Développement C: Développement d’un hash breaker

Bastien Bodart, Guillerme Duvillié, Didier Valentin

Introduction

Depuis la découverte de la fuite de 92 millions d'adresses mail des clients d'AOL en 2004 [1], considérée comme l'une des premières fuites de données numériques de grande ampleur, les fuites de données numériques font partie intégrante du paysage de l'informatique moderne. Que ce soit Yahoo! en 2013 [2] ou 2014 [3], Facebook et Microsoft en 2019 [5] ou Instagram, TikTok et Youtube en 2020 [6], un grand nombre d'informations personnelles d'utilisateurs ont été obtenues puis diffusées sur les internet. Si toutes les fuites de données ne contiennent pas les identifiants des utilisateur·trice·s, quelques unes d'entre elles ont donné naissance à des listes contenant les logins et les mots de passe hachés des utilisateur·trice·s impacté·e·s. La dernière étape avant de pouvoir exploiter ces informations consiste à retrouver les mots de passe correspondant aux *hash* contenus dans ces listes.

Principes

Fonction de Hachage

Selon Menezes et al. [7], une fonction de hachage peut être définie comme suit:

> Une fonction de hachage est une fonction efficace en terme de calcul associant une chaîne binaire de taille arbitraire à une chaîne binaire de taille fixe, appelée hash, empreinte ou condensat.

En d'autres termes, une fonction de hachage prend en entrée une donnée binaire quelconque et calcule à partir de cette dernière, une chaîne de caractères de taille fixe [1]. Pour qu'elle soit considérée comme sûre cryptographiquement, elle doit répondre aux propriétés suivantes[7]:

1. la fonction est **déterministe**: une même entrée donne toujours le même condensat,
2. elle est à **sens unique**: étant donné un condensat, il est impossible de construire de manière efficace un message produisant ce condensat,
3. elle présente une grande **dispersion**: deux messages très proches donnent des condensats très différents,
4. elle est **résistante aux collisions**: il est difficile de trouver deux messages différents ayant le même condensat.

De la propriété 2 découle le fait que, étant donné un condensat obtenu à l'aide d'une fonction de hachage cryptographique, l'un des seuls moyens permettant de trouver une chaîne de caractères générant ce condensat est de tester successivement l'ensemble des chaînes de caractères possibles, de calculer leur condensat et de le comparer au condensat initial. C'est cet algorithme dit exhaustif, qu'il vous sera demandé de développer en C.

1. Des fontions de hachage plus récentes telles que SHA3 permettent d'obtenir des hashs de taille variable.

Utilisation des sockets

L'exploration exhaustive de l'espace des chaînes de caractères étant une opération très coûteuse en calcul, le programme que vous avez à développer devra se présenter une architecture client/serveur, afin de pouvoir répartir la charge de calcul sur plusieurs machines.

Serveur

Le serveur possède la liste des condensats à casser. Il se charge d'accepter les connexions, de récupérer les mots de passe en clair et de distribuer des condensats aux clients en faisant la demande.

Client

Le client est en charge de l'exploration de l'espace des chaînes de caractères. Il se connecte au serveur, effectue la demande d'un condensat, récupère ce dernier puis effectue les calculs nécessaires pour récupérer le mot de passe en clair. Une fois ce dernier obtenu, il renvoie au serveur le condensat et le mot de passe en clair correspondant.

Évaluation

Acquis d'apprentissage

L'évaluation a pour objectif de valider les acquis d'apprentissage tels que décrits dans la fiche UE du cours:

- Développer une application C dans le contexte de la sécurité informatique
- Comprendre les enjeux de la gestion de la mémoire en C
- Argumenter les choix d'implémentation

Critères minimum de réussite

Votre système devra au minimum respecter les contraintes suivantes

- Programmes qui compilent et fonctionnent sans erreurs.
- Respect des consignes de l'énoncé.
- Respect des bonnes pratiques vues durant les cours et laboratoires.

- Le projet se compose de deux programmes C, un jouant le rôle de serveur, un autre jouant le rôle de client. Le programme client est capable d'envoyer des messages au programme serveur à l'aide des **sockets**.
- Le programme serveur est capable de communiquer de manière synchrone avec un unique client, de lui transmettre un condensat et de récupérer le résultat du client.
- Le programme client est capable de récupérer un condensat **MD5** auprès du serveur, de le casser et de transmettre au serveur le résultat de ses calculs.
- Le programme client effectue une exploration de l'espace des chaînes de caractères de manière récursive.
- Le numéro de port et l'adresse IP à utiliser par le serveur pour *binder* le **socket** doivent être renseignés à l'aide des arguments (`argv, argc`). Cette adresse et ce numéro de port devront être renseignés au programme client de la même manière.
- Ces programmes sont compatibles avec un machine Linux type Debian/Kali Linux.
- Respect de la deadline.

Critères de dépassement

Une fois les critères de base rencontrés, libre aux étudiants de dépasser ces fonctionnalités et d'ajouter une touche personnelle à leur système. Ci-dessous une liste non exhaustive de critères de dépassement :

- Gestion de plusieurs clients par le serveur en utilisant des communications asynchrones,
- Possibilité pour le serveur d'envoyer des ensembles de condensats aux clients,
- Utilisation de structures de données appropriées par le programme client pour optimiser le cassage de hash lors de la réception d'ensembles de condensats,
- Gestion de plusieurs fonctions de hachage (SHA1, SHA2, SHA3, SAH256, …),
- Chiffrement des communications,
- Gestion propre des erreurs,
- Utilisation d'une base de données côté serveur,
- …

Avertissement

Le plus important pour nous est que l'étudiant soit capable de démontrer sa compréhension du code réalisé davantage que l'implémentation d'une multitude de fonctionnalités. Un bon code dont l'étudiant serait incapable de justifier l'implémentation n'est pas suffisant pour réussir. Veillez donc à bien comprendre chaque ligne de code avant d'aller plus loin. Nous préférons de loin un code simple

et propre avec le minimum de fonctionnalités de pair avec une bonne défense de l'étudiant plutôt qu'un code complexe difficile à défendre.

Modalités pratiques

- Le projet doit être réalisé par groupe de deux,
- L'étudiant doit être capables de défendre chaque partie du système.
- De plus, l'étudiant sera interrogé de manière individuelle sur des questions théoriques liées au cours.
- Toute tentative de triche ou de plagiat sera sanctionnée d'un zéro.
- L'étudiant utilisera deux machines virtuelles pour faire une démonstration de leur système. Attention de travailler avec un réseau virtuel interne pour ne pas dépendre d'une connexion internet le jour de l'examen.
- Le dépôt du projet dans les délais fait office d'inscription à l'examen.

Bibliographie

1. CNN. AOL employee arrested and charged with stealing list. *https://money.cnn.com/2004/06/23/technology/aol_spam/, Dernier accès: 2023-09-14*

2. Times N-Y. All 3 billion Yahoo accounts were affected by 2013 attack. *https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html, Dernier accès: 2023-09-14*

3. Post TW. Yahoo data breach casts 'cloud' over Verizon deal. *https://www.washingtonpost.com/news/the-switch/wp/2016/09/22/report-yahoo-to-confirm-data-breach-affecting-hundreds-of-millions-of-accounts/?hpid=hp_hp-top-table-main_yahoo-switch-330pm%3Ahomepage%2Fstory, Dernier accès: 2023-09-14*

4. CNET. Millions of Facebook user phone numbers exposed online, security researchers say. *https://www.cnet.com/news/privacy/millions-of-facebook-user-phone-numbers-exposed-online-security-researchers-say, Dernier accès: 2023-09-14*

5. Engadget. Microsoft accidently exposed 250 million customer service records. *https://www.engadget.com/2020-01-22-microsoft-database-exposure.html, Dernier accès: 2023-09-14*

6. Forbes. 235 Million Instagram, TikTok and YouTube user profiles exposed in massive data leak. *https://www.forbes.com/sites/daveywinder/2020/08/19/massive-data-leak235-million-instagram-tiktok-and-youtube-user-profiles-exposed/, Denier accès: 2023-09-14*

7. Menezes AJ., Van Oorschot PC., Vanstone SA. Handbook of applied cryptography. CRC press; 2001.