NSS Lab Assignment 3

Kaustav Vats 2016048
Sushant Kumar Singh 2016103

Task 1

1)Creating File

- First we created the required file '**kaustav-sushant.1.txt**' (using echo and input redirection)

2)Encryption

- Then used the dd command to create a key file 'key1.txt'
  - Input (if) to dd was taken from /dev/urandom as we want radom key to be generated
  - Output file was selected as 'key1.txt'
  - We used byte size(bs) as 16, since we are using aes-128-cbc which uses 16 bytes key
  - Count for blocks (count) is 1 as we want only one block of 16 bytes as the key
- Similarly we created initialization vector using dd command into file 'iv1.txt'

```
kvats@kvats:~$ echo -e "Kaustav Vats 2016048\nSushant Kumar Singh 2016103" > kaustav-sushant.1.txt
kvats@kvats:~$ cat kaustav-sushant.1.txt
Kaustav Vats 2016048
Sushant Kumar Singh 2016103
kvats@kvats:~$ dd if=/dev/
Display all 192 possibilities? (y or n)
kvats@kvats:~$ dd if=/dev/urandom of=key1.txt bs=16 count=1
1+0 records in
1+0 records out
16 bytes copied, 0.00016435 s, 97.4 kB/s
kvats@kvats:~$ dd if=/dev/urandom of=iv1.txt bs=16 count=1
1+0 records in
1+0 records out
16 bytes copied, 0.000162598 s, 98.4 kB/s
```

2)(continued)

- Then converted the files key1.txt and iv1.txt to hex to pass to openssl enc command to files key2.txt and iv2.txt
  - **xxd** command was used to generate the hex dump
  - -l 32 option was used to generate hex dump, 16 hex bytes have 32 characters in hex dump (2 for each byte e.g. \xaa to ‘aa’)
  - we then used **head** -c 32 to extract exactly first 32 bytes of this dump into **key3.txt** and **iv3.txt** as xxd had added trailing \n by default to its output in key2.txt and iv2.txt which is not part of the key
  - **So finally key3.txt and iv3.txt are the key and initialization vectors to used by openssl**
- Then we used openssl enc command to encrypt the file **‘kaustav-sushant.1.txt’** supplying the key3.txt as key and iv3.txt as iv
  - Saved output of encryption to **‘encrypted.txt’**

Following snapshot shows how keys are generated and encryption is done

```
kvats@kvats:~$ xxd -l 32 -ps key1.txt > key2.txt
kvats@kvats:~$ xxd -l 32 -ps iv1.txt > iv2.txt
kvats@kvats:~$ head -c 32 key2.txt > key3.txt
kvats@kvats:~$ head -c 32 iv2.txt > iv3.txt
kvats@kvats:~$ cat key3.txt
0e063250e91a2bd1d1403d62d8116ecbkvats@kvats:~$
kvats@kvats:~$ cat iv3.txt
38e27fad5a46502d9d82ac54042f5c5ckvats@kvats:~$
kvats@kvats:~$ openssl enc -aes-128-cbc -in kaustav-sushant.1.txt -K `cat key3.txt` -iv `cat iv3.txt
` -out encrypted.txt -p
salt=C0A840D5D2550000
key=0E063250E91A2BD1D1403D62D8116ECB
iv =38E27FAD5A46502D9D82AC54042F5C5C
kvats@kvats:~$ _
```

3)

- To create the **HMAC** we used sha1 algo with the **hex key ‘key3.txt’** generated in previous step
  - Output stored into **hmac_enc.txt**

4)

- We then created the copy of ‘**encrypted.txt’** named **‘copy_enc.txt’**

5)

- We then changed the file and tried hmac validation
  - Changed one byte (first) of the file by opening and modifying it
  - Used **cmp** to check that the two files differ
  - Calculated hmac of **copy_enc.txt** to file **hmac_verify.txt** using the same steps as in **3)**
  - Compared the HMAC for hmac validation

```
kvats@kvats:~$ openssl sha1 -mac hmac -macopt hexkey:`cat key3.txt` encrypted.txt > hmac_enc.txt
kvats@kvats:~$ cp encrypted.txt copy_enc.txt
kvats@kvats:~$ cat hmac_enc.txt
HMAC-SHA1(encrypted.txt)= 63e4ef6217436b55eca1b2f3530ed312babe1b97
kvats@kvats:~$
```

Creation of hmac and copy of encrypted file

```
kvats@kvats:~$ cmp -b -c encrypted.txt copy_enc.txt
encrypted.txt copy_enc.txt differ: byte 1, line 1 is 311 M-I 106 F
```

Encrypted.txt and copy_enc.txt are different as shown

```
kvats@kvats:~$ ls
copy_enc.txt   hmac_enc.txt     iv1.txt  iv3.txt                key1.txt  key3.txt
encrypted.txt  hmac_verify.txt  iv2.txt  kaustav-sushant.1.txt  key2.txt
kvats@kvats:~$ cmp -c -b hmac_enc.txt hmac_verify.txt
hmac_enc.txt hmac_verify.txt differ: byte 1, line 1 is  66 6  61 1
kvats@kvats:~$ cat hmac_enc.txt
63e4ef6217436b55eca1b2f3530ed312babe1b97
kvats@kvats:~$ cat hmac_verify.txt
154d07e79b4775a33b019b1e7bbe050d11e60048
kvats@kvats:~$ _
```

This shows that the HMACs don’t match for the files and **HMAC Validation Failed**

6)

- Then performed the decryption using openssl enc **-d** (for decryption) option after supplying the required key and iv
  - Decryption performed on ‘ecrypted.txt’ and ‘copy_enc.txt’
  - As expected the ‘encrypted.txt’ was decrypted but ‘copy_enc.txt’ could not be decrypted as the file was damaged

Decryption of ‘encrypted.txt’ successful, showing the contents

```
kvats@kvats:~$ openssl enc -d -aes-128-cbc -in encrypted.txt -K `cat key3.txt` -iv `cat iv3.txt` -ou
t decrypted.txt -p
salt=C098086622560000
key=0E063250E91A2BD1D1403D62D8116ECB
iv =38E27FAD5A46502D9D82AC54042F5C5C
kvats@kvats:~$ cat decrypted.txt
Kaustav Vats 2016048
Sushant Kumar Singh 2016103
kvats@kvats:~$
```

Below is the decryption for copy_enc.txt (which was modified in previous step)

```
kvats@kvats:~$ openssl enc -d -aes-128-cbc -in copy_enc.txt -K `cat key3.txt` -iv `cat iv3.txt` -out
 decrypted.txt -p
salt=C048EABBFA550000
key=0E063250E91A2BD1D1403D62D8116ECB
iv =38E27FAD5A46502D9D82AC54042F5C5C
bad decrypt
140367906374080:error:0606506D:digital envelope routines:EVP_DecryptFinal_ex:wrong final block lengt
h:../crypto/evp/evp_enc.c:525:
kvats@kvats:~$ _
```

Task 2

Server- 10.0.0.7
Client- 10.0.0.5

1) First we setup the vms using ifconfig command to have the ips for same internal network. Installed nc on both the VM's using apt.
2) Created the script server **server.sh** as shown below and started it using **sh server.sh**
   a) infinite loop of while true
   b) Used **echo** to output the current http response and the date and piped to netcat
   c) Used nc -lv 80 i.e -l for listening to port 80, -v for verbose output of the server
3) Ran the command lynx http://10.0.0.7 on client to fetch the output as shown

Server.sh script and its execution

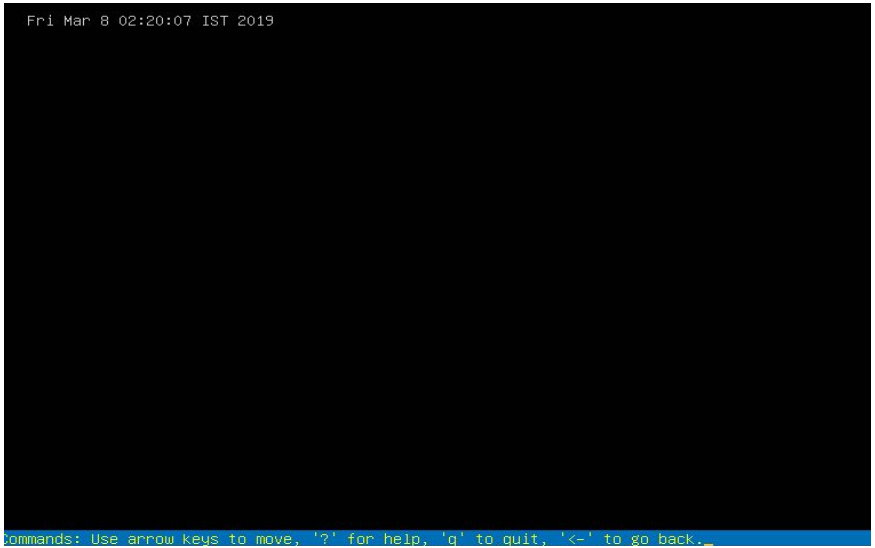
```
sushant@sushant:~/nss_lab_3$ cat server.sh
while true;
do
        { echo "HTTP/1.1 200 OK\\r\\n"; echo "$(date)";} | nc -lv 80;
done
sushant@sushant:~/nss_lab_3$ sudo sh server.sh
Listening on [0.0.0.0] (family 0, port 80)
Connection from [10.0.0.5] port 80 [tcp/http] accepted (family 2, sport 47056)
Listening on [0.0.0.0] (family 0, port 80)
Connection from [10.0.0.5] port 80 [tcp/http] accepted (family 2, sport 47058)
GET / HTTP/1.0
Host: 10.0.0.7
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q
=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17

Listening on [0.0.0.0] (family 0, port 80)
^Csushant@sushant:~/nss_lab_3$
```

```
sushant@beehive:~$ nc 10.0.0.7 80
HTTP/1.1 200 OK

Fri Mar  8 02:19:13 IST 2019
^C
```

Received page on lynx shows date as expected



<- Trying to receive output on nc before trying on lynx

4)

- For encrypted the traffic we first generated the key and iv as in Task 1
  - Shared the same key and iv to client for decryption (using nc itself).
  - (We have shared the files key3.tx, iv3.txt so that we dont need to type the whole key in decryption stage of the next step)

Creation of key and iv

```
sushant@sushant:~/nss_lab_3$ dd if=/dev/urandom of=key1.txt bs=16 count=1
1+0 records in
1+0 records out
16 bytes (16 B) copied, 0.0303234 s, 0.5 kB/s
sushant@sushant:~/nss_lab_3$ dd if=/dev/urandom of=iv1.txt bs=16 count=1
1+0 records in
1+0 records out
16 bytes (16 B) copied, 0.000239035 s, 66.9 kB/s
sushant@sushant:~/nss_lab_3$ xxd -l 32 -ps key1.txt > key2.txt
sushant@sushant:~/nss_lab_3$ xxd -l 32 -ps iv1.txt > iv2.txt
sushant@sushant:~/nss_lab_3$ head -c 32 key2.txt > key3.txt
sushant@sushant:~/nss_lab_3$ head -c 32 iv2.txt > iv3.txt
sushant@sushant:~/nss_lab_3$
```

Shared the key and iv from server. On client received the key as shown.

```
sushant@sushant:~/nss_lab_3$ nc -l 8000 < key3.txt
```

```
sushant@beehive:~$ nc 10.0.0.7 8000 > key3.txt_
```

4)

- Created the script **secure_server.sh** (shown in snapshot) to send the encrypted response to client
  - We used similar script as in previous step with little change i.e. :
  - Piped the echo commands output to openssl enc for encrypting it before sending then finally piped it to netcat server

5)

- On client side we piped the output from client netcat to openssl enc -d command to decrypt the response
  - We used -w1 option with nc 10.0.0.7 80 just to ensure that after 1 second client closes the connection

```
sushant@sushant:~/nss_lab_3$ cat secure_server.sh
#!/bin/bash
while true;
do
        { echo "HTTP/1.1 200 OK\\r\\n"; echo "$(date)";} | openssl enc -aes-128-
cbc -K `cat key3.txt` -iv `cat iv3.txt` | nc -lv 80;
done
sushant@sushant:~/nss_lab_3$ sudo sh secure_server.sh
Listening on [0.0.0.0] (family 0, port 80)
Connection from [10.0.0.5] port 80 [tcp/http] accepted (family 2, sport 48290)
Listening on [0.0.0.0] (family 0, port 80)
```

Server side script for encrypting the response

```
sushant@beehive:~$ nc 10.0.0.7 80 -w1
NR♦♦♦_*♦♦f~t♦♦♦N♦$
                  ♦_yp♦♦♦#O♦♦♦m♦♦♦[sushant@beehive:~$
sushant@beehive:~$ nc 10.0.0.7 80 -w1 | openssl enc -d -aes-128-cbc -K $(cat key3.txt) -iv $(cat iv3
.txt)
HTTP/1.1 200 OK

Fri Mar  8 09:18:15 IST 2019
sushant@beehive:~$
```

Encrypted response received on client side, which was then decrypted in next step

6)

- On client side we then piped decrypted output from previous step to a local server hosted on **port 8081**
  - This server was used to show the decrypted incoming response from the actual server 10.0.0.7

7)

- Then we used lynx to send get response to the local server at 8081 port
  - As expected The output was decrypted and date and time was shown
  - The ip address of localhost is 127.0.0.1 so we do lynx http://127.0.0.1:8081

Client side decryption of servers response and sending to the localhost server

```
sushant@beehive:~$ cat client.sh
#!/bin/bash
nc 10.0.0.7 80 -w1 | openssl enc -d -aes-128-cbc -K $(cat key3.txt) -iv $(cat iv3.txt) | nc -lv -w1
8081
sushant@beehive:~$ bash client.sh
Listening on [0.0.0.0] (family 0, port 8081)
Connection from localhost.localdomain 55622 received!
GET / HTTP/1.0
Host: 127.0.0.1:8081
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17
```

Below snapshot shows Response of get request to localhost, which is received from actual server and decrypted and then send here

```
sushant@beehive:~$ lynx http://127.0.0.1:8081_
```

```
Fri Mar 8 09:30:14 IST 2019

Commands: Use arrow keys to move. '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

8)

- While running the client side command for receiving server response we had also captured the packets using **tcpdump**
  - Captured packets both on **interface enp0s8** which is in the subnet as server and on **interface lo** ie. localhost
  - As expected the packet capture shows the **encrypted** packets received from server to client i.e. interface **enp0s8**
  - Also the **decrypted** packets which are piped to **localhost** server

```
sushant@beehive:~$ cat capture.pcap
♦õ♦♦n♦\4JE<♦@@♦♦♦4♦<4♦♦♦♦0♦♦
p♦P♦♦n♦\WJE<@@<♦♦♦42♦@<4♦♦♦♦0♦♦
p♦P♦p♦P♦♦n♦\eBE4♦@@♦♦♦4♦<42♦@ ♦V♦(
p♦P♦p♦P♦♦n♦\    qEc♦,@@|f♦♦42♦@ <4♦V♦W
p♦P♦p♦P♦HTTP/1.1 200 OK

Fri Mar  8 10:13:47 IST 2019
♦n♦\"   BE4♦@@♦♦♦4♦<42♦@O♦V♦(
p♦P♦p♦P♦♦n♦\2   FE8♦@@♦♦♦4♦<42♦@O♦V♦,
p♦P♦p♦P♦GET / HTTP/1.0
Host: 127.0.0.1:8081
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17

♦n♦\4   BE4♦-@@|♦♦♦42♦@O<5♦^♦(
p♦P♦p♦P♦♦n♦\♦   BE4♦.@@|♦♦♦42♦@O<5♦^♦(
p♦T♦p♦P♦♦n♦\_   BE4♦@@♦♦♦4♦<52♦@P♦V♦(
p♦T♦p♦T♦♦n♦\♦   BE4♦/@@|♦♦♦42♦@P<5♦^♦(
p♦T♦p♦T♦sushant@beehive:~$ sudo tcpdump -i enp0s8 -i lo -w capture.pcap
```

Task 3

- We created a **server side** script **secure_server_hmac.sh**(shown in screenshot) similar to **secure_server.sh** but with provision to send the HMAC of the response
  - We first store the encrypted response to be sent in a shell variable “**a**”
  - Then calculated the HMAC of “a” i.e. the encrypted response into variable **“b”**
  - Finally piped this encrypted response followed by HMAC=<calculated HMAC> to netcat server
- We also created a script on **client side** called **client_hmac.sh** (shown in screenshot)
  - Here we first receive the response from server using nc 10.0.0.7:80 -w1 into variable response
  - Then the hmac is extracted by finding “HMAC” substring (using **grep**)in response
  - First part is encrypted payload and following “HMAC” substring is its value
  - Then we decrypt the payload i.e. the encrypted response
  - Calculate HMAC of payload
  - compare it with HMAC received
  - If equal then send it to hosted localhost
  - Else we send the msg **MITM detected** to localhost

Client side script **client_hmac.sh**

```
#!/bin/bash

response=$(nc 10.0.0.7 80 -w1);
#header=$(echo "$response" | sed -n 1p);

hmac_start_ind=$(echo "$response" | grep -aob "HMAC=" | grep -oE '[0-9]+' );
#echo "$hmac_start_ind";

payload=$(echo "$response" | head -c $hmac_start_ind);
#echo "payload ->$payload<--";

hmac=$(echo "$response" | tail -c +$hmac_start_ind | cut -d "=" -f 2 );
hmac_payload=$(echo -n "$payload" | openssl sha1 -mac hmac -macopt hexkey:$(cat key3.txt) | cut -d '
 " -f 2);

#echo "response_-->$response<--";
echo "received hmac $hmac";
echo "calculated hmac $hmac_payload";

if [ "$hmac" = "$hmac_payload" ]
then
        echo "HMAC validation successful";
        decrypted_payload=$(echo -n "$payload" | openssl enc -d -aes-128-cbc -K $(cat key3.txt) -iv
$(cat iv3.txt));
else
        echo "HMAC validation failed"
        decrypted_payload="MITM detected\n"
fi
echo "$decrypted_payload" | nc -lv -w1 8081
~
~
```

Server side script **secure_server_hmac.sh**

```
#!/bin/bash
while true;
do
        a=$({ echo "HTTP/1.1 200 OK\r\n"; echo "$(date)"; } | openssl enc -aes
-128-cbc -K $(cat key3.txt) -iv $(cat iv3.txt));
        b=$(echo -n "$a" | openssl sha1 -mac hmac -macopt hexkey:$(cat key3.txt)
 | cut -d " " -f 2);
        echo "->$a<-";
        echo "->$b<-";
        { echo -n "$a"; echo "HMAC=$b";} | nc -lv 80;
done
~
~
```

Task 3 (continued)

- We setup a third vm with ip 10.0.0.9 as the router
  - Added route rule on Server for Router as gateway for reaching host client
  - Added route rule on Client for Router as gateway for reaching host Server
  - Enabled forwarding on Router VM
- we run the server script on server **sh secure_server_hmac.sh**
- Then we run **sh client_hmac.sh** to receive the output from server
- Lynx on localhost

Server- 10.0.0.7
Client- 10.0.0.5
Router- 10.0.0.9

(Snapshot below) As expected the debugging output from script shows that HMAC validation is **successful** on client when router **has not changed** the encrypted packet. Also the lynx on localhost shows the date and time in clear text

```
sushant@beehive:~$ sh client_hmac.sh
received hmac 7d727c26d8e893614ed48604911a568bb41e910f
calculated hmac 7d727c26d8e893614ed48604911a568bb41e910f
HMAC validation successful
Listening on [0.0.0.0] (family 0, port 8081)
Connection from localhost.localdomain 47886 received!
GET / HTTP/1.0
Host: 127.0.0.1:8081
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17

sushant@beehive:~$
```

Lynx http://127.0.0.1:80 shows date and time

Task 3 (continued)

1. We then added a rule on Router VM to redirect packets to port 5555
2. Then used netsed to modify the packets coming on port 5555
3. Received response on client side by running **sh client_hmac.sh**

```
sushant@beehive2:~$ sudo iptables -t nat -A PREROUTING -p tcp -j REDIRECT --to 5555
```

```
sushant@sushant:~/nss_lab_3$ sudo sh secure_server_hmac.sh
->♦R♦♦♦_♦♦♦f♦♦o♦♦♦(♦mi♦@♦Y0y♦♦6`♦fI♦♦<-
->16bdcdcd03faaec09324d7ca676f4d24e3e01a0d<-
Listening on [0.0.0.0] (family 0, port 80)
Connection from [10.0.0.9] port 80 [tcp/http] accepted (family 2, sport 52976)
```

← 1.)This is **server VM**. Encrypted response and HMAC sent to client is shown.

```
^Csushant@beehive2:~$ sudo netsed tcp 5555 0 0 s/R/c/1
netsed 1.2 by Julien VdG <julien@silicone.homelinux.org>
      based on 0.01c from Michal Zalewski <lcamtuf@ids.pl>
[*] Parsing rule s/R/c/1...
[+] Loaded 1 rule...
[+] Using dynamic (transparent proxy) forwarding.
[+] Listening on port 5555/tcp.
[+] Got incoming connection from 10.0.0.5,50152 to 10.0.0.7,8081
[*] Forwarding connection to 10.0.0.7,8081
[!] Cannot connect to remote server, dropping connection.
[+] Got incoming connection from 10.0.0.5,43182 to 10.0.0.7,80
[*] Forwarding connection to 10.0.0.7,80
[+] Caught server -> client packet.
    Applying rule s/R/c...
    (rule just expired)
[*] Done 1 replacements, forwarding packet of size 94 (orig 94).
```

← 2.)This is **Router VM**. It replaced the character R in encrypted text with c, and thus modified the payload.

```
sushant@beehive:~$ lynx http://10.0.0.7
MITM detected
Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

MITM detected msg on localhost on doing lynx http://10.0.0.7

```
sushant@beehive:~$ sh client_hmac.sh
received hmac 16bdcdcd03faaec09324d7ca676f4d24e3e01a0d
calculated hmac de2bdc0d6c6c1d7d1b1fff37c0b60ec175666469
HMAC validation failed
Listening on [0.0.0.0] (family 0, port 8081)
Connection from localhost.localdomain 47856 received!
GET / HTTP/1.0
Host: 127.0.0.1:8081
Accept: text/html, text/plain, text/sgml, text/css, application/xhtml+xml, */*;q=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.9dev.16 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.5.17
```

← 3.)This is **Client VM**. as expected HMAC validation failed. Note the HMAC received is same as sent by server but calculated HMAC is different