Rapport de projet :

Sécurisation du Projet Gestion-Universitaire Contre les Injections SQL

Réalisé par :

OUATTARA Sié Hans

Professeur :

Mr ratli

Année scolaire 2024-2025

SOMMAIRE

Objectif : Protéger l'application contre les injections SQL en renforçant les requêtes SQL, la validation des entrées, et la gestion des fichiers.

Contexte : Application PHP/MySQL avec une interface utilisateur pour la gestion des étudiants, enseignants, cours, examens, et spécialités, utilisant Bootstrap, Light Bootstrap Dashboard, et SweetAlert2.

1. Introduction

Les injections SQL sont une vulnérabilité critique où un attaquant peut manipuler des requêtes SQL en injectant des données malveillantes via des formulaires, URL, ou autres entrées utilisateur. Dans ton projet, des fichiers comme ajouter_seance.php (version initiale) utilisaient des requêtes directes (e.g., "SELECT * FROM accounts WHERE email='$email'"), exposant l'application à ce risque. Ce rapport propose des solutions pour sécuriser tous les fichiers PHP interagissant avec la base de données, en s'appuyant sur les meilleures pratiques : requêtes préparées, validation des entrées, échappement des sorties, et gestion sécurisée des fichiers.

2. Structure du Projet

L'arborescence fournie indique une application bien organisée avec des fichiers dédiés à l'authentification, la gestion des données, et l'interface utilisateur. Les fichiers clés pour la sécurisation incluent :

- Authentification : connexion.php, login.php, logout.php, unauthorized.php.
- Tableaux de bord : dashboard.php, dashboard_enseignant.php, dashboard_etudiant.php.
- Gestion de données : ajouter_*.php, modifier_*.php, supprimer_*.php, deposer_examen.php, inscription.php, notes.php, liste_*.php.
- Frontend : index.css, index.js, navtop.php, sidenav.php, footer.php.
- Uploads : uploads/ (pour deposer_examen.php).
- Schéma de la base de données (déduit) :

- accounts : id, email, nom, prenom, role (admin, enseignant, etudiant), password.
- specialite : id, nom_specialite, description, departement, date_de_creation, type_seance (CM, TP, TD).
- seances : id, titre, date_seance (DATE), duree (TIME), commentaire, enseignant_id, specialite_id.
- examen : id, titre, description, date_exam (DATETIME), enseignant_id, specialite_id, fichier.

Autres: etudiants, cours, notes, inscriptions.

3. Problèmes Identifiés

Requêtes SQL directes : Dans ajouter_seance.php ,des requêtes comme SELECT * FROM accounts WHERE email='$email' permettent l'injection (e.g., email=' OR '1'='1).

Manque de validation : Peu ou pas de validation des entrées dans certains fichiers (e.g., login.php, deposer_examen.php).

Exposition des erreurs : Les erreurs SQL (e.g., $conn->error) peuvent révéler des informations sensibles.

Gestion des fichiers : deposer_examen.php peut être vulnérable si les uploads ne sont pas validés.

Fichiers non sécurisés : modifier_*.php, supprimer_*.php, inscription.php, etc., risquent d'utiliser des requêtes non préparées.

4. Recommandations Générales

Pour rendre ton projet résistant aux injections SQL, applique ces principes à tous les fichiers PHP interagissant avec la base de données :

4.1. Utiliser des Requêtes Préparées

Remplace toutes les requêtes directes ($conn->query("SELECT ...")) par des requêtes préparées avec MySQLi ou PDO.

Exemple :

```
$stmt = $conn->prepare("SELECT * FROM accounts WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
```

Fichiers concernés : login.php, dashboard_*.php, ajouter_*.php, modifier_*.php, supprimer_*.php, deposer_examen.php, inscription*.php, liste_*.php, notes.php, data.php.

4.2. Valider et Nettoyer les Entrées

Valide les entrées utilisateur pour vérifier le format, le type, et la plage (e.g., email, ID, date).

Utilise filter_var, trim, (int), et des expressions régulières.

Exemple :

```
$email = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);
$id = (int)($_GET['id'] ?? 0);
if (!$email || $id <= 0) {
    die("Entrée invalide.");
}
```

Fichiers concernés : Tous les formulaires et paramètres URL (e.g., ajouter_seance.php, modifier_etudiant.php, deposer_examen.php).

4.3. Échapper les Sorties

Protège contre les XSS (non lié à SQL, mais complémentaire) en échappant les données affichées.

Exemple :

```
echo htmlspecialchars($row['nom']);
```

Fichiers concernés : dashboard_*.php, liste_*.php, details_etudiant.php, ajouter_*.php.

4.4. Limiter les Privilèges de la Base de Données

Configure un utilisateur MySQL avec des permissions minimales :

```
GRANT SELECT, INSERT, UPDATE, DELETE ON inscription_universitaire.* TO 'user'@'localhost';
```

Fichier concerné : connexion.php.

4.5. Gérer les Erreurs Sécuritairement

Masque les erreurs SQL pour les utilisateurs, enregistre-les dans des logs.

Exemple :

```
if (!$stmt->execute()) {
    error_log("SQL Error: " . $stmt->error);
    die("Une erreur est survenue.");
}
```

Fichiers concernés : Tous les fichiers avec requêtes SQL.

4.6. Sécuriser les Uploads de Fichiers

Valide le type, la taille, et l'extension des fichiers uploadés (e.g., PDF, DOCX).

Utilise des noms de fichiers générés par le serveur.

Exemple :

```
$file = $_FILES['fichier'];
$allowed = ['pdf', 'doc', 'docx'];
$ext = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));
if (!in_array($ext, $allowed) || $file['size'] > 5 * 1024 * 1024) {
    die("Fichier invalide.");
}
$filename = uniqid('exam_') . '.' . $ext;
move_uploaded_file($file['tmp_name'], 'uploads/' . $filename);
```

Fichiers concernés : deposer_examen.php, download.php.

4.7. Considérer PDO (Optionnel)

PDO offre une abstraction portable et une syntaxe claire.

Exemple :

```
$pdo = new PDO("mysql:host=localhost;dbname=inscription_universitaire", "user", "pass");
$stmt = $pdo->prepare("SELECT * FROM accounts WHERE email = :email");
$stmt->execute(['email' => $email]);
```

Fichiers concernés : Tous, si migration.

5. Recommandations Spécifiques par Fichier

Voici des modifications concrètes pour les fichiers clés, basées sur les exemples fournis et l'arborescence.

5.1. connexion.php

Objectif : Établir une connexion sécurisée.

Code suggéré :

```php
<?php
// connexion.php
define('DB_HOST', 'localhost');
define('DB_USER', 'your_db_user');
define('DB_PASS', 'your_db_password');
define('DB_NAME', 'inscription_universitaire');

try {
    $conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
    if ($conn->connect_error) {
        error_log("Connection failed: " . $conn->connect_error);
        die("Erreur de connexion à la base de données.");
    }
    $conn->set_charset("utf8mb4");
    date_default_timezone_set('Europe/Paris');
} catch (Exception $e) {
    error_log("Connection error: " . $e->getMessage());
    die("Erreur serveur.");
}
?>
```

Sécurité : Masque les erreurs, utilise UTF-8, définit le fuseau horaire.

5.2. login.php

Objectif : Authentification sécurisée.

Code suggéré :

```
<?php
// login.php
session_start();
require_once 'connexion.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = filter_var($_POST['email'] ?? '', FILTER_VALIDATE_EMAIL);
    $password = $_POST['password'] ?? '';

    if (!$email || empty($password)) {
        $error = "Email ou mot de passe invalide.";
    } else {
        $stmt = $conn->prepare("SELECT id, email, nom, prenom, role, password
FROM accounts WHERE email = ?");
        $stmt->bind_param("s", $email);
        $stmt->execute();
        $result = $stmt->get_result();
        $user = $result->fetch_assoc();
        $stmt->close();

        if ($user && password_verify($password, $user['password'])) {
            $_SESSION['email'] = $user['email'];
            $_SESSION['role'] = $user['role'];
            $_SESSION['user_id'] = $user['id'];
            session_regenerate_id(true);
            header("Location: dashboard.php");
            exit();
        } else {
```

```
            $error = "Identifiants incorrects.";
        }
    }
}
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Connexion</title>
    <?php include 'navtop.php'; ?>
</head>
<body>
    <div class="container">
        <h2>Connexion</h2>
        <?php if (isset($error)): ?>
            <div class="alert alert-error"><?php echo htmlspecialchars($error); ?></div>
        <?php endif; ?>
        <form method="POST" action="login.php">
            <div class="form-group">
                <label>Email</label>
                <input type="email" name="email" required>
            </div>
            <div class="form-group">
                <label>Mot de passe</label>
                <input type="password" name="password" required>
            </div>
            <button type="submit" class="submit-btn">Se connecter</button>
```

```
    </form>
  </div>
  <?php include 'footer.php'; ?>
</body>
</html>
```

Sécurité : Requêtes préparées, validation d'email, hachage de mot de passe, régénération de session.

5.3. dashboard_enseignant.php

Objectif : Afficher des données sécurisées.

Code existant (déjà sécurisé dans la version fournie) :

Utilise des requêtes préparées pour accounts, seances, et specialite.

Valide role et échappe les sorties.

A Appliquer le même modèle à dashboard.php, dashboard_etudiant.php, liste_*.php :

```
$stmt = $conn->prepare("SELECT COUNT(*) FROM seances WHERE enseignant_id = ? AND date_seance >= CURDATE()");
$stmt->bind_param("i", $user_id);
$stmt->execute();
```

5.4. ajouter_seance.php

Objectif : Ajouter des séances sécurisées.

Code existant (déjà sécurisé) :

Utilise des requêtes préparées pour insertion et sélection.

Valide titre, date_seance, duree, specialite_id.

Appliquer à ajouter_*.php

```
$stmt = $conn->prepare("INSERT INTO etudiants (nom, prenom, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $nom, $prenom, $email);
```

5.5. deposer_examen.php

Objectif : Gérer les uploads et insertions sécurisées.

Code

```
<?php
// deposer_examen.php
session_start();
require_once 'connexion.php';

if (!isset($_SESSION['email']) || $_SESSION['role'] !== 'enseignant') {
    header("Location: unauthorized.php");
    exit();
}

$email = $_SESSION['email'];
$stmt = $conn->prepare("SELECT id FROM accounts WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$user = $stmt->get_result()->fetch_assoc();
$stmt->close();
$enseignant_id = $user['id'];

$specialites = [];
$stmt = $conn->prepare("SELECT id, nom_specialite FROM specialite ORDER
BY nom_specialite");
$stmt->execute();
$result = $stmt->get_result();
while ($row = $result->fetch_assoc()) {
    $specialites[] = $row;
}
$stmt->close();
```

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $titre = trim($_POST['titre'] ?? '');
    $description = trim($_POST['description'] ?? '');
    $date_exam = $_POST['date_exam'] ?? '';
    $specialite_id = (int)($_POST['specialite_id'] ?? 0);

    $errors = [];
    if (empty($titre)) $errors[] = "Titre requis.";
    if (!preg_match("/^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}$/", $date_exam)) $errors[] = "Date invalide.";
    if ($specialite_id <= 0) $errors[] = "Spécialité invalide.";
    if (!isset($_FILES['fichier']) || $_FILES['fichier']['error'] !== UPLOAD_ERR_OK) {
        $errors[] = "Fichier requis.";
    }

    if (empty($errors)) {
        $file = $_FILES['fichier'];
        $allowed = ['pdf', 'doc', 'docx'];
        $ext = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));
        if (!in_array($ext, $allowed)) {
            $errors[] = "Type de fichier non autorisé.";
        } elseif ($file['size'] > 5 * 1024 * 1024) {
            $errors[] = "Fichier trop volumineux (max 5MB).";
        } else {
            $filename = uniqid('exam_') . '.' . $ext;
            $upload_path = 'uploads/' . $filename;
            if (move_uploaded_file($file['tmp_name'], $upload_path)) {
                $stmt = $conn->prepare("INSERT INTO examen (titre, description, date_exam, enseignant_id, specialite_id, fichier) VALUES (?, ?, ?, ?, ?, ?)");
                $stmt->bind_param("sssiss", $titre, $description, $date_exam, $enseignant_id, $specialite_id, $filename);
```

```
            if ($stmt->execute()) {
                $success_message = "Examen déposé avec succès.";
            } else {
                error_log("SQL Error: " . $stmt->error);
                $errors[] = "Erreur lors du dépôt.";
                unlink($upload_path);
            }
            $stmt->close();
        } else {
            $errors[] = "Erreur lors de l'upload.";
        }
    }
  }
}
?>

<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Déposer un Examen</title>
  <?php include 'navtop.php'; ?>
</head>
<body>
  <div class="wrapper">
    <?php include 'sidenav.php'; ?>
    <div class="main-panel">
      <div class="container">
        <h2>Déposer un Examen</h2>
```

```
        <?php if (isset($success_message)): ?>
            <div class="alert alert-success"><?php echo htmlspecialchars($success_message); ?></div>
        <?php endif; ?>
        <?php if (!empty($errors)): ?>
            <div class="alert alert-error"><?php echo htmlspecialchars(implode('<br>', $errors)); ?></div>
        <?php endif; ?>
        <form method="POST" action="deposer_examen.php" enctype="multipart/form-data">
            <div class="form-group">
                <label>Titre</label>
                <input type="text" name="titre" required>
            </div>
            <div class="form-group">
                <label>Description</label>
                <textarea name="description"></textarea>
            </div>
            <div class="form-group">
                <label>Date de l'examen</label>
                <input type="datetime-local" name="date_exam" required>
            </div>
            <div class="form-group">
                <label>Spécialité</label>
                <select name="specialite_id" required>
                    <option value="">Sélectionner</option>
                    <?php foreach ($specialites as $spec): ?>
                        <option value="<?php echo htmlspecialchars($spec['id']); ?>">
                            <?php echo htmlspecialchars($spec['nom_specialite']); ?>
                        </option>
                    <?php endforeach; ?>
```

```
            </select>
        </div>
        <div class="form-group">
            <label>Fichier (PDF, DOC, DOCX)</label>
            <input type="file" name="fichier" accept=".pdf,.doc,.docx" required>
        </div>
        <button type="submit" class="submit-btn">Déposer</button>
    </form>
</div>
<?php include 'footer.php'; ?>
</div>
</div>
</body>
</html>
```

Sécurité : Requêtes préparées, validation des fichiers, nettoyage des uploads échoués.

5.6. modifier_*.php et supprimer_*.php

Objectif : Modifier ou supprimer des entités sécurisées.

Code

```
<?php
// modifier_etudiant.php
session_start();
require_once 'connexion.php';

if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
    header("Location: unauthorized.php");
    exit();
```

```
}

$id = (int)($_GET['id'] ?? 0);
if ($id <= 0) {
    header("Location: liste_etudiants.php");
    exit();
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $nom = trim($_POST['nom'] ?? '');
    $prenom = trim($_POST['prenom'] ?? '');
    $email = filter_var($_POST['email'] ?? '', FILTER_VALIDATE_EMAIL);

    $errors = [];
    if (empty($nom)) $errors[] = "Nom requis.";
    if (empty($prenom)) $errors[] = "Prénom requis.";
    if (!$email) $errors[] = "Email invalide.";

    if (empty($errors)) {
        $stmt = $conn->prepare("UPDATE etudiants SET nom = ?, prenom = ?, email = ? WHERE id = ?");
        $stmt->bind_param("sssi", $nom, $prenom, $email, $id);
        if ($stmt->execute()) {
            header("Location: liste_etudiants.php?success=1");
            exit();
        } else {
            error_log("SQL Error: " . $stmt->error);
            $errors[] = "Erreur lors de la modification.";
        }
        $stmt->close();
    }
```

```
}

$stmt = $conn->prepare("SELECT nom, prenom, email FROM etudiants WHERE id = ?");
$stmt->bind_param("i", $id);
$stmt->execute();
$etudiant = $stmt->get_result()->fetch_assoc();
$stmt->close();
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modifier Étudiant</title>
    <?php include 'navtop.php'; ?>
</head>
<body>
    <div class="wrapper">
        <?php include 'sidenav.php'; ?>
        <div class="main-panel">
            <div class="container">
                <h2>Modifier Étudiant</h2>
                <?php if (!empty($errors)): ?>
                    <div class="alert alert-error"><?php echo htmlspecialchars(implode('<br>', $errors)); ?></div>
                <?php endif; ?>
                <form method="POST" action="modifier_etudiant.php?id=<?php echo $id; ?>">
                    <div class="form-group">
```

```
            <label>Nom</label>
            <input type="text" name="nom" value="<?php echo htmlspecialchars($etudiant['nom'] ?? ''); ?>" required>
        </div>
        <div class="form-group">
            <label>Prénom</label>
            <input type="text" name="prenom" value="<?php echo htmlspecialchars($etudiant['prenom'] ?? ''); ?>" required>
        </div>
        <div class="form-group">
            <label>Email</label>
            <input type="email" name="email" value="<?php echo htmlspecialchars($etudiant['email'] ?? ''); ?>" required>
        </div>
        <button type="submit" class="submit-btn">Modifier</button>
      </form>
    </div>
    <?php include 'footer.php'; ?>
  </div>
 </div>
</body>
</html>
```

php

```
<?php
// supprimer_etudiant.php
session_start();
require_once 'connexion.php';

if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
```

```
    header("Location: unauthorized.php");
    exit();
}

$id = (int)($_GET['id'] ?? 0);
if ($id <= 0) {
    header("Location: liste_etudiants.php");
    exit();
}

$stmt = $conn->prepare("DELETE FROM etudiants WHERE id = ?");
$stmt->bind_param("i", $id);
if ($stmt->execute()) {
    header("Location: liste_etudiants.php?success=1");
} else {
    error_log("SQL Error: " . $stmt->error);
    header("Location: liste_etudiants.php?error=1");
}
$stmt->close();
$conn->close();
?>
```

Sécurité : Requêtes préparées, validation d'ID, restriction d'accès.

5.7. Autres Fichiers

Liste (liste_*.php, notes.php) :

php

```
$stmt = $conn->prepare("SELECT id, nom, prenom FROM etudiants WHERE specialite_id = ?");
$stmt->bind_param("i", $specialite_id);
```

Inscriptions (inscription.php, inscription_specialite.php) :

php

```
$stmt = $conn->prepare("INSERT INTO inscriptions (etudiant_id, specialite_id) VALUES (?, ?)");

$stmt->bind_param("ii", $etudiant_id, $specialite_id);
```

AJAX (data.php) :

```
$specialite_id = (int)($_POST['specialite_id'] ?? 0);

$stmt = $conn->prepare("SELECT id, nom FROM etudiants WHERE specialite_id = ?");

$stmt->bind_param("i", $specialite_id);
```

9. Conclusion

En appliquant ces recommandations, le projet sera protégé contre les injections SQL grâce à l'utilisation systématique de requêtes préparées, la validation rigoureuse des entrées, et une gestion sécurisée des fichiers. Des tests réguliers et une configuration appropriée du serveur MySQL garantiront une sécurité durable.