LISV Laboratoire d'ingénierie des systèmes de Versailles
UVSQ université PARIS-SACLAY

Programmation sécurisée (Secure coding)
Examen TP 3
Développement d'une application web vulnérable en PHP

Halim Djerroud

révision 1.0

Objectif du TP

L'objectif de ce TP est de concevoir une application web en PHP permettant de gérer une liste de tâches (*To-Do List*) en s'appuyant sur une base de données SQLite. L'application sera volontairement développée avec certaines vulnérabilités courantes afin de servir de banc de test pour des outils d'analyse de sécurité tels que *ZAP Proxy* de l'OWASP.

Ce TP permettra aux étudiants de :
— Comprendre les failles de sécurité les plus courantes en développement web.
— Expérimenter l'exploitation de ces failles dans un environnement contrôlé.
— Apprendre à sécuriser une application en appliquant les bonnes pratiques de développement.

Description de l'application

L'application devra offrir les fonctionnalités suivantes aux utilisateurs :
— Ajouter, modifier et supprimer des tâches dans une liste.
— Rechercher des tâches spécifiques.

Elle sera développée avec des vulnérabilités intentionnelles afin d'illustrer les risques liés à la sécurité applicative. Les principales failles intégrées sont les suivantes :

1. Injection SQL

L'application ne filtrera pas correctement les entrées utilisateur, permettant ainsi l'injection de requêtes SQL malveillantes. Cette faille pourra être exploitée pour :
— Accéder à des données non autorisées.
— Modifier ou supprimer des enregistrements dans la base de données.
— Bypasser les mécanismes d'authentification.

Exemple d'exploitation : Une requête malveillante dans un champ de connexion permettant d'accéder au compte administrateur sans connaître son mot de passe.

2. Cross-Site Scripting (XSS)

Les entrées utilisateur ne seront pas échappées avant d'être affichées, permettant l'injection de code JavaScript malveillant. Cette vulnérabilité pourra être exploitée pour :
— Voler des informations sensibles (cookies, tokens de session).
— Rediriger l'utilisateur vers une page malveillante.
— Défigurer l'application en injectant du contenu arbitraire.

Exemple d'exploitation : Un utilisateur malveillant insérant un script qui affiche une fausse page de connexion pour voler les identifiants des utilisateurs.

3. Gestion non sécurisée des sessions

L'application présentera des lacunes dans la gestion des sessions utilisateur, telles que :
— Absence de régénération de l'ID de session après connexion.

— Utilisation de cookies non sécurisés (*httponly*, *secure*, *SameSite* non définis).
— Sessions non expirées après une certaine période d'inactivité.

Exemple d'exploitation : Un attaquant récupérant un cookie de session via une attaque XSS et prenant le contrôle du compte utilisateur.

Travail demandé

Les étudiants devront :
— **Développer une version vulnérable de l'application**, intégrant les failles mentionnées ci-dessus.
— **Assurer le bon fonctionnement de l'application**, avec une interface basique permettant d'interagir avec la base de données (ajout, modification, suppression et recherche de tâches).
— **Tester les vulnérabilités** en réalisant des attaques sur l'application à l'aide des méthodes et outils appropriés.
— **Documenter les exploits possibles**, en détaillant les étapes pour chaque attaque ainsi que leur impact sur la sécurité.
— **Proposer des correctifs** pour sécuriser l'application et implémenter une version corrigée (dans une autre branche git) .

Livrables

À la fin du TP, les étudiants devront fournir :
— **Le code source** de l'application vulnérable ainsi que sa version corrigée et sécurisée.
— **Un rapport détaillé** comprenant :
  — La description des failles exploitées et leurs conséquences.
  — Les étapes d'exploitation de chaque vulnérabilité.
  — Les correctifs proposés et implémentés pour sécuriser l'application.
— **Un lien GitHub** vers le projet contenant l'ensemble du code et du rapport.

Outils recommandés

1. Environnement de développement

— **XAMPP/WAMP/LAMP** pour installer un serveur Apache, PHP et SQLite.
— **Un serveur PHP intégré** (fourni avec PHP) si XAMPP/WAMP n'est pas utilisé.

2. Base de données

— **SQLite**, avec un fichier `.db` géré en PHP.

3. Outils d'analyse et de test de sécurité

— **ZAP Proxy** (OWASP) : Scanner de sécurité pour détecter les vulnérabilités.
— **Burp Suite** : Analyse et manipulation des requêtes HTTP pour tester les failles.
— **SQLite Browser** : Visualisation et modification de la base de données SQLite.
— **Postman** : Test des requêtes API et interactions avec la base de données.

Critères d'évaluation

Les étudiants seront évalués sur :

1. **Le bon fonctionnement de l'application vulnérable** (respect des fonctionnalités demandées).
2. **La pertinence des vulnérabilités intégrées** et leur exploitabilité.
3. **La qualité de l'exploitation des failles**, avec des scénarios bien documentés.
4. **L'implémentation des correctifs** et la justification des choix de sécurisation.
5. **La qualité du rapport final**, en termes de clarté, d'organisation et de contenu technique.