Projet Cybersécurité

Rapport d’audit de sécurité

Ewan KERLOC’H--MICHAUD
Tarek AHMED BELHADJ
Année universitaire 2025/2026

Groupe Matthieu – Line

Vulnérabilités :

- **Injection SQL**
  Le fichier add.php fait une requête SQL qui n'est pas préparée, ce qui donne la possibilité de réaliser une injection SQL.

  Code source de la faille dans **/inc/add.php** :
  $q = 'insert into cracks (content, owner, datesend) ' . ' values("'.nl2br($char_content).'", "'.$_SESSION['userid'].'", '.time().')';
  $db->query($q);

  Toutefois, la commande précédente utilise la fonction htmlspecialchars() pour échapper les caractères spéciaux de la saisie utilisateur, ce qui réduit grandement la possibilité de réussir une injection SQL.

- **Absence de politique de mot de passe**
  Il n'existe aucune contrainte sur la complexité des mots de passe utilisateur, ce qui peut mener à des attaques de type brute force sur le formulaire de connexion.

Groupe Ange – Armelle

Vulnérabilités :

- **Bypass de l'authentification :**
  Le système d'authentification accepte la valeur d'un cookie nommé authbypass pour authentifier l'utilisateur, en sachant qu'il est possible de modifier la valeur de ce cookie. Un attaquant peut donc se connecter à n'importe quel compte du site sans avoir à fournir d'identifiant, seulement en plaçant l'UID du compte utilisateur qu'il souhait usurper dans le cookie.

  Code source de la faille dans **Auth.php** :

```
const COOKIENAME = 'authbypass';

  protected function __construct(){
    session_start();
    if(isset($_COOKIE[self::COOKIENAME])) {
      $this->log($_COOKIE[self::COOKIENAME]);
    }
  }
```

  La faille est référencée dans une issue du repository GitHub, mais il le semble que le code source n'ait pas encore été corrigé.

- **Injection SQL**
  Le code concatène directement les entrées utilisateur dans une requête SQL. Cela rend l'application vulnérable aux injections SQL : un attaquant pourrait manipuler la valeur de $login pour modifier la structure de la requête et contourner l'authentification, lire/modifier/supprimer des données, ou exécuter d'autres actions selon les privilèges SQL.

  Code source de la faille dans **Auth.php** :

```
public function tryLog($login, $pwd): bool {

        global $db;

        $q = 'select * from users where login="'.$login.'"';

        $stmt = $db->query($q, PDO::FETCH_ASSOC);

        $user = $stmt ? $stmt->fetch() : null;
```

- **Injection SQL**
  Le fichier add.php fait une requête SQL qui n'est pas préparée, ce qui donne la possibilité de réaliser une injection SQL.

  Code source de la faille dans **/inc/add.php** :

```
$q = 'insert into cracks (content, owner, datesend) ' . ' values("'.nl2br($char_content).'",
"'.$_SESSION['userid'].'", '.time().')';
$db->query($q);
```

Groupe Numa

Vulnérabilité :

- **Possibilité de déni de service sur le champ d'un formulaire**
  Pas de temporisation sur le formulaire de recherche de cracks, ce qui donne la possibilité de faire une attaque par déni de service sur le serveur web qui porte l'application. L'outil Zaproxy permet de réaliser une attaque de ce type. La faille se trouve dans le code source de la page **search.php**