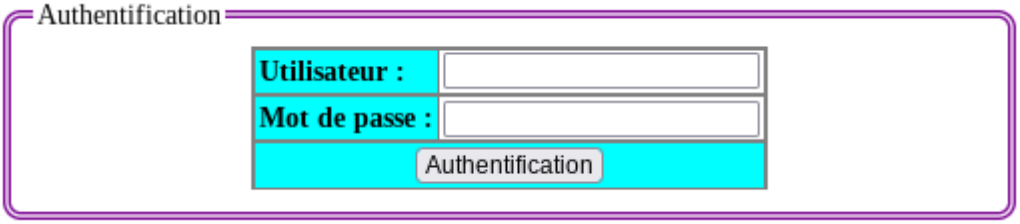
SAE23 - TD/TP n° 01

Authentification par base de données

Le but de ce TD/TP est de programmer une authentification WEB sécurisée.



1 Authentification WEB :

1. Mettre en place les fichiers fournis sur l'ENT, dans le sous répertoire `sae23-tp1` de `public_html` de votre compte sur la machine 194.199.227.110 (`https://r207.borelly.net`).
2. Tester les pages `page1.php`, `page2.php`, ... `page4.php`. Quelles sont les pages accessibles sans authentification (pour lesquelles on doit voir le titre « Page X » où X est 1, 2, 3 ou 4) ?
3. Comment les rendre toutes sécurisées en obligeant les utilisateurs à s'authentifier ?
4. Étudier le fonctionnement du script `login.php` et de la fonction **`auth`**`($user,$pass)` définie dans le fichier `funcs-auth.php`. Quels sont les identifiants à utiliser pour valider l'authentification ?
5. Comment garder le nom de l'utilisateur dans la session (pour qu'il soit visible sur toutes les pages `pageX.php` après authentification) ? On pourra alors l'afficher dans le lien de déconnexion en fin du script `login.php`.

Utilisateur [toto] authentifié !

Déconnecter toto

Page 2

6. Modifier la fonction d'authentification (script `funcs-auth.php`) pour que le mot de passe n'apparaisse plus en clair dans les sources (utiliser par exemple la fonction **non réversible** `sha1()` : facile de calculer `sha1(x)`, mais pour un `y` donné, « impossible » de trouver `x` tel que `sha1(x)=y`).

   En PHP : **`sha1`**`('titi')`
   Dans un shell Linux : `echo -n titi |` **`sha1sum`**
   `f7e79ca8eb0b31ee4d5d6c181416667ffee528ed`

7. On veut maintenant réaliser une authentification avec une base de données MySQL. En utilisant phpMyAdmin, importer les requêtes SQL du ficher `auth-mysql.sql` dans votre base de données sur le serveur `r207.borelly.net`.
8. Donner et tester une requête SQL permettant de vérifier si il y a un utilisateur `'alice'` avec le mot de passe `'abcd1234'`. Comment déduire du résultat (ou non) de l'exécution de cette requête, si l'authentification a marché ou pas.
9. Dé-commenter les lignes 2 à 4 du fichier `config-db.php` en adaptant les valeurs du compte et de la base de données MySQL à utiliser. Puis modifier la fonction `auth()` en adaptant la requête SQL précédente pour les variables `$user` et `$pass` avec PHP PDO :

```
...
$stmt=$pdo->query("SELECT * FROM users ...");
$rows=$stmt->fetchAll();
$stmt->closeCursor();
var_dump($rows); // Pour vérification uniquement
if (count($rows)>0) {
  // Auth OK
```

   Tester et valider cette authentification avec plusieurs utilisateurs.
10. Vérifier que cela fonctionne aussi même si on ne respecte pas la casse des valeurs (tout en majuscule ou bien majuscule en 2ème lettre par exemple).
11. Comment corriger le problème en modifiant le type des données (voir `BINARY`, `VARBINARY` ou `TINYBLOB`...) ?
12. La fonction MySQL `PASSWORD()` comme `sha1()` est également **non réversible**. Créer un second champ *passwd* de type VARCHAR(50) dans la table *users* pour y sauvegarder les valeurs hachées des mots de passe (afin qu'ils n'apparaissent plus en clair). Donner, en commentaire dans les sources, les requêtes SQL permettant de créer et d'initialiser le champ *passwd*.
13. Modifier la requête SQL de la fonction `auth()`pour utiliser le champ *passwd*.
14. Vérifier que l'on a plus le bug du §1.10 avec des mots de passe ne respectant pas la casse (tout en majuscule ou bien majuscule en 2ème lettre par exemple).

15. Que se passe-t-il si on rentre dans la case utilisateur du formulaire, la valeur « `admin' --` » (Avec un espace à la fin) ?
16. Corriger le problème en remplaçant l'appel à `pdo->query()` par les méthodes `prepare()` et `execute()` de PHP PDO :

```
...
$stmt=$pdo->prepare("SELECT * FROM users WHERE user=:u");
$stmt->execute(array('u'=>'toto'));
...
```

17. Comment utiliser plutôt une base SQLite ? Utiliser l'outil en ligne de commandes `sqlite3` pour créer/afficher la base SQLite `auth.sqlite` avec de nouveaux utilisateurs sur le serveur.
18. Modifier le fichier `config-db.php` pour se connecter à cette base SQLite et vérifier le fonctionnement de l'authentification.
19. La base SQLite est elle sensible au bug sur la casse comme au 1.10 ?
20. A quel endroit faut-il placer la base dans l'arborescence pour la sécurité du site ? Tester en indiquant `auth.sqlite` en fin d'URL par exemple !