2024-2025

TASKSTEP : AUDIT DE SECURITE

1PEUDINSPI :

- Tristan DAL MOLIN
- Moulay-Wassim ALAOUI
- Jules DUTRION
- Matteo DE MARCO
- Wassim DIOURI

BUT2 Informatique

IUT DIJON AUXERRE NEVERS
UNIVERSITÉ DE BOURGOGNE
Révélateur de talents depuis 1968

UNIVERSITÉ DE BOURGOGNE

Présentation détaillée des résultats : Sécurité

Préciser quelle checklist standard a été utilisée.
Détailler vos résultats :

- Présenter la liste des contrôles/mesures vérifiés considérés comme critères d'audit.
- présenter les bonnes pratiques identifiées.
- présenter la liste de vulnérabilités,

Vous pouvez utiliser des tableaux synthétiques comme le modèle ci-dessous :

Domaine	Critères d'audit	Résultats de l'audit (constats)	Description des vérifications effectuées (tests, conditions de test, etc)
	A.5.1 Orientations de la direction en matière de sécurité de l'information		
A.5 Politiques de la sécurité de l'information	A.5.1.1 Politiques de sécurité de l'information	Bonnes pratiques identifiées	
		Bonne pratique 1	
		Bonne pratique 2	
		....	
		Vulnérabilités enregistrées	
		Vulnérabilité 1 - Référence de la vulnérabilité 1	
		....	
	A.5.1.2 Revue des politiques de sécurité de l'information	Bonnes pratiques identifiées	
		Bonne pratique 1	
		Bonne pratique 2	
		....	
		Vulnérabilités enregistrées	
		Vulnérabilité 1 - Référence de la vulnérabilité 1	

Tableau 3: Exemple détail résultat

Pour chaque vulnérabilité non acceptable enregistrée indiquer :

- Description :
- Preuve(s) d'audit : un imprim écran du résultat obtenu
- Recommandation :

- **MODALITES DE RENDU :**
- Ce rapport détaillé doit être intégré au rapport global sous forme d'annexe
- **Il doit également être envoyé par mail (format docx) à Matthieu Simonet.**
- Le dépôt doit être effectué **le vendredi 18 avril avant minuit**

I. Description de la situation :

Liste des types d'utilisateurs

Il n'existe qu'un type d'utilisateur : ceux qui verront leurs tâches être affichées dans Taskstep

Liste des composants de l'application

Nom du composant	Localisation	Restriction d'accès
BDD	Lieu d'hébergement du serveur	Administrateur du service et serveur
Serveur	Lieu d'hébergement du serveur	Administrateur du service
Client	Chez l'utilisateur	Public

À noter que l'application peut être utilisée localement, auquel cas tout se trouvera sur la même machine.

Liste des données manipulées

Données	**Accès en lecture**			**Accès en écriture**		
	Utilisateurs cibles	**Risques**	**Sensibilité**	**Utilisateurs cibles**	**Risques**	**Sensibilité**
Login de l'utilisateur	Utilisateur concerné	X	Pas sensible	X	Perte de fonctionnalité	Sensible
Mot de passe	Utilisateur concerné	Usurpation d'identité	Très sensible	Utilisateur concerné	Perte de l'usage du mot de passe	Très sensible
Item	Utilisateur concerné	X	Sensible	Utilisateur concerné	X	Aucune sensibilité
Paramètre	Utilisateur concerné	X	Pas de risque	Utilisateur concerné	X	Aucune sensibilité

II. Analyse des communications

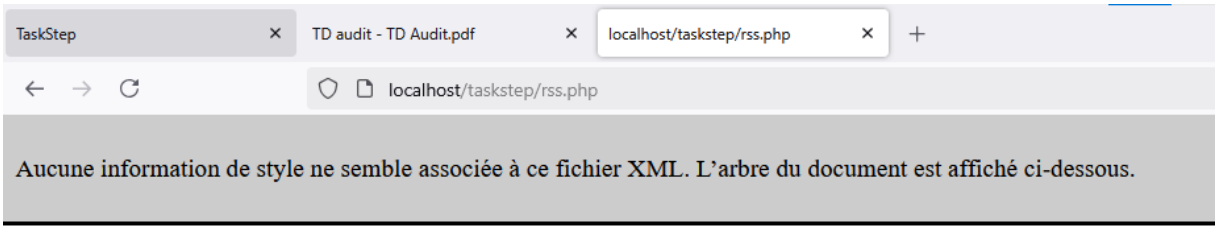
Liste des communications entre composants

Communication de ... vers ...	Type de communication	Données transmises	Risques	Sensibilité	Protocole utilisé
Client vers serveur	HTTP	Données web	Interception mot de passe / item	Très sensible	HTTP
Serveur vers BDD	SQL	Données SQL	Interception mot de passe / item	Très sensible	MySQL

Analyse des vulnérabilités

La communication se passe sans chiffrement donc les mots de passe se retrouve en clair et peuvent être intercepté lors de la communication. Il pourrait y avoir un problème pour intercepter les requêtes du côté serveur si le mot de passe protégeant la base.

De même, le serveur ne limite en rien les requêtes que l'utilisateur a le droit de faire ou non, impliquant qu'on peut ouvrir n'importe quel fichier sur le serveur. Cela n'est, pour le moment, pas critique car nous ne stockons rien de dangereux, mais si jamais nous venions à stocker des fichiers plus sensibles (base de données sous fichier db par exemple), l'utilisateur pourrait y avoir accès sans même être connecté en trouvant simplement le bon lien.




```
−<rss version="2.0">
  −<channel>
      <title>TaskStep</title>
      <link>http://localhost/taskstep/</link>
      <description>TaskStep Items Feed</description>
      <language>en-us</language>
      <generator>IceMelon RSS Feeder</generator>
    −<item>
        <title>Sample task</title>
        <link>http://localhost/taskstep/edit.php?id=1</link>
        <description>2007-08-27 | SampleProject | SampleContext | Notes</description>
      </item>
  </channel>
</rss>
```

Recommandations

Passer les communications entre le client et le serveur par https pourrait donc éviter les différentes failles, niveau de priorité critique.

Limiter l’accès aux fichiers pour l’utilisateur à l’aide de fichier .htaccess. Faille pour le moment peu critique mais pouvant le devenir selon ce qu’on stocke sur le serveur et étant facilement réparable.

III. Analyse de la gestion de permissions

Analyse de l'identification

La connexion est faite par un mot de passe en simple facteur. Le mot de passe n'a pas de règle détaillée (ni de taille imposée, de symboles ou majuscule, de durée de vie, etc.), et rien n'est mis en place dans le cas où l'utilisateur oublierait son mot de passe avant de se connecter.

Au moment de la connexion, on peut essayer autant de mots de passe que l'on veut, ce qui rend donc le site vulnérable aux brutes forces.

Exemple de brute force : on utilisera le logiciel Hydra via invite de commande sur WSL a été utilisé pour faire les tests liés aux brutes force avec une wordlist contenant le mot de passe pour permettre la connexion via le mot de passe :

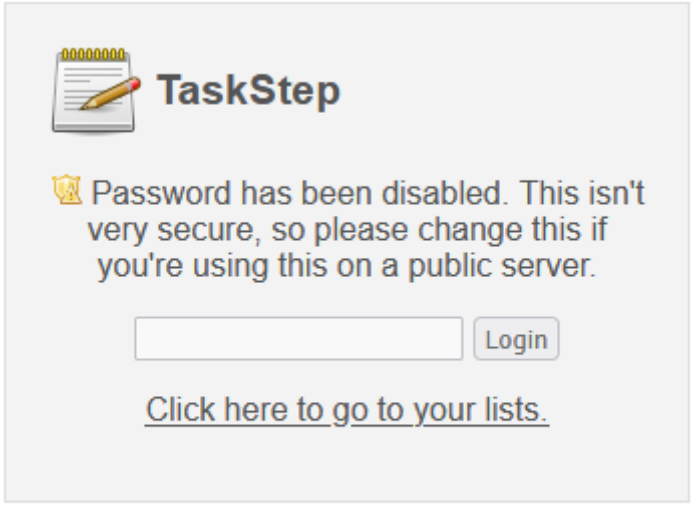
```
jules@DESKTOP-II5QK5U:/mnt/c/Users/myste/Downloads$ hydra -l "" -P wordlist_fr_5d.txt -f 192.168.56.1 http-post-form "/taskstep-mas
ter/login.php:password=^PASS^&submit=Login:Incorrect password."
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for ille
gal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-16 09:26:58
[DATA] max 16 tasks per 1 server, overall 16 tasks, 11875 login tries (l:1/p:11875), ~743 tries per task
[DATA] attacking http-post-form://192.168.56.1:80/taskstep-master/login.php:password=^PASS^&submit=Login:Incorrect password.
[80][http-post-form] host: 192.168.56.1   password: taskstep
[STATUS] attack finished for 192.168.56.1 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-04-16 09:27:17
```

Le mot de passe "taskstep" a été placé dans les millièmes lignes et le mot de passe a donc fini par être cracké de cette façon.

Le mot de passe est stocké haché et salé en base de données.

De même, il existe un mode de fonctionnement du site qui ne nécessite aucun mot de passe, ce qui, dans le cadre d'une application multi-utilisateurs, est très dangereux.

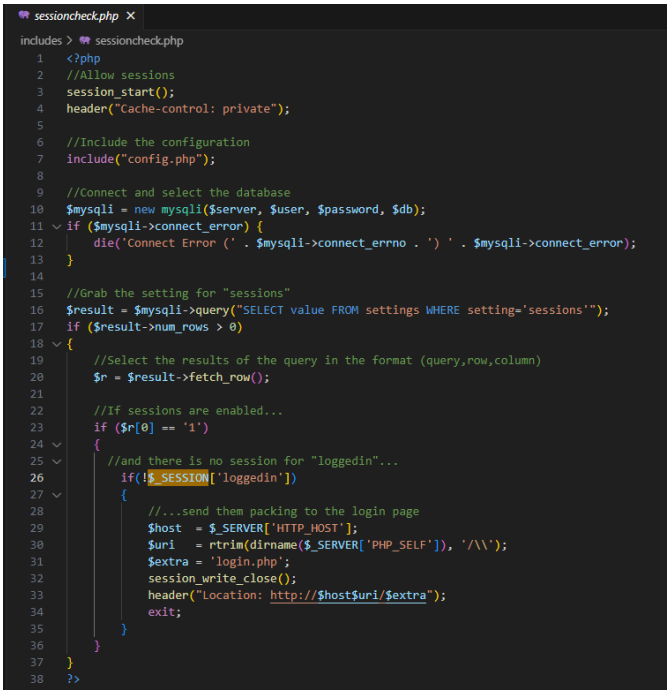


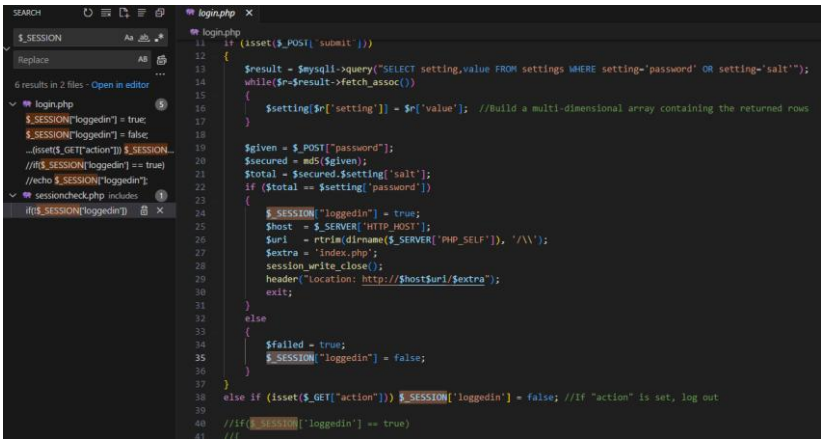
Analyse de l'authentification

L’utilisateur ne peut pas modifier ses propres droits dans l’application, uniquement son mot de passe.

L’application ne génère rien pour gérer l’authentification de l’utilisateur (ni token, ni session créée pour limiter dans le temps la connexion).

Il n’y a aucune expiration à la connexion de l’utilisateur : une fois connecté, on l’est pour toujours (voir figures ci-dessous pour preuves dans le code)





Recommandation

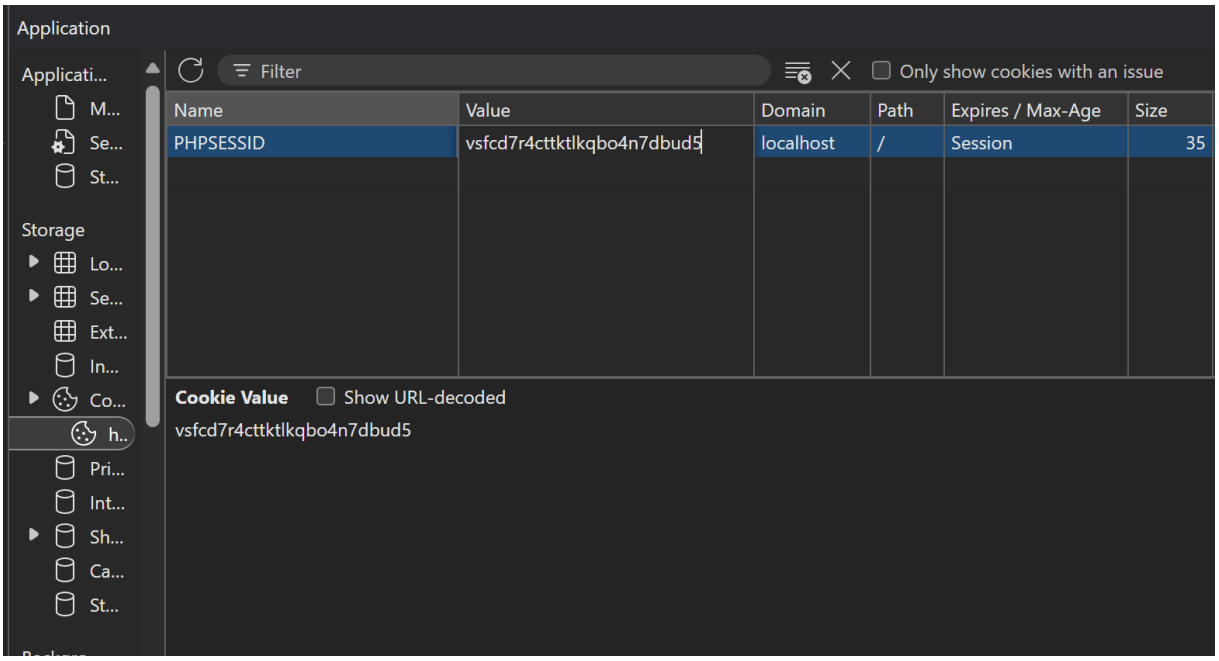
Concernant les attaques par brutes force, il faudrait mettre un compte à rebours entre chaque erreur de mot de passe pour rendre l’attaque lente et en définitive décourager les attaquants, en plus d’intégrer une limite d’un certain nombre d’essais par heure ou jour afin d’empêcher de telles attaques d’arriver. En effet actuellement le site peut facilement être attaqué par brute force via n’importe quel logiciel, et l’efficacité peut être démultipliée par l’utilisation de “wordlist” complète.

De même, la connexion de l'utilisateur ne s'arrête jamais, il serait nécessaire d'intégrer une expiration à sa connexion afin d'éviter d'éventuelles intrusion sur le site à l'aide d'une simple connexion maintenue ouverte.

Le mode sans mot de passe du site web sera également à retirer pour renforcer sa sécurité.

Enfin, le mot de passe en lui-même pourrait nécessiter des ajustements tels qu'imposer majuscule, chiffre et caractère spécial voire une durée de vie au bout de laquelle il faudrait changer de mot de passe.

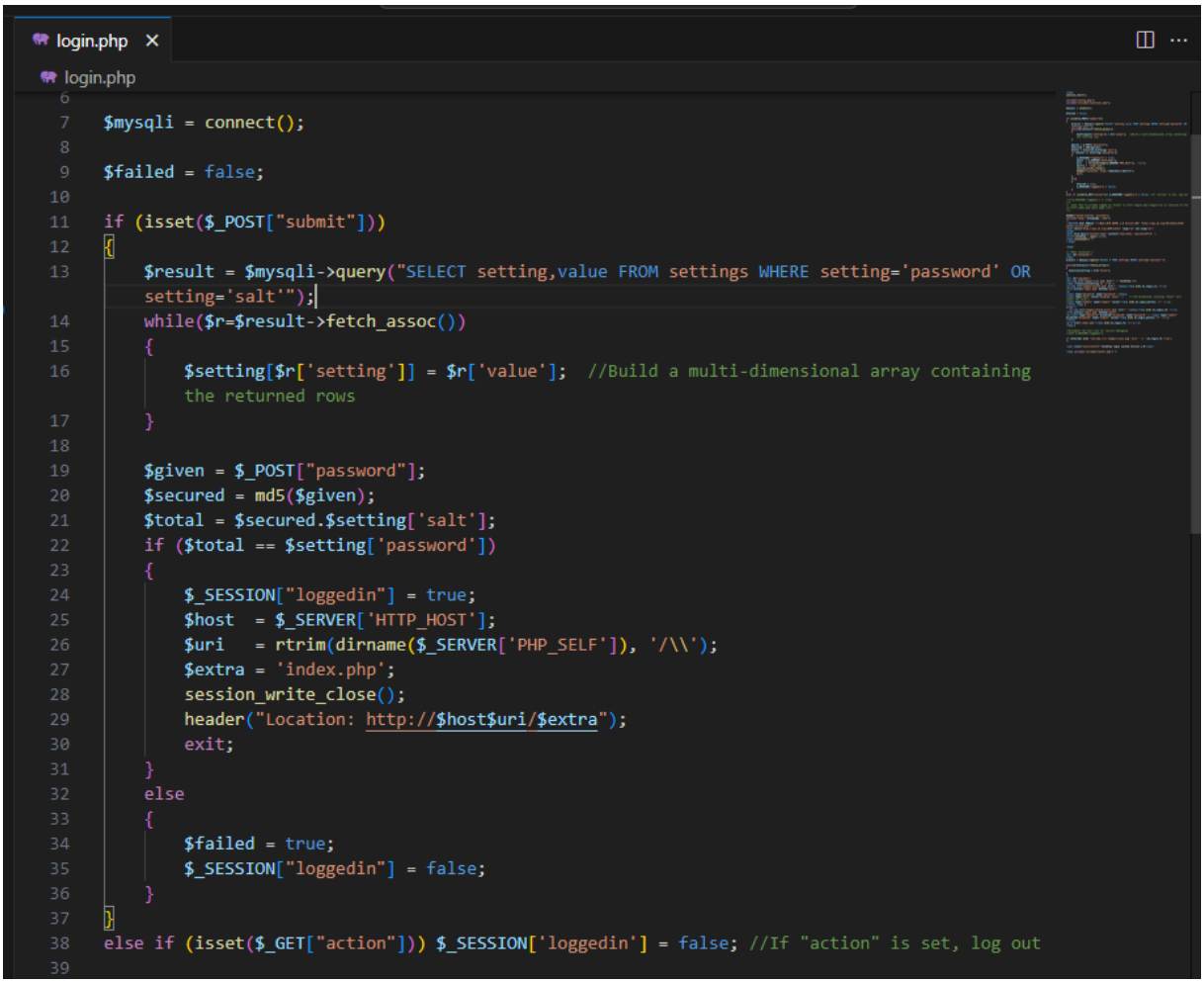
Le site rencontre aussi une nouvelle faille. Il y a un cookie sur le site qui permet de stocker l'identifiant d'une session PHP qui n'est pas supprimée. En modifiant la valeur du cookie par un id de session PHP qui est stockée, la page de connexion peut donc être directement contournée et les actions du site deviennent directement accessibles.



IV. Analyse du stockage des informations

Vérification des données stockées

Une connexion est mise en place pour ce qui est de la connexion : étant donné qu'il n'existe qu'un seul mot de passe à l'application, il est récupéré en BDD et ensuite comparé au mot de passe entré, ce qui évite une possible injection SQL. Il est cependant haché en MD5, qui n'est plus un algorithme de hachage sécurisé.

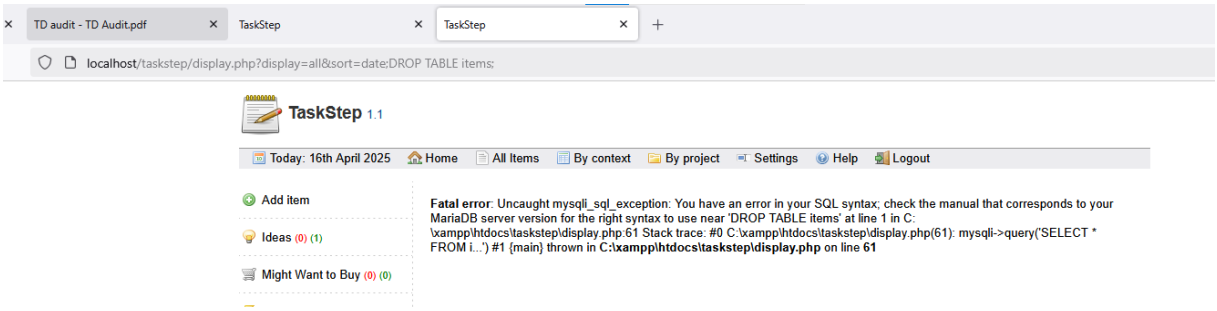


Pour le reste des requêtes, aucune requête de sélection n'est protégée face aux injections et est donc à sujet de nombreuses failles dans la base de données.

```php
$display = (isset($_GET["display"])) ? $_GET["display"] : '';
$sortby = (isset($_GET["sort"])) ? $_GET["sort"] : 'date';
$section = (isset($_GET["section"])) ? $_GET["section"] : '';
$tid = (isset($_GET["tid"])) ? $_GET["tid"] : '';

switch ($display)
{
    case "section":
        //Massively cleaned up section which obtains section titles
        from the language file
        foreach($l_sectionlist as $key=>$value){
            if($section==$key){
                $currentsection = $key;
                $sectiontitle = $value;
            }
        }
        $result = $mysqli->query("SELECT * FROM items WHERE
        section='$currentsection' ORDER BY $sortby");
        echo "<div id='sectiontitle'><h1>$sectiontitle</h1></div>";
        $noresultsurl = '?section=' . $section;
    break;
    case "project":
    case "context":
        $idresult = $mysqli->query("SELECT title FROM {$display}s
        WHERE id='$tid'");
        $disptitle = $idresult->fetch_row()[0];
        $result = $mysqli->query("SELECT * FROM items WHERE
        $display='$disptitle' ORDER BY $sortby");
        echo "<div id='sectiontitle'><h1>$disptitle</h1></div>";
        $noresultsurl = '?tid=' . $tid;
    break;
    case "all":
        $result = $mysqli->query("SELECT * FROM items ORDER BY
        $sortby");
        echo "<div id='sectiontitle'><h1>".$l_nav_allitems."</h1></
        div>";
        $noresultsurl = '';
    break;
    case "today":
        $today = date("Y-m-d");
        $todayf = date($menu_date_format);
        $result = $mysqli->query("SELECT * FROM items WHERE
        date='$today' ORDER BY $sortby");
        echo "<div id='sectiontitle'><h1>".$l_nav_today.": $todayf</
        h1></div>";
        $noresultsurl = '';
    break;
```

On peut donc effectuer n'importe quelle injection SQL si on le souhaite à partir de l'URL du site :



MySQL protège par défaut des requêtes multiples dans un même élément mais il reste possible d'exploiter cette faille d'une façon ou d'une autre afin d'attaquer la base de données du site.

Chiffrement des données stockées

Données	Sensibilité	Chiffrement utilisé	Accès à la donnée	Niveau d'adéquation
Tâches	Peu sensible	Chiffré par MySQL	Sur les pages listant les tâches	Bon car nous n'avons pas besoin de plus de chiffrement de données que celui de la BDD de base
Items	Peu sensible	Chiffré par MySQL	Sur les pages listant les items	Bon car nous n'avons pas besoin de plus de chiffrement de données que celui de la BDD de base
Login	Peu sensible	Chiffré par MySQL	Par la base de données et lors de la connexion	Bon car nous n'avons pas besoin de plus de chiffrement de données que celui de la BDD de base
Mot de passe	Très sensible	MD5 + salage	/	Non car MD5 plus sécurisé
Sel	Très sensible	Chiffré par MySQL	/	Mauvais car pourrait être accessible sous injection SQL, peut être stocké à part de la BDD

Recommandations

Toutes les requêtes du programme seront à sécuriser si on souhaite éviter les injections SQL, cela n'est actuellement pas le cas et il est possible de trouver des informations sensibles de la base de données à l'aide de la bonne requête. En mettant le code à jour, il serait donc préférable d'échapper les paramètres et de les vérifier avant d'exécuter des requêtes SQL.

Toutes fois, les failles SQL via le mot de passe sont empêchées lors de la connexion.

Le sel, servant à renforcer la sécurité derrière l'authentification, se trouve également en BDD, ce qui implique qu'une fuite de la base de données nous donnerait accès, non seulement au mot de passe haché, mais aussi à son sel et n'empêchera donc pas l'authentification. Il devrait donc se trouver dans un fichier séparé ou purement en code afin d'éviter une telle faille de sécurité.

De même, la fonction de hachage MD5 est aujourd'hui sujette à de majeures vulnérabilités et continuer de l'utiliser est un risque pour les utilisateurs de cette application. Il serait donc recommandable de modifier l'algorithme de hachage pour un plus robuste, le SHA256 étant plus actuel et optimisé dans notre cas.