

Projet : passe ton hack d'abord !

1 Contexte

Suite à des fuites de données, M. X, hacker inexpérimenté, a récupéré des informations d'authentification concernant différents services web. Son objectif est d'essayer de « craquer » certains mots de passe. Votre objectif est de l'y aider de manière efficace.

Vous disposez de plusieurs fichiers texte, chacun correspondant aux données d'authentification d'une application web.

Les fichiers **tetedamisXX.txt** et **slogramXX.txt** contiennent pour chaque utilisateur dont les informations ont fuité, le login et le mot de passe haché (cf Section 4 pour comprendre de quoi il s'agit) des services web **tetedamis** et **slogram**. Le service web **depensetout** était moins sécurisé et le fichier **depensetoutXX.txt** contient pour chaque utilisateur son login et son mot de passe en clair.

Tous les logins ont ici une taille de 8 caractères.

Les données ont fuité plusieurs fois et le « XX » à la fin du nom du fichier indique le numéro de la fuite.

Une ligne sur deux de chaque fichier correspond à un login et une ligne sur deux à un mot de passe (haché ou non selon le cas). Le login d'un utilisateur précède son mot de passe.

Voici un extrait d'un des fichiers avec des mots de passe hachés :

```
rlisliso
KZWKqna9rUz0/rMZNv+TXbfqAeAtH1RmTiKezkjsJrE=
smallasz
zGuQ6Ct5zkbXvzzY8YPkQkNG1WrU8KOujs1SoOEouVM=
dvincoll
7FxBwSekh0vJzAs8opQ1RiPxFYjvPHGjoUjhsZ+7ed0=
```

Vous pouvez lire les mots de passe hachés des utilisateurs **rlisliso**, **smallasz** et **dvincoll**. Il s'agit respectivement de

- _ KZWKqna9rUz0/rMZNv+TXbfqAeAtH1RmTiKezkjsJrE=,
- _ zGuQ6Ct5zkbXvzzY8YPkQkNG1WrU8KOujs1SoOEouVM=
- _ 7FxBwSekh0vJzAs8opQ1RiPxFYjvPHGjoUjhsZ+7ed0=

Voici maintenant un extrait d'un des fichiers du service **depensetout** :

```
drostern
5e5'+3
xverlin0
bonjour1
mvelparl
0jf+?n)IZTRZ{4:nuP/"<!I>wka,R'C$-<mct&LH
```

Vous pouvez lire les mots de passe en clair des utilisateurs **drostern**, **xverlin0** et **mvelparl**. Il s'agit respectivement de

- 5e5`+3,
- bonjour1
- 0jf+?n)IZTRZ{4:nuP/"<! I>wka,R'C\$-<mct&LH

On sait que la taille des mots de passe peut varier de 6 à 128 caractères et que les caractères autorisés dans ces mots de passe sont les caractères ASCII dont les codes sont compris entre 33 correspondant au point d'exclamation « ! » et 126 correspondant au tilde « ~ ».

On vous fournit également un fichier `french_passwords_top20000.txt` contenant les mots de passe français les plus couramment choisis.

Le hacker vous pose les questions suivantes : est-il raisonnable de tenter une attaque en force brute, c'est à dire de tester tous les mots de passe possibles ? Sinon, peut-on tenter des attaques en force brute partielles ? c'est à dire tester un sous-ensemble de mots générés automatiquement, et si oui quel sous-ensemble ? Existe-t-il d'autres pistes pour casser tout ou partie des mots de passe avec les données dont il dispose ?

2 Travail attendu

Vous devez d'abord répondre à la question du hacker en justifiant votre réponse¹.

Par ailleurs, vous devez imaginer les types de données utiles et les fonctions nécessaires pour que M. X puisse exploiter ces données et idéalement casser quelques mots de passe. Il doit notamment pouvoir :

- fusionner dans une même variable les informations contenues dans les fichiers correspondant à plusieurs fuites d'une même application en éliminant les doublons (même login et même mot de passe) ;
- déterminer si un même login est présent dans plusieurs fuites de données (donc dans les fichiers correspondant à plusieurs applications web) et dans ce cas déterminer si les mots de passe sont identiques ;
- déterminer si un même mot de passe haché est présent dans plusieurs fuites de données et savoir à quels logins ils sont associés ;
- Étant donnée une liste de mots de passe en clair, extraire la liste des couples (application web, login) pour lequel le mot de passe haché associé au login correspond au haché d'un des mots de passe en clair.

À l'aide de ces fonctions et peut-être d'autres, vous devez arriver à craquer certains mots de passe et extraire une liste de triplets (application web,login,mot de passe craqué).

Évidemment, pour vous assurer de la justesse de vos fonctions, vous devrez les tester sur de plus petits jeux de données que celles contenues dans les fichiers eux-mêmes.

Vous pouvez écrire autant de fonctions que nécessaires. Votre code doit être bien structuré, modulaire, lisible et commenté. Les fonctions doivent être écrites dans un ou plusieurs fichiers, le code permettant de tester vos fonctions sur des exemples simples doit être dans un ou plusieurs fichiers spécifique et celui permettant de les utiliser sur les fichiers de données fournis doit également être dans un fichier à part.

Par ailleurs, les fonctions doivent être assez génériques pour pouvoir être utilisées sur n'importe quels fichiers de données respectant le format décrit précédemment et pas seulement sur les fichiers fournis.

3 Organisation et modalités d'évaluation

Le projet est à réaliser exclusivement par binôme. Votre binôme doit être dans votre groupe de TP.

1. Des éléments de réponse ont été fournis lors du dernier cours de technologie du web l'an dernier...

Le projet donnera lieu à deux évaluations :

- Une première note évaluera le travail proprement dit. On attend une archive `login1_login2.zip` où `login1` et `login2` sont les logins des membres du binôme dans l'ordre alphabétique, contenant :
 - tous vos fichiers sources (fichiers suffixés par `.ml`) : vos noms doivent être indiqués en commentaire en début de chaque fichier ;
 - un petit rapport au format pdf qui consiste en un mode d'emploi pour le hacker, répondant aux questions qu'il a posées, lui expliquant à la fois l'utilité des différentes fonctions écrites et la manière de les utiliser (sur les fichiers fournis ou sur d'autres) et lui donnant une idée sur le temps et la place mémoire nécessaires à l'exécution de votre programme en fonction de la taille des données d'entrée.
 - si vous avez utilisé des ressources extérieures pour écrire vos fonctions, vous devez les référencer dans une partie « Ressources ». Si ce sont des livres vous indiquez les références complètes, pour des sites web, vous donnez les URL. Et si vous avez utilisé un outil IA de génération automatique de code, vous devez indiquer le prompt utilisé et la réponse de l'IA. Attention toutefois, vous n'aurez pas accès à une IA le jour du contrôle sur table et ce projet est aussi l'occasion d'entraîner votre intelligence personnelle qui n'a, elle, rien d'artificielle.
- Une deuxième note sera donnée dans le cadre du dernier contrôle de l'année sur table pour lequel une partie du sujet consistera à répondre à des questions sur le projet.

Nous nous réservons le droit de convoquer à un oral autant de groupes que nécessaires pour donner toute précision qui pourrait s'avérer utile sur le travail rendu.

4 Un mot sur le hachage en général et le hachage de mots de passe en particulier

De manière générale, en informatique, le hachage est une technique qui consiste à transformer une donnée (texte, nombre, fichier, etc.) en une séquence de caractères de longueur fixe appelée valeur de hachage ou haché de la donnée. Cette technique est utilisée dans plusieurs contextes : pour assurer un stockage sécurisé des informations d'authentification d'un utilisateur (c'est ce qui nous intéresse ici), pour vérifier l'intégrité de données ou empreintes numériques, assurer l'authenticité de données mais aussi pour construire des structures de données permettant un accès efficace aux données qu'elles contiennent.

La transformation des données se fait via l'application d'une fonction de hachage. Il existe plusieurs approches pour en construire une. Une bonne fonction de hachage doit posséder les caractéristiques suivantes :

- **longueur fixe du résultat de la transformation** : une fonction de hachage prend en entrée une donnée de longueur variable et produit en sortie une valeur de longueur fixe (par exemple, l'algorithme de hachage connu sous le nom de SHA-256 génère un haché de 256 bits)
- **déterministe** : une même donnée en entrée produit toujours la même sortie, ainsi, par exemple, si deux utilisateurs ont le même mot passe, ils auront également le même haché ;
- **difficulté voire impossibilité de retrouver la donnée d'entrée à partir de son seul haché** : les fonctions de hachage sont construites pour être à sens unique. Ainsi disposer du haché d'un mot de passe ne permet pas de calculer le mot de passe lui-même.
- **sensibilité aux modifications des valeurs en entrée** : une légère modification dans les données d'entrée (par exemple la modification d'un caractère dans un mot de passe) entraîne une modification radicale de la sortie. Par exemple, avec une certaine fonction de hachage, le haché du mot « informatique » est `19D4t1gDY2dlulmpaAm5UN8jk169/RtSFqe753zF2oc=` et le haché du mot « informagique » est `Neq/z9tJVZr8bG3dtxF4jxunoK4A7dxtteyypSW5v6NQ=`.

- **minimisation des collisions** : par construction de ces fonctions, il peut exister plusieurs données différentes qui ont le même haché², on dit alors qu'il y a collision ; une bonne fonction de hachage minimise les collisions.

Ainsi pour protéger ses utilisateurs, un site web ne devrait jamais stocker les mots de passe ni en clair ni dans une version chiffrée mais déchiffrable. Il devrait uniquement stocker le résultat de la transformation du mot de passe par une fonction de hachage bien choisie. Lorsque l'utilisateur veut se connecter, il saisit son mot de passe, la fonction de hachage lui est appliquée et c'est le haché qui est comparé aux informations stockées dans la base de données du site web. Ainsi si la base de données des mots de passe est compromise, le mot de passe lui-même ne l'est pas...sauf si l'utilisateur a utilisé le même mot de passe sur un site moins sécurisé...

Ici une seule fonction de hachage est utilisée pour les données manipulées. Le hacker l'a récupérée et elle vous est fournie.

5 Code fourni

On vous fournit dans le fichier `tools.ml` les fonctions suivantes :

- `read_data_from_file file` qui lit le fichier texte dont le nom est passé en paramètre, et construit et renvoie la liste dont chaque élément est une chaîne de caractères correspondant au contenu d'une ligne du fichier ;
- `hash_password pwd` qui renvoie le haché du mot de passe `pwd` passé en paramètre. Cette fonction nécessite que les paquets `cryptokit` et `base64` de Ocaml aient été installés (en utilisant `opam` par exemple).

Vous pouvez évaluer les fonctions contenues dans le fichier `tools.ml` en utilisant la directive `#use "tools.ml" ;;`

2. Pour utiliser un vocabulaire mathématiques, les fonctions de hachage ne sont pas injectives