Sécurisation des mots de passe

La persistance des mots de passe au sein d'une application se fait généralement via une base de données, ou une base d'annuaire LDAP.

Ce stockage doit être réalisé de manière sécurisée. En effet, en cas de compromission de cette base (si elle a été récupérée ou rendue publique par un attaquant), les mots de passe seront directement révélés s'ils sont stockés en clair, ou s'il sont insuffisamment protégés.

Défense en profondeur

1. Sécurisation du système de stockage

L'accès à la base de données doit bien évidemment être sécurisé :

- Chiffrement des données échangées ;
- Mise en place d'une authentification avec droits (pour réduire la surface d'attaque).



Cette sécurisation est insuffisante et peut ne pas empêcher l'accès aux données (mots de passe).

En effet, les principales failles viennent de l'application en elle même, à partir d'utilisateurs authentifiés, de manière régulière, ou avec usurpation :

- Injections SQL
- Injections noSQL
- Injections LDAP

La sécurisation du système de stockage doit donc s'accompagner d'une méthode de sécurisation des données sensibles, au cas ou le premier rempart de protection aurait sombré ou aurait été contourné (principe de défense en profondeur).

2. Méthodes de protection des mots de passe

2.1 Chiffrement ou cryptage

Principe :

Le chiffrement (cryptage) est réversible : une données chiffrée peut être remise en clair (décryptage).

Il se base sur deux éléments : l'algorithme et la clé.

L'algorithme est un ensemble de règles qui détermine le mode de chiffrement. La clé permet de déterminer la configuration du chiffrement. Pour déchiffrer, il faut connaître l'algorithme et la clé (en case de chiffrement symétrique : 1 seule clé).

Le chiffrement de mots de passe n'est pas sécurisé, dans la mesure où :

toute la protection repose sur la non divulgation de la clé, la clé peut être découverte à partir d'un mot de passe en clair et sa version chiffrée. Pour les autres données (autres que le mot de passe) qui nécessite une

sécurisation, le chiffrement est une bonne solution.

Algorithmes DES : Data Encryption Standard (dep) AES : Advanced Encryption Standard (128, 192, 256) MARS RSA (clés asymétriques)

2.2 Hachage

Le hachage est un procédé qui utilise un algorithme pour convertir du texte en clair en valeurs numériques :

son inversion est impossible la forme hachée est appelée empreinte Deux textes identiques produisent la même empreinte Avec stockage de l'empreinte du mot de passe, le mécanisme d'authentification est le suivant :

Réception du mot de passe à tester Hachage de ce mot de passe Comparaison avec l'empreinte stockée en base Si les 2 mots de passe sont identiques sous leur forme hachée, le mot de passe est bon. 2.2.1 Algorithmes MD5 : Message-Digest Algorithm 5 SHA : Secure Hash Algorithm (sha1 (déprécié), sha-256, sha-384, sha-512) Argon2 (memory hard) bcrypt (avec sel, basé sur Blowfish)

2.2.2 Failles

Vétusté de certains algorithmes :

Exemple :

possibles collisions avec MD5 : 2 mots différents produisent la même empreinte Rainbow tables : Tables de correspondance entre mots et empreintes (consultables sur internet) Comparaison possible des hashs et déduction si toute la base est accédée 2.3 Salage Pour contrer les deux failles précédentes, la solution est d'ajouter une chaîne (grain de sel), de préférence aléatoire, au mot de passe avant de le hacher.

Le même mot de passe ne produit plus jamais la même empreinte, mais comme la chaîne utilisée comme sel est accessible au programme, on peut toujours hacher un mot de passe fourni pour le comparer au mot de passe stocké.

Il est recommandé d’utiliser un sel choisi aléatoirement pour chaque compte et d’une longueur d’au moins 128 bits. Utiliser une fonction de dérivation de mots de passe memory-hard pour pénaliser les attaques par force brute. Références : Guide ANSSI authentification et mots de passe

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/sio/bloc3/passwords?rev=1677094862

Last update: **2023/02/22 20:41**