



TLS Assignment

23.04.2019

Om Sitapara
CS16BTECH11036

Harshit Patel
CS16BTECH11017

Plagiarism Statement:

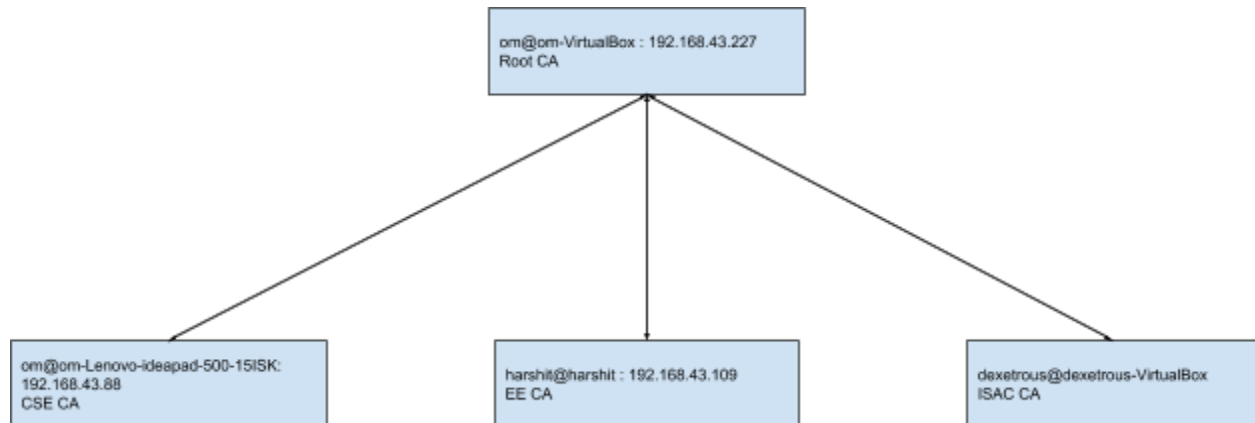
We certify that this assignment/report is our own group's work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, we understand my responsibility to report honour violations by other students if we become aware of it.

Names of the Students in the group: Om Sitapara, Harshit Patel

Date: 23/04/2019

Signature: OS, HP

Here



Here are the screenshots for the transfer.

- Sending EE csr:

[illegible]

- Receiving EE.pem by root:

```

bm@om-VirtualBox:~$ scp ee.pem harshit@192.168.43.109:/home/harshit/
The authenticity of host '192.168.43.109 (192.168.43.109)' can't be established.
ECDSA key fingerprint is SHA256:YZcl45iIf1PVb4i2eZWihp8p8Dn59FCm7Xny0FvmbGg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.109' (ECDSA) to the list of known hosts.
harshit@192.168.43.109's password:
ee.pem
100% 1314 18.7KB/s 00:00

```

Similarly here are the other screenshots of transfer and signing:

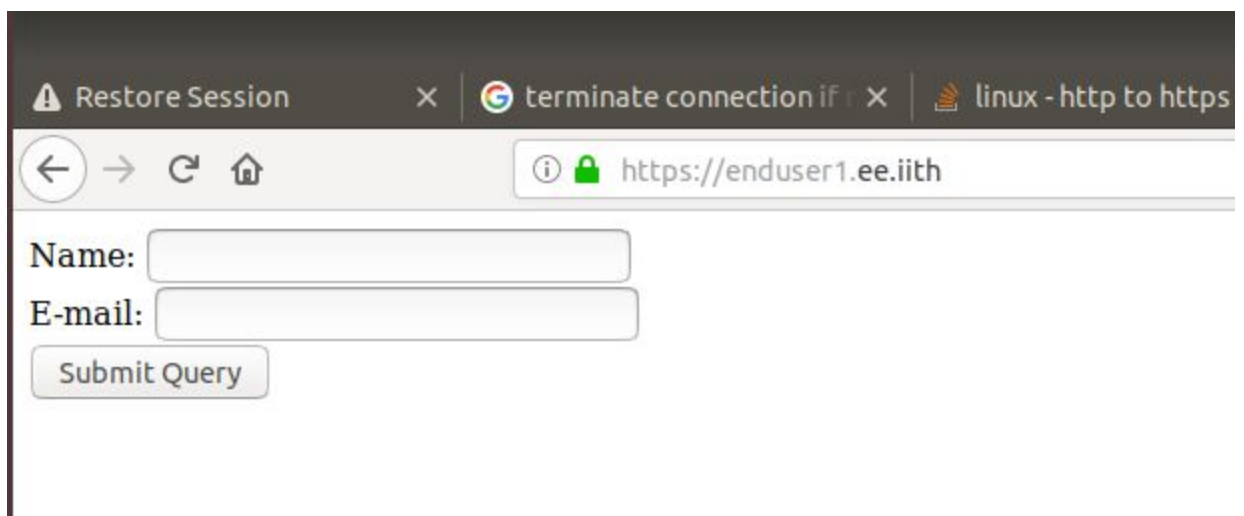
```
om@om-VirtualBox:~$ scp isac.pem dexetrous@192.168.43.95:/home/dexetrous/
The authenticity of host '192.168.43.95 (192.168.43.95)' can't be established.
ECDSA key fingerprint is SHA256:SxgNwd36LQKNRhuuNeoJkH4eL07Fk3LW3taxP848SE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.95' (ECDSA) to the list of known hosts.
dexetrous@192.168.43.95's password:
isac.pem 100% 1322 65.6KB/s 00:00

dexetrous@dexetrous-VirtualBox:~$ scp isac.csr om@192.168.43.227:/home/om/
The authenticity of host '192.168.43.227 (192.168.43.227)' can't be established.
ECDSA key fingerprint is f3:2d:88:65:69:37:2d:0d:77:7a:b9:28:be:6c:24:ed.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.227' (ECDSA) to the list of known hosts.
om@192.168.43.227's password:
isac.csr 100% 1041 1.0KB/s 00:00
```

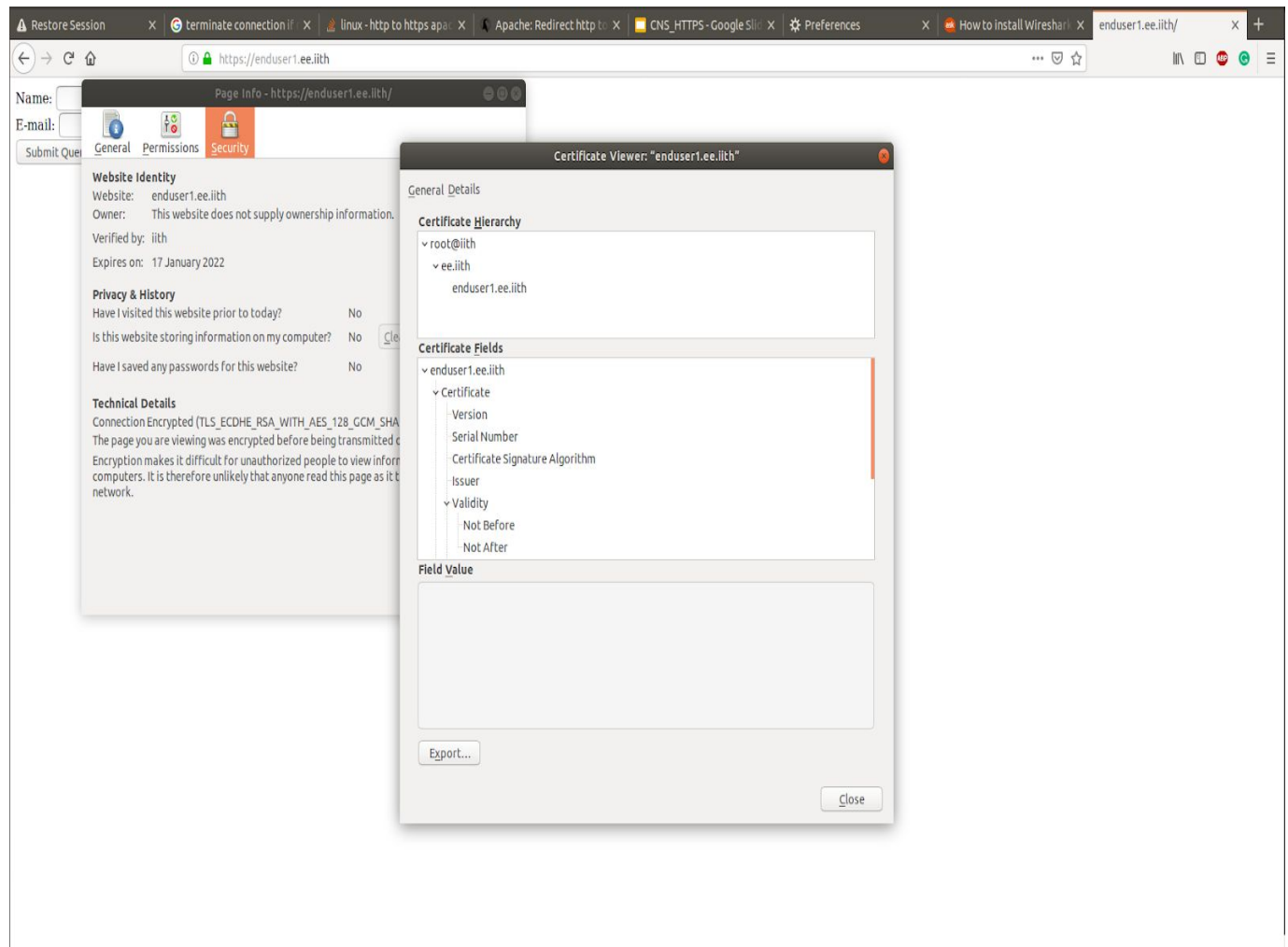
Question-2

For this part we have hosted a website called enduser1.ee.iith on ip 192.168.43.109. We have made this server to redirect all traffic to https. Now on ip 192.168.43.88 we did the following things:

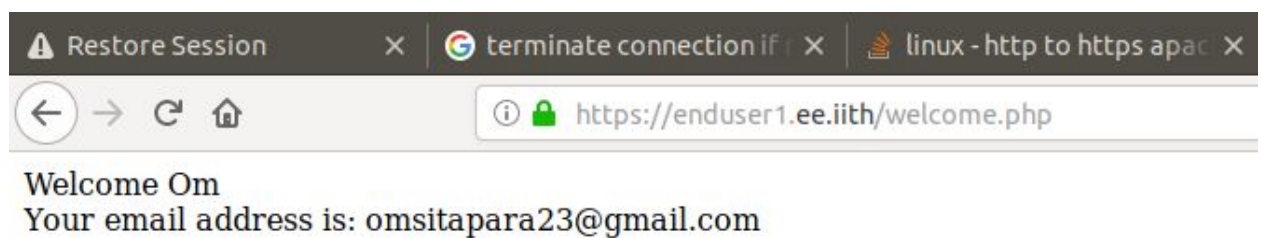
- Installed the **root CA** of iith
- Added the entry **192.168.43.109 enduser1.ee.iith** in **/etc/hosts/** for accessing the site using name (performing local dns)
- Now accessing the site it opens like this:



- The certificate can be viewed :



- Now entering the name and password : This shows that the server has received the entered data



- The messages passed are encrypted :

(ip.src == 192.168.43.88 or ip.src == 192.168.43.109) and (ip.dst == 192.168.43.88 or ip.dst == 192.168.43.109)

▶ Frame 315: 423 bytes on wire (3384 bits), 423 bytes captured (3384 bits) on interface 0
 ▶ Ethernet II, Src: IntelCor_e5:9e:26 (dc:53:60:e5:9e:26), Dst: HonHaiPr_10:f5:f5 (68:14:01:10:f5:f5)
 ▶ Internet Protocol Version 4, Src: 192.168.43.88, Dst: 192.168.43.109
 ▶ Transmission Control Protocol, Src Port: 48374, Dst Port: 443, Seq: 631, Ack: 157, Len: 357
 ▼ Secure Sockets Layer
 ▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
 Content Type: Application Data (23)
 Version: TLS 1.2 (0x0303)
 Length: 352
 Encrypted Application Data: 000000000000000001db0979e3716a94877c554c317a2927d9...

2b 6d bc f6 01 bb 8e 34 67 b2 b1 bb 5a 29 80 18	+m.....4 g...Z)...
00 ed 44 21 00 00 01 01 08 0a 99 0a f1 8d 07 85	..D!.....
56 69 17 03 03 01 60 00 00 00 00 00 00 01 db	Vi....
09 79 e3 71 6a 94 87 7c 55 4c 31 7a 29 27 d9 ed	..y.qj... UL1z)'...
9c f2 24 58 d6 e7 5a e1 f2 4a 9e a2 50 33 dd 58	..\$X...Z...J...P3-X
d3 99 da 87 99 08 61 88 92 54 5a 53 b3 e3 39 9fa...TZS...9.
de f7 72 99 51 2a fb e8 c5 04 07 f6 73 5f 5e bf	..r.Q*... ..s_^.
8d b0 e0 2f dc ff 95 77 11 1e 98 1e f0 a6 9f 08	.../...w
f3 18 29 21 4c cd 71 69 a2 92 49 11 07 f5 12 fd	..)!L.qi ..I.....
24 96 99 24 5d 09 17 68 04 52 c9 6f 2a c2 da 49	\$..\$]...h ..R.o*...I
fe 68 00 97 52 c3 90 06 fa e7 75 e5 29 a3 8b 71	..h..R... ..u.)...q
f7 ea e3 6b e2 5a 3f e5 16 35 a6 de d5 c7 bd dc	...k.Z?..5.....
88 04 a2 5d ff b1 b2 f7 30 e5 8d 22 a8 b0 53 0d	...]}....0..."S.
78 23 33 7d d2 d0 05 ac 9b c6 c9 34 35 3e 5e e4	x#3}.... ..45>^.
bf b7 ee ed eb eb 9c 5b dc b0 2e 01 81 73 c9 1e[.....s.
0b 4a 02 45 7c 96 68 5b 67 bf db da 7e 75 cf 9e	..J.E ..h[g...~u..
e8 f9 70 01 0a 78 b7 53 59 cd 22 4f fd 92 90 6a	..p...x.S Y.."O...j
d3 de 15 2f b0 d3 f5 52 fe 6e dc 20 45 18 5f 9c	.../...R ..n. E._.
3c 16 b4 29 b8 3b 05 d5 d1 2b 61 4c ab 52 fc 1e	<...).;... ..+aL.R..
3a a5 d3 75 aa ab 47 eb 63 d3 5f b4 c6 aa 9c d4	:...u..G..c.....
48 c3 a8 80 71 2a 93 eb 99 69 44 d7 4b 91 77 10	H...q*... ..iD.K.w.
8a a2 8f 8e c5 40 76 f6 9f 43 41 d6 08 74 15 d9@v...CA...t..
ee 8f 4f f7 b7 b0 db 17 13 f2 71 1c 96 2f 86 01	..O..... ..q.../..
a7 a9 1e e9 8d bc 77 90 ff e2 bc c8 c7 d0 03 65w.....e
71 c5 1f 7e 25 aa 91	q...~%...

Question-3

We have 3 code files:

Peer.cpp

clientSSL.cpp

serverSSL.cpp

Command to run : `g++ peer.cpp -o peer -lssl -lcrypto -lpthread`
`./peer`

This will ask to be server or client. Now set the fields asked accordingly.

In the code for each peer, I have loaded openssl algorithms:

OpenSSL_add_all_algorithms();

And the error strings using

SSL_load_error_strings();

Then we have created a **SSL_CTX*** named **ctx** and passed the method created for each peer into this context. For each peer, mode for **SSL_CTX_set_verify()**

Is set to **SSL_VERIFY_PEER**.

Verify location is set using: **SSL_CTX_load_verify_locations()**

Which will be used to verify the received certificate.

Next certificate file and key(to check if certificate matches the key) are loaded into the SSL context.

Then a TCP connection is created using `connect()` and `accept()`.

After the TCP connection is established we create a SSL connection using this tcp connection using **SSL_connect()** and **SSL_accept()**.

Then the certificates are verified for the other peer. Suitable errors are generated if the verification process fails.

After the verification succeeds, each peer creates 2 threads for sending and receiving messages.

Messages are send using SSL_write() and received using SSL_read().

To terminate the SSL session and close it both the peers have to send "exit" message.

Demonstrating the chat application:

PEER1 sends certificate : enduserEE1.pem , uses key enduserEE1.key

Certificate to verify : eechain.pem

PEER1 sends certificate : enduserEE.pem , uses key enduserEE.key

Certificate to verify : eechain.pem

Note: eechain.pem is a chain of EE CA and root CA

Here are the screenshots for the chat:

```
harshit@harshit:~/Documents/CNS Prog$ g++ peer.cpp -o peer -lssl -lcrypto -lpthread
harshit@harshit:~/Documents/CNS Prog$ ./peer
Enter your choice :
1 for Server, 2 for Receiver
1
Enter Port No :3000
Enter certificate(enduser) to send :enduserEE1.pem
Enter private key(enduser) path :enduserEE1.key
Enter certificate to verify with :eechain.pem
Enter PEM pass phrase:
4
Certificate validation completed :
Start chatting:

hi I am peer 1
    hi i am peer 2

how are you
    ca va. et toi?

Nice, CNS is fun
exit  exit
```



```

om@om-Lenovo-Ideapad-500-15ISK:~/Downloads$ ./peer
Enter your choice : 
1 for Server, 2 for Receiver
2
Enter Port No :3000
Enter certificate(client) to send :enduserEE.pem
Enter private key(enduser) path :enduserEE.key
Enter certificate to verify with :eechain.pem
Enter PEM pass phrase:
connected
Certificate validation completed :
Start chatting:

    hi I am peer 1

hi i am peer 2
how are you

ca va. et toi?
Nice, CNS is fun

exit
exit

```

Now here we can see that the peers send the request for others certificate after they send their own certificate:

192.168.43.109	192.168.43.88	TLSv1.2	3363 Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
192.168.43.88	192.168.43.109	TCP	66 50460 → 3000 [ACK] Seq=177 Ack=3298 Win=35840 Len=0 TSval=2571756416 TSecr=130304108
192.168.43.88	192.168.43.109	TCP	1514 50460 → 3000 [ACK] Seq=177 Ack=3298 Win=35840 Len=1448 TSval=2571756430 TSecr=130304108
192.168.43.88	192.168.43.109	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]

Here we can see that the message are transfered in an encrypted fashion:

0000	dc 53 60 e5 9e 26 68 14	01 10 f5 f5 08 00 45 00	·S`··&h· ·····E·
0010	01 50 ae 34 40 00 40 06	b3 5d c0 a8 2b 6d c0 a8	·P·4@·@· ·]··+m·
0020	2b 58 0b b8 c5 1c ac f5	43 c3 b3 2c 6b c8 80 18	+X····· C··,k··
0030	01 5c 99 0f 00 00 01 01	08 0a 07 c4 97 f6 99 4a	·\····· ·····J
0040	20 b9 17 03 03 01 17 06	11 1a 29 7a 4d 1a 62 54	····· ··)zM·bT
0050	24 98 12 1b 9c 3e 97 5e	92 e0 40 c6 51 04 59 05	\$····>·^ ··@·Q·Y·
0060	b9 27 72 6f f3 db f4 f8	69 f9 1b e3 cc e9 56 7a	·'ro···· i····Vz
0070	c3 52 2d 90 65 39 cc ff	95 17 53 3c 68 7c 6a a6	·R·-e9·· ··S<h j·
0080	03 c3 c0 9b 3c e3 b4 3c	ec c6 98 78 00 16 cc ca	····<··< ···x····
0090	06 b1 58 68 3b e4 b5 a8	0a e0 83 f6 fb ff 51 39	··Xh;··· ·····Q9
00a0	51 62 ff b7 cc a8 49 51	24 08 df 35 d2 be be c9	Qb···· IQ \$··5····
00b0	81 31 33 ee 1b 8e b6 8d	29 90 68 ca b3 32 41 75	·13·····)·h··2Au
00c0	63 a5 aa 63 e4 44 83 92	d1 6f ef 78 9f eb 4d 4e	c··c·D·· ·o·x··MN
00d0	68 93 43 cb 86 21 7b 4e	20 5a 42 13 f5 ff 94 c2	h·C··!{N ZB····
00e0	7c b4 ce bb 43 54 db cd	ce 58 37 cb 40 45 b2 18	···CT·· ·X7·@E··
00f0	bd 6c c7 c3 2a 46 b7 87	31 30 36 2a 15 11 69 7a	·1··*F·· 106*··iz
0100	2b 36 e5 4f 43 a0 17 06	79 0a 7b 17 a3 59 37 26	+6·0C··· y·{··Y7&
0110	88 d1 5d 57 86 0c 1a 5f	05 1c 6f e4 c8 5b bf bc	··]W··_ ··o··[··
0120	77 94 73 db 2e 62 ed 75	c4 8b ac 5d 6e 2b c6 38	w·s·.b·u ···]n+·8
0130	b3 3f 9a cc 11 7f 94 f1	be 64 c0 ac 80 6f 69 55	·?····· ·d··oiU
0140	2d 49 eb 37 d0 13 65 c9	22 af 16 d6 d4 2a 86 40	-I·7··e· "····*·@
0150	44 d4 b4 a4 f7 ef a6 bb	df 91 49 0f 11 a7	D····· ··I···