UQAC
Université du Québec
à Chicoutimi

TRAVAIL PRATIQUE #2

Automne 2024

Sécurité des Réseaux et du Web
(8INF138)

Groupe 01

Local : P1-4270

Département d'Informatique et de Mathématiques (DIM)

Florentin Thullier

Courriel : **fthullie@uqac.ca**

Bureau : **P3-5040**

Téléphone : **(418) 545-5011 poste 2355**

Introduction

Au cours de la session, nous avons vu, ou nous allons voir, plusieurs notions en lien avec la cryptographie et l'authentification qui sont importantes pour vous de maîtriser. Ainsi, ce travail a pour objectif de vous permettre de mettre en pratique ces éléments.

Énoncé

Pour ce travail, vous allez devoir réaliser **en équipe de deux MAXIMUM**, une application de gestion des mots de passe très simplifiée en mode console (CLI).

Pour ce faire, je vous laisse le choix du langage de programmation que vous pouvez utiliser parmi les options suivantes :

- Python *(version >= 3.8) pas de notebook jupyter !*
- JavaScript/TypeScript *(avec le runtime que vous voulez, p.ex. node, bun, etc.)*
- Golang
- Rust
- C++ *(version >= 11)*
- Java *(version >= 1.8)*
- C#

Attention : Vous pouvez, cette fois, utiliser des librairies hors de la librairie standard du langage que vous allez choisir et ceci est même encouragé. Rappelez-vous que vous n'êtes pas des cryptographes accomplis et qu'en ce qui concerne la cryptographie, un des concepts importants est de ne jamais redévelopper sa propre solution !

Note : Si vous n'avez pas une grosse expérience en programmation, choisissez un langage qui va vous faciliter la vie, ne vous lancez pas dans un programme en C++ par exemple, parce que vous en faites dans un autre cours. Cependant, je ne vous empêcherai pas de le faire si c'est vraiment ce que vous voulez faire. Par ailleurs, ayez en tête que vous n'aurez qu'une seule séance d'aide en classe et que je ne peux pas tous vous aider individuellement, vous êtes 80 étudiants ! Mais cela ne veut pas dire que vous ne devez pas m'envoyer un courriel si vous avez malgré tout des difficultés. L'objectif est que tout le monde parvienne à compléter son TP !

Fonctionnement de la CLI

Votre programme doit fonctionner de la manière suivante :

- Enregistrer un nouvel utilisateur sur votre application :

```
$> passmanager -r <username>

/!\ Enter <username> master password: <master_password>
```

- Ajouter un nouveau mot de passe pour un utilisateur enregistré donné. *Attention, si l'utilisateur n'a pas été enregistré au préalable, il ne devrait pas être possible d'ajouter le mot de passe.*

```
$> passmanager -u <username> -a <label> <password>

/!\ Enter <username> master password: <master_password>

--> password <label> successfully saved!
```

- Afficher un mot de passe donné pour un utilisateur donné. *Attention, si l'utilisateur n'existe pas, c'est-à-dire, qu'il n'a pas été enregistré au préalable, ou que le label n'existe pas dans la base de données, vous devez afficher un message d'erreur.*

```
$> passmanager -u <username> -s <label>

/!\ Enter <username> master password: <master_password>

--> password <label> is: <plain_text_password>
```

Consignes

1. Vous devez utiliser une **base de données SQLite** pour stocker les données (informations utilisateurs, mots de passe, *etc.*). **Le contenu de cette base de données doit impérativement être chiffré**. Pour ce faire, vous pouvez utiliser les variables d'environnement *via* un fichier **`.env`** pour définir et stocker le mot de passe de la base de données.

2. Le mot de passe principal (**`master_password`**) des utilisateurs doit être chiffré en utilisant l'algorithme **SHA-256**. Vous devez également produire un **sel cryptographique de 16 octets** qui sera concaténé au mot de passe principal, mais aussi stocké indépendamment dans la base de données. Pour ce faire, les deux éléments doivent être sauvegardés dans la base de données en **base64** tels que :

    - **`password = base64(sha256(password + salt))`**
    - **`salt = base64(salt)`**

3. Les mots de passe sauvegardés doivent être chiffrés en utilisant **AES-256**. La version chiffrée des mots de passe doit être enregistrée en **base64** dans la base de données.

4. **La clé de chiffrement AES-256** doit être générée à partir du mot de passe principal de l'utilisateur (**`master_password`**) grâce à l'algorithme **PBKDF2.** Pour ce faire, vous devez utiliser un **sel cryptographique de 16 octets**, **100000 itérations** et **HMAC-SHA256** comme **algorithme de dérivation**.

Note : L'énoncé de ce TP laisse volontairement des zones floues pour vous inciter à chercher par vous-même et à vous faire vous poser des questions. Au besoin, n'hésitez surtout pas à me poser des questions sur les points pour lesquels des précisions auraient besoin de vous être apportées. Je suis là pour ça !

Remise

Vous devrez remettre ce second travail dans **une archive « .zip »** que vous nommerez **« password-manager.zip »** dans l'espace de dépôt prévu à cet effet sur Moodle, et ce, avant le cours du lundi 25 novembre 2024, soit **au maximum à 23h59 le dimanche 24 novembre 2024**.

L'organisation de votre dossier de rendu doit respecter la structure de fichiers comme montré ci-dessous :

```
password-manager/
├── .env
├── .env.example
├── db/
│   └── data.sqlite
├── src/
│   └── **/*.py || **/*.js || **/*.go || **/*.ts || **/*.java ...
└── README.md
```

- Le fichier **`.env`** doit contenir votre variable d'environnement qui définit le mot de passe de la base de données SQLite. **Ce fichier n'est pas à rendre dans l'archive de remise. Il doit être gardé secret, uniquement pour vous.**

- Le fichier **`.env.example`** doit contenir un exemple de la définition de votre variable d'environnement qui définit le mot de passe, de sorte que je n'ai pas à chercher comment vous avez nommé votre variable pendant des heures lors de la correction.

- Le dossier **`db/`** doit contenir votre base de données SQLite. **Vous n'avez pas à rendre ce dossier dans l'archive de remise. Cependant, faites bien attention à avoir, dans votre application, un bloc d'instruction permettant de créer une base de données à cet emplacement exact si celle-ci n'existe pas lors de l'exécution de votre programme (car ce sera mon cas lorsque je vais corriger)**

- Le dossier **`src/`** doit contenir toute l'arborescence des fichiers et dossiers que vous allez mettre en place pour votre code.

- Le fichier **`README.md`** doit contenir toutes les informations importantes quant à votre projet. Je vous donne le minimum à inclure, mais sentez-vous libre d'ajouter toutes les informations supplémentaires que vous jugerez pertinentes. Vous pouvez vous référer à https://www.markdownguide.org/cheat-sheet/ pour vous aider à formater convenablement ce fichier.

Le contenu minimal de votre fichier **`README.md`** doit être le suivant :

```
# Travail Pratique #2: Password Manager

## Auteurs
 - Code permanent, nom, prénom
 - Code permanent, nom, prénom

## Compatibilité

Langage - version <!-- p.ex. Python - 3.10 -->

## Utilisation

> incluez dans cette section toutes les instructions nécessaires pour
faire fonctionner votre projet. Quels sont les préalables : est-ce
que je dois installer un environnement d'exécution, si oui, précisez
sa version, donnez le lien de download ; est-ce que j'ai des
dépendances à installer, si oui quelle est la commande pour ce faire
; etc. (supprimez ce commentaire pour votre rendu!)
```

Remarque : Toute autre méthode de remise pour ce travail ne sera pas acceptée. Votre travail sera alors considéré comme non remis. De plus, tout non-respect des consignes de remise entrainera une perte de point.

Bonus

Si vous souhaitez étendre les fonctionnalités qui sont minimalement demandées dans l'énoncé de ce TP, vous êtes libre de le faire comme vous le souhaitez. Idéalement, décrivez les fonctionnalités supplémentaires dans votre fichier **`README.md`**. J'ajouterai autant de points bonus que nécessaire en fonction de la pertinence et de la qualité des fonctionnalités ajoutées jusqu'à concurrence d'un total de 5 points bonus.

Attention : Aucun point bonus ne sera accordé si les exigences minimales de base ne sont pas fonctionnelles. Répondez d'abord aux objectifs du TP, puis ajoutez-y les éléments bonus dans un second temps.

Pénalité pour retard

Tout travail remis en retard sans motif valable sera évalué sur 50%. Une pénalité de 10% additionnels par jour supplémentaire sera appliquée après le premier jour de retard (*p. ex.* 3 jours de retard, le travail est évalué sur 30%).