Gestionnaire de mots de passe distribué avec Java RMI

Abdelmouiz bensbai / Reda hamdi

17 juin 2025

Table des matières

Chapitre 1

Introduction

La sécurité des mots de passe est un enjeu majeur dans le monde numérique actuel. Ce projet vise à développer une application de gestion de mots de passe distribuée, permettant aux utilisateurs de stocker et de récupérer leurs mots de passe de manière sécurisée à l'aide de la technologie Java RMI (Remote Method Invocation).

Chapitre 2

Présentation de Java RMI

2.1 Définition

Java RMI (Remote Method Invocation) est une technologie Java qui permet à un objet situé dans une machine virtuelle Java (JVM) d'invoquer des méthodes sur un objet situé dans une autre JVM, potentiellement sur une machine distante. RMI facilite la création d'applications distribuées en masquant la complexité de la communication réseau.

2.2 Fonctionnement général

- **Interface distante :** Définit les méthodes accessibles à distance.
- **Implémentation distante :** Fournit le code des méthodes de l'interface distante.
- **Serveur RMI :** Publie l'objet distant dans le registre RMI.
- **Client RMI :** Recherche l'objet distant dans le registre et invoque ses méthodes.

2.3 Avantages de RMI

- Transparence des appels distants (comme des appels locaux)
- Passage d'objets complexes grâce à la sérialisation
- Gestion automatique des exceptions distantes
- Sécurité configurable via des politiques Java

Chapitre 3

Architecture du projet

3.1 Description générale

Le projet est basé sur une architecture client-serveur :
— **Serveur :** Centralise la gestion et le stockage sécurisé des mots de passe.
— **Client :** Permet à l'utilisateur d'interagir avec le gestionnaire de mots de passe via une interface graphique.

3.2 Diagramme d'architecture

(Insérer ici un schéma illustrant la communication entre le client, le serveur et le registre RMI.)

3.3 Composants principaux

— **Interface distante (RMIInterface) :** Définit les méthodes de gestion des mots de passe accessibles à distance.
— **Serveur RMI :** Implémente l'interface distante et publie l'objet dans le registre.
— **Client RMI :** Se connecte au serveur et utilise les méthodes distantes.
— **Gestionnaire de mots de passe :** Gère le hachage et le stockage local/centralisé des mots de passe.

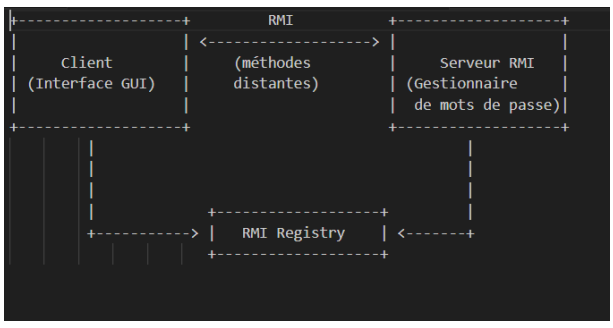

FIGURE 3.1 – architecture

Chapitre 4

Fonctionnalités du projet

4.1 Authentification

— Inscription et connexion sécurisées
— Hachage des mots de passe avec SHA-256

4.2 Gestion des mots de passe

— Ajout, suppression, modification et récupération de mots de passe
— Stockage centralisé et sécurisé

4.3 Sécurité

— Hachage des mots de passe
— Possibilité d'ajouter du chiffrement (AES, etc.)
— Politiques de sécurité Java (fichiers `policy`)

Chapitre 5

Interface Client

5.1 Présentation

Décrivez ici l'interface graphique du client. Ajoutez des captures d'écran si possible.

5.2 Fonctionnalités principales

— Connexion/inscription
— Ajout et gestion des mots de passe
— Affichage des mots de passe enregistrés

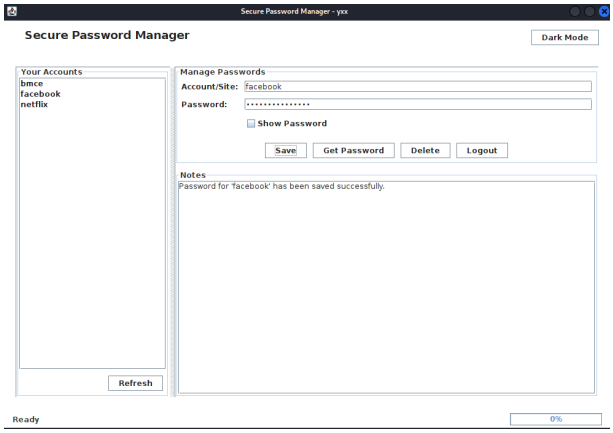

FIGURE 5.1 – client

Chapitre 6

Interface Serveur

6.1 Présentation

Décrivez ici l'interface d'administration du serveur, s'il y en a une. Ajoutez des captures d'écran si possible.

6.2 Fonctionnalités principales

— Supervision des connexions clients
— Gestion des utilisateurs
— Logs et statistiques

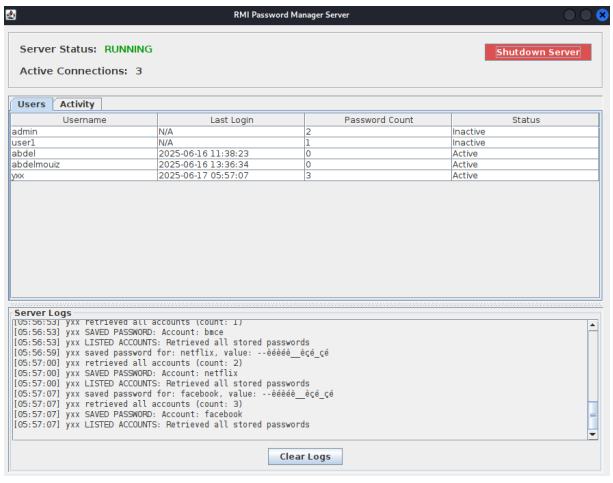

FIGURE 6.1 – server

Chapitre 7

Conclusion

Ce projet met en œuvre une application distribuée sécurisée grâce à Java RMI, illustrant la simplicité et la puissance de cette technologie pour la gestion distante de données sensibles.

Chapitre 8

Annexes

8.1 Extraits de code

```
// Exemple d'interface distante
public interface RMIInterface extends Remote {
    boolean storePassword(String username, String service, String password) throws Re
    String retrievePassword(String username, String service) throws RemoteException;
}
```

8.2 Instructions de compilation et d'exécution

— Compilation : `./compile_and_run.sh compile`
— Lancement du serveur : `./compile_and_run.sh server`
— Lancement du client : `./compile_and_run.sh client`