TP1 : Crypto et authentification

Rendu : dimanche 1er octobre à 23H59

Pensez aussi à lire les instructions décrites plus bas !!!!

Énoncé

Nous avons d'ores et déjà étudié les principes régissant l'authentification et quelques méthodes de cryptographie. Ce TP a pour but de vous faire pratiquer la plupart des éléments vus en cours. Vous allez, au cours de ce devoir, créer une application **console** de gestion des mots de passe.

Cas d'utilisation

Avant toute chose le (ou les) utilisateur(s) de votre application devront s'enregistrer sur votre application avec la commande suivante :

dotnet run -r USERNAME MASTER_PASSWORD

Ensuite, ils pourront ajouter des mots de passe associés à des tags comme ceci :

dotnet run -a USERNAME MASTER_PASSWORD TAG PASSWORD

Évidemment ils doivent pouvoir récupérer les mots de passe avec la commande (cette commande affiche sur la console le mot de passe en clair) :

dotnet run -g USERNAME MASTER_PASSWORD TAG

La suppression, fonction importante de ce type d'application, se fera de la manière suivante :

dotnet run -d USERNAME MASTER_PASSWORD TAG

Pour faciliter la correction, vous devez fournir deux fonctions d'affichage très simples :

dotnet run -t USERNAME doit retourner le mot de passe hashé et le sel cryptographique enregistrés dans la base de données de l'utilisateur au format suivant : SALT:HASH

dotnet run -t USERNAME TAG doit retourner la version chiffrée encodée en Base64 du mot de passe correspondant au tag

Si une opération se déroule comme prévu, et qu'aucun retour n'est attendu (ajout, suppression, etc.) vous devez répondre **OK** en cas de problème écrivez **ERROR**.

Instructions

Il est possible (très probable) qu'une partie des points de votre TP soit attribuée automatiquement par un serveur exécutant une série de tests sur vote application donc pensez à bien respecter les instructions suivantes :

1. Pour ce premier TP nous utiliserons le langage **C#** avec les technologies **.NET Core 2.0**
2. Votre application **doit** utiliser un backend **SQLite** pour le stockage des informations (utilisateurs, mots de passe, etc.)
3. Les mots de passe principaux des utilisateurs doivent être stockés en utilisant les algorithmes **SHA-256 et base64** en respectant la formule suivante :
    a. $Save = base64(SHA256(PASSWORD + SALT))$
    b. Le sel cryptographique doit être de 16 octets et stocké lui aussi en utilisant le format base64 : $SaveSalt = base64(SALT)$
4. Les mots de passe sauvegardés doivent être chiffrés en utilisant AES-256 et la version enregistrée doit être le base64 de la version chiffrée **précédée** de l'IV utilisé.
5. La clé de chiffrement **AES-256** doit être générée à partir du mot de passe en utilisant l'algorithme **PBKDF2** avec un sel cryptographique identique à celui utilisé pour hasher le mot de passe de l'utilisateur, **10000** itérations et un algorithme de dérivation **HMAC-SHA256**.
6. Rappelez-vous la compilation est vitale, votre application **DOIT** marcher alors préférez une application n'implémentant pas toutes les fonctionnalités, mais fonctionnelle plutôt que l'inverse.

Tips

https://docs.microsoft.com/en-us/ef/core/get-started/netcore/new-db-sqlite

https://www.nuget.org/packages/Microsoft.AspNetCore.Cryptography.KeyDerivation/2.0.0