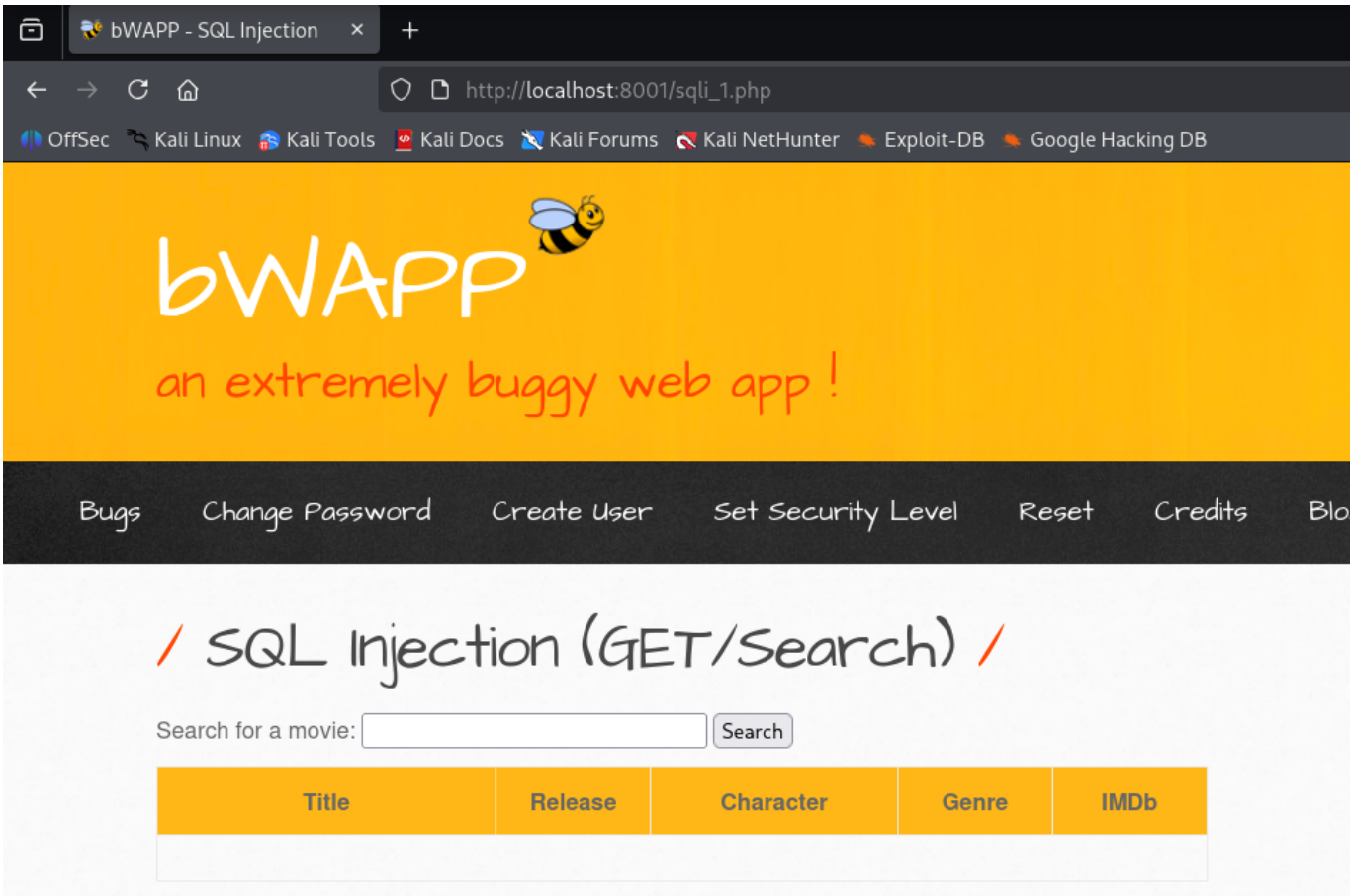
Análisis de Vulnerabilidad - SQL Injection

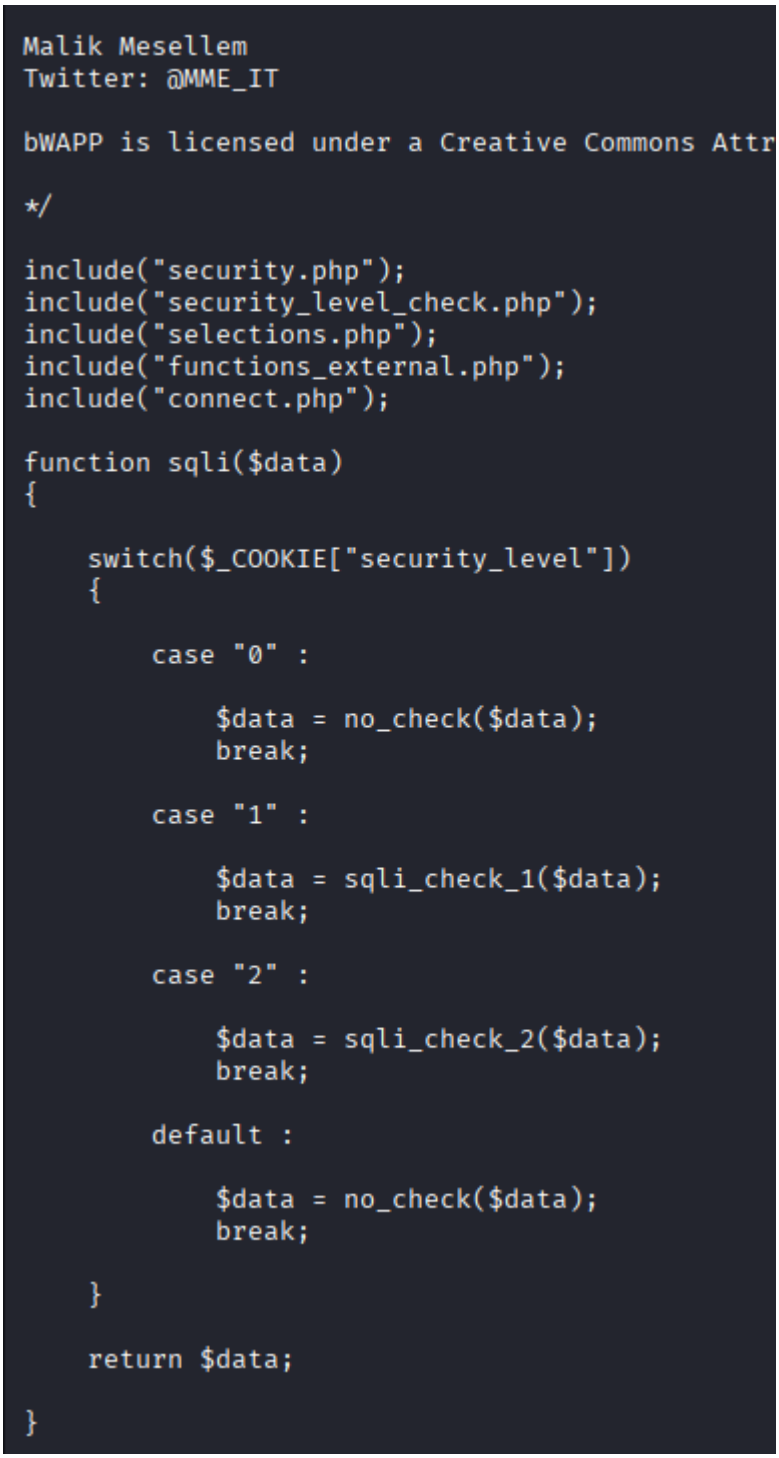
Selección del tipo de ataque



En esta captura se muestra la interfaz donde elegimos el tipo de ataque (SQL Injection) y podemos ver el nombre del fichero vulnerable que será analizado.

Análisis del código fuente

Primera inspección del fichero vulnerable



```
Malik Mesellem
Twitter: @MME_IT

bWAPP is licensed under a Creative Commons Attr

*/

include("security.php");
include("security_level_check.php");
include("selections.php");
include("functions_external.php");
include("connect.php");

function sqli($data)
{

    switch($_COOKIE["security_level"])
    {

        case "0" :

            $data = no_check($data);
            break;

        case "1" :

            $data = sqli_check_1($data);
            break;

        case "2" :

            $data = sqli_check_2($data);
            break;

        default :

            $data = no_check($data);
            break;

    }

    return $data;

}
```

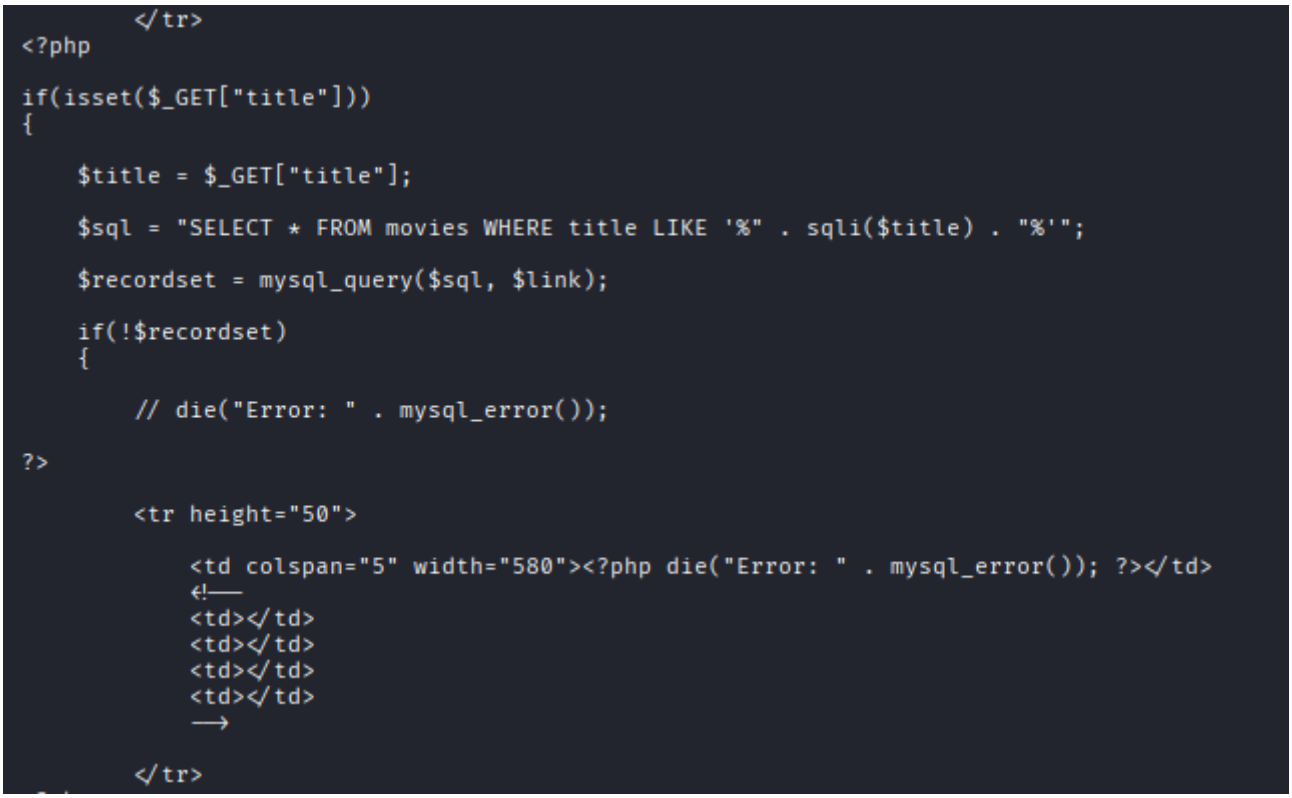
Esta imagen muestra el contenido inicial del fichero vulnerable donde se puede observar la estructura básica del código y las funciones que serán examinadas.

Segunda inspección - Ejecución de la consulta SQL

```
        </tr>
<?php

    }

    if(mysql_num_rows($recordset) != 0)
    {

        while($row = mysql_fetch_array($recordset))
        {

            // print_r($row);

?>

        <tr height="30">

            <td><?php echo $row["title"]; ?></td>
            <td align="center"><?php echo $row["release_year"]; ?></td>
            <td><?php echo $row["main_character"]; ?></td>
            <td align="center"><?php echo $row["genre"]; ?></td>
            <td align="center"><a href="http://www.imdb.com/title/<?php echo $row["imdb"]; ?>" target="_blank">Link</a></td>

        </tr>
<?php

        }

    }

    else
    {

?>

        <tr height="30">

            <td colspan="5" width="580">No movies were found!</td>
            <!--
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            -->

        </tr>
<?php

    }

    mysql_close($link);

}

else
{

?>

        <tr height="30">
```

Esta captura muestra cómo se ejecuta la consulta SQL contra la base de datos y se procesan los resultados. Se puede observar el bucle que recorre los registros obtenidos y cómo se muestran los datos en la página sin validación adicional.

Tercera inspección - Consulta SQL vulnerable



```
        </tr>
<?php

if(isset($_GET["title"]))
{

    $title = $_GET["title"];

    $sql = "SELECT * FROM movies WHERE title LIKE '%" . sqli($title) . "%'";

    $recordset = mysql_query($sql, $link);

    if(!$recordset)
    {

        // die("Error: " . mysql_error());

?>

        <tr height="50">

            <td colspan="5" width="580"><?php die("Error: " . mysql_error()); ?></td>
            <!--
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            -->

        </tr>
```

Esta imagen muestra la consulta SQL donde se evidencia claramente que **el valor de `$_GET['title']` se concatena directamente dentro del string SQL sin ningún tipo de validación o escapado**, lo que permite la inyección de código SQL.

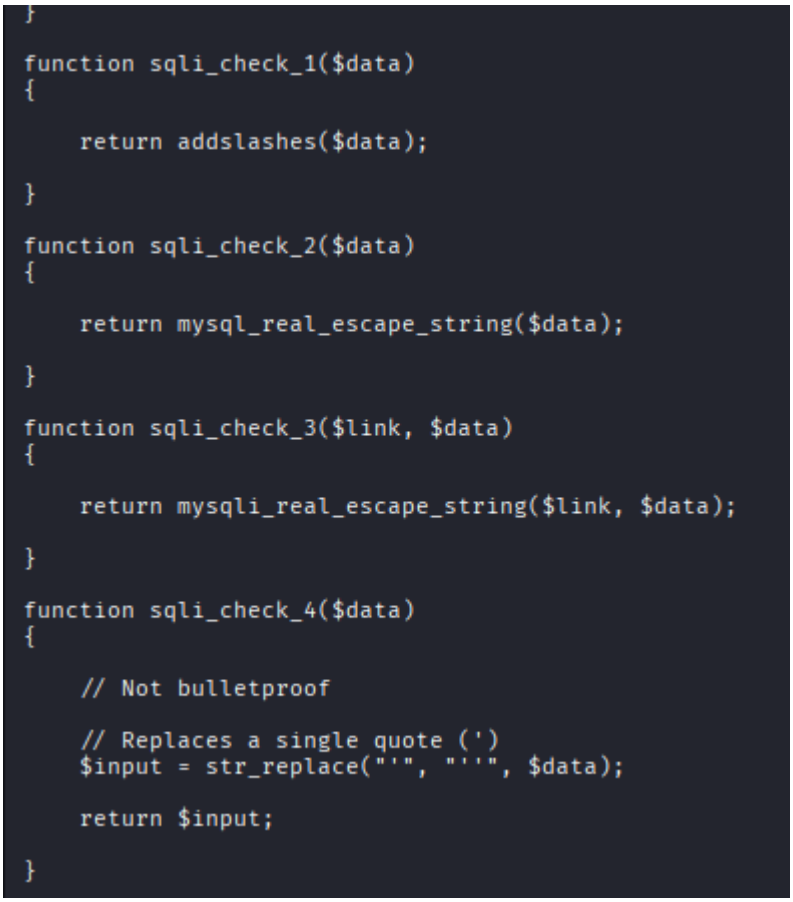
Análisis de funciones externas

Lectura del fichero de funciones

```php
*/

function no_check($data)
{

    return $data;

}

function email_check_1($data)
{

    return preg_match("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^", $data);

}

function email_check_2($data)
{

    return filter_var($data, FILTER_VALIDATE_EMAIL);

}

function maili_check_1($data)
{

    // URL decoding %0A → \n and %0D → \r
    $input = urldecode($data);

    $input = str_replace("\n", "", $input);
    $input = str_replace("\r", "", $input);
    $input = str_replace("bcc:", "", $input);

    return $input;

}

function maili_check_2($data)
{

    // URL decoding %0A → \n and %0D → \r
    $input = urldecode($data);

    $input = filter_var($input, FILTER_SANITIZE_EMAIL);

    return $input;

}

function url_check_1($data)
{

    // URL decoding %0A → \n and %0D → \r
    $input = urldecode($data);
```

Al hacer `cat` del fichero incluido en `functions_external.php`, se visualizan los métodos completos que el fichero vulnerable estaba llamando, permitiendo entender el flujo completo de la aplicación.

Funciones de validación disponibles pero no utilizadas



```
}

function sqli_check_1($data)
{

    return addslashes($data);

}

function sqli_check_2($data)
{

    return mysql_real_escape_string($data);

}

function sqli_check_3($link, $data)
{

    return mysqli_real_escape_string($link, $data);

}

function sqli_check_4($data)
{

    // Not bulletproof

    // Replaces a single quote (')
    $input = str_replace("'", "''", $data);

    return $input;

}
```

Esta captura muestra varias funciones de validación que existen en el fichero de funciones externas. A pesar de tener estas herramientas disponibles, el código vulnerable no las utiliza, dejando la aplicación expuesta a inyecciones SQL.