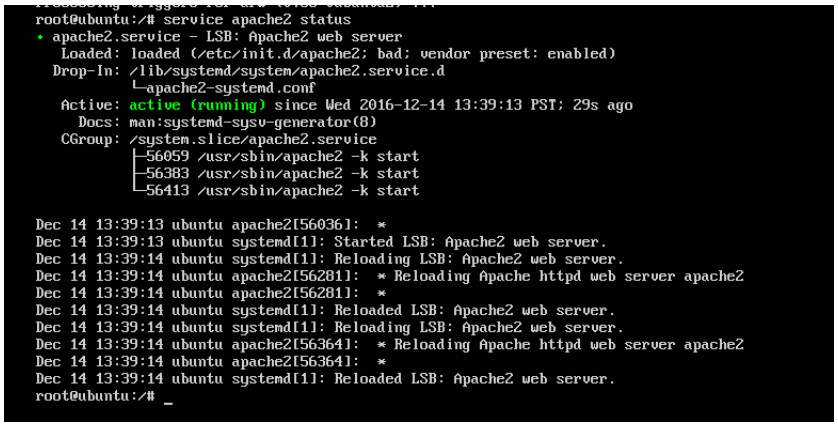

Figure 97: Apache2 running

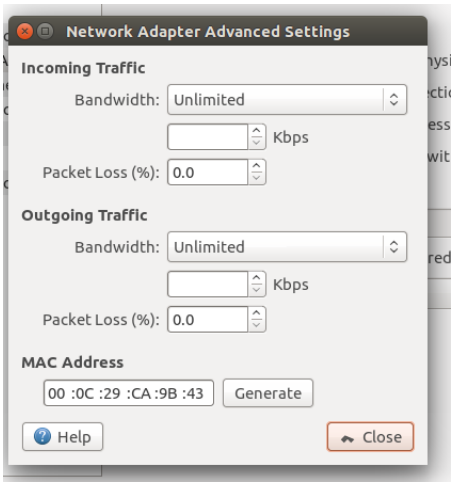

Figure 98: Router Bandwidth set to unlimited

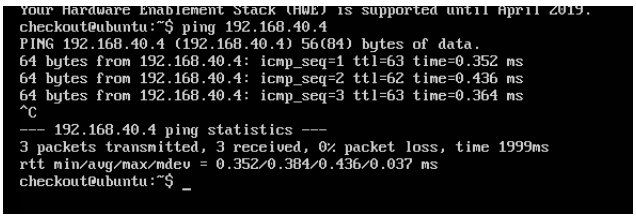

Figure 99: Ping from bot to the apache2 server

We then checked t ensure the iptables had no rules as seen below. We also

looked to see the http connections to the server as seen in 101.

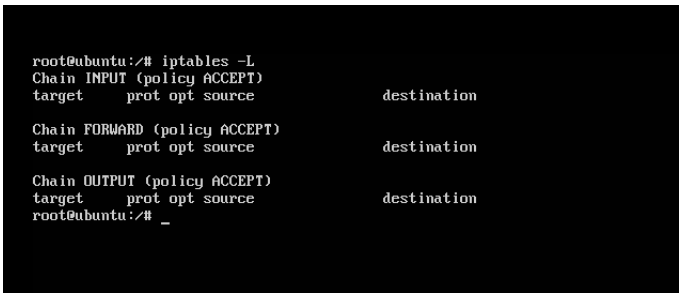

Figure 100: Iptables -L before dos attack

Figure 101: http connections before dos attack

We then prformed a DOS with the hping3 command on a single bot aimed for the Server We then checked the http connections being mae again as seen below.

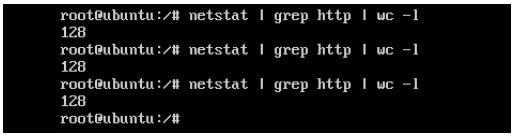

Figure 102: Http connections after dos attack initiated

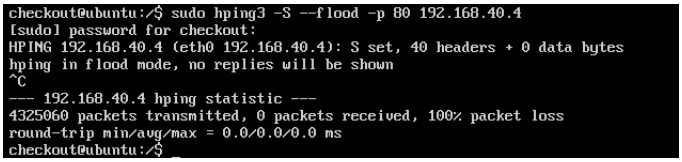

Figure 103: hping3 command initiated on Bot and ended

3.4.1.2 Step 2: DDOS Deflate

We used the wget command to download DDOS Deflate into /usr/local/src and unzipped the file. Then we navigated ito the ddos-deflate-master file and ran the installation.

Figure 104

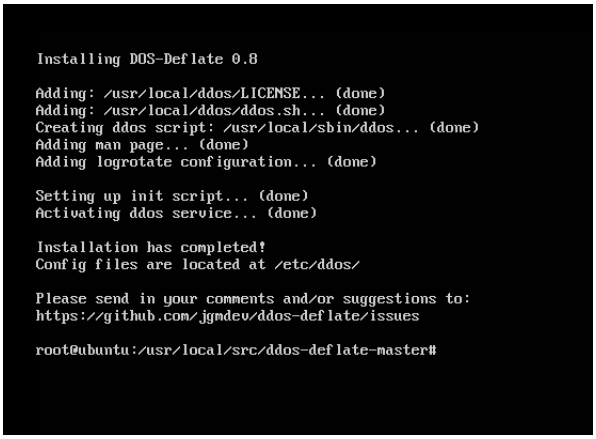

Figure 105

After receiving the indicated Installtion complete message we took note of the different commands available within DDOS deflate with the command `ddos -h`.

Q1: What is DDOS Deflate?**A1:** A bash file that works with a dedicated

system such as iptables or advanced policy firewall to block ip addresses

We then checked the status of ddos as seen below. Once again we started the dos attack with the hping3 command. While this command was running, we checked the iptables again. We then checked the active conntections to the server through the ddos -v command and then looked at the banned ip addresses. At this point there were none. We then used the command `ddos -k 100` to ban any ip address that was making more than 100 connections to the server. We then ensured that this had been saved to the banned ip address list and the iptables.

Q3: What command displays the active connections to the server through DDOS deflate? Banned ip addresses? How many connections are from the bot?**A3:** ddos -v, ddos -b, and 128 connections

Q4: Explain the ddos -kill XX command. What number did you choose?**A4:** 100 since there are 128 connections from the malicious ip address.

Q5: What are the advantages and disadvantages of this approach to a DDOS attack?**A5:** This approach only blocks the ip address which can easily be changed. There is a need to detect and drop packets based on the syn flag since ths is a syn attack.

Q4: Explain the ddos -kill XX command. What number did you choose?**A4:** 100 since there are 128 connections from the malicious ip address.

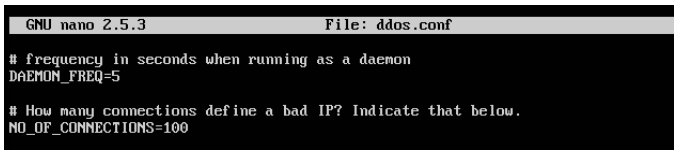

Figure 106

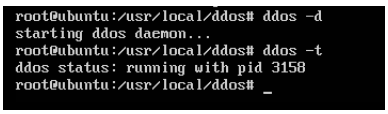

Figure 107

Figure 108

3.4.1.3 Step 3: Advanced Policy Firewall

In this step we use apf in conjunction with DDOS Deflate. First we installed apf and cofigured it as seen below.

Figure 109

```
root@ubuntu:/# wget http://www.rfxn.com/downloads/apf-current.tar.gz
--2016-12-13 22:15:16--  http://www.rfxn.com/downloads/apf-current.tar.gz
Resolving www.rfxn.com (www.rfxn.com)... 129.121.132.46
Connecting to www.rfxn.com (www.rfxn.com)|129.121.132.46|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 107699 (105K) [application/x-gzip]
Saving to: 'apf-current.tar.gz'

apf-current.tar.gz       100%[=================================>] 105.17K   423KB/s    in 0.2s

2016-12-13 22:15:16 (423 KB/s) - 'apf-current.tar.gz' saved [107699/107699]

root@ubuntu:/# tar xfz apf-current.tar.gz
root@ubuntu:/# cd apf-9.7-2
root@ubuntu:/apf-9.7-2# ./install.sh_
```

Figure 110

```
root@ubuntu:/apf-9.7-2# ./install.sh
Installing APF 9.7-2: Completed.

Installation Details:
  Install path:         /etc/apf/
  Config path:          /etc/apf/conf.apf
  Executable path:      /usr/local/sbin/apf

Other Details:
  Listening TCP ports: 22,6010
  Listening UDP ports:
  Note: These ports are not auto-configured; they are simply presented for information purposes. You
 must manually configure all port options.
root@ubuntu:/apf-9.7-2#
```

Figure 111

```
##
# [Main]
##
# !!! Do not leave set to (1) !!!
# When set to enabled; 5 minute cronjob is set to stop the firewall. Set
# this off (0) when firewall is determined to be operating as desired.
DEVEL_MODE="0"

# The installation path of APF; this can be changed but it is not recommended.
INSTALL_PATH="/etc/apf"

# Untrusted Network interface(s); all traffic on defined interface will be
# subject to all firewall rules. This should be your internet exposed
# interfaces. Only one interface is accepted for each value.
IFACE_IN="eth0"
IFACE_OUT="eth0"
```

Figure 112

```
# Configure inbound (ingress) accepted services. This is an optional
# feature; services and customized entries may be made directly to an ip's
# virtual net file located in the vnet/ directory. Format is comma separated
# and underscore separator for ranges.
#
# Example:
IG_TCP_CPORTS="21,22,25,53,80,443,110,143,6000_7000"
IG_UDP_CPORTS="20,21,53,123"
IG_ICMP_TYPES="3,5,11,0,30,8"
```

Figure 113

```
# Common outbound (egress) TCP ports
EG_TCP_CPORTS="21,25,80,443,43"

# Common outbound (egress) UDP ports
EG_UDP_CPORTS="20,21,53"

# Common ICMP outbound (egress) types
# 'internals/icmp.types' for type definition; 'all' is wildcard for any
EG_ICMP_TYPES="all"
```

Figure 114

APF needed to be hard coded to accept our kernel as seen in Figure 115.

```
        MEXT="ko"
elif [[ "$KREL" =~ "4.4" ]]; then
        MEXT="ko"
$EL" =~ "4.4"]] ; then
        if [ ! "$SET_VERBOSE" == "1" ]; then
                echo "kernel version not compatible or netfilter support missing, aborting."
        fi
```

Figure 115

We then attached a video with our submission to show how the dos could be automatically stopped by banning the ip address.

3.4.2 Key Learning/Takeaways:

The key learning of the exercise were to learn how to block DOS attacks not only through straight forward iptables as we had before, but a more extensive approac. We learned to mitigate a DOS attack that was in progress by manually 'killing' the ip address or adding it to the banned ip address list. We also learned how to set up ddos deflate to automatically ban ip addresses associate with DOS activity.

4 Lab Additional Questions & List of Attachments

4.1 Additional Questions

Did you provide answers to any additional questions? None in this lab!

4.2 List of Attachments

Did you provide a list of Attachments (FILTERED pcap files, GNS3 project files, etc.)?

5 Lab Observations, Suggestions & Best Practices

5.1 Observations

Did you provide your constructive observations about the lab? No Complaining and No general comments. You need to be specific. For example, when you say typos, you need to identify all the typos you have encountered.

5.2 Suggestions

Did you provide suggestions for improvements & additional exercises?

5.3 Best Practices

Did you provide your Best Practices?

Work together outside the lab. Label the images. Tag the answers. Learn more about ShareLatex.

6 Lab References

[1] J. Talamantes. (). The plugbot: Hardware botnet research project, [Online]. Available: https://www.redteamsecure.com/the-plugbot-hardware-botnet-research-project/.

[2] ——, (). Plugbot-plug issues, [Online]. Available: https://github.com/redteamsecurity/PlugBot-Plug/issues.

[3] ——, (). Plugbot-c2c issues, [Online]. Available: https://github.com/redteamsecurity/PlugBot-C2C/issues.

[4] M. McDowell. (). Understanding denial-of-service attacks, [Online]. Available: https://www.us-cert.gov/ncas/tips/ST04-015.

[5] VMWare. (). Lan segment requirements, [Online]. Available: https://www.vmware.com/support/ws5/doc/ws_team_lan_requirements.html.

7 Acknowledgements

We would like to acknowledge Dr. Salib for recommending the DDWRT Virtual Machine. It allowed us to do more experimentation, and divide up the subnets more effectively than with a single hardware DDWRT AP. It was also a valuable learning experience with routing.

Did you provide a well written acknowledgment, that is, We wish to acknowledge so and so (including TAs, Lab Manager, online communities, etc.) and for what? [No boiler plate kind of acknowledgment, be specific as for who and for what you are offering your acknowledgments]

8 Lab Extra Credit Exercises

8.1 Extra Credit 1:

8.2 Extra Credit 2: