Passman Project 2024

Παπαδάκης Κωνσταντίνος Φώτιος

kpapadak@ece.auth.gr

AEM:10371

27 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* υπό την έλλειψη καθαρισμού της εισόδου:

```
<!-- For user input on the dashboard page such as: -->
<script>alert('XSS Attack example!');</script>
<!-- we can execute javascript code -->
<!-- Instead now, after sanitization: -->
$new_website = htmlspecialchars(trim($_POST["new_website"]), ENT_QUOTES, 'UTF-8');
<!-- it is going to be displayed as the following text: -->
<!-- &lt;script&gt;alert('XSS Attack example!')&lt;/script&gt; -->
```

Παράδειγμα επίθεσης *SQL Injection* χωρίς παραμετροποίηση:

```
SELECT * FROM login_users WHERE username = 'admin' OR '1' = '1';
-- This will return all the users in the database
-- Can be avoided through proper parameterization
```

Παράδειγμα εκμετάλλευσης πρακτικής μη κρυπτογράφησης κωδικών:

```
SELECT * FROM login_users;
-- This will return all the users and their passwords in the database
-- Can be avoided by hashing the passwords before storing them
```

Παράδειγμα επίθεσης χρήστη με όλα τα δικαιώματα:

```
DROP DATABASE pwd_mgr;
-- This will delete the entire database and all its data
-- Can be avoided by using a user with limited privileges
```

Περιγραφή Αλλαγών Κώδικα

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

connection.php

Εδώ προδιαγράφουμε τη σύνδεση με τη βάση δεδομένων μας.

```
<?php
$servername = "localhost";
$username = "limited_user"; // Use a non-admin user
$password = "limited_password";
```

```php
$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';
-- Also laying the groundwork to later incorporate hashing
ALTER TABLE pwd_mgr.login_users MODIFY password VARCHAR(255);
```

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.

```php
<?php
// 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 έτσι ώστε να μην μπορεί να αξιοποιηθεί από κάποιον επιτιθέμενο η πληροφορία αυτή. Για να χωράει ο *hashed* κωδικός εντός της βάσης αλλάζουμε τον τύπο δεδομένων του πεδίου *password* σε $VARCHAR(255)$, κάτι το οποίο είδαμε μεταξύ των αλλαγών που κάναμε στη βάση δεδομένων.

```php
<?php
$hashed_password = password_hash($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.

```php
<?php
$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:

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

Παραμετροποίηση:

```php
<?php
// 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:

```
<?php
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:

```
<?php
$new_website = htmlspecialchars(trim($_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:

```
<?php
$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:

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

Παραμετροποίηση:

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