Лабораторна робота виконана на основі клієнт-серверної програми для лабораторної роботи sensetive_information.

Протоколом шифрування каналів був обраний TLS версії 1.2 (TLSv1_2). Це найсучашніша версія і найкращий вибір для максимального захист. В цій версії були заборонені старі криптографічні хеш-функції MD5 і SHA-1, а саме їх замінили на SHA-256.

Для цього спочатку згенеруємо сертифікат за допомгою *openssl*:

```
(.venv) [jetraid@jetraid-pc password_storage]$ openssl req -x509 -newkey rsa:409
6 -nodes -out cert.pem -keyout key.pem -days 365
Generating a RSA private key
.....................................................................
.....................................................+++
.....................................................................
.++++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:UA
State or Province Name (full name) [Some-State]:Kyiv Oblast
Locality Name (eg, city) []:Kyiv
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:KPI FICS
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:
(.venv) [jetraid@jetraid-pc password_storage]$
```

Після цього спробуємо додати його до нашого веб застосунку використовуючи біліотеку *ssl* для створення контексту для *Flask app.*

```python
if __name__ == '__main__':
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
    context.load_cert_chain('./data/cert.crt', './data/key.pem')
    app.run(debug=False, ssl_context=context)
```

Нові версії фласку підтримують ssl нативно, тому просто передаємо контекст.

Як бачимо для створення контексту потрібно вказати шлях до файлів з сертифікатом та ключем.

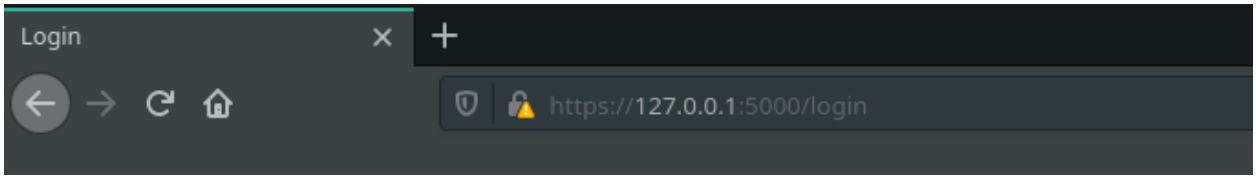
Після продіяного можна спробувати запустити застосунок, але ми всеодно не отримаємо бажаного результату, оскільки сертифікат повинен бути *trusted.* Тобто таким якому довіряє браузер, оскільки це *self-signed* сертифікат то найпростіше що ми можемо зробити це додати його до списку корньових довірених сертифікатів.

```
[jetraid-pc password_storage]# cat cert.crt  >> /etc/ssl/certs/ca-certificates.c
rt
[jetraid-pc password_storage]#
```

Після цього вже можна спробувати запустити застосунок.

```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a produc
nt.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on https://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Можемо спостерігати що тепер він спілкується по протоколу HTTPs.



І браузер вказує що сайт зашихений, проте й повідомляє що сертифікат само підписаний.

Щодо зберігання сертифікату та ключа. Сертифікат є публічною інформацією, тобто немає різниці де його зберігати. А ключ потрібно захистити. Найчастіше це роблять таким чином: створюють окремого користувача (або просто використовують root), або групу яка має права на читання приватних ключів якими користується сервер, і змінюють права на доступ до нього за допомогою *chmod.* В нашому випадку сертифікат та ключ зберігаються поряд із застосунком у директорії *data.*