Inscription des utilisateurs

Mise en place de la connexion à la base de données

Créez un fichier `database.php` :

```
<?php

try {
  $pdo = new PDO('mysql:host=localhost;dbname=test', 'root', 'MOT DE PASSE');
} catch (PDOException $e) {
  echo $e->getMessage();
}

return $pdo;
```

Mise en place du formulaire d'inscription

Nous modifions `register.php` :

```
<?php
$pdo = require_once './database.php';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $input = filter_input_array(INPUT_POST, [
    'username' => FILTER_SANITIZE_FULL_SPECIAL_CHARS,
    'email' => FILTER_SANITIZE_EMAIL,
  ]);
  $error = '';
  $username = $input['username'] ?? '';
  $password = $_POST['password'] ?? '';
  $email = $input['email'] ?? '';
  if (!$username || !$password || !$email) {
    $error = 'ERROR';
  } else {
    $statement = $pdo->prepare('INSERT INTO user VALUES (
      DEFAULT,
      :email,
      :username,
      :password
    )');
    $statement->bindValue(':email', $email);
    $statement->bindValue(':username', $username);
    $statement->bindValue(':password', $password);
    $statement->execute();
```

```
    header('Location: /login.php');
  }
}

?>

<!DOCTYPE html>
<html lang="fr">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <nav>
    <a href="/">Accueil</a>
    <a href="/login.php">Connexion</a>
    <a href="/logout.php">Déconnexion</a>
    <a href="/profile.php">Profil</a>
    <a href="/register.php">Inscription</a>
  </nav>

  <h1>Inscription</h1>

  <form action="/register.php" method="post">
    <input type="text" name="username" placeholder="username"><br><br>
    <input type="text" name="email" placeholder="email"><br><br>
    <input type="text" name="password" placeholder="password"><br><br>
    <?php if ($error) : ?>
      <h1 style="color:red;"><?= $error ?></h1>
    <?php endif; ?>
    <button type="submit">Valider</button>
  </form>
</body>
```

Hachage du mot de passe

Nous allons voir comment mettre en place le hachage de nos mots de passe pour l'inscription et la connexion.

Qu'est-ce que le hachage ?

Le hachage permet de transformer de manière irréversible une valeur en une chaîne de caractères appelée `hash`.

La transformation s'effectue par un algorithme appelé fonction de `hash` qui utilise plusieurs paramètres : généralement la version de l'algorithme, un coût (mémoire, itérations etc) et une chaîne de caractères générée

aléatoirement appelée `salt`.

Le `hash` et les paramètres sont stockés dans la base de données comme une chaîne de caractères :

Comment dès lors comparer le mot de passe d'inscription et le mot de passe de connexion ?

Il est impossible de comparer directement les deux mots de passe, par contre il est possible de comparer les `hash` obtenus à partir du même mot de passe (et des mêmes paramètres).

Ainsi, lors de l'inscription le serveur va transformer le mot de passe en un `hash` indéchiffrable de manière irréversible.

Pour cela, il va générer une valeur aléatoire appelée `salt` qui sera nécessaire pour arriver au même `hash` à partir de la même valeur et des mêmes autres paramètres.

Lors de la connexion, le mot de passe sera transformé de même en un `hash` (en utilisant les mêmes paramètres dont le même `salt` qui a été stocké avec le `hash`) et comparé avec le `hash` stocké dans la base de données.

De cette manière votre application ne stockera jamais aucun mot de passe en clair.

Cela ne veut pas dire que les mots de passe sont totalement protégés !

Si un attaquant dérobe votre base de données, il aura tous les `hash` et va essayer de les `brute-force` en essayant des millions de mots de passe en utilisant les paramètres.

Est-ce que deux mots de passe identiques auront le même `hash` ? La réponse est non, grâce au `salt` aléatoire.

Les `hash` générés à partir de la même valeur mais avec deux `salt` différents sont en effet complètement différents.

Pour obtenir un `hash` il faut donc obligatoirement connaître l'algorithme, le coût (nous y reviendrons) et le `salt`.

Avec ces trois options en entrée nous obtenons un `hash` qui correspond à un mot de passe.

Par cette technique, il n'y aura jamais deux `hash` identiques qui rendraient trop simple les recherches des mots de passe basiques comme `123456` etc.

Est-ce que les paramètres sont secrets ? Non absolument pas, ils sont en clair.

L'attaquant peut facilement connaître l'algorithme, les options de coût et le `salt` nécessaires au hachage dès lors qu'il a accès à votre base de données.

C'est là qu'intervient le paramètre du coût : la fonction de hachage fait des calculs complexes pour obtenir le `hash` à partir d'une valeur.

Plus le coût est élevé (en mémoire et en `CPU`), plus la quantité d'opérations pour `hasher` une valeur sera élevée, donc plus les calculs nécessaires au `brute-force` des mots de passe seront importants. Il devient rapidement impossible d'utiliser le `brute-force` pour découvrir un mot de passe car il faudrait une quantité d'opération dissuasive.

Avec `PHP`, les options de coût des algorithmes de `hash` ont des valeurs par défaut recommandées. Il n'y a donc pas besoin de les paramétrer.

Attention ! Hacher un mot de passe ne rend pas les mots de passe peu sécurisés incrackables.

Si votre mot de passe est dans la liste des 1000 mots de passe les plus utilisés, l'attaquant n'aura qu'à essayer au maximum 1000 tentatives de hachage avec les trois paramètres.

Vous savez maintenant pourquoi il ne faut pas utiliser des mots de passe triviaux.

A titre informatif les 20 mots de passe les plus utilisés sont : 123456, Password, 12345678, qwerty, 12345, 123456789, letmein, 1234567, football, iloveyou, admin, welcome, monkey, login, abc123, starwars, 123123, dragon, passw0rd, master.

Soyez donc vigilants dans vos choix de mot de passe ! Les vols de base de données sont plus courants que l'on croit, il en existe des centaines : Uber, Yahoo, Ebay, JP Morgan, Facebook, British Airways, Google plus y sont tous passés. (Et ce ne sont que celles qui ont été dévoilées !)

Un mot de passe non trivial avec caractères spéciaux et d'une longueur suffisante est en revanche inviolable, (y compris par la NSA). Mais il existe bien sûr d'autres types d'attaques !

Modification de `register.php`

Nous mettons en place le hachage :

```
<?php
$pdo = require_once './database.php';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $input = filter_input_array(INPUT_POST, [
    'username' => FILTER_SANITIZE_FULL_SPECIAL_CHARS,
    'email' => FILTER_SANITIZE_EMAIL,
  ]);
  $error = '';
  $username = $input['username'] ?? '';
  $password = $_POST['password'] ?? '';
  $email = $input['email'] ?? '';
  if (!$username || !$password || !$email) {
    echo 'HERE';
    $error = 'ERROR';
  } else {
    $hashedPassword = password_hash($password, PASSWORD_ARGON2ID);
    $statement = $pdo->prepare('INSERT INTO user VALUES (
      DEFAULT,
      :email,
      :username,
      :password
    )');
    $statement->bindValue(':email', $email);
    $statement->bindValue(':username', $username);
    $statement->bindValue(':password', $hashedPassword);
    $statement->execute();

    header('Location: /login.php');
  }
}

?>
```

Essayez maintenant de créer un utilisateur et allez dans `Workbench`.

Dans la colonne `password` vous aurez quelque chose comme :

```
$argon2id$v=19$m=65536,t=4,p=1$NUNYaEJWby50RXMzMy5Ebw$BLNQzPP9wWoBN8gk7PrH7TF3ZQMoc01xmm50SXC+/1I
```

`$argon2id` est l'algorithme de hachage utilisé.

`$v=19` est la version de l'implémentation de l'algorithme.

`$m=65536,t=4,p=1` sont les paramètres par défaut pour le coût mémoire, le nombre d'itération et le parallélisme. Nous n'avons pas besoin de modifier ces options par défaut.

`$NUNYaEJWby50RXMzMy5Ebw` est le `salt` dont nous avons parlé.

`$BLNQzPP9wWoBN8gk7PrH7TF3ZQMoc01xmm50SXC+/1I` est le `hash`.