**ACIT 4630 – Assignment 3 – Transport Layer Security**

TLS, Transport Layer Security, is a standard that allows different applications to communicate with one another securely through the internet. Most web servers these days are using HTTPS, which is built on top of TLS.

Please download and unzip Labsetup.zip files in your SEED VM. Enter the Labsetup folder, and use the docker-compose.yml file to set up the lab environment.

*Take screenshots of your code snippets and important results and explain what you see.*

## TLS handshake

Before a client and a server can communicate securely, several things need to be set up first, including what encryption algorithm and key will be used, what MAC algorithm will be used, what algorithm should be used for the key exchange, etc. These cryptographic parameters need to be agreed upon by the client and the server. That is the primary purpose of the TLS **Handshake** Protocol.

On the client container, navigate to the `volumes` folder and run `handshake.py` code to communicate with a real HTTPS-based web server (e.g. www.bcit.ca (TLS v1.2 or www.google.com TLSv1.3), the address of the server needs to be specified as the command-line argument for the python code.

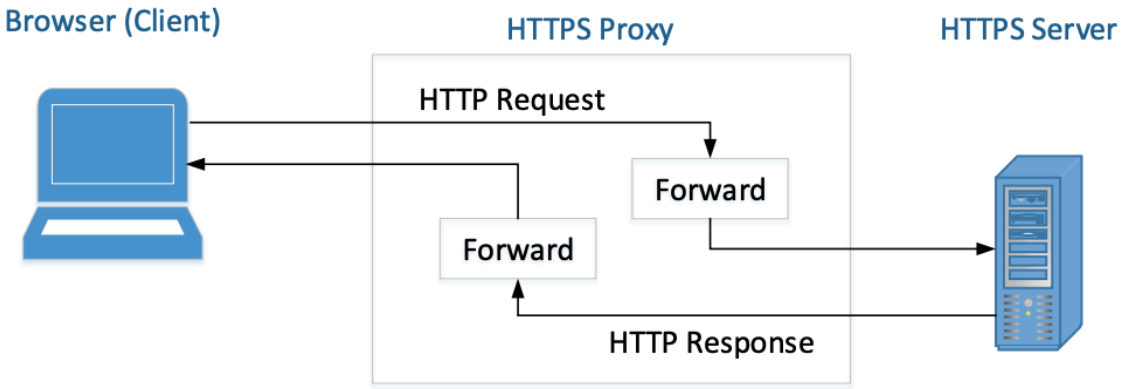
- **Q1**. What is the cipher used between the client and the server?
- Note the server's public certificate in the printed data. Check its validity.
- **Q2**. What is the server's public certificate used for after the client verifies the server's identity and its public key?
- Look in the `handshake.py` code and explain the purpose of `cadir = '/etc/ssl/certs'`
- On the VM use Wireshark to capture the network traffics during the execution of the TLS handshake. Compare the handshake process in TLS v1.2 and v1.3
    - To find the correct interface for the container network on Wireshark, you can do one of the following:
        - Use `ifconfig` on the VM to list network interfaces. Look for the IP address 10.9.0.1, that the IP address assigned to

our VM in the new network created to connect the VM and the containers.
        - Use `docker network ls` command on the VM to find out the network ID. You could find the network name in the compose file.
    - Note the TCP handshake before the TLS handshake. TLS runs on top of some reliable transport protocols (e.g., TCP).

**A Simple HTTPS Proxy**

TLS can protect against the Man-In-The-Middle attack, but only if the underlying public-key infrastructure is secured. In this task, we will demonstrate the Man-In-The-Middle attack against TLS servers if the PKI infrastructure is compromised, i.e., some trusted CA is compromised or the server's private key is stolen.

We will implement a simple HTTPS proxy which is actually a combination of the TLS client and server programs. To the browser, the TLS proxy is just a server program, which takes the HTTP requests from the browser (the client) and returns HTTP responses to it. The proxy does not generate any HTTP responses; instead, it forwards the HTTP requests to the actual web server and then gets the HTTP responses from the webserver. To the actual web server, the TLS proxy is just a client program. After getting the response, the proxy forwards the response to the browser, the real client.




Note that a company can also use such proxy to decrypt and inspect the encrypted information going in/out (TLS Inspection). This is basically a man-in-the-middle attack against their own user!

- Use the self-signed CA you created in the PKI lab to create a (fake but valid) certificate for www.example.net and put the certificate and the key in the `volumes/server-certs/` folder inside the LabSetup folder (name them proxy.crt and proxy.key)
- Run `tls_proxy.py` on the proxy container to simulate using a fake certificate for www.example.net to capture the traffic between the browser and www.example.net
    - You need to provide the web server name as a command-line argument
    - The terminal hangs waiting for a connection.
- Update `/etc/hosts` file on the VM to map www.example.net to the IP of the proxy container
    - In real-world, this would be done via a DNS attack
- Due to the change above, on the proxy container, the IP address to www.example.net is also mapped to 10.9.0.143. This is a problem because the proxy needs to communicate with the actual web server. We have to update`/etc/resolv.conf` file on the proxy container (not the one on the VM). The file has one or multiple `nameserver` entries. Change the first one to 8.8.8.8, which is the public DNS server provided by Google. (You can also use other public DNS servers)
- On the browser on the VM, navigate to https://www.example.net .
- Go back to the proxy container and show that your proxy container is seeing the communication (even though it was supposed to be encrypted!) in the same command window when you ran the tls_proxy.py code
    - You might need to add the certificate for the root CA to the browser if not already there from the PKI lab

BONUS: Repeat these steps to intercept data with a website that requires login, and then use your MITM proxy to steal the password. Many popular servers, such as facebook, have complicated login mechanisms, so feel free to find a server that has simple login mechanisms. Please remember to hide your password in your lab report if you are using a real password.

**Submission:**

Submit your report, with the screenshots of your steps and answers to the questions above, to the Assignment 3 dropbox on the Learning Hub before the Week 9 class.