Task 3 - Transport Layer Security Walkthrough

Since your proxy is based in Docker, you need to get into the docker container to perform these steps.

But before we go into this. You'll notice that this container has a port mapping for only 80. If you type `docker ps -a`, you'll notice that there is a 80->80 mapping. This means port 80 on the *host* will be mapped to port 80 on the *container*. This is good, but we need an additional port because we're now adding HTTPS capabilities. HTTP runs on 80 as you can see here. HTTPS runs on 443. And so, we need another port mapping otherwise, we won't be able to send traffic to the proxy. This could only be done through other containers on the same Docker network. Different story for a different time. The point is that we will need to remake the container and add this port mapping.

Sadly, Docker does not provide us a way to dynamically add port mappings easily, and so, we need to remake the container.

You can run:

```
docker rm -f proxy
```

And then run

```
docker run -d --name proxy --hostname proxy --network website_proxy_network -p 80:80 -p 443:443 -v /etc/nginx/nginx.conf:/etc/nginx/nginx.conf nginx
```

This command takes some prior knowledge of Docker, but I'll explain the pieces here.

1. --name proxy : Gives the name "proxy" to the container
2. --hostname proxy: Gives the hostname "proxy" to the container. This the network name.
3. --network website_proxy_network: Attaches the container to the specific Docker network to have connectivity to the website.
4. -p 80:80: maps port 80 on the host to port 80 on the container
5. -p 443:443: maps port 443 on the host to port 443 on the container.
6. -v /etc/nginx/nginx.conf: /etc/nginx/nginx.conf: This will mount the configuration file on the left (source) to the path/file specified on the container (destination). This mapping will make your configurations **persistent**. Meaning that when you delete the container or stop the container, the changes will not be lost and will be available on the host machine from at the *source* location.
7. nginx: This refers to the Docker image that this container will be based off of.

Now, you have an nginx container that can be contacted through port 80 and 443. We can now begin securing our proxy.

So first thing's first, let's get into the container:

```
docker exec -it bash proxy
```

Now, you have a command line into the container. Next, you'll want to install certbot and an extra python3-based version for certbot.

```
apt-update -y && apt-get install certbot && apt-get install python3-certbot-nginx
```

This will download all the needed components of certbot to make this extremely easy.

Now, depending on how you did this, certbot may use your /etc/nginx/nginx.conf or the default.conf file in /etc/nginx/conf.d/default.conf file. It will take care of modifying the file for you. The only thing that you need to add is the server_name directive to the server listening on 80 and 443.

```
server {
  listen 80;
  server_name teamX.umlcyber.club;

  ...
}
```

Now, you can save this file and then verify the syntax of your configuration.

```
nginx -t && nginx -s reload
```

And here is the part that we've been waiting for, using certbot to secure our communications.

```
certbot --nginx -d teamX.umlcyber.club
```

The -d flag means that you're trying to get a certificate for a specific domain; you can do this for multiple domains in the same line, but you don't need to do that right now. Once you enter the email and not send your IP to the data collectors, you will see a congratulations message that will tell you that the communications to this proxy are now secured through HTTPS.

Now you can simply access your website through https://teamX.umlcyber.club and you can see the lock on the left of the search bar!