INF3005 – Programmation web avancée

Authentification

Jacques Berger

Objectifs

Mise en place d'un mécanisme d'authentification dans une application web

Suivre les bonnes pratiques en matière de sécurité

Prérequis

Flask

Authentification

Identifier un utilisateur, avec preuve

Technique habituelle : nom d'utilisateur et mot de passe

Mots de passe

La gestion sécuritaire des mots de passe est une responsabilité importante et complexe

Plusieurs applications web préfèrent maintenant utiliser des services d'authentification externe plutôt que d'avoir cette responsabilité
(ex. Log in with Facebook)

Mots de passe

Les mots de passe sont difficiles à rendre sécuritaires car :

1) Les utilisateurs choisissent des mots de passe faciles à retenir, donc facile à deviner

2) Les utilisateurs utilisent le même mot de passe pour plusieurs applications

Mots de passe

3) Les mécanismes de récupération de mot de passe sont souvent peu sécuritaires (ex.
    Questions secrètes
    Envoie du mot de passe par courriel
    Etc.

4) Plusieurs utilisateurs peuvent avoir les mêmes mots de passe

Mots de passe

5) Il peut être difficile de contrôler qui a accès à la liste des utilisateurs/mot de passe et ce qu'ils font avec cette liste

Mots de passe

Aucun mécanisme de sécurité ne peut empêcher qu'une personne devine le mot de passe d'un utilisateur

Par contre, on doit protéger notre organisation contre une fuite éventuelle de données

Force brute

Une attaque par force brute peut briser n'importe quelle application

Force brute : Essayer toutes les combinaisons possibles jusqu'à ce qu'une d'entre elles fonctionne

Force brute

Il y a 2 choses qu'on peut faire contre la force brute

1) Limiter le nombre de tentatives de connexion (ex. guichet automatique)

2) Rendre le processus de connexion lent, de façon à rendre une attaque par force brute très longue

Force brute

Les applications bancaires ont tendance à choisir la première option

La deuxième option est beaucoup plus fréquente et plus facile à mettre en oeuvre

Attaque dictionnaire

L'attaque dictionnaire tente de deviner le mot de passe d'un utilisateur en essayant des mots de passe fréquemment utilisés et des mots du dictionnaire

Comme les mots de passe sont souvent les mêmes et souvent des mots faciles à retenir, c'est une attaque efficace

Attaque dictionnaire

Comme la force brute, il s'agit d'une attaque à répétition, mais qui fonctionne bien avec un bon nombre de mots de passe

Stockage

Le mot de passe, ou une donnée dérivée du mot de passe, doit être stocké dans une base de données

En clair

Beaucoup d'applications stockent les mots de passe en clair

En clair signifie tel quel, sans encryption, ni manipulation

En clair

Cette façon de faire est à éviter

Toute personne ayant accès à la base de données a immédiatement accès aux mots de passe de tous les utilisateurs

Pourrait entraîner une fuite de données (volontaire ou involontaire)

En clair

Certains développeurs apprécient avoir les mots de passe en clair car c'est simple et ça leur permet de fournir rapidement le mot de passe d'un utilisateur qui l'aurait perdu, ou encore utiliser le mot de passe pour des tests

Comme un utilisateur utilise souvent le même mot de passe sur plusieurs sites, cette situation n'est pas acceptable

Encryption

Une solution un peu plus sécuritaire serait d'encrypter les mots de passe dans la base de données

Encryption

Fonctionne avec une ou deux clés d'encryption

Chiffrement symétrique : une seule clé d'encryption

Chiffrement asymétrique : deux clés (clé publique et clé privée), une servant à l'encryption, l'autre à la décryption, et vice versa

Encryption

Plus sécuritaire que les mots de passe en clair

S'il y a une fuite des mots de passe, il faut encore trouver l'algorithme d'encryption et les clés

Encryption

Permet quand même à des développeurs d'accéder aux mots de passe et de les fournir aux utilisateurs

Toujours aussi vulnérable à l'ingénierie sociale

Encryption

En cas de fuite, si l'algorithme est découvert et que les clés ont été trouvées, tous les mots de passe sont compomis

Bien que plus sécuritaire, ce n'est pas non plus une technique recommandée

Hachage

Le hachage permet de calculer une empreinte (une valeur, non unique) à partir d'un mot de passe

Une fonction de hachage va toujours donner le même résultat pour une même donnée en entrée

Hachage

Le hachage altère grandement la donnée originale, de sorte qu'il est impossible de construire la donnée originale à partir du haché

Hachage

La chaîne 'INF3005' hachée avec SHA-256 donne

840cffacc5c529cf0370274e7a8fe0814e18ece78a
9cce35e5909c3e1af32543

Hachage

Au lieu de stocker les mots de passe encryptés, on peut stocker uniquement les hachés et comparer les hachés ensembles

Ainsi, on ne stocke jamais le vrai mot de passe dans la BD

Hachage

La faiblesse du hachage c'est que plusieurs données en entrée pourraient donner le même haché (des collisions)

Vulnérable à la force brute (par rainbow table)

Pour cette raison, on n'utilise pas le hachage seul

Salt

La technique de stockage de mot de passe de passe avec un salt est la technique recommandée

Un salt est une chaîne aléatoire

Salt

On fait une concaténation du mot de passe avec le salt et on envoie le résultat dans une fonction de hachage

Haché = hachage(MdP + salt)

Salt

Le haché et le salt sont stockés dans la BD mais pas le mot de passe

Il est impossible de reconstruire le mot de passe à partir du haché et du salt

Salt

Comme le salt n'est pas connu de l'utilisateur, il peut être long et complexe

Plus le salt est long, plus une attaque par force brute va être longue à donner un résultat

Salt

On génère un salt unique pour chaque utilisateur, ainsi, en cas de fuite de données, il est impossible d'identifier les utilisateurs ayant le même mot de passe

Grâce au salt, les risques de collisions sont presque inexistants