ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ: ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ασφάλεια Πληροφοριακών Συστημάτων

Εργασία 2025-2026

Παπαδόπουλος Παναγιώτης
10697
ppapadoe@ece.auth.gr

1. Εισαγωγή

1.1 Περιγραφή του Προβλήματος

Στα πλαίσια της παρούσας εργασίας, δόθηκε προς εξέταση μια διαδικτυακή εφαρμογή διαχείρισης κωδικών πρόσβασης (Password Manager) υλοποιημένη σε PHP και MySQL. Η αρχική έκδοση της εφαρμογής ("Passman") σχεδιάστηκε σκόπιμα με πολλαπλές ευπάθειες ασφαλείας, οι οποίες καθιστούσαν τα δεδομένα των χρηστών ευάλωτα σε υποκλοπή και αλλοίωση.

Σκοπός της εργασίας ήταν:

1. Ο εντοπισμός και η τεκμηρίωση των κενών ασφαλείας μέσω τεχνικών ελέγχου διείσδυσης (Penetration Testing).
2. Ο σχεδιασμός και η εφαρμογή διορθωτικών ενεργειών στον πηγαίο κώδικα (Remediation).
3. Η επαλήθευση της ασφάλειας του τελικού συστήματος.

1.2 Μεθοδολογία

Η προσέγγιση που ακολουθήθηκε βασίστηκε σε δύο φάσεις:

- **Φάση Ανάλυσης (Black-box & White-box Testing):** Αρχικά πραγματοποιήθηκαν επιθέσεις στην εφαρμογή για την επιβεβαίωση των ευπαθειών και στη συνέχεια μελετήθηκε ο πηγαίος κώδικας για τον εντοπισμό των αιτιών.
- **Φάση Υλοποίησης:** Εφαρμόστηκαν βέλτιστες πρακτικές ασφαλούς προγραμματισμού (Secure Coding Practices) για την εξάλειψη των κινδύνων.

2. Ανάλυση Ευπαθειών (Problem Detection)

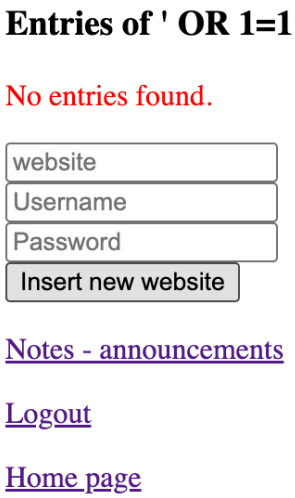
Στην ενότητα αυτή παρουσιάζονται τα προβλήματα που εντοπίστηκαν, συνοδευόμενα από τον ευπαθή κώδικα και τα αποτελέσματα της εκμετάλλευσής τους.

2.1 SQL Injection (SQLi) - Παράκαμψη Αυθεντικοποίησης

Κατά τον έλεγχο της φόρμας εισόδου (login.php), διαπιστώθηκε ότι η εφαρμογή κατασκεύαζε το ερώτημα SQL συνενώνοντας απευθείας τις εισόδους του χρήστη χωρίς έλεγχο.

- **Ευπαθές Αρχείο:** login.php
- **Απόσπασμα Κώδικα:**
  `$sql_query = "SELECT * FROM login_users WHERE username='{$username}' AND password='{$password}';";`
- **Σενάριο Επίθεσης:** Εισαγωγή της συμβολοσειράς `' OR 1=1 #` στο πεδίο Username.
- **Αποτέλεσμα:** Το σύστημα ταυτοποίησε τον εισβολέα ως τον πρώτο χρήστη της βάσης (Administrator) χωρίς να απαιτηθεί κωδικός πρόσβασης.

Εικόνα 2.1: Επιτυχής παράκαμψη εισόδου με SQL Injection.



2.2 Stored Cross-Site Scripting (XSS)

Στη σελίδα ανακοινώσεων (notes.php), η εφαρμογή επέτρεπε την εισαγωγή και αποθήκευση ακατέργαστου κώδικα HTML/JavaScript στη βάση δεδομένων. Κατά την προβολή των σημειώσεων, ο κώδικας αυτός εκτελούνταν στον περιηγητή κάθε επισκέπτη.

- **Ευπαθές Αρχείο:** notes.php
- **Απόσπασμα Κώδικα:**
  `echo "<div class='note-content'>" . $row["note"] . "</div>";`
- **Σενάριο Επίθεσης:** Εισαγωγή κακόβουλου script: `<script>alert('XSS Attack');</script>`.

- **Αποτέλεσμα:** Εμφάνιση αναδυόμενου παραθύρου (alert box) σε όλους τους χρήστες, αποδεικνύοντας τη δυνατότητα εκτέλεσης κώδικα (π.χ. για υποκλοπή cookies).

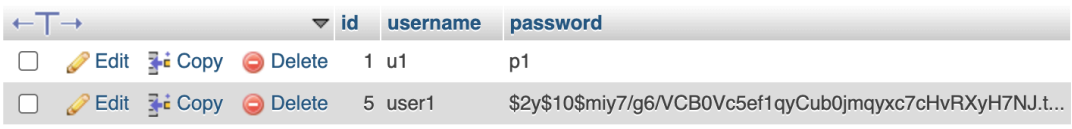
Εικόνα 2.2: Εκτέλεση κακόβουλου κώδικα JavaScript (XSS).

2.3 Μη Ασφαλής Αποθήκευση Κωδικών (Plaintext Storage)

Μέσω ελέγχου της βάσης δεδομένων, διαπιστώθηκε ότι οι κωδικοί πρόσβασης αποθηκεύονταν ως απλό κείμενο.

- **Ευπαθές Αρχείο:** register.php
- **Απόσπασμα Κώδικα:**
```
$sql_query = "INSERT INTO login_users (username,password) VALUES ('{$new_username}','{$new_password}');";
```
- **Αποτέλεσμα:** Σε περίπτωση διαρροής της βάσης δεδομένων, όλοι οι λογαριασμοί χρηστών θα παραβιάζονταν άμεσα.

Εικόνα 2.3: Αποθήκευση κωδικών σε απλή μορφή.



2.4 Παραβίαση Αρχής Ελαχίστων Προνομίων

Η εφαρμογή συνδεόταν στη βάση δεδομένων χρησιμοποιώντας τον λογαριασμό root χωρίς κωδικό πρόσβασης σε όλα τα αρχεία PHP. Αυτό σημαίνει ότι οποιαδήποτε επιτυχής επίθεση SQL Injection θα έδινε στον επιτιθέμενο πλήρη έλεγχο ολόκληρου του διακομιστή βάσης δεδομένων (όχι μόνο της συγκεκριμένης εφαρμογής).

3. Προτεινόμενες Λύσεις & Υλοποίηση (Solution)

Για την αντιμετώπιση των ανωτέρω προβλημάτων, πραγματοποιήθηκαν οι παρακάτω δομικές και λογικές αλλαγές.

3.1 Κεντρική Διαχείριση Σύνδεσης Βάσης Δεδομένων

Δημιουργήθηκε ένα νέο αρχείο (db_connect.php) για την αποφυγή επαναλαμβανόμενου κώδικα και τη χρήση ασφαλούς χρήστη.

- **Αλλαγή:** Αντικατάσταση του χρήστη root με έναν περιορισμένο χρήστη (app_user) που έχει δικαιώματα μόνο *SELECT*, *INSERT*, *UPDATE*, *DELETE* στη βάση pwd_mgr.

- **Κώδικας (db_connect.php):**

```
$conn = new mysqli("localhost", "app_user", "secure_password", “pwd_mgr");
if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error); }
```

3.2 Χρήση Prepared Statements

Σε όλα τα σημεία αλληλεπίδρασης με τη βάση δεδομένων (login.php, register.php, dashboard.php, notes.php), αντικαταστάθηκε η απλή εκτέλεση ερωτημάτων με **Prepared Statements**. Αυτό διαχωρίζει πλήρως τα δεδομένα από τις εντολές SQL, εξαλείφοντας τον κίνδυνο SQL Injection.

- **Παράδειγμα Υλοποίησης (login.php):**

```
// SECURITY FIX: Use Prepared Statement
$stmt = $conn->prepare("SELECT id, username, password FROM login_users WHERE username = ?");
$stmt->bind_param("s", $username);
$stmt->execute();
```

3.3 Κρυπτογράφηση Κωδικών (Hashing)

Εφαρμόστηκε η συνάρτηση password_hash() (με αλγόριθμο bcrypt) κατά την εγγραφή και η password_verify() κατά την είσοδο. Πλέον, στη βάση αποθηκεύεται μόνο το hash του κωδικού και όχι ο ίδιος ο κωδικός.

- **Παράδειγμα Υλοποίησης (register.php):**
```
// SECURITY FIX: Hash the password
$hashed_password = password_hash($new_password, PASSWORD_DEFAULT);
$stmt->bind_param("ss", $new_username, $hashed_password);
```

3.4 Εξυγίανση Εξόδου (Output Encoding)

Για την πρόληψη επιθέσεων XSS, εφαρμόστηκε η συνάρτηση htmlspecialchars() πριν την εμφάνιση δεδομένων χρήστη στην οθόνη. Αυτή η συνάρτηση μετατρέπει ειδικούς χαρακτήρες (όπως <, >, ") σε ασφαλείς οντότητες HTML, εμποδίζοντας την εκτέλεση σεναρίων.

- **Παράδειγμα Υλοποίησης (notes.php):**
```
// SECURITY FIX: Output Encoding
safe_note = htmlspecialchars($row["note"], ENT_QUOTES, 'UTF-8');
echo "<div class='note-content'>" . $safe_note . "</div>";
```

4. Αρχεία που Τροποποιήθηκαν

Συνοπτικά, τροποποιήθηκαν ή δημιουργήθηκαν τα ακόλουθα αρχεία για την επίτευξη της ασφάλειας:

1. **db_connect.php (Νέο):** Περιέχει τις ρυθμίσεις σύνδεσης με τη ΒΔ.
2. **register.php:** Προσθήκη hashing κωδικών και Prepared Statements.
3. **login.php:** Έλεγχος hash κωδικών (αντί plaintext) και Prepared Statements.
4. **dashboard.php:** Εφαρμογή Prepared Statements για εισαγωγή/διαγραφή ιστοσελίδων και htmlspecialchars στην προβολή.

5. **notes.php:** Εφαρμογή Prepared Statements στην εισαγωγή και αυστηρή χρήση htmlspecialchars στην προβολή για αποτροπή XSS.

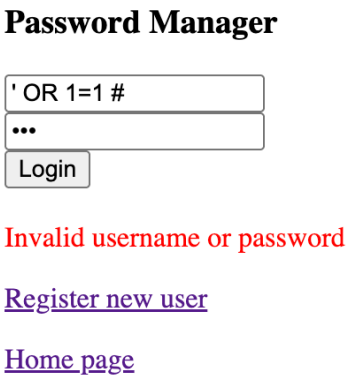
Σημείωση: Τα αρχεία στον φάκελο xss/ και τα βοηθητικά αρχεία test_*.php δεν τροποποιήθηκαν, σύμφωνα με τις οδηγίες της εκφώνησης.

5. Έλεγχος και Επαλήθευση (Testing)

Μετά την εφαρμογή των διορθώσεων, επαναλήφθηκαν τα σενάρια επίθεσης για την επιβεβαίωση της ασφάλειας.

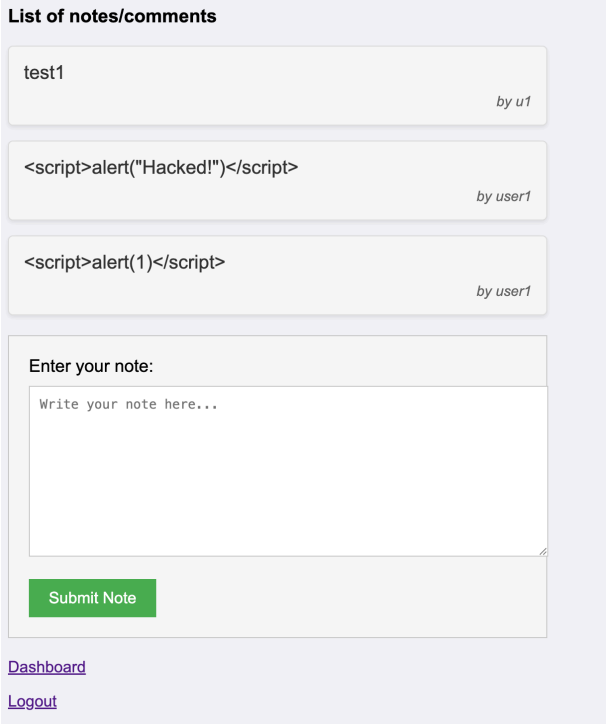
1. **Έλεγχος SQL Injection:**
   - Η εισαγωγή ' OR 1=1 # στο πεδίο Login απέτυχε. Η εφαρμογή εμφάνισε το μήνυμα "Invalid username or password", καθώς το σύστημα αναζήτησε χρήστη με αυτό το κυριολεκτικό όνομα.

Εικόνα 5.1: Η εφαρμογή απορρίπτει την SQL Injection επίθεση.



2. **Έλεγχος XSS:**
   - Νέα σημείωση με περιεχόμενο *<script>alert(1)</script>* καταχωρήθηκε επιτυχώς.
   - Κατά την προβολή, ο κώδικας εμφανίστηκε ως απλό κείμενο στην οθόνη και **δεν εκτελέστηκε**.

Εικόνα 5.2: Ο κακόβουλος κώδικας εμφανίζεται ως απλό κείμενο (Sanitization).



3. **Έλεγχος Βάσης Δεδομένων:**

   - Εγγραφή νέου χρήστη και έλεγχος μέσω HeidiSQL. Επιβεβαιώθηκε ότι στο πεδίο password υπάρχει πλέον μια κρυπτογραφημένη συμβολοσειρά (hash) και όχι το κείμενο του κωδικού.

Εικόνα 5.3: Οι κωδικοί αποθηκεύονται κρυπτογραφημένοι.

		id	username	password
☐	Edit Copy Delete	1	u1	p1
☐	Edit Copy Delete	5	user1	$2y$10$miy7/g6/VCB0Vc5ef1qyCub0jmqyxc7cHvRXyH7NJ.t...

6. Περιορισμοί της Λύσης

Παρόλο που η νέα υλοποίηση καλύπτει τις βασικές απαιτήσεις ασφαλείας της άσκησης, υπάρχουν ορισμένοι περιορισμοί που οφείλονται είτε στη φύση του τοπικού περιβάλλοντος ανάπτυξης είτε στο εύρος της εργασίας:

- **Έλλειψη HTTPS:** Η εφαρμογή τρέχει σε τοπικό περιβάλλον HTTP (localhost). Σε πραγματικές συνθήκες, η χρήση TLS/SSL είναι απαραίτητη για την κρυπτογράφηση της κίνησης δικτύου.
- **Προστασία CSRF:** Δεν υλοποιήθηκαν μηχανισμοί προστασίας (π.χ. CSRF tokens) για την αποτροπή πλαστογράφησης αιτημάτων μεταξύ ιστοτόπων, καθώς δεν ήταν μέρος των ζητουμένων.
- **Πολιτική Κωδικών:** Δεν εφαρμόστηκε έλεγχος πολυπλοκότητας (password strength policies) κατά την εγγραφή χρηστών.