Laboratorium 9

Ataki na skrypty PHP cz. 1

W sprawozdaniu zamieść zrzuty ekranu z istotnych etapów oraz odpowiedzi na pytania.

Taki symbol oznacza, że trzeba w sprawozdaniu dodać zrzut ekranu (najczęściej 1) z wyniku działania polecenie.

Taki symbol oznacza, że należy dodać opis (najczęściej 1 zdanie) z wyniku działania polecenia.

Celem niniejszego laboratorium jest zapoznanie się z najczęstszymi podatnościami aplikacji webowych napisanym w języku PHP, takimi jak SQL Injection oraz Cross-Site Scripting (XSS). Poprzez praktyczne odwzorowanie błędów programistycznych w kontrolowanym środowisku, nauczysz się, w jaki sposób hakerzy mogą przejąć kontrolę nad bazą danych lub sesją użytkownika. Zrozumienie mechanizmu ataku jest kluczowym krokiem do nauki poprawnego zabezpieczania kodu źródłowego.

XAMPP to darmowy, wieloplatformowy pakiet, który pozwala w kilka minut zamienić Twój komputer (w tym przypadku Kali Linux) w lokalny serwer testowy. Instaluje się go, ponieważ do uruchomienia skryptów z instrukcji potrzebujesz całego ekosystemu współpracujących ze sobą narzędzi.

I. Przygotowanie maszyny

1. Uruchom maszynę wirtualną z systemem Kali Linux.

2. Otwórz przeglądarkę internetową i przejdź na stronę XAMPP.

3. Pobierz wersję XAMPP przeznaczoną dla systemu Linux.

4. Otwórz terminal w systemie Kali Linux.

5. Przejdź do katalogu Downloads (Pobrane), wpisując polecenie:

```
cd Downloads
```

6. Nadaj pobranemu plikowi instalacyjnemu uprawnienia do uruchomienia, zastępując `nazwa-pliku-instalatora.run` właściwą nazwą pliku:

```
sudo chmod +x nazwa-pliku-instalatora.run
```

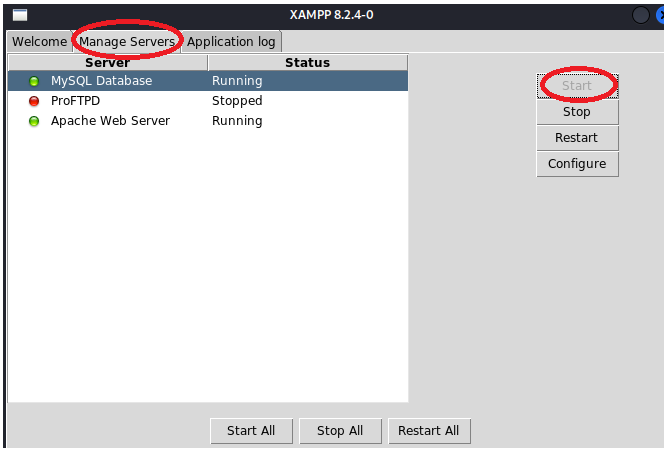
7. Uruchom instalator XAMPP poleceniem:

```
sudo ./nazwa-pliku-instalatora.run
```

8. Przechodź przez kolejne etapy instalacji, klikając przycisk `Next` (nie wprowadzaj żadnych zmian).

9. Po zakończeniu instalacji uruchom aplikację XAMPP.

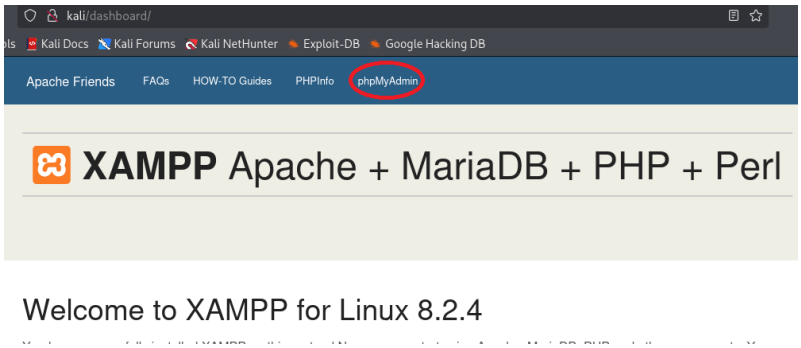
10. Przejdź do zakładki **`Manage Servers`**, wybierz **`MySQL Database`**, a następnie uruchom bazę danych, klikając **`Start`**.



11. Zminimalizuj okno aplikacji XAMPP.

12. W przeglądarce internetowej przejdź pod adres: http://kali/dashboard/.

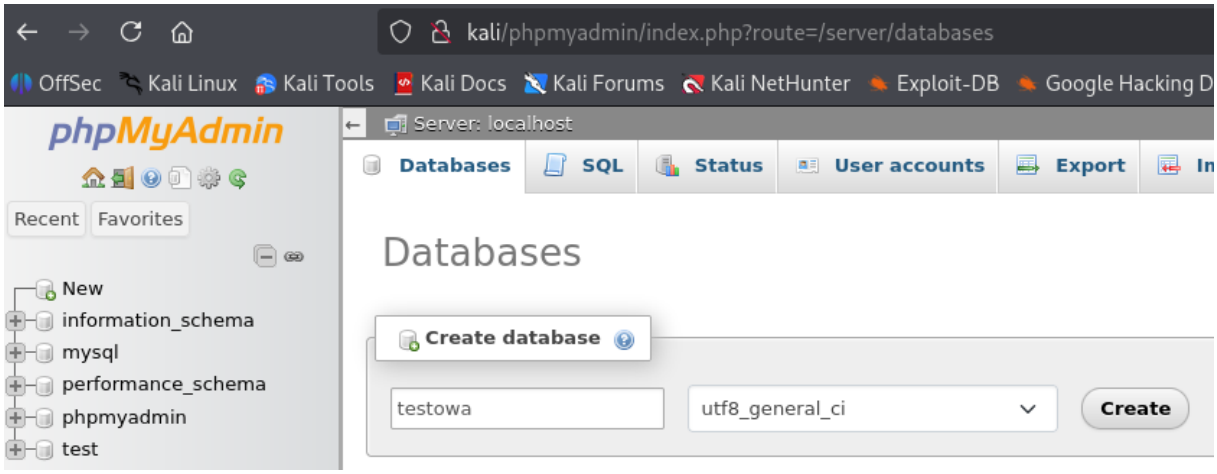
13. Przejdź do zakładki **`phpMyAdmin`**.



14. W lewym panelu kliknij zakładkę **`New`** (Nowa).

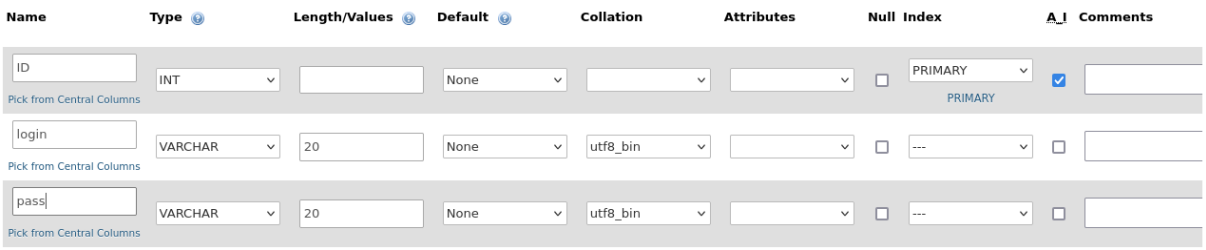
15. W polu **`Nazwa bazy danych`** wpisz **`testowa`**.

16. Jako metodę porównywania napisów wybierz **`utf8_general_ci`**, a następnie kliknij **`Create`**.



17. Po utworzeniu bazy danych utwórz w niej tabelę o nazwie **`users`** z trzema kolumnami.

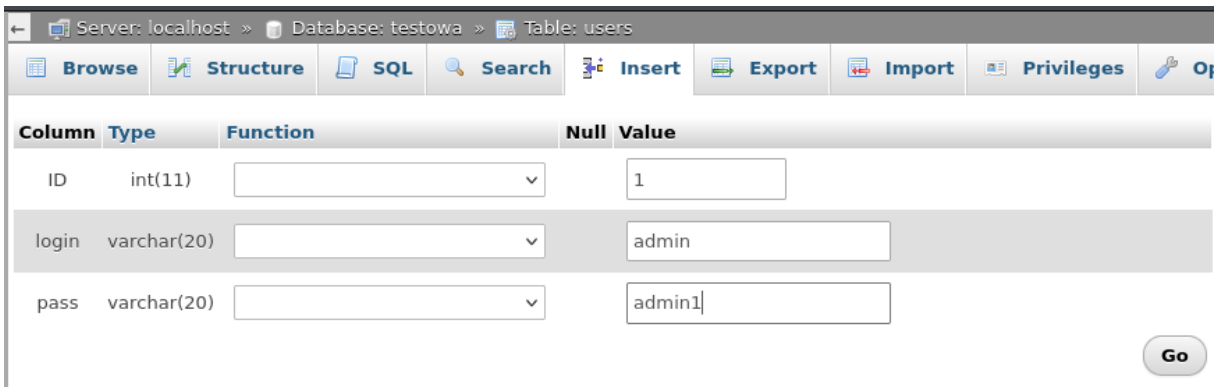
18. W nowym oknie skonfiguruj strukturę tabeli, tak jak poniżej i kliknij **`Save`**:



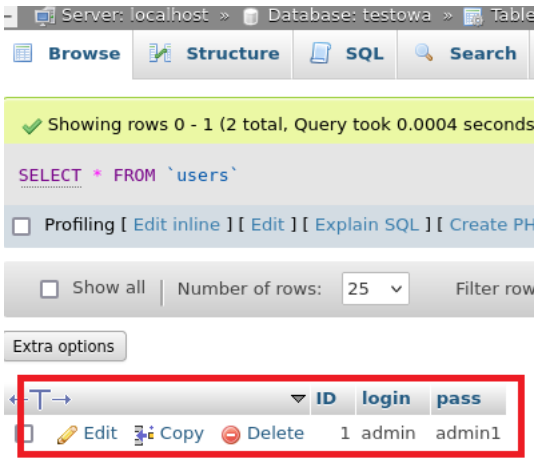
19. Kliknij tabelę **`users`** w lewym panelu, a następnie wybierz zakładkę **`Insert`** (Wstaw).

20. Wprowadź następujące dane i kliknij **`Go`**:

- **`id:`** 1
- **`login:`** admin
- **`pass:`** admin1



21. Przejdź do zakładki **`Browse`** i upewnij się, że rekord użytkownika został poprawnie dodany do tabeli.



II. SQL Injection

Atak SQL Injection polega na wstrzyknięciu złośliwego kodu do zapytania wysyłanego do bazy danych SQL. Dzięki wykorzystaniu znaków specjalnych możliwe jest modyfikowanie treści zapytania w taki sposób, aby ominąć mechanizmy zabezpieczeń aplikacji.

1. Wróć do terminala.

2. rzejdź do katalogu **`htdocs`**, który pełni rolę głównego katalogu dokumentów serwera Apache:

```
cd /opt/lampp/htdocs
```

3. Utwórz katalog o nazwie **`ataki`**:

```
sudo mkdir ataki
```

4. Przejdź do nowo utworzonego katalogu:

```
cd ataki
```

5. Utwórz plik **`formularz.html`**:

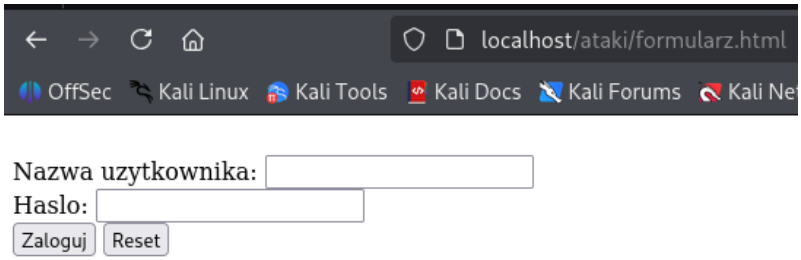
```
sudo nano formularz.html
```

6. Wypełnij plik następującą treścią:

```
<form action="logowanie.php" method="post">
    <br />Nazwa uzytkownika:
    <input type="text" id="username" name="login" />
    <br/>Haslo:
    <input type="password" id="password" name="pass" />
    <br />
    <input type="submit" value="Zaloguj" />
    <input type="reset" value="Reset" />
</form>
```

7. Zapisz plik i wyjdź z edytora.

8. Sprawdź działanie formularza – wpisz w przeglądarce: http://localhost/ataki/formularz.html.



9. Wróć do terminala.

10. Utwórz plik logowanie.php:

```
sudo nano logowanie.php
```

11. Wypełnij plik następującą treścią:

```
<?php

// Włączanie raportowania błędów
ini_set('display_errors', 1);
error_reporting(E_ALL);

$hostname = 'localhost';
$user = 'root';
$password = '';
$database = 'testowa';

// Połączenie z bazą danych
$link = mysqli_connect($hostname, $user, $password, $database);

if (!$link) {
    die("Błąd połączenia: " . mysqli_connect_error());
}

// Odbieranie danych z formularza - BEZ FILTROWANIA (celowo dla testu)
$log = $_POST['login'];
$pas = $_POST['pass'];

// Zapytanie SQL podatne na wstrzyknięcie
$query = "SELECT id FROM users WHERE login = '$log' AND pass = '$pas'";

// Wyświetlamy zapytanie na ekranie, aby widzieć jak działa atak
echo "Wykonane zapytanie: <b>" . $query . "</b><br><br>";

$result = mysqli_query($link, $query);

if ($result && mysqli_num_rows($result) > 0) {
    echo "<h2 style='color:green'>Zalogowano pomyślnie!</h2>";
} else {
    echo "<h2 style='color:red'>Błąd logowania</h2>";
}
mysqli_close($link);

?>
```

12. Zapisz plik i wyjdź.

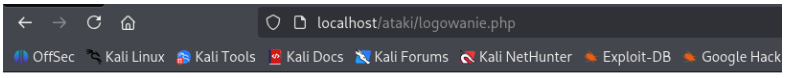
Po przesłaniu danych z pliku formularz.html, dane są przetwarzane przez skrypt logowania. Na początku skryptu definiowane są zmienne umożliwiające połączenie z przykładową bazą danych testowa, utworzoną na lokalnym serwerze przy użyciu programu XAMPP.

W dalszej części skrypt nawiązuje połączenie z bazą danych i wykonuje zapytanie SQL, którego celem jest pobranie ID użytkownika z tabeli users, którego pola login i pass odpowiadają wartościom przesłanym w formularzu.

Na końcu sprawdzane jest, czy zapytanie zwróciło jakiekolwiek wyniki (czy liczba wierszy jest większa niż 0). Jeśli tak, użytkownik zostaje zalogowany. W przedstawionym przykładzie logowanie ogranicza się jedynie do wyświetlenia komunikatu tekstowego. W rzeczywistych zastosowaniach mogłoby to obejmować np. przyznanie odpowiednich uprawnień, ustawienie ciasteczek, przekierowanie na inną stronę lub konfigurację zmiennych sesji.

13. Wróć do przeglądarki i spróbuj się zalogować, używając poprawnych danych.

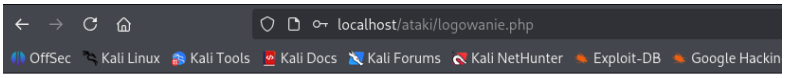
login: `admin` hasło: `admin1`



Wykonane zapytanie: **SELECT id FROM users WHERE login = 'admin' AND pass = 'admin1'**

Zalogowano pomyślnie!

14. Spróbuj zalogować się ponownie, podając błędne hasło.



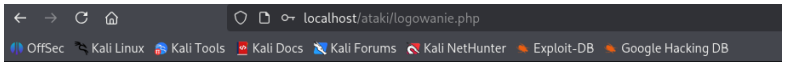
Wykonane zapytanie: **SELECT id FROM users WHERE login = 'admin' AND pass = 'admin1f'**

Błąd logowania

Skrypt działa poprawnie i zabezpiecza nas przed nieautoryzowanym dostępem, ale czy na pewno?

15. Sprawdź, co stanie się, gdy w polu hasła wpiszesz:

```
cokolwiek' or 1 = '1
```



Wykonane zapytanie: **SELECT id FROM users WHERE login = 'admin' AND pass = 'cokolwiek' or 1 = '1'**

Zalogowano pomyślnie!

W tym przypadku aplikacja zwróci komunikat świadczący o poprawnym zalogowaniu, mimo braku znajomości prawidłowego hasła. Jest to przykład skutecznie przeprowadzonego ataku SQL Injection.

O metodach zabezpieczania aplikacji przed tego typu atakami dowiemy się na kolejnych zajęciach.