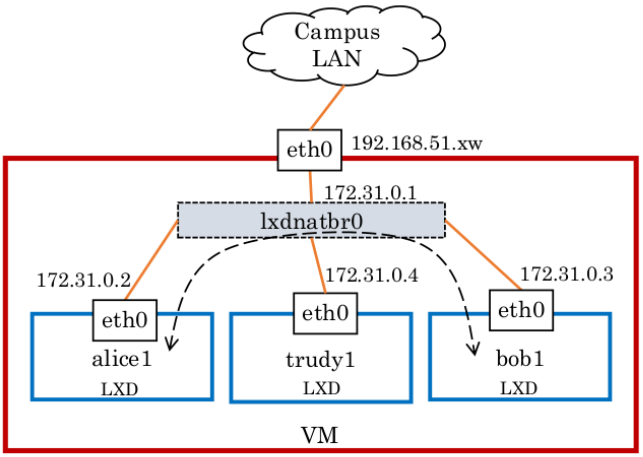
Assignment 6: Secure chat using openssl and MITM attacks

It's a group project with Max of 3 students per group playing the roles of Alice/Bob/Trudy! Check Appendix A to know your group details.

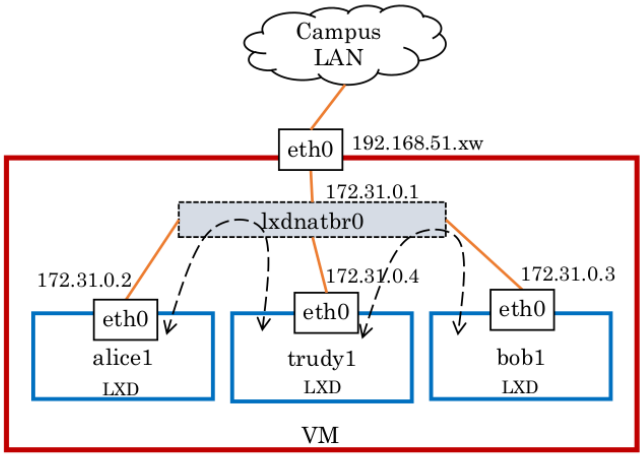
In this programming assignment, your group will implement a secure peer-to-peer chat application using openssl in C/C++/Python and demonstrate how Alice and Bob could chat with each other using it. Plus you will also implement evil Trudy who is going to intercept the chat messages between Alice and Bob by launching various MITM attacks.

Setup:

Each group will be given a dedicated QEMU-KVM VM with the IP address (192.168.51.xw) in NeWS lab's cloud for completing this assignment. Refer Appendix B for some help on how to work with containers. The VM runs three LXD containers (one LXD container each for Alice, Trudy and Bob in the VM provided) which are configured in a star topology i.e., with the switch at the center, Alice, Bob, Trudy are connected to switch ports. **Make a note of hostnames assigned to Alice, Trudy and Bob from your VM.** DNS is already set up properly so ping using hostnames from Alice to Bob and vice-versa works. Under normal circumstances, Trudy does not come in the traffic forwarding path between Alice and Bob. But when DNS is poisoned by Trudy, all traffic between Alice, and Bob is intercepted by Trudy as the MITM attacker. You can also launch a similar MITM attack using ARP cache poisoning.



When DNS is secure



When DNS is poisoned by trudy1

Please upload ssh public key to your github profile and share your github user-id with TAs to get access to your group's VM through IITH's Wireguard VPN (if you are not on the campus). The three LXDs are reachable from inside the VM, and cannot be reached from campus LAN.

Task 1: (10M)

Use the OpenSSL commands to create a root CA certificate (V3 X.509 certificate, self-signed using 512-bit ECC Private Key of the root), a certificate of Alice (issued i.e., signed by the root CA) and a certificate of Bob (issued by the root CA). Ensure that you provide realistic meta-data while creating these X.509 V3 certificates like values for CN/SAN, OU, L, Country, etc of your choice with appropriate key usage/constraints. Save these certificates as root.crt, alice.crt and bob.crt, save their CSRs and key-pairs in .pem files and verify that they are valid using openssl. You can complete this task either on the VM provided (recommended) or on your local machine.

How to communicate among LXDs and between VM and LXD?
You can use the Linux based commands like SCP for transferring the files like CSRs and certs. You have to be sure about the integrity and clearly, show that the files transferred are indeed sent by the intended sender and received by the intended receiver. For example, if you send the CSR to the Signing Authority (root CA), then the signing authority should be able to verify that it is sent by the intended sender and similarly when receiving the certificate back it should be verified that it is indeed signed and sent by the actual authority. You can use signing and verification concepts applied in the Openssl tutorial for this.

Task 2: (20M)

Write a peer-to-peer application (`secure_chat_app`) for chatting which uses TLS 1.3 and TCP as the underlying protocols for secure and reliable communication. *Note that the* `secure_chat_app` *works like HTTPS except that here it's a peer-to-peer paradigm where Alice plays the role of the client and Bob plays the role of the server (half-duplex communication)*. The same program should have different functions for server and client code which can be chosen using command line options "-s" and "-c <serverhostname>" respectively. Feel free to define your own chat headers (if necessary) and add them to the chat payload before giving it to TLS/TCP. Make sure that the application uses only the hostnames for communication between Alice and Bob but not hard-coded IP addresses (refer `gethostbyname(3)`). The application should perform the following operations:

a) Establish a TCP connection between Alice and Bob. Bob starts the app using "`secure_chat_app -s`", and Alice starts the app using "`secure_chat_app -c bob1`"
b) Alice sends a *chat_hello* message to Bob and Bob replies with a *chat_reply* message. It works like a handshake at the application layer. Note that these messages are sent in plain-text. Show that it is indeed the case by capturing pcap traces at Alice-LXD/Bob-LXD.
c) Alice initiates a secure chat session by sending out a *chat_STARTTLS* message and getting *chat_STARTTLS_ACK* from Bob. Your program should load the respective private key and certificate for both Alice and Bob. Furthermore, each of them should have pre-loaded the certificate of the root CA in their trust stores.
    i) For example, if Alice sends a *chat_STARTTLS* message to Bob, upon parsing

the message, Bob initiates replies with *chat_STARTTLS_ACK.* Upon parsing this ACK from Bob, Alice initiates **TLS 1.3** handshake by first sending a *client_hello* message as we discussed in the TLS lesson.

ii) Alice gets the certificate of Bob and verifies that. She also provides her certificate to Bob for verification. So, Alice and Bob use their certificates to perform mutual aka two-way authentication.

d) Upon establishing a secure TLS 1.3 pipe, it is used by Alice and Bob to exchange their encrypted chat messages. Show that it is indeed the case by capturing pcap traces at Alice-LXD/Bob-LXD.

e) Either of them sends a *chat_close* message which triggers closure of TLS connection and finally TCP connection.

Task 3: STARTTLS downgrade attack (20M)

Downgrade attack by Trudy by blocking the chat_STARTTLS message from Alice (Bob) to Bob (Alice).

a) If Alice receives *chat_STARTTLS_NOT_SUPPORTED* message after sending *chat_STARTTLS* to Bob, it is assumed that Bob does not want to have secure chat communication.

b) In this attack, Trudy blocks *chat_STARTTLS* from reaching Bob and sends a fake reply message *chat_STARTTLS_NOT_SUPPORTED* to Alice and thereby forcing Alice and Bob to have unsecure chat communication for successfully intercepting their communication. Show that it is indeed the case by capturing pcap traces at Trudy/Alice/Bob LXDs.

c) You need to write a program (`secure_chat_interceptor`) to launch this downgrade attack (`-d` command line option) from Trudy-LXD. For this task, you can assume that Trudy poisoned the /etc/hosts file of Alice (Bob) and replaced the IP address of Bob (Alice) with that of her. It's a kind of DNS spoofing for launching MITM attacks. In this attack, Trudy only plays with *chat_STARTTLS* message(s) and forwards the rest of the traffic as it is.

To poison /etc/hosts file of Alice, Bob containers, use the following command from inside the VM

```
bash ~/poison-dns-alice1-bob1.sh
```

To revert back the /etc/hosts file of Alice, Bob containers, use the following command from inside the VM.

```
bash ~/unpoison-dns-alice1-bob1.sh
```

To start a downgrade attack, start the secure chat interceptor program using the following command from inside the Trudy LXD.

```
./secure_chat_interceptor -d alice1 bob1
```

Task 4: (30M)

Active MITM attack by Trudy to tamper the chat communication between Alice and Bob. For this task also, you can assume that Trudy poisoned the /etc/hosts file of Alice (Bob) and replaced the IP address of Bob (Alice) with that of her.

a) Also assume that Trudy hacks into the server of root CA and issues fake/shadow certificates for Alice and Bob. Save these fake certificates as fakealice.crt and fakebob.crt, save their CSRs and key-pairs in .pem files and verify that they are indeed valid using openssl.
b) Rather than launching the STARTTLS downgrade attack, in this case Trudy sends the fake certificate of Bob when Alice sends a client_hello message and vice versa. This certificate is indeed signed by the root CA, so it's verification succeeds at Alice. So, two TLS 1.3 pipes are set up: one between Alice and Trudy; the other between Trudy and Bob. Trudy is now like a malicious proxy and decrypts messages from Alice to Bob (and from Bob to Alice) and re-encrypts them as-it-is or by altering message contents as she desires! Show that it is indeed the case by capturing pcaps at Trudy/Alice/Bob LXDs.
c) Enhance the secure_chat_interceptor program to launch this active MITM attack from Trudy-LXD.
   To start the MITM attack, start the secure chat interceptor program using the following command from inside the Trudy LXD.
   `./secure_chat_interceptor -m alice1 bob1`

When you build the tar file with filename as <RollNoX|RollNoY|RollNoZ>.tgz, have separate subfolders for Alice, Bob, Trudy and the data (i.e., the certificates, keys, pcaps, etc). Each of Alice/Bob/Trudy folders should have their own Makefiles with the targets described in README instructions file.

Deliverables in GC as a tar ball:

- **Readable Report cum Design Document enumerating steps followed with screenshots for each of the important steps (10M)**
  - **Details of your chat protocol like its headers and typical message flow. For example, HTTP uses GET/POST/OK methods for message flow between client and server.**
  - **Details on how various MITM attacks are realized by Trudy**
  - **Credit Statement (2-pager): share an accurate and detailed description of each of the group member's contributions to the assignment in terms of coding, report writing, bug fixes, etc.**

The report should be self sufficient to understand what your group has done. You can add screenshots to show the working of the system. All the tools and softwares used should be mentioned with references. Reports should also list the important code snippets with explanations. Simply attaching code without any explanation will not receive credits

- **README file, Pcap traces collected with appropriate names and all Source code (well documented like in GitHub - openssl/openssl: TLS/SSL and crypto library), keys, certificates, etc in appropriate folders (10M)**

PLAGIARISM STATEMENT <Include it in your report>

We certify that this assignment/report is our own 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, packages, datasets, 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/project in any other course lab, 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 plagiarized 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 honor violations by other students if we become aware of it.

Names:
Date:
Signature: <keep your initials here>

References:

1. **OpenSSL Cookbook: Chapter 1. OpenSSL Command Line (feistyduck.com)**
2. **/docs/man1.1.1/man3/index.html (openssl.org)**
3. **OpenSSL client and server from scratch, part 1 – Arthur O'Dwyer – Stuff mostly about C++ (quuxplusone.github.io)**
4. **ssl — TLS/SSL wrapper for socket objects — Python 3.9.2 documentation**
5. **Secure programming with the OpenSSL API – IBM Developer**
6. **Simple TLS Server - OpenSSLWiki**
7. **The /etc/hosts file (tldp.org)**
8. **PowerPoint Presentation (owasp.org)**
9. **SEED Project (seedsecuritylabs.org)**

The assignment will be evaluated by the TAs on a scheduled date and time. If you do not appear for this offline evaluation/viva, the assignment will be treated as not submitted.

Late Submission Policy: 10% penalty for each late day beyond the buffer days.

Bonus Marks (30 M): In this assignment, you emulated DNS poisoning by running **`poison-dns-alice1-bob1.sh script`** written by TAs to manipulate the entries in the /etc/hosts file. **Instead you need to implement one of the following to get the bonus marks:**

1. You need to first ensure that Alice/Bob sends DNS queries (over UDP) to a local resolver which in turn contacts an emulated DNS infra (root servers and Authoritative Name servers) and gets DNS response. Trudy tampers these responses so that the DNS cache of the resolver is poisoned.
2. ARP cache poisoning where Trudy sends gratuitous fake ARP messages to Alice/Bob. Refer this assignment https://seedsecuritylabs.org/Labs_20.04/Networking/ARP_Attack/ for launching this real MITM attack in your set up.

Appendix A: (Group Details)

Group no.	Member 1	Member 2	Member 3
1	Sai Varshittha Ponnam	Akash Tadwai	Amit Kumar
2	Revanth Rokkam	Arkadeb Ghosh	Divya Pathak
3	Ayan Kumar Pahari	Harinder Kaur	Nilesh Shivanand Kale
4	Gurpreet Singh	K Shiv Kumar	Devang Deviprasad Dubey
5	Pradhumn Kanase	Pallavi Saxena	Kamal Shrestha
6	Ravi Chandra Duvvuri	Kishan Nayanbhai Bhinde	Pratik Abhijeet Bendre
7	Raguru Sai Sandeep	Satvik Padhiyar	KOMMANA VIKAS
8	Nisha M	Priyansha Tiwari	Koustav Choudhury
9	VINTA REETHU	Madhvendra Singh Chouhan	Anwesha Kar
10	Suranjan Daw	Rishabh Dongre	MUDAVATH SHATHANAND SAI
11	Piyush Madhukar Dadgal	Visakh Vijayan	Sampath Kumar Sivampeta
12	Pamuluri Ravi Sankar	PELLURI SRIVARDHAN	TATIPELLY VAMSHI
13	Siddhesh Pratim Sovitkar	Supriya Kumari	Srivathsa L Rao
14	Yogesh Ahirwar	Unnati Dixit	Praveen Chandrahas
15	Bharti Sahu		

Appendix B: Some help regarding the setup

- To login to the VM allocated, type **`ssh ns@192.168.51.xw`** command.
- To list out the containers inside the VM, use **`lxc ls`** command from inside the VM.
- To login to a container, use **`lxc exec <containername> bash`** command from inside the VM.
- To logout from the respective container use **`exit`** command.
- To capture packet traces inside a container (eg. trudy1), use **`lxc exec trudy1 -- sudo tcpdump -i eth1 -nn not tcp port 22`**
- Root file-system of a container (eg alice1) is located as a subtree under its storage pool **`/var/snap/lxd/common/lxd/storage-pools/default/containers/alice1/rootfs`**
- To copy files from local system to remote system:
  `scp <filename> root@<destination-ip>:~/`
  To copy files from remote system to local system
  `scp root@<source-ip>:~/<filename> .`