1 Fonctionnement du programme

Avant toute chose, veillez à bien télécharger les bibliothèques nécessaires dans le dossier "requirements.txt" et à lancer le serveur, via le fichier web_server.py.

Ensuite, le programme est lancé à partir du fichier main.py. À partir de là, le programme vous guidera à travers son fonctionnement à l'aide d'inputs et de retours visuels.

D'abord, il vous faut rentrer vos identifiants UNILIM afin d'accéder au service. Ensuite, vous choisirez entre la création d'une attestation (1) ou la vérification d'une attestation (2).

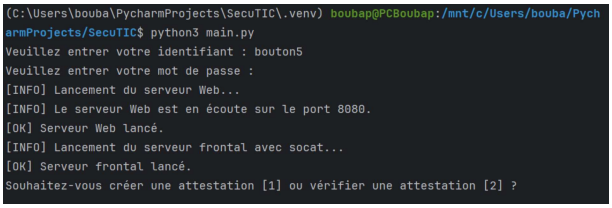

FIGURE 1 – Le choix entre création et vérification

1.0.1 Création d'une attestation

Vous devrez rentrer un nom, prénom et un intitulé de certification afin de créer votre certification. Par la suite, le serveur s'occupera de générer une attestation à partir des informations données. Comme demandé, il commencera par concaténer les informations NOM|PRENOM|INTITULE, qu'il signera et ajoutera à la certification dans un QR Code. De plus, il va également créer un timestamp à partir de ces informations concaténées, et complétées pour atteindre 64 caractères. Il ajoutera par la suite la concaténation suivie du timestamp à l'intérieur de la certification, par stéganographie. L'attestation est désormais créée !

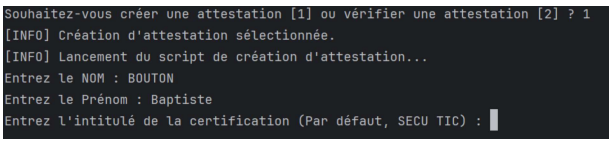

FIGURE 2 – Création d'une attestation

1.0.2 Vérification d'une attestation

La vérification est d'autant plus simple : il ne vous sera demandé qu'un nom et un prénom (afin d'éviter de taper le chemin entier de l'image...). Chaque attestation possède le schéma de nom suivant : attestation_NOM_Prénom.png. Ainsi, vous pouvez facilement adapter le "NOM" et le "Prénom" pour correspondre au chemin souhaité. Après cela, le programme s'occupera de vérifier la signature présente dans le QR Code. Il récupérera

aussi les informations dissimulées par stéganographie et, à partir de ces informations, va pouvoir recréer un fichier timestamp request (.tsq), et va utiliser le timestamp pour recréer le fichier originel timestamp (.tsr). C'est avec ces fichiers qu'il pourra vérifier l'authenticité du timestamp auprès de l'autorité d'horodatage. Il indiquera finalement si l'attestation est authentique ou erronée.

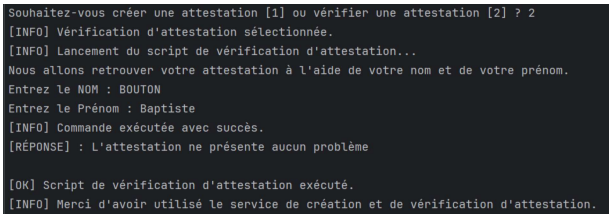

FIGURE 3 – Vérification d'une attestation

2 Analyse de risques

2.1 Actifs primaires

— **Attestations numériques** : ces fichiers ne doivent être ni compromis, ni perdus, car ils sont la source de confiance entre les tiers les ayant demandées et leurs potentiels clients.

— **Confiance dans le système de vérification** : les utilisateurs doivent pouvoir s'appuyer sur le résultat donné par notre service de vérification. Une perte de fiabilité (erreurs dans les réponses) rend l'outil inutilisable.

— **Autorités d'horodatages et de certifications** : les autorités externes qui nous permettent d'obtenir des certifications ou des horodatages se doivent d'être elles-mêmes des tiers de confiance sans lesquels l'architecture de notre projet s'effondre.

2.2 Actifs de support

— **Serveur web** : il permet de faire fonctionner l'ensemble du logiciel de création et de vérification, sa compromission rend alors l'ensemble du programme inutilisable.

— **Base de données** : elle stocke les informations liées aux attestations et à leur vérification. Des problèmes d'accès ou d'intégrité des données peuvent entraîner des résultats incorrects, ou, simplement, des pertes de données (et donc une inutilité du logiciel en lui-même...). Dans un cadre réel, une non-fiabilité de la base de données pourrait être la source de problèmes d'ordre juridiques, notamment liées aux contraintes RGPD de la protection des données des utilisateurs.

— **Module d'horodatage** : composant essentiel pour prouver l'antériorité. En cas d'indisponibilité ou de désynchronisation, la crédibilité des données temporelles est

compromise, et il est alors impossible d'utiliser les attestations, car elles ne peuvent plus être vérifiées. Dans le cas de notre projet, cet actif est externe à notre CA factice, et donc en dehors de notre portée.

— **Tunnel TLS** : le serveur frontal fonctionnant comme un reverse proxy est une cible potentielle à frapper lors d'attaques. Dans notre cas, précisément, l'utilisation d'un simple tunnel TLS/TCP vers TCP à l'aide de socat manque cruellement de protection et représente sans doute un risque majeur lors de la transmission de données sensibles entre le serveur frontal et le serveur web.

— **Clés cryptographiques** : même si elles sont fictives dans ce projet, elles modélisent une dépendance à la sécurité de l'authentification. Une mauvaise gestion pourrait compromettre l'ensemble de la sécurité du système.