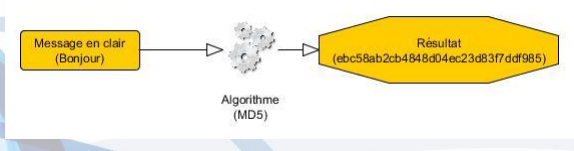
PHP – Hachage

Christel Ehrhart
contact@ce-formation.com

Définition

- Algorithme qui va, à partir d'une donnée de départ, proposer une empreinte irréversible
- Impossible à décrypter

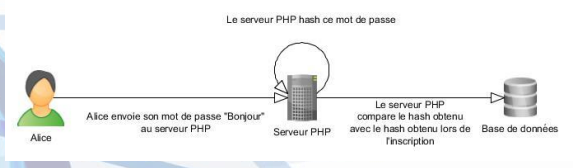


2

Pourquoi hacher les mots de passe ?

- Sans le hachage , chaque mot de passe stocké peut être volé si le support de stockage (typiquement une base de données) est compromis.

⇒Récupération des informations du site
⇒Récupération des informations d'autres site (utilisation du même mot de passe)



- Attention : Le hashage ne fait que protéger les mots de passe dans la base, pas leur éventuelle interception alors qu'ils sont envoyés à l'application par l'utilisateur

MD5() – SHA1() – SHA256()

- Ils sont destinés à être rapides et efficaces
- Dictionnaire et Banque de données md5
- De nombreux experts en sécurité considèrent ces algorithmes comme faibles et les déconseillent fortement pour hasher un mot de passe utilisateur.

Crypt()

- Permet de hacher une chaîne de caractères.

```
<?php
        $hashed_password = crypt('mypassword');

        if (hash_equals($hashed_password, crypt($user_input, $hashed_password))) {
                echo "Mot de passe correct !";
        }
?>
```

- hash_equals() permet de comparer 2 mots hachés.

crypt (string $str [, string $salt]) : string

- Attention, le 2ème paramètre est facultatif, mais provoque une notice si manquant depuis PHP 5.6

- Depuis PHP 5.5, une API de hachage existe, utilisant crypt()

API de hachage

- Les fonctions de hachage sont :
  - password_get_info — Retourne des informations à propos du hachage fourni
  - password_hash — Crée une clé de hachage pour un mot de passe
  - password_needs_rehash — Vérifie que le hachage fourni est conforme à l'algorithme et aux options spécifiées
  - password_verify — Vérifie qu'un mot de passe correspond à un hachage

Utilisation simple :

```
<?php
        // Hachage du mot de passe
        $hash =  password_hash("ici_mon_mot_de_passe", PASSWORD_DEFAULT);
        // Test sur le mot de passe
        if (password_verify("ici_mon_mot_de_passe", $hash)) {
                echo 'Le mot de passe est valide !';
        } else {
                echo 'Le mot de passe est invalide.';
        }
?>
```

Fonction password_hash

- Pour l'inscription afin de pouvoir insérer le mot de passe dans la base de données :

```
// retourne le mot de passe hashé pour le mettre en base de données
password_hash($_POST['password'], PASSWORD_DEFAULT);
```

  - Exemple de ce qui est stocké en base :

```
$2y$10$O3VEaboKqjKbC7bgP2tNp.U7Kz7rpXYpkifH.sZpg3KSFK9GdnvRe
```

- Pour l'authentification :

```
// retourne vrai si égaux
password_verify($_POST['password'], $bdd['password_in_database']);
```

7

Le Sel

- "Saler" consiste à ajouter des caractères au texte avant de le hasher
- Permet de complexifier

```
<?php
        $strPass = "MOT_DE_PASSE";

        // On sale la chaîne
        $strPass = "Need" .$strPass. "Coffee";

        // Puis on hash
        $strPass = hash('sha512', $strPass);
?>
```

- Il est également intéressant d'ajouter du sel dynamique

Exemple : ajouter la longueur de la chaine

```
<?php
    // On calcule la longueur de la chaîne
    $intLong = strlen($strPass);

    // On sale et on hash
    $strPass = $intLong.$strPass;
    $strPass = hash('sha512', $strPass);
?>
```

8

Travaux Pratiques

- Ajoutez dans la page profile.php les champs de modification du mot de passe de l'utilisateur connecté
- Enregistrez le nouveau mot de passe en utilisant du hachage
- Déconnectez l'utilisateur, puis essayez de vous reconnecter