# Passman Project 2024

Παπαδάκης Κωνσταντίνος Φώτιος

AEM:10371

26 Ιανουαρίου 2025

# Περιγραφή προβλήματος

Η εφαρμογή *Passman* αποτελεί μια απλουστευμένη υλοποίηση διαχειριστή κωδικών. Στα πλαίσια της εργασίας μας καλούμαστε να επισημάνουμε τα κενά ασφαλείας της εν λόγω εφαρμογής, να παρουσιάσουμε ευπάθειες, να προτείνουμε τρόπους αντιμετώπισης και να εφαρμόσουμε τις απαραίτητες αλλαγές, όπου αυτό είναι δυνατόν.

# Περιγραφή διορθώσεων

Για να ανταπεξέλθουμε στις απαιτήσεις της εργασίες επιστρατεύσαμε διάφορες τακτικές τις οποίες θα εξηγήσουμε στη συνέχεια.

- **Περιορισμοί χρηστών βάσης:** Συνδεόμαστε στη βάση ως ένας χρήστης με περιορισμένα δικαιώματα.
- **Κρυπτογράφηση κωδικών:** Χρησιμοποιούμε *hash* αντί να παραθέτουμε τους κωδικούς στη βάση ως απλό κείμενο.
- **Παραμετροποιημένα ερωτήματα:** Προστασία από *SQL Injections*.
- **Καθαρισμός εισόδου:** Προστασία από *XSS* επιθέσεις.

# Παραδείγματα εκμετάλλευσης ευπαθειών

Παράδειγμα επίθεσης *XSS*:

```
<script>alert('XSS Attack example!');</script>
<-- Instead now it is going to be displayed as the following text: -->
<-- &lt;script&gt;alert('XSS Attack example!')&lt;/script&gt; -->
```

# Περιγραφή Αλλαγών Κώδικα

Στη συνέχεια πρόκειται να παραθέσουμε τον παλιό μαζί με τον καινούριο κώδικα και να εστιάζοντας στις αλλαγές που πραγματοποιήσαμε καθώς και το νόημα αυτών.

## *connection.php*

Εδώ προδιαγράφουμε τη σύνδεση με τη βάση δεδομένων μας.

```
$servername = "localhost";
$username = "limited_user"; // Use a non-admin user
$password = "limited_password";
$dbname = "pwd_mgr";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
```

Έτσι οι χρήστες αλληλεπιδρούν με τη βάση με τα προνόμια του χρήστη *limited_user* και όχι του *root*. Η δημιουργία του χρήστη *limited_user* και η ανάθεση των *privileges* του γίνεται ως εξής:

```
CREATE USER 'limited_user'@'localhost' IDENTIFIED BY 'limited_password';
GRANT SELECT, INSERT, UPDATE, DELETE ON pwd_mgr.* TO 'limited_user'@'localhost';
```

Του δίνουμε πρόσβαση μόνο στη βάση *pwd_mgr* και μόνο για τις $SELECT$, $INSERT$, $UPDATE$ και $DELETE$ ενέργειες.

## *register.php*

Η πρώτη αλλαγή στο συγκεκριμένο αρχείο απαιτεί τον καθαρισμό του ονόματος χρήστη και του κωδικού από ειδικούς χαρακτήρες στην αρχή και το τέλος του αλφαριθμητικού κατά την εισαγωγή τους στη βάση.

```
// Get user submitted information
$new_username = trim($_POST['new_username']);
$new_password = trim($_POST['new_password']);
```

Σημαντικό είναι να σημειωθεί ότι στο ενδιάμεσο γίνεται και η κρυπτογράφηση του κωδικού.

```
$hashed_password = password_hash($new_password, PASSWORD_BCRYPT);
```

Στη συνέχεια ¨προετοιμάζουμε' το παραμετροποιημένο ερώτημά μας προστατεύοντας από $SQL\ Injections$. Ουσιαστικά αυτό καθιστά αδύνατον να ερμηνευτεί η είσοδος του χρήστη ως $SQL$ εντολή.

```
$stmt = $conn->prepare("INSERT INTO login_users (username, password) VALUES (?, ?)");
$stmt->bind_param("ss", $new_username, $hashed_password);
```

### *login.php*

Ομοίως, και εδώ καθαρίζουμε και παραμετροποιούμε το ερώτημά μας ενώ παράλληλα επαληθεύουμε τον κωδικό του χρήστη μέσω του *hash* του.
**Καθαρισμός εισόδου:**

```
// Get user submitted information
$username = trim($_POST['username']);
$password = trim($_POST['password']);
```

**Παραμετροποίηση:**

```
// Prepare an SQL query
$stmt = $conn->prepare("SELECT password FROM login_users WHERE username = ?");
$stmt->bind_param("s", $username);
$stmt->execute();
$stmt->store_result();
```

**Επαλήθευση κωδικού:**

```
if ($stmt->num_rows > 0) {
    $stmt->bind_result($hashed_password);
    $stmt->fetch();

    if (password_verify($password, $hashed_password)) {
        $_SESSION['username'] = $username;
        $_SESSION['loggedin'] = true;
        header("Location: dashboard.php");
        exit;
    } else {
        $login_message = "Invalid username or password";
    }
} else {
    $login_message = "Invalid username or password";
}
```

### *dashboard.php*

Στον κώδικα της κεντρικής σελίδας καθαρίζουμε την είσοδο του χρήστη πριν την αποθηκεύσουμε στη βάση. Επειδή πρόκειται να προβάλουμε τα στοιχεία αυτά στην σελίδα μέσω *html* είναι απαραίτητο να τα περάσουμε και από την συνάρτηση *htmlspecialchars* για να αποφύγουμε τυχόν επιθέσεις $XSS$ όπου εκτελείται κώδικας *javascript* μέσα στη σελίδα:

```
$new_website = htmlspecialchars(trim($_POST["new_website"]), ENT_QUOTES, 'UTF-8');
$new_username = htmlspecialchars(trim($_POST["new_username"]), ENT_QUOTES, 'UTF-8');
$new_password = htmlspecialchars(trim($_POST["new_password"]), ENT_QUOTES, 'UTF-8');
```

Παραμετροποιούμε το ερώτημα:

```
$stmt = $conn->prepare("INSERT INTO websites (login_user_id, web_url, web_username, web_password) V
$stmt->bind_param("ssss", $username, $new_website, $new_username, $new_password);
```

Ομοίως για τα ερωτήματα $DELETE$ και $SELECT$.

*notes.php*

Τελευταία έχουμε το αρχείο υπεύθυνο για την διαχείρηση σημειώσεων από τον χρήστη. Επαναλαμβάνονται οι ίδιες τεχνικές που είδαμε παραπάνω:

**Καθαρισμός εισόδου:**

```
$new_note = htmlspecialchars(trim($_POST["new_note"]), ENT_QUOTES, 'UTF-8');
```

**Παραμετροποίηση:**

```
$stmt = $conn->prepare("INSERT INTO notes (login_user_id, note) VALUES ((SELECT id FROM login_users
$stmt->bind_param("ss", $username, $new_note);
```

# Σχόλια