Hachage des mots de passe avec PHP

La sécurité sur le Web est un enjeu très important. On entend fréquemment parler de vols de données personnelles, d'usurpation d'identité, d'arnaques, etc.

Or, comment assurer la sécurité d'un site Web? La base minimale est d'offrir une inscription basée sur un nom d'utilisateur associé à un mot de passe. Ces comptes d'utilisateurs sont stockés dans une base de données. Qu'arrive-t-il lorsque le contenu de la table des comptes est volé, que ce soit par des développeurs du site ou par des hackers? Il y a deux possibilités :

a) Les comptes ont été enregistrés en clair. Le malfaiteur a alors accès directement à ces informations stratégiques
b) Les informations ont été encryptées par un algorithme de hachage (**adressage calculé** de codage unidirectionnelle) et alors personne ne peut utiliser directement ces informations.

L'option b) est nettement la plus sécuritaire. L'idée est simple mais puissante : seul le possesseur du compte verra son mot de passe en clair. Personne d'autre, pas même l'administrateur de la BD, ne pourra voir les mots de passe en clair.

Password_hash() et Password_verify()

Techniquement voici ce qu'il faut faire en PHP :

À la **création** d'un compte, lorsque l'utilisateur fournit le mot de passe désiré, on passe cette information à une fonction de hachage et c'est le résultat du hachage qui est sauvegardé en BD.

À la **connexion**, on saisira son mot de passe et on le passera dans une fonction de vérification qui comparera la correspondance du mot de passe fourni avec celui encrypté en BD.

PHP fournit une paire de fonctions offrant ce service de hachage-vérification : **password_hash()** et **password_verify()** . Leur utilisation est simple. Voici un petit script PHP utilisant le Password_Hash :

```
<?php
echo "<h1>Hachage de mots de passe</h1>";
$motPasse="bonjour";
  $motPasseHache1 = password_hash($motPasse, PASSWORD_DEFAULT);
  echo "$motPasse : $motPasseHache1<br>";

  $motPasseHache2 = password_hash($motPasse, PASSWORD_DEFAULT);
  echo "$motPasse : $motPasseHache2<br>";

  $motPasseHache3 = password_hash($motPasse, PASSWORD_DEFAULT);
  echo "$motPasse : $motPasseHache3<br>";
```

Voici son output à l'écran:

Remarquez que le même mot de passe (bonjour) est haché trois fois et que les trois résultats sont différents. Contrairement à d'autres fonctions de cryptage plus anciennes (par exemple MD5()) qui donnaient toujours le même résultat de hachage pour la même string.

En effet password_hash() prend en compte la temporalité pour exécuter son encryptage : le timestamp est encodé dans la string hachée.

La fonction inverse, Password_verify() est capable de retrouver le timestamp, de faire le hachage approprié et de vérifier si une string correspond à un résultat de hachage.

Voici un exemple d'utilisation de Password_verify() :

```
<?php
echo "<h1>Hachage de mots de passe</h1>";
$motPasse="bonjour";
$motPasseHache = password_hash($motPasse, PASSWORD_DEFAULT);
  echo "mot de passe haché : $motPasseHache<br>";

  echo "<h2>vérification</h2>";

  if (password_verify($motPasse, $motPasseHache))
        echo "OK 1<br>";
  else
        echo "Erreur 1<br>";

  if (password_verify($motPasse . '1', $motPasseHache))
        echo "OK 2<br>";
  else
        echo "Erreur 2<br>;
```

Et son rendu à l'écran:

Application dans vos sites

Pour utiliser cette technique simple, il ne vous reste plus qu'à

1- Modifier le champ MotDePasse de votre table Client pour la mettre en varchar de 225. L'encodage actuel du password_hash() est une string de 60 caractères, mais ce type de technique est appelée à évoluer (MD5() retournait des strings de hachage de 32 caractères) et 255 caractères vous donne une bonne marge de manœuvre.
2- Lorsque l'utilisateur fournit son mot de passe à la création de son compte, vous le passez à la fonction password_hash()
3- Vous stockez en BD le retour de password_hash().

De même lorsque l'utilisateur veut se connecter :

1- Vous saisissez le mot de passe dans un champ de formulaire
2- Vous récupérez son mot de passe haché encrypté de la BD
3- Vous passez ces deux paramètres à la fonction password_verify(). Cette dernière fonction est booléenne : s'il y a un match elle retourne vrai, sinon retourne faux.

Remarques finales

Vous n'êtes pas devenus cryptologues après avoir lu ce cours texte, mais la protection des données de votre site s'est élevée d'un cran.

La fonction de hachage est à sens unique, il n'y a pas de méthodes pour décrypter le résultat de password_hash(). Théoriquement, il pourrait y avoir un autre texte qui donnerait le même résultat de hachage, mais les probabilités sont de 1 / (53 exposant 64). C'est considéré comme très sécuritaire.

Finalement le deuxième paramètre de password_hash() peut varier plus que le PASSWORD_DEFAULT qu'on a utilisé dans nos exemples. Voici plus de documentation si les détails de password_hash vous intéressent : https://www.php.net/manual/fr/function.password-hash.php.