Ticket n°5
Titre du ticket : amélioration du chiffrement des mots de passe

Type du ticket : incident *(évolution/incident)*	**Niveau de gravité** : ☐Bloquant ☒ Majeur ☐ Mineur
Émetteur : Nicolas BOURGEOIS *(nom de l'émetteur)*	**Date signalement** : 25/09/2023 *(jj/mm/aaaa)*
Assignation : Martin PLISSONNEAU *(nom du membre de l'équipe en charge du ticket)*	**Date de résolution souhaitée** : 02/10/2023 *(jj/mm/aaaa)*
Application concernée : *R3st0.fr*	**Version** : 1.0 initiale – septembre 2023

Description du problème *(avec éventuelles captures d'écran, messages d'erreurs)* :

Par souci de renforcer la sécurité de l'application, les maîtres d'œuvre souhaitent faire évoluer le traitement des mots de passe des utilisateurs (authentification, nouvelle inscription) pour respecter les préconisations de PHP en la matière.

Actuellement, le chiffrement des mots de passe utilise la fonction crypt() avec un sel simpliste (="sel").

PHP conseille l'usage du couple de fonctions password_hash / password_verify avec l'algorithme de hachage par défaut BCRYPT.

Références :

- password_hash https://www.php.net/manual/fr/function.password-hash.php
- crypt https://www.php.net/manual/fr/function.crypt.php

Avantages de l'utilisation de password_hash :

- meilleur algorithme de hachage par défaut (BCRYPT)
- évolutive (adaptation automatique aux améliorations des algorithmes)
- salage efficace
- compatibilité avec crypt, donc les anciens mots de passe resteront utilisables, même s'il sera préférable de les modifier pour générer une meilleure empreinte.

Lexique

fonction de hachage *: calcule une empreinte numérique non réversible.*

salage *: renforce la sécurité du hachage en y ajoutant une donnée supplémentaire (le sel) afin d'empêcher que deux informations identiques conduisent à la même empreinte => protège des attaques par force brute et par table arc-en-ciel.*

coût *: rend l'algorithme arbitrairement lent et contribue à dissuader les attaques par table arc-en-ciel et par force brute.*

table arc-en-ciel *: table comportant un grand nombre d'empreintes connues, permettant de retrouver un mot de passe à partir de son empreinte.*

Solution *(diagnostic, localisation, modification, test) :*

En observant les fichiers « authentification.inc.php » et « UtilisateurDAO.class.php », on remarque l'utilisation de la fonction crypt pour crypter les mots de passes des utilisateurs :

Exemple :

```
public static function updateMdp(int $idU, string $mdpClair): bool {
    $ok = false;
    try {
    $requete = "UPDATE utilisateur SET mdpU = :mdpU WHERE idU = :idU";
    $stmt = Bdd::getConnexion()->prepare($requete);
    $mdpUCrypt = crypt($mdpClair, "sel");
    $stmt->bindValue(':mdpU', $mdpUCrypt, PDO::PARAM_STR);
    $stmt->bindValue(':idU', $idU, PDO::PARAM_INT);
    $ok = $stmt->execute();
    } catch (PDOException $e) {
        throw new Exception("Erreur dans la méthode " . get_called_class() . "::updateMdp : <br/>" . $e->getMessage());
    }
    return $ok;
}
```

L'objectif est donc de remplacer cette méthode de hashage par une méthode plus sécurisée comme password_hash/password_verify. Pour cela, il faut donc remplacer chaque méthode « crypt » par la méthode « password_hash » dans la méthode « updateMdp » du fichier « UtilisateurDAO » :

```
public static function updateMdp(int $idU, string $mdpClair): bool {
$ok = false;
try {
    $requete = "UPDATE utilisateur SET mdpU = :mdpU WHERE idU = :idU";
    $stmt = Bdd::getConnexion()->prepare($requete);
    $mdpUCrypt = password_hash($mdpClair, PASSWORD_BCRYPT); // Génère un hachage sécurisé avec BCRYPT
    $stmt->bindValue(':mdpU', $mdpUCrypt, PDO::PARAM_STR);
    $stmt->bindValue(':idU', $idU, PDO::PARAM_INT);
    $ok = $stmt->execute();
} catch (PDOException $e) {
    throw new Exception("Erreur dans la méthode " . get_called_class() . "::updateMdp : <br/>" . $e->getMessage());
}
return $ok;
```

Pour l'authentification, on transforme la méthode « crypt » par la méthode password_verify pour vérifier si le mot de passe saisi correspond au mot de passe "haché" de la BDD :

```
// Si le mot de passe saisi correspond au mot de passe "haché" de la BDD
if (password_verify($mdpU, $mdpBD)) {
    // le mot de passe est celui de l'utilisateur dans la base de donnees
    $_SESSION["idU"] = $idU;        // la clef est idU désormais
    $_SESSION["mailU"] = $mailU;
    $_SESSION["mdpU"] = $mdpBD;
}
```

Après avoir crée un nouvel utilisateur, on se connecte avec son identifiant et son mot de passe. Si la connexion est réussi, cela indique que la fonction « password_hash » à bien hash le mot de passe et la fonction « password_verify » à bien vérifier si le mot de passe correspond bien au mot de passe issu de la BDD.