[Bonus] Partie Intégration

Compétences C2 & C3

Contexte

La boîte à outils développée comporte des différents scripts et programmes en langage C. Dès lors, le but est de rendre l'ensemble cohérent, ergonomique avant de l'enrichir davantage en nouvelles fonctionnalités.

Pour rappel, tous les objectifs décrits dans ce document sont optionnels : ne pas les rendre ne rend pas défaillant, mais ne pas les rendre diminue la note maximale atteignable.

Objectifs

Intégration des programmes

Les programmes `decipher` et `findkey` ne sont pas encore utilisés par les scripts existants.

Créer un script `restore-archive.sh` qui :

1. Accepte un argument un chemin jusqu'à un dossier de destination pour écrire les fichiers récupérés
2. Demande à l'utilisateur quelle archive est à restaurer
3. Détecte les fichiers qui ont été chiffrés
4. Appelle le programme `findkey` pour retrouver la clé de déchiffrement
5. Stocke la clé dans le fichier `archives` au bon endroit
6. Appelle le programme `decipher` pour récupérer le contenu des fichiers

Le script doit respecter les contraintes suivantes :

- Si le dossier de destination n'existe pas, il est créé.
- Le chemin relatif des fichiers restaurés depuis l'archive est conservé dans le dossier de destination.
- Si un fichier déchiffré existe déjà dans le dossier de destination, l'utilisateur confirme son écrasement. Si l'utilisateur ne confirme pas, l'ancien fichier déchiffré est conservé, le nouveau est ignoré.

Le script utilisent les codes de retour suivants :

- 0 si la restauration des fichiers a été réalisée avec succès

- 1 si le dossier `.sh-toolbox` n'existe pas
- 2 si le dossier de destination n'a pas pu être créé
- 3 si la mise à jour du fichier `archives` a échoué
- 4 si un des fichiers n'a pas pu être restauré

Exemple

Le fichier chiffré `exemple1.txt` se trouve dans le dossier `abc`. Nous souhaitons le restaurer dans le dossier de destination `out`. Une fois restauré, le fichier `exemple1.txt` se trouvera au chemin relatif suivant : `out/abc/exemple1.txt`

Amélioration du déchiffrement

Certaines clés de déchiffrement obtenues avec `findkey` ne contiennent pas uniquement des caractères imprimables. De ce fait, la clé ne doit plus être directement stockée dans le fichier `archives` si elle contient des caractères non imprimables. Sauvegarder la clé dans un fichier permet de contourner le problème.

Le chemin vers ce fichier doit être passé en paramètre à l'appel du programme `findkey` après l'argument `-o` (cf. exemple).

Exemple

```
./findkey client1-20250411-1311.tar.gz -o /tmp/KEY
```

La clé retrouvée sera stockée dans le fichier KEY dans le dossier temporaire `/tmp`

Intégration de l'amélioration du déchiffrement

La gestion du fichier `archives` doit être enrichie pour prendre en charge ce scénario. Pour chaque archive importée, le type de clé doit être précisée. Afin d'être rétrocompatible avec les précédentes fonctionnalités, le quatrième élément de chaque ligne indique si la clé est stockée dans le fichier `archives` (« s ») ou si la clé est stockée dans un fichier (« f »).

Exemple du contenu du fichier `archives`

La structure du fichier `archives` sera la suivante :

```
2
client1-20250411-1311.tar.gz:20251104-131504:CleSAE2025:s
client2-20250411-1341.tar.gz:20251104-134407::f
```

Dans l'exemple ci-dessus :

- La clé de l'archive du client 1 est stockée dans le fichier `archives`.
- La clé de l'archive du client 2 est stockée dans un fichier indépendant.

Si la clé est stockée dans un fichier indépendant, ce dernier se trouve dans un sous-dossier du dossier `.sh-toolbox`. Le nom de ce sous-dossier correspond au nom de l'archive associée à la clé, sans l'extension `.tar.gz`.

Exemple de l'arborescence associée

```
.
└── .sh-toolbox
    ├── client2-20250411-1341
    │   └── KEY
    ├── archives
    ├── client1-20250411-1311.tar.gz
    └── client2-20250411-1341.tar.gz
```

Dans l'exemple ci-dessus, la clé est enregistrée dans le fichier KEY.

Amélioration de l'initialisation de l'environnement de travail

En complément des fonctionnalités déjà présentes dans le script `init-toolbox.sh`, il est demandé de :

1. Vérifier la présence des binaires `decipher` et `findkey`
2. Assembler automatiquement les binaires s'ils sont manquants

Les codes de retour seront adaptés pour intégrer ces changements :

- 10 si les fichiers sources n'existent pas
- 11 si le compilateur n'est pas disponible
- 12 si la compilation a échoué

L'arborescence de l'environnement de travail aura la structure suivante :

```
.
├── .sh-toolbox
│   └── archives
├── check-archive.sh
├── decipher
├── findkey
├── import-archive.sh
├── init-toolbox.sh
├── ls-toolbox.sh
├── restore-archive.sh [Bonus]
├── restore-toolbox.sh [Bonus]
└── src
    ├── Makefile [Bonus]
    ├── decipher.c
    └── findkey.c
```

D'autres fichiers peuvent exister, en fonction des réponses apportées aux questions bonus.

Historique des modifications du sujet

Version 1.0.0 – 202511141400

Version initiale du sujet.