

eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Sistemas de Gestión de Seguridad de Sistemas de Información

Ingeniería Informática de Gestión y Sistemas de
Información

Sistema Web

Autores:

Xabier Gabiña
Ainhize Martinez
Marcos Martín

26 de noviembre de 2023

Índice general

1. Introducción	3
2. Primera auditoría	4
2.1. ZAP	4
2.2. sqlmap	6
2.3. nmap	8
2.4. Metaexploit	9
3. Vulnerabilidades	11
3.1. Rotura de control de acceso	11
3.1.1. Acceso mediante URL	11
3.1.2. Acceso a cuenta personal	12
3.2. Fallos criptográficos	16
3.2.1. Cifrado de extremo a extremo	16
3.2.2. Almacenamiento de contraseñas	18
3.2.3. Configuración errónea de las Cookies	20
3.3. Inyección	21
3.3.1. SQL Injection	21
3.3.2. Cross-Site Scripting (XSS)	22
3.3.3. Cross-Site Request Forgery	23
3.4. Diseño inseguro	24
3.4.1. Reutilización de códigos seguros	24
3.5. Configuración de seguridad insuficiente	26
3.5.1. Despliegue seguro	26
3.5.2. Cabecera CSP	28
3.5.3. Cabecera Cache-Control	29
3.5.4. Cabecera HSTS	30
3.5.5. Cabecera X-XSS-Protection	31
3.5.6. Cabecera X-Content-Type-Options	32
3.5.7. Cabecera X-Frame-Options	33
3.5.8. Cabecera X-Permitted-Cross-Domain-Policies	34
3.5.9. Cabecera X-DNS-Prefetch-Control	35
3.5.10. Cabecera X-Download-Options	36
3.5.11. Cabecera Referrer-Policy	37

3.5.12.	Cabecera Feature-Policy	38
3.5.13.	Cabecera Expect-CT	39
3.5.14.	Configuración PHP	40
3.5.15.	Configuración Apache	41
3.6.	Componentes vulnerables y obsoletos	42
3.6.1.	Control de versiones de los componentes	42
3.7.	Fallos de identificación y autenticación	43
3.7.1.	Captcha	43
3.7.2.	Contraseñas débiles o por defecto	44
3.7.3.	Invalidación de sesiones	45
3.8.	Fallos en la integridad de datos y software	46
3.8.1.	Copias de seguridad	46
3.8.2.	CI/CD	47
3.9.	Fallos en la monitorización de la seguridad	48
3.9.1.	Auditorías de seguridad	48
3.9.2.	Configuración de logs del sistema	49
4.	Segunda auditoría	50
4.1.	ZAP	50
4.2.	sqlmap	51
4.3.	nmap	53
4.4.	Metaexploit	54
5.	Conclusiones	55
6.	Bibliografía	56

Introducción

La finalidad de esta etapa del proyecto es garantizar la seguridad y robustez de la página web desarrollada en la fase inicial del proyecto. Con el propósito de lograr este objetivo, se recurrirá a la aplicación de diversas herramientas y técnicas que han sido abordadas y exploradas a lo largo del curso. El enfoque principal de esta fase se centra en identificar y abordar posibles vulnerabilidades y debilidades de seguridad en el sistema, a fin de implementar medidas correctivas y fortalecer la integridad de la plataforma web.

Este proceso implica la utilización de herramientas especializadas, como ZAP, SQLMap, Nmap y Metasploit, que permiten evaluar y poner a prueba la infraestructura de seguridad de la página web. A través de su empleo, se llevará a cabo un exhaustivo análisis que permitirá descubrir posibles fallos o vulnerabilidades que puedan ser aprovechadas por actores maliciosos, poniendo en riesgo la integridad de los datos y la privacidad de los usuarios.

El objetivo último es la identificación, documentación y solución de posibles vulnerabilidades, lo que contribuirá a elevar la seguridad y confiabilidad de la página web. A medida que se avance en este proceso, se aplicarán estrategias y medidas correctivas para abordar los hallazgos identificados y se establecerán salvaguardias que minimicen el riesgo de futuros incidentes de seguridad.

Este proyecto representa un ejercicio académico valioso que no solo demuestra la comprensión de los conceptos de seguridad informática, sino que también ofrece la oportunidad de aplicar de manera práctica las habilidades adquiridas en la detección y mitigación de riesgos en entornos web. A través de esta experiencia, se espera adquirir un conocimiento más profundo y aplicable en el campo de la seguridad informática y estén mejor preparados para enfrentar desafíos en el mundo real.

Primera auditoría

El objetivo principal de esta auditoría es la de encontrar los fallos de seguridad que tiene nuestro sistema web para poder solucionarlos en el siguiente capítulo de este documento.

2.1. ZAP

Para empezar con la primera auditoría ejecutaremos el proxy ZAP como se nos ha propuesto en clase.

ZAP es una herramienta de seguridad de aplicaciones web de código abierto que se utiliza para encontrar automáticamente vulnerabilidades en las aplicaciones web. Está diseñado para ser accesible a usuarios con diversos niveles de experiencia y conocimientos en seguridad, y como tal es ideal para desarrolladores y probadores funcionales que son nuevos en el ámbito de la seguridad de aplicaciones web.

En nuestro caso usamos AJAX por lo que debemos especificar al ejecutar AJAX que usa la SPIDER AJAX en ZAP.

Al ejecutarla en nuestra página web, nos encontramos con el siguiente listado de errores y fallos de configuración:

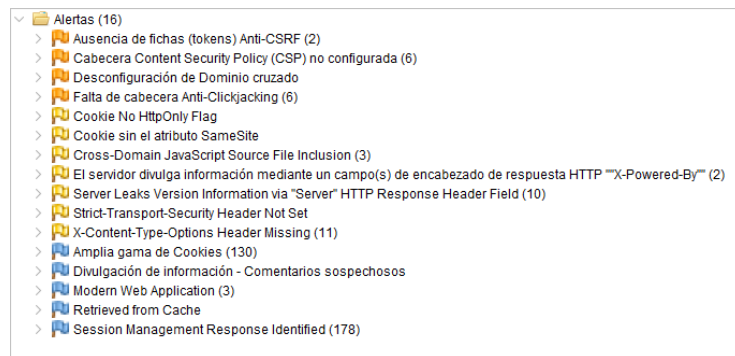


Figura 2.1: Listado de errores de la primera auditoría

Como podemos ver en la imagen, tenemos un total de 16 errores. Estos errores son por lo general de configuraciones tanto del servidor web como de la página web.

2.2. sqlmap

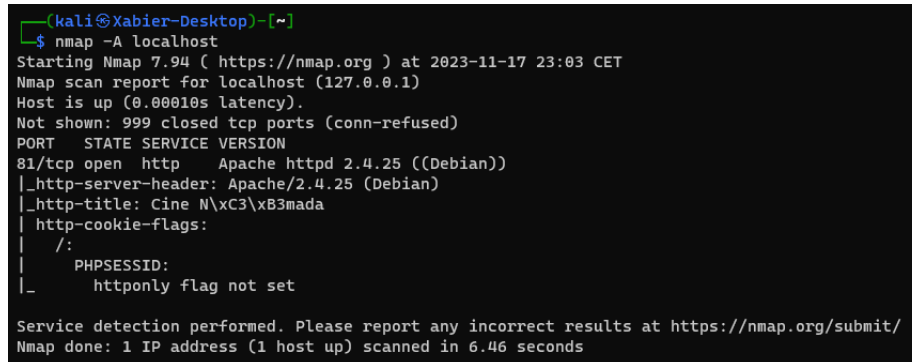
La siguiente herramienta que usaremos será sqlmap, la cual nos permitirá encontrar fallos de seguridad en la base de datos y en los formularios que hacen uso de ella.

[illegible]

Figura 2.2: Output de sqlmap 1

2.3. nmap

La siguiente herramienta que usaremos será nmap. Nmap es utilizada para descubrir hosts y servicios en una computadora en una red, creando un mapa de la misma. Para lograr su propósito, Nmap envía paquetes de datos sin procesar a través de la red y luego analiza las respuestas.



```
(kali@Xabier-Desktop)~$ nmap -A localhost
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-17 23:03 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00010s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
81/tcp    open  http      Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Cine N\xC3\xB3mada
|_http-cookie-flags:
|_  /:
|_  PHPSESSID:
|_  httponly flag not set

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.46 seconds
```

Figura 2.6: Output de nmap 1

Como podemos ver nmap ha detectado el servidor web y nos ha arrojado información como su puerto y su versión. Esto es una vulnerabilidad, ya que al saber la versión se puede hacer uso de bases de datos, como la de exploitDB, para encontrar exploits que puedan ser usados contra nuestro servidor.

2.4. Metaexploit

En este caso, he usado la herramienta Metaexploit para comprobar si nuestro sistema web es vulnerable a algún exploit conocido. Metaexploit es un framework de pruebas de penetración que permite a los usuarios crear, probar y desplegar exploits contra sistemas de seguridad.

Tras algunas búsquedas, se ha encontrado un exploit que mediante diccionarios de contraseñas intenta loguearse en phpMyAdmin.

```
kali@Xabier-Desktop: ~
kali@Xabier-Desktop: /usr/sha
$ msfconsole
Metasploit tip: View a module's description using info, or the enhanced
version in your browser with info -d

# cowsay++
< metasploit >

      \      /
      (oo)\_____)
      (__)|       )\/
      ||----w |
      ||     ||

      =[ metasploit v6.3.40-dev ]
+ -- --=[ 2370 exploits - 1229 auxiliary - 414 post ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > shearch phpMyAdmin
[-] Unknown command: shearch
msf6 > search phpMyAdmin

Matching Modules
=====

#  Name                                     Disclosure Date  Rank    Check  Description
-  -
0  exploit/unix/webapp/phpmyadmin_config    2009-03-24      excellent No      PhpMyAdmin Config Fil
e Code Injection
1  auxiliary/scanner/http/phpmyadmin_login  normal          No      PhpMyAdmin Login Scan
ner
2  post/linux/gather/phpmyadmin_credsteal   normal          No      Phpmyadmin credential
s stealer
3  auxiliary/admin/http/telpho10_credentia dump 2016-09-02      normal  No      Telpho10 Backup Crede
ntials Dumper
4  exploit/multi/http/zpanel_information_disclosure_rce 2014-01-30      excellent No      Zpanel Remote Unauthen
ticated RCE
5  exploit/multi/http/phpmyadmin_3522_backdoor 2012-09-25      normal  No      phpMyAdmin 3.5.2.2 se
rver_sync.php Backdoor
6  exploit/multi/http/phpmyadmin_lfi_rce    2018-06-19      good    Yes     phpMyAdmin Authentica
ted Remote Code Execution
7  exploit/multi/http/phpmyadmin_null_termination_exec 2016-06-23      excellent Yes     phpMyAdmin Authentica
ted Remote Code Execution
8  exploit/multi/http/phpmyadmin_preg_replace 2013-04-25      excellent Yes     phpMyAdmin Authentica
ted Remote Code Execution via preg_replace()

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/http/phpmyadmin_preg_replace

msf6 > use 1
```

Figura 2.7: Output de metasploit 1

```

msf6 auxiliary(scanner/http/phpmyadmin_login) > set RHOSTS localhost
RHOSTS => localhost
msf6 auxiliary(scanner/http/phpmyadmin_login) > set RPORT 8890
RPORT => 8890
msf6 auxiliary(scanner/http/phpmyadmin_login) > run

[*] PhpMyAdmin Version: Not Detected
[*] Error: 127.0.0.1: Metasploit::Framework::LoginScanner::Invalid Cred details can't be blank, Cred details can't be blank (Metasploit::Framework::LoginScanner::PhpMyAdmin)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/phpmyadmin_login) > set PASS_FILE /usr/share/wordlists/metasploit/http_default_pass.txt
PASS_FILE => /usr/share/wordlists/metasploit/http_default_pass.txt
msf6 auxiliary(scanner/http/phpmyadmin_login) > set USER_FILE /usr/share/wordlists/metasploit/http_default_users.txt
USER_FILE => /usr/share/wordlists/metasploit/http_default_users.txt
msf6 auxiliary(scanner/http/phpmyadmin_login) > run

[*] PhpMyAdmin Version: Not Detected
[-] 127.0.0.1:8890 - Failed: 'root:admin'
[!] No active DB -- Credential data will not be saved!
[-] 127.0.0.1:8890 - Failed: 'root:password'
[-] 127.0.0.1:8890 - Failed: 'root:manager'
[-] 127.0.0.1:8890 - Failed: 'root:letmein'
[-] 127.0.0.1:8890 - Failed: 'root:cisco'
[-] 127.0.0.1:8890 - Failed: 'root:default'
[+] 127.0.0.1:8890 - Success: 'root:root'
[-] 127.0.0.1:8890 - Failed: 'admin:admin'
[-] 127.0.0.1:8890 - Failed: 'admin:password'
[-] 127.0.0.1:8890 - Failed: 'admin:manager'
[-] 127.0.0.1:8890 - Failed: 'admin:letmein'
[-] 127.0.0.1:8890 - Failed: 'admin:cisco'
[-] 127.0.0.1:8890 - Failed: 'admin:default'
[-] 127.0.0.1:8890 - Failed: 'admin:root'
[-] 127.0.0.1:8890 - Failed: 'admin:apc'
[-] 127.0.0.1:8890 - Failed: 'admin:pass'
[-] 127.0.0.1:8890 - Failed: 'admin:security'
[-] 127.0.0.1:8890 - Failed: 'admin:user'
[-] 127.0.0.1:8890 - Failed: 'admin:system'
[-] 127.0.0.1:8890 - Failed: 'admin:sys'
[-] 127.0.0.1:8890 - Failed: 'admin:none'
[-] 127.0.0.1:8890 - Failed: 'admin:xampp'
[-] 127.0.0.1:8890 - Failed: 'admin:wampp'
[-] 127.0.0.1:8890 - Failed: 'admin:ppmax2011'
[-] 127.0.0.1:8890 - Failed: 'admin:turnkey'
[-] 127.0.0.1:8890 - Failed: 'admin:vagrant'
[+] 127.0.0.1:8890 - Success: 'admin:test'
[-] 127.0.0.1:8890 - Failed: 'manager:admin'
[-] 127.0.0.1:8890 - Failed: 'manager:password'
[-] 127.0.0.1:8890 - Failed: 'manager:manager'

```

Figura 2.8: Output de metasploit 2

Al tener una contraseña tan débil, enseguida ha encontrado las credenciales de phpMyAdmin tanto para root como para el usuario admin.

Esto es una vulnerabilidad del sistema ya que, al tener acceso a phpMyAdmin, se puede acceder a la base de datos y modificarla a placer.

Vulnerabilidades

3.1. Rotura de control de acceso

La rotura de control de acceso es una vulnerabilidad que permite a un atacante acceder a recursos restringidos o privilegiados, ya sea por un error en la implementación de la autenticación y autorización o por un error en la lógica de control de acceso.

3.1.1. Acceso mediante URL

Descripción

El acceso mediante URL se refiere a la práctica de manipular los parámetros presentes en una URL con el objetivo de eludir o evadir los controles de acceso implementados en un sistema o aplicación.

Solución

Esto no es un error en nuestra página web ya que al contenido no se accede mediante redirecciones en la página web, sino que se hace mediante el uso de AJAX.

AJAX es una técnica de programación que permite que una página web se comunique con el servidor, sin necesidad de hacer redirecciones ni recargas de la página.

Esto hace que si alguien entra en un archivo como login.php al no cargarse mediante AJAX, tampoco se cargan los archivos necesarios para que funcione por lo que es inútil.

3.1.2. Acceso a cuenta personal

Descripción

En nuestro sistema, un usuario puede modificar sus datos personales, pero también puede modificar los datos de otros usuarios. Esto es un fallo de rotura de control de acceso ya que un usuario no debería poder modificar los datos de otro usuario.

PoC

Pongamos en el ejemplo que tenemos dos usuarios, Admin y Xabier con sus respectivas IDs.

id	nombre
1	Xabier
10	Admin

Figura 3.1: Datos de Usuarios

Si desde inspeccionar elementos hacemos click sobre el botón 'Perfil', este nos mostrará el link el cual se ve así:

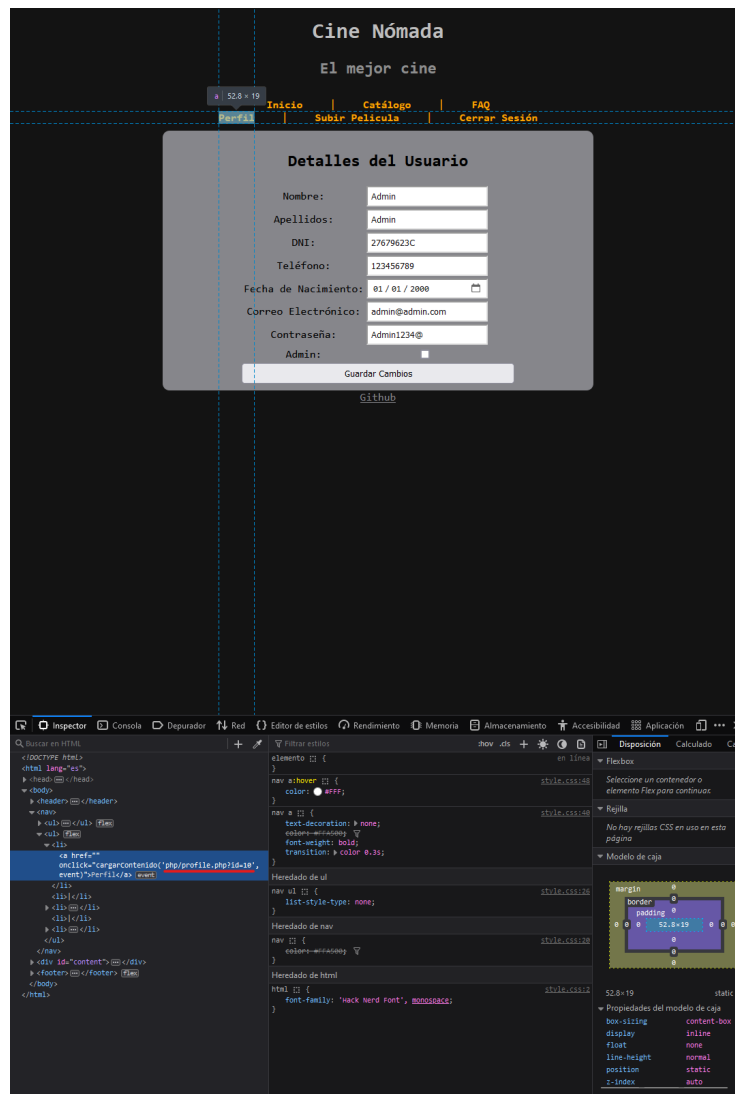


Figura 3.2: Perfil de Admin

Si alteramos el valor de 'id=X' por, en este caso, la id de Xabier (La ID 1) podemos acceder a sus datos.

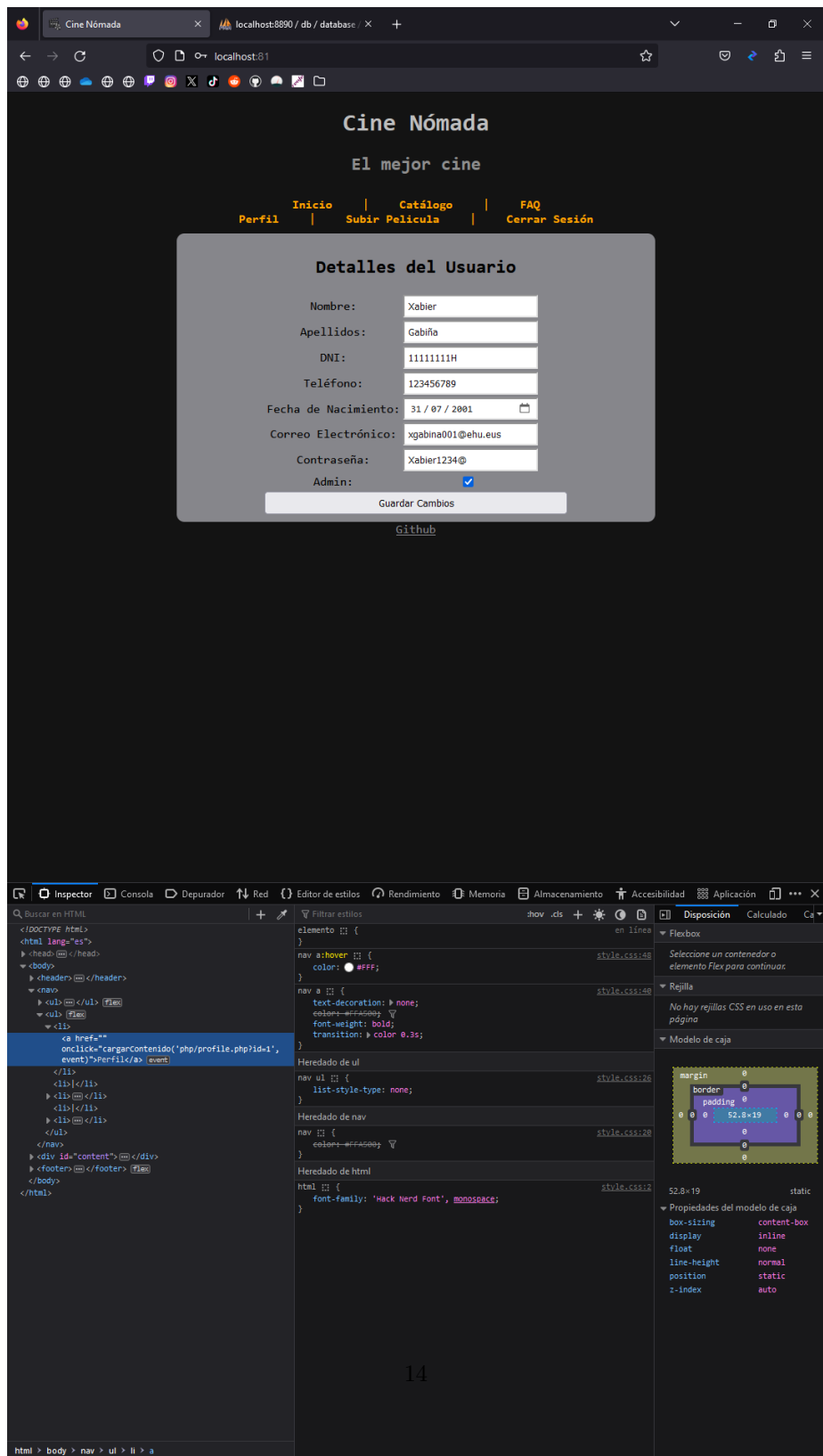


Figura 3.3: Perfil de Xabier

Solución

Para solucionar este problema, hemos añadido una comprobación en el código que verifica si el usuario que está intentando modificar los datos es el mismo que el usuario que está logueado en el sistema.

```
// Verificar si se recibió un ID válido a través de la URL
if (isset($_GET['id']) && is_numeric($_GET['id']))
{
    // Verificar si el usuario es el mismo que el de la sesión
    if ((int)$_GET['id'] == (int)$_SESSION['user_id'])
    {
        $userId = $_SESSION['user_id'];
    }
}
```

Figura 3.4: Comprobación de usuario

Este mismo error también ha sido corregido en el catálogo.

3.2. Fallos criptográficos

Los "fallos criptográficos" se refieren a debilidades o errores en el diseño, implementación o uso de algoritmos criptográficos que comprometen la seguridad de la información protegida mediante técnicas de cifrado y protección.

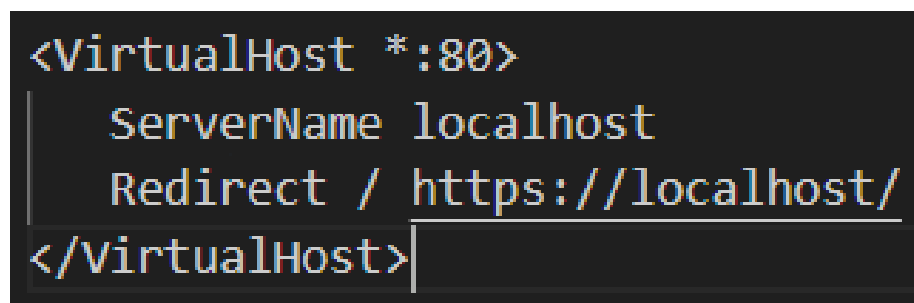
3.2.1. Cifrado de extremo a extremo

Descripción

El cifrado de extremo a extremo es un método de seguridad informática en el que la información se cifra en el punto de origen y solo se descifra en el punto de destino. Esto significa que los datos están protegidos durante su transmisión y solo son legibles para la parte destinataria que posee la clave de descifrado correspondiente.

Solución

Para solucionar este problema, configuraremos nuestro servidor para que cifre y redirija todo el tráfico a HTTPS. Creamos un certificado SSL autofirmado dentro del Dockerfile. Creamos una regla para redirigir el tráfico entrante del puerto 80 al puerto 443.



```
<VirtualHost *:80>
    ServerName localhost
    Redirect / https://localhost/
</VirtualHost>
```

Figura 3.5: Redirección de tráfico

También debemos decir al puerto 443 que use el certificado SSL que hemos creado.

```
<VirtualHost *:443>
  ServerName localhost
  DocumentRoot /var/www/html

  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
  SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
```

Figura 3.6: Certificado SSL

3.2.2. Almacenamiento de contraseñas

Descripción

En nuestro sistema, no se almacenan las contraseñas de los usuarios de forma segura, lo que permite que un atacante pueda obtener las contraseñas de los usuarios.

PoC

Si un atacante consigue acceso a la base de datos, podría obtener las contraseñas de los usuarios en texto plano.

email	passwd
xabierland@gmail.com	Xabier1234@
admin@admin.com	Admin1234@

Figura 3.7: Contraseñas en texto plano

Solución

Para solucionar este problema de la mejor forma posible, debemos tener tres puntos bien definidos:

1. Configurar el factor de costo apropiado
 - El factor de costo (work factor) en bcrypt determina el número de iteraciones utilizadas en el cálculo del hash. Un valor mayor implica una contraseña más segura, pero también requiere más tiempo para calcular el hash. Un valor razonable es 12 o más, dependiendo del hardware y las necesidades de rendimiento.
2. Usar un algoritmo de cifrado seguro.
 - En nuestro caso, usaremos BCrypt. CRYPT_BLOWFISH se usa para crear el hash. Producirá un hash estándar compatible con crypt() utilizando el identificador "\$2y\$". El resultado siempre será un string de 60 caracteres, o false en caso de error.
3. Generar un semilla aleatoria para cada usuario.

- BCrypt ya gestiona las semillas de forma automática y en el manual de PHP no recomiendan su uso de otra manera. Mas información en: <https://www.php.net/manual/es/function.password-hash.php>

```
// Hashear la contraseña  
$options=['cost'=>12,];  
$hashedPassword = password_hash($passwd, PASSWORD_BCRYPT, $options);
```

Figura 3.8: Contraseñas encriptadas

3.2.3. Configuración errónea de las Cookies

Descripción

La configuración errónea de las cookies se refiere a la práctica de establecer parámetros o atributos de las cookies de manera inadecuada, lo que puede tener consecuencias negativas en términos de seguridad, privacidad y funcionalidad en una aplicación web.

Solución

Para solucionar este problema, hemos añadido una configuración segura a las cookies de nuestro sitio web.

```
<?php
session_start([
    'cookie_lifetime' => 0,           // La sesión expira cuando se cierra el navegador.
    'cookie_path' => '/',            // Disponible en todo el dominio.
    'cookie_secure' => true,         // Solo se envía la cookie sobre conexiones HTTPS.
    'cookie_httponly' => true,       // La cookie solo es accesible a través de HTTP.
    'cookie_samesite' => 'Lax',      // Define la política de SameSite (puede ser 'Lax' o 'Strict').
]);
?>
```

Figura 3.9: Configuración de las cookies

3.3. Inyección

En el ámbito de la seguridad informática y el desarrollo de software, el término 'inyección' se refiere a una clase de vulnerabilidades que permite a un atacante introducir y ejecutar código malicioso en una aplicación o sistema.

3.3.1. SQL Injection

Descripción

La inyección SQL es una vulnerabilidad de seguridad informática que permite a un atacante modificar las consultas SQL que se ejecutan en una base de datos subyacente. Esto puede permitir a un atacante obtener información confidencial, alterar o eliminar datos, o incluso comprometer completamente el sistema.

Solución

Para solucionar este problema, hemos modificado el código que procesa las consultas SQL para que no se puedan inyectar consultas SQL.

```
// SQL seguro utilizando consultas preparadas
$sql = "INSERT INTO usuarios (nombre, apellidos, passwd, dni, fechaN, email, telefono) VALUES (?, ?, ?, ?, ?, ?, ?)";

if ($stmt = $conn->prepare($sql)) {
    // Vincula las variables a los marcadores de posición
    $stmt->bind_param("sssssss", $nombre, $apellidos, $hashedPassword, $dni, $fechaNacimiento, $email, $telefono);

    // Ejecuta la consulta preparada
    if ($stmt->execute()) {
        echo "Registrado con éxito";
    } else {
        echo "Error al ejecutar la consulta: " . $stmt->error;
    }
} else {
    echo "Error al preparar la consulta: " . $conn->error;
}
```

Figura 3.10: Parametrizar consulta SQL

3.3.2. Cross-Site Scripting (XSS)

Descripción

El Cross-Site Scripting (XSS) es una vulnerabilidad de seguridad informática que permite a un atacante inyectar código malicioso (por lo general JavaScript) en páginas web que son vistas por otros usuarios. El XSS permite a los atacantes ejecutar scripts en el navegador de un usuario, lo que puede secuestrar sesiones de usuario, desfigurar sitios web, robar información confidencial y distribuir malware.

Solución

La solución a este problema se verá más adelante en la sección 3.5.5.

3.3.3. Cross-Site Request Forgery

Descripción

El Cross-Site Request Forgery (CSRF) es una vulnerabilidad de seguridad informática que permite a un atacante forzar a un usuario autenticado a ejecutar acciones no deseadas en una aplicación web en la que confía. El CSRF puede utilizarse para realizar acciones como transferencias de fondos, cambios de contraseñas, compras, etc.

Solución

Para solucionar este problema, hemos añadido un token CSRF a nuestro sitio web.

```
// Generar token CSRF si no existe
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

Figura 3.11: Token CSRF

```
// Verificar el token CSRF
if (isset($_POST['csrf_token']) && $_POST['csrf_token'] === $_SESSION['csrf_token']) {
    // Token CSRF válida, continúa con el registro
}
```

Figura 3.12: Comprobación de token CSRF

3.4. Diseño inseguro

El "diseño inseguro" se refiere a la presencia de debilidades fundamentales en el diseño de una aplicación web que pueden ser explotadas por atacantes para comprometer la seguridad.

3.4.1. Reutilización de códigos seguros

Descripción

La reutilización de código seguro se refiere a la práctica de aprovechar componentes de software previamente probados y seguros en lugar de escribir código desde cero. Esto no solo puede acelerar el desarrollo de software, sino que también puede reducir el riesgo de introducir vulnerabilidades de seguridad.

Solución

En nuestro proyecto, se han implementado estas prácticas mediante la reutilización del archivo `validar.js` en el que se encuentran las funciones de validación de todos los formularios de nuestro sitio web.

```
// validaciones.js
function validarNumeroTelefono(telefono) {
    // Expresión regular para validar números de teléfono en formato de 9 dígitos
    var regex = /^[0-9]{9}$/;

    // Utiliza test() para verificar si el número cumple con la expresión regular
    return regex.test(telefono);
}

function validarCorreoElectronico(correo) {
    // Expresión regular para validar direcciones de correo electrónico
    var regex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    // Utiliza test() para verificar si el correo cumple con la expresión regular
    return regex.test(correo);
}

function validarContrasena(contrasena) {
    // Expresión regular para validar contraseñas
    var regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*])[A-Za-z\d!@#$%^&]{8,}$/;

    // Utiliza test() para verificar si la contraseña cumple con la expresión regular
    return regex.test(contrasena);
}

function validarDNI(nif) {
    // Expresión regular para validar un NIF español
    var regex = /^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]$/i;

    // Utiliza test() para verificar si el número cumple con la expresión regular
    if (!regex.test(nif)) {
        return false; // El formato del NIF es incorrecto
    }

    // Obtiene los números del NIF (los primeros 8 caracteres)
    var numerosNIF = nif.substr(0, 8);

    // Obtiene la letra proporcionada en el NIF (el último carácter)
    var letraProporcionada = nif.charAt(8).toUpperCase();

    // Array con las letras correspondientes a los números del NIF
    var letrasNIF = 'TRWAGMYFPDXBNJZSQVHLCKE';

    // Calcula la letra correcta para los números del NIF
    var letraCorrecta = letrasNIF[numerosNIF % 23];

    // Compara la letra proporcionada con la letra correcta
    return letraProporcionada === letraCorrecta;
}
```

Figura 3.13: Validación de formularios

3.5. Configuración de seguridad insuficiente

La configuración de seguridad incorrecta o insuficiente se refiere a ajustes o parámetros de seguridad que no están adecuadamente configurados para proteger un sistema, aplicación, red o servicio. Puede haber varias áreas en las que la configuración de seguridad puede ser insuficiente o incorrecta, lo que podría dejar un sistema vulnerable a amenazas y ataques.

3.5.1. Despliegue seguro

Descripción

Es importante que a la hora de montar nuestro servidor web no haya archivos que puedan ser accesibles desde el exterior y que puedan contener información sensible como contraseñas. Esto a nosotros nos ocurre en el `docker-compose.yml`

Solución

Hemos creado un `.env` donde se almacenan las contraseñas y el resto de información sensible y hemos añadido el archivo al `.gitignore` para que no se suba al repositorio. Dado que esto es un trabajo, el `.env` será incluido para poder comprobar que funciona correctamente.

```

version: '3'
services:
  web:
    container_name: web
    build:
      context: .
      dockerfile: Dockerfile
    image: web
    environment:
      - ALLOW_OVERRIDE=true
    ports:
      - "81:80"
      - "443:443"
    links:
      - db
    volumes:
      - ./web/src:/var/www/html/

  db:
    container_name: db
    image: mariadb:11.1.2
    restart: always
    volumes:
      - ./mysql:/var/lib/mysql
    environment:
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    ports:
      - "8889:3306"

  phpmyadmin:
    container_name: phpmyadmin
    image: phpmyadmin/phpmyadmin:5.2.1
    links:
      - db
    ports:
      - 8890:80
    environment:
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}

```

Figura 3.14: docker-compose sin información sensible

3.5.2. Cabecera CSP

Descripción

Las cabeceras CSP son una medida de seguridad utilizada en la programación web para mitigar los riesgos asociados con ataques de Cross-Site Scripting (XSS) y otros tipos de ataques de inyección de código malicioso en páginas web.

Solución

Para solucionar este problema, hemos añadido una cabecera 'Content-Security-Policy' con los siguientes valores:

```
<meta http-equiv="Content-Security-Policy" content="
default-src 'self' https://www.google.com/ 'unsafe-inline';
script-src 'self' https://code.jquery.com/ https://www.google.com/ https://www.gstatic.com 'unsafe-inline';
style-src 'self';
img-src 'self' data;;
form-action 'self';
">
```

Figura 3.15: Cabecera CSP

En este caso, nos hemos visto obligados a usar la etiqueta 'unsafe-inline' ya que, si no la usamos, no se cargan las bibliotecas de jQuery y reCAPTCHA. Esto podría ser un problema con librerías inseguras pero en nuestro caso no lo es ya que las librerías que usamos son de confianza.

3.5.3. Cabecera Cache-Control

Descripción

Las cabeceras 'Cache-Control' son utilizadas en las respuestas HTTP enviadas por un servidor web para controlar cómo los recursos web deben ser almacenados en la memoria caché del navegador o en servidores intermedios (como proxies) y cómo se deben comportar en términos de almacenamiento y actualización. Estas cabeceras son fundamentales para gestionar la eficiencia de la carga de páginas web, la seguridad y la privacidad.

Solución

Para solucionar este problema, crearemos una cabecera 'Cache-Control' con el valor 'no-store' para que el navegador no almacene en caché la página.

```
<!DOCTYPE html>
<html Lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  ⚡ <meta http-equiv="Cache-Control" content="no-store, no-cache, must-revalidate">
  <title>Cine Nómada</title>
  <link rel="stylesheet" href="css/style.css">
  <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

Figura 3.16: Cabecera Cache-Control

3.5.4. Cabecera HSTS

Descripción

Las cabeceras HSTS (HTTP Strict Transport Security) son una medida de seguridad utilizada en la programación web para garantizar que las comunicaciones entre un navegador web y un sitio web se realicen a través de una conexión segura y encriptada utilizando el protocolo HTTPS (HTTP Secure). HSTS ayuda a prevenir ataques de tipo man-in-the-middle (MitM) que podrían exponer datos sensibles o comprometer la seguridad de la comunicación.

Solución

Para solucionar este problema, hemos añadido una cabecera 'Strict-Transport-Security' con los siguientes valores:

```
# Cabecera HSTS
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
```

Figura 3.17: Cabecera HSTS

3.5.5. Cabecera X-XSS-Protection

Descripción

Las cabeceras "X-XSS-Protection" son una medida de seguridad utilizada en la programación web para ayudar a prevenir ataques de tipo Cross-Site Scripting (XSS). Los ataques XSS ocurren cuando un atacante inyecta código JavaScript malicioso en una página web, que luego se ejecuta en el navegador de un usuario sin su conocimiento. Estas cabeceras se utilizan para controlar y mitigar estos ataques.

Solución

Para solucionar este problema, hemos añadido una cabecera 'X-XSS-Protection' con los siguientes valores:

```
# Cabecera X-XSS-Protection
Header always set X-XSS-Protection "1; mode=block"
```

Figura 3.18: Cabecera X-XSS-Protection

3.5.6. Cabecera X-Content-Type-Options

Descripción

Las cabeceras "X-Content-Type-Options" son una medida de seguridad utilizada en la programación web para mitigar ciertos tipos de ataques, como ataques de tipo MIME-sniffing. Estas cabeceras se utilizan para controlar cómo el navegador web interpreta y muestra el contenido de una página web.

Solución

Para solucionar este problema, hemos añadido una cabecera 'X-Content-Type-Options' con los siguientes valores:

```
# Cabeceras X-Content-Type-Options
Header always set X-Content-Type-Options "nosniff"
```

Figura 3.19: Cabecera X-Content-Type-Options

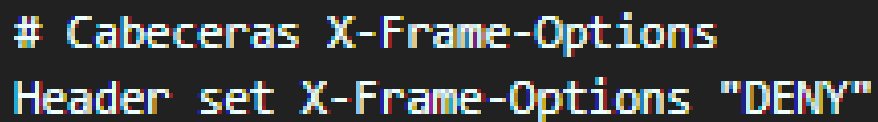
3.5.7. Cabecera X-Frame-Options

Descripción

Las cabeceras "X-Frame-Options" son una medida de seguridad utilizada en la programación web para controlar si una página web puede ser incrustada o mostrada dentro de un marco (frame) de otro sitio web. Estas cabeceras se utilizan para prevenir ataques de clickjacking, en los cuales un atacante puede ocultar contenido malicioso detrás de una página legítima y engañar a los usuarios para que hagan click en elementos sin su consentimiento.

Solución

Para solucionar este problema, hemos añadido una cabecera 'X-Frame-Options' con los siguientes valores:



```
# Cabeceras X-Frame-Options  
Header set X-Frame-Options "DENY"
```

Figura 3.20: Cabecera X-Frame-Options

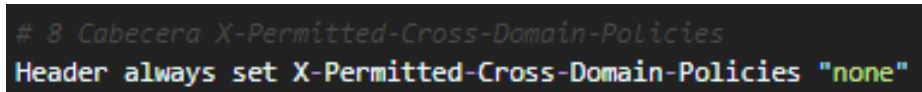
3.5.8. Cabecera X-Permitted-Cross-Domain-Policies

Descripción

La cabecera X-Permitted-Cross-Domain-Policies permite a los propietarios del contenido controlar cómo los documentos son tratados en contextos de navegación cruzada.

Solución

Para solucionar este problema, crearemos una cabecera 'X-Permitted-Cross-Domain-Policies' con el valor 'none' para que el navegador no permita que se usen características y APIs en los contenidos de una página.

A screenshot of a code editor with a dark background. The first line is a comment: `# 8 Cabecera X-Permitted-Cross-Domain-Policies`. The second line is a configuration command: `Header always set X-Permitted-Cross-Domain-Policies "none"`.

```
# 8 Cabecera X-Permitted-Cross-Domain-Policies
Header always set X-Permitted-Cross-Domain-Policies "none"
```

Figura 3.21: Cabecera X-Permitted-Cross-Domain-Policies

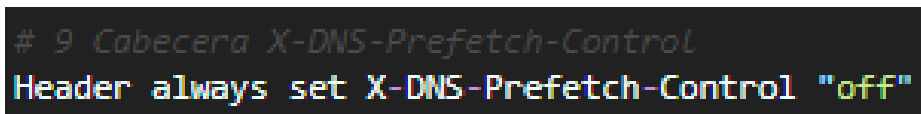
3.5.9. Cabecera X-DNS-Prefetch-Control

Descripción

La cabecera X-DNS-Prefetch-Control es una cabecera HTTP que se utiliza para controlar la funcionalidad de prefetching DNS en los navegadores. El prefetching DNS es una técnica que los navegadores utilizan para anticiparse a las solicitudes DNS antes de que un usuario haga click en un enlace. Esto puede acelerar la carga de páginas al preresolver los nombres de dominio antes de que se soliciten explícitamente. Esto puede provocar que el navegador realice peticiones a dominios que no son de confianza, dar información de dominios sensibles, etc.

Solución

Para solucionar este problema, crearemos una cabecera 'X-DNS-Prefetch-Control' con el valor 'off' para que el navegador no permita que se precargue contenido de la página.



```
# 9 Cabecera X-DNS-Prefetch-Control  
Header always set X-DNS-Prefetch-Control "off"
```

Figura 3.22: Cabecera X-DNS-Prefetch-Control

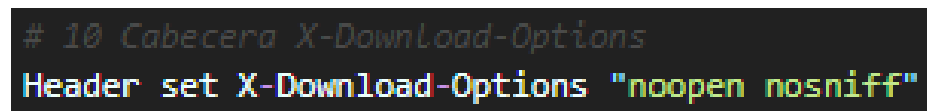
3.5.10. Cabecera X-Download-Options

Descripción

La cabecera X-Download-Options es una cabecera HTTP que se utiliza para controlar cómo ciertos navegadores manejan la descarga de archivos. Esta cabecera tiene un propósito específico en el contexto de Internet Explorer (IE) y se utiliza para mitigar el riesgo de ejecución automática de archivos descargados que podrían ser maliciosos.

Solución

Para solucionar este problema, crearemos una cabecera 'X-Download-Options' con el valor 'noopen' para que el navegador no permita que se ejecuten los archivos descargados.



```
# 10 Cabecera X-Download-Options  
Header set X-Download-Options "noopen nosniff"
```

Figura 3.23: Cabecera X-Download-Options

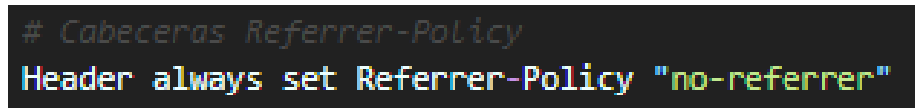
3.5.11. Cabecera Referrer-Policy

Descripción

Esta cabecera controla cómo se incluye el encabezado Referer en las solicitudes. Puedes configurarlo para reducir la cantidad de información sensible enviada en las solicitudes de referencia.

Solución

Para solucionar este problema, crearemos una cabecera 'Referrer-Policy' con el valor 'no-referrer' para que el navegador no envíe información sensible en las solicitudes de referencia.

A screenshot of a code editor with a dark background. It shows two lines of code. The first line is a comment: `# Cabeceras Referrer-Policy`. The second line is an instruction to set the header: `Header always set Referrer-Policy "no-referrer"`.

```
# Cabeceras Referrer-Policy
Header always set Referrer-Policy "no-referrer"
```

Figura 3.24: Cabecera Referrer-Policy

3.5.12. Cabecera Feature-Policy

Descripción

La cabecera 'Feature-Policy' permite que un sitio controle qué características y APIs pueden ser utilizadas por los contenidos de una página.

Solución

Para solucionar este problema, crearemos una cabecera 'Feature-Policy' con el valor 'none' para que el navegador no permita que se usen características y APIs en los contenidos de una página.

```
# Cabeceras Feature-Policy  
Header always set Feature-Policy "geolocation 'self'; camera 'none'";
```

Figura 3.25: Cabecera Feature-Policy

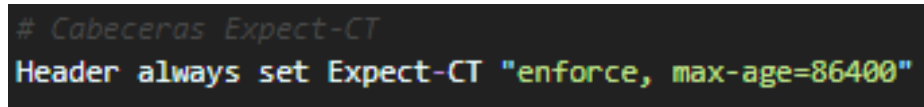
3.5.13. Cabecera Expect-CT

Descripción

Las cabeceras 'Expect-CT' ayuda a proteger al usuario contra ataques de certificate transparency.

Solución

Para solucionar este problema, crearemos una cabecera 'Expect-CT' con el valor 'max-age=86400' para que el navegador no permita que se usen características y APIs en los contenidos de una página.

A screenshot of a code editor with a dark background. The first line is a comment: `# Cabeceras Expect-CT`. The second line is a configuration directive: `Header always set Expect-CT "enforce, max-age=86400"`.

```
# Cabeceras Expect-CT
Header always set Expect-CT "enforce, max-age=86400"
```

Figura 3.26: Cabecera Expect-CT

3.5.14. Configuración PHP

Descripción

Cuando usamos PHP en un servidor web, es importante configurarlo de forma segura para evitar que se filtre información que pueda ayudar a los atacantes a encontrar una vulnerabilidad en nuestro sistema.

Solución

Hemos ocultado la versión de PHP en las peticiones a nuestro servidor para evitar que un atacante pueda usar esta información para encontrar una vulnerabilidad en nuestro sistema. Para ello, hemos creado el php.ini

```
; Decides whether PHP may expose the fact that it is installed on the server
; (e.g. by adding its signature to the Web server header). It is no security
; threat in any way, but it makes it possible to determine whether you use PHP
; on your server or not.
; https://php.net/expose-php
expose_php = Off
```

Figura 3.27: Ocultar versión de PHP

3.5.15. Configuración Apache

Descripción

Al igual que con PHP en el punto anterior, es importante configurar Apache para que muestre la mínima información al exterior.

Solución

Para ello, al igual que con PHP, hemos ocultado las versiones del servidor para evitar en la medida de lo posible que el atacante encuentre vulnerabilidad en nuestro servidor. Para ello, hemos editado el `apache2.conf`

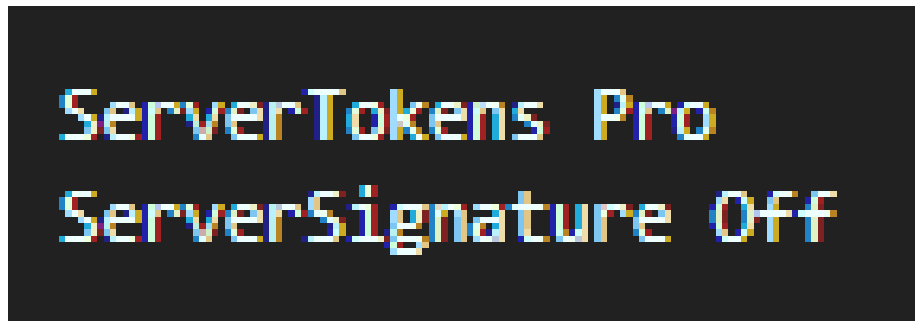


Figura 3.28: Ocultar versión de Apache

3.6. Componentes vulnerables y obsoletos

Se refiere a partes o elementos dentro de un sistema, software o hardware, que presentan vulnerabilidades de seguridad debido a su antigüedad o a la falta de actualizaciones.

3.6.1. Control de versiones de los componentes

Descripción

Es crítico mantener actualizados los sistemas operativos y el software de seguridad con las últimas actualizaciones y parches de seguridad.

Solución

Para solucionar este problema, hemos actualizado todos los componentes a sus últimas versiones.

- PHP 7.2.2 → 8.2
- MariaDB 10.8.2 → 11.1.2
- phpMyAdmin 5.1.1 → 5.1.1
- jQuery 3.6.0 → 3.7.1

En estos casos, es recomendable no usar latest ya que en el pasado ha sido un problema de seguridad. Esto se debe a que un atacante puede tener acceso a la imagen y alterándola provoca que todo el que use latest se descargue la versión infectada.

3.7. Fallos de identificación y autenticación

Los fallos de identificación y autenticación se refieren a deficiencias o problemas en los procesos y mecanismos diseñados para verificar la identidad de un usuario y asegurar que la persona o entidad que intenta acceder a un sistema o recurso es realmente quien afirma ser.

3.7.1. Captcha

Descripción

Un captcha es un tipo de prueba de Turing que se utiliza para determinar si el usuario es o no humano. Los captchas se utilizan para prevenir ataques de tipo bot, como el spam y el abuso de servicios web.

Solución

Para solucionar este problema, hemos añadido un captcha a nuestro formulario de registro e inicio de sesión. Hemos añadido para nuestro proyecto el reCAPTCHA v3 de Google ya que es el más fácil de implementar y el que menos molesta al usuario. La puntuación se basa en las interacciones con su sitio y le permite tomar las medidas adecuadas para su sitio. Es por esto que, ha diferencia de otros captchas, este no muestra ninguna imagen al usuario pero si que se ejecuta en segundo plano y, si detecta que el usuario es un bot, se le bloquea el acceso.

Nota: reCAPTCHA v3 utiliza un sistema de puntuaciones que gestiona la propia Google. Esto puede hacer que al conectarte desde una VPN o una red pública te bloquee el acceso. Ten esto en cuenta si estás accediendo desde la red de la universidad.

3.7.2. Contraseñas débiles o por defecto

Descripción

Las contraseñas débiles o por defecto son contraseñas que son fáciles de adivinar o que se utilizan en muchos sistemas diferentes. Esto puede permitir a un atacante obtener acceso no autorizado a un sistema o aplicación.

Solución

Para solucionar este problema, hemos hecho dos cosas:

- Añadir un JS que asegura que las contraseñas de nuestros usuarios cumplan con los requisitos mínimos de seguridad.

```
function validarContrasena(contrasena) {  
  // Expresión regular para validar contraseñas  
  var regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*])[A-Za-z\d!@#$%^&*]{8,}$/  
  
  // Utiliza test() para verificar si la contraseña cumple con la expresión regular  
  return regex.test(contrasena);  
}
```

Figura 3.29: JS de validación de contraseñas

- Cambiar las contraseñas de nuestros servicios como phpMyAdmin, MariaDB, etc.

```
MYSQL_ROOT_PASSWORD: root  
MYSQL_USER: admin  
MYSQL_PASSWORD: test  
MYSQL_DATABASE: database
```

Figura 3.30: Contraseña antes del cambio

- Estas contraseñas se ven dentro del .env el cual no se debería subir, pero al ser un trabajo se ha subido al repositorio.

3.7.3. Invalidación de sesiones

Descripción

La invalidación de sesiones se refiere a la práctica de terminar una sesión de usuario cuando el usuario sale de una aplicación o sitio web. Esto puede ayudar a prevenir ataques de tipo secuestro de sesión, en los que un atacante puede tomar el control de una sesión de usuario activa.

Solución

En este punto, también hemos puesto dos soluciones al problema:

1. Hemos añadido un botón de cerrar sesión en la página de perfil de usuario.
2. La cookie se cierra automáticamente al cerrar el navegador.

Con estas dos soluciones, si un usuario se olvida de cerrar sesión, el sistema lo hará automáticamente y, si un atacante consigue la sesión de un usuario, esta se invalidará de igual forma.

3.8. Fallos en la integridad de datos y software

Los fallos en la integridad de datos y software pueden tener graves consecuencias en la seguridad y confiabilidad de sistemas y aplicaciones. Estos fallos pueden ocurrir debido a diversas razones, y es importante abordarlos adecuadamente para mantener la integridad de los datos y el funcionamiento del software.

3.8.1. Copias de seguridad

Descripción

Las copias de seguridad son una medida de seguridad que permite a los usuarios y administradores de sistemas restaurar datos y sistemas a un estado anterior en caso de que se produzca una pérdida de datos o un fallo del sistema. Las copias de seguridad pueden ser útiles para recuperar datos perdidos debido a errores humanos, ataques de malware, fallos de hardware, desastres naturales, etc.

Solución

Para solucionar este problema, debemos realizar una copia de seguridad de nuestra base de datos cada cierto periodo de tiempo. En nuestro caso, accedemos a la base de datos mediante el phpMyAdmin y exportaremos la base de datos en un archivo .sql

3.8.2. CI/CD

Descripción

CI/CD es un conjunto de prácticas y herramientas que permiten a los equipos de desarrollo de software entregar cambios de código de forma rápida y fiable. CI/CD es un acrónimo de Integración Continua (CI) y Despliegue Continuo (CD). CI/CD es una práctica fundamental para DevOps.

Solución

Para solucionar este problema, podemos generar un pipeline de CI/CD en GitHub Actions mediante los Workflows.

```
name: Test

on: push

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
        with:
          context: ./Docker
```

Figura 3.31: Pipeline de CI/CD

Este no es un pipeline completo con todas las pruebas unitarias ni la creación de las imágenes ni una despliegue como el que podría ser Kubernetes, pero sirve como ejemplo para este trabajo.

3.9. Fallos en la monitorización de la seguridad

Los fallos en la monitorización de la seguridad se refieren a deficiencias o problemas en los sistemas y procesos diseñados para vigilar y evaluar la seguridad de una red, sistema informático, aplicación o entorno en general.

3.9.1. Auditorías de seguridad

Descripción

Una auditoría de seguridad es un proceso sistemático de evaluación y revisión de sistemas, redes, aplicaciones o entornos tecnológicos con el propósito de identificar vulnerabilidades, debilidades y riesgos de seguridad. El objetivo principal de una auditoría de seguridad es garantizar que los controles de seguridad estén implementados adecuadamente y cumplan con los estándares de seguridad, y proporcionar recomendaciones para mejorar la protección de activos y datos contra amenazas y ataques cibernéticos.

Solución

La solución a este problema, es realizar un plan de seguridad para nuestro sistema y realizar auditorías de seguridad cada cierto tiempo para comprobar que no se han introducido nuevos problemas de seguridad.

3.9.2. Configuración de logs del sistema

Descripción

Los logs del sistema son una herramienta de seguridad que permite a los administradores de sistemas y a los equipos de seguridad supervisar y analizar la actividad de los sistemas y las redes. Los logs del sistema pueden utilizarse para detectar y analizar incidentes de seguridad, así como para identificar y prevenir posibles amenazas.

Solución

Para solucionar este problema, hemos implementado una lógica del log del lado del servidor para guardar los logs de los usuarios al cometer errores.

```
<?php
// Ruta del archivo de registro
$logFile='../log/log.json';

// Verifica si la ruta del archivo no está vacía antes de intentarlo
if (!empty($logFile)) {
    $archivo = fopen($logFile, 'a');

    // Verifica si la apertura del archivo fue exitosa antes de escribir en él
    if ($archivo) {
        // Obtiene la fecha y hora actual
        $date = date('Y-m-d H:i:s');

        // Elimina el último elemento del array $_POST
        array_pop($_POST);

        // Obtiene la dirección IP del cliente
        // $ip = $_SESSION['ip'];

        // Crea un array con la fecha, hora y datos del $_POST
        $logData = [
            'date' => $date,
            // 'ip' => $ip,
            'postData' => $_POST,
        ];

        // Convierte el array a formato JSON de manera legible
        $mensaje = json_encode($logData, JSON_PRETTY_PRINT);
        fwrite($archivo, $mensaje . "\n");
        fclose($archivo);
    } else {
        echo "Error al abrir el archivo de registro.";
    }
} else {
    echo "La ruta del archivo de registro está vacía.";
}
?>
```

Figura 3.32: Logs del lado del servidor

Segunda auditoría

Tras arreglar los problemas de la primera auditoría, hemos vuelto a realizar una auditoría de seguridad para comprobar que no se han introducido nuevos problemas de seguridad.

4.1. ZAP

Para llevar la misma estructura que en la primera auditoría, hemos usado la misma herramienta para realizar la segunda.

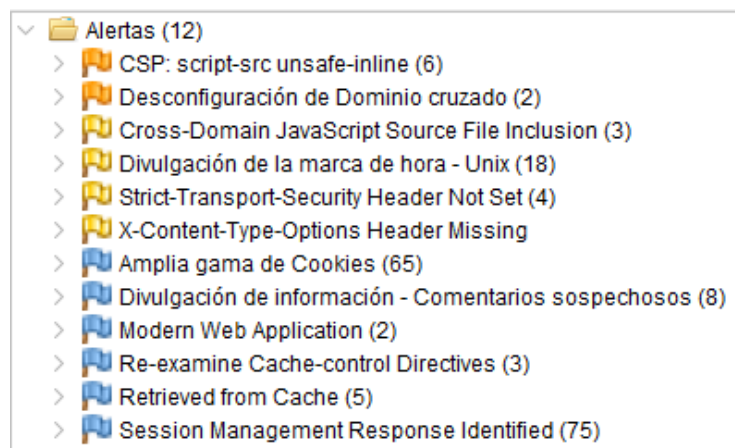


Figura 4.1: Listado de errores de ZAP de la segunda auditoría

Aunque veamos que todavía hay errores, estos en realidad son dados porque ZAP no puede comprobar algunas de las librerías que usamos en nuestro proyecto como son jQuery y reCAPTCHA.

Aun así, ambas librerías hacen que nuestra web sea mas segura por lo que no las vamos a eliminar de nuestro proyecto solo para que ZAP no de errores.

4.2. sqlmap

Para comprobar que los problemas de inyección SQL están solucionados, hemos usado sqlmap.

```
kali@Xabier-Desktop: ~  
└─(kali@Xabier-Desktop)-[~]  
$ sqlmap -u https://localhost/html/login.php --wizard --flush-session  
  
+-----+  
| H |  
+-----+ {1.7.10#stable}  
| . |  
| . |  
| . |  
| V... |  
+-----+ https://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 13:11:59 /2023-11-18/  
  
[13:11:59] [INFO] starting wizard interface  
POST data (--data) [Enter for None]:  
[13:12:01] [WARNING] no GET and/or POST parameter(s) found for testing (e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will search for forms  
Injection difficulty (--level/--risk). Please choose:  
[1] Normal (default)  
[2] Medium  
[3] Hard  
> 3  
Enumeration (--banner/--current-user/etc). Please choose:  
[1] Basic (default)  
[2] Intermediate  
[3] All  
> 3  
  
sqlmap is running, please wait..  
  
[1/1] Form:  
POST https://localhost/php/login.php  
POST data: email=&passwd=Eg-recaptcha-response=&csrf_token=  
do you want to test this form? [Y/n/q]  
> Y  
Edit POST data [default: email=&passwd=Eg-recaptcha-response=&csrf_token=] (Warning: blank fields detected): email=&passwd=Eg-recaptcha-response=&csrf_token=  
do you want to fill blank fields with random values? [Y/n] Y  
POST parameter 'csrf_token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [y/N] N  
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found.  
Do you want to reduce the number of requests? [Y/n] Y
```

Figura 4.2: Output de sqlmap

Como podemos ver, sqlmap no ha encontrado ninguna vulnerabilidad en nuestro sistema gracias al token csrf que está impidiendo que se tramiten las peticiones maliciosas al servidor.

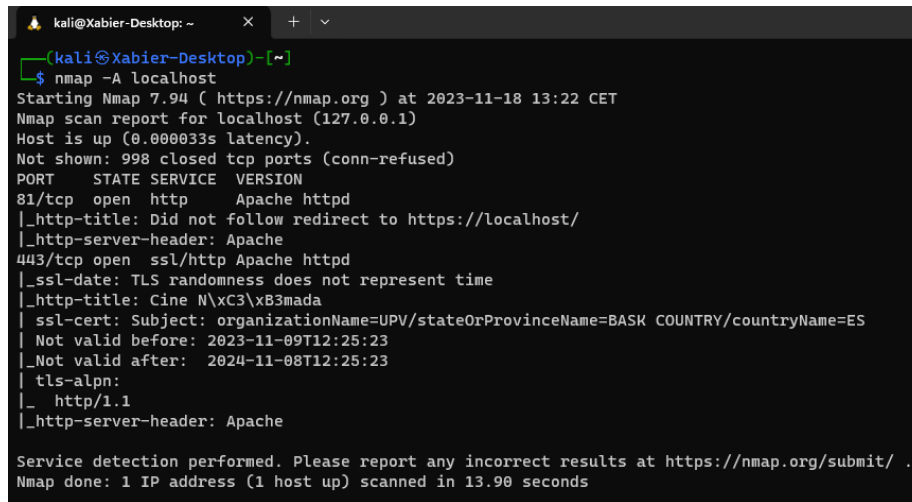
```
88] "POST /php/login.php HTTP/1.1" 200 2301 "https://localhost/php/login.php" "sqlmap/1.7.10#stable (https://sqlmap.org)"
[client 172.23.0.1:51248] PHP Warning: Undefined array key "csrf_token" in /var/www/html/php/login.php on line 7, referer: https://localhost/php/login.php
```

Figura 4.3: Logs del servidor

Esto se debe a que nuestra página web está pensada para cargar el contenido dinámicamente mediante AJAX y no mediante peticiones GET o POST. Por lo tanto, al no acceder antes al index.php, no se crea la sesión en el navegador con los tokens reCAPTCHA, CSRF y la cookie de sesión. Esto hace que sqlmap no pueda acceder a las páginas que requieren autenticación y, por lo tanto, no pueda comprobar si hay vulnerabilidades.

4.3. nmap

Para comprobar que nuestro servidor web ya no aporta información sensible, hemos usado nmap.

A terminal window titled 'kali@Xabier-Desktop: ~' with a dark background. The prompt is '(kali@Xabier-Desktop)-[~]'. The command '\$ nmap -A localhost' has been entered. The output shows the start of Nmap 7.94 at 2023-11-18 13:22 CET, a scan report for localhost (127.0.0.1), and that the host is up with 0.000033s latency. It notes 998 closed TCP ports. The open ports are 81/tcp (http, Apache) and 443/tcp (ssl/http, Apache). Detailed information for port 443 is shown, including a TLS certificate from UPV/Bask Country and a redirect to http://localhost/. The scan took 13.90 seconds.

```
(kali@Xabier-Desktop)-[~]
$ nmap -A localhost
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-18 13:22 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000033s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
81/tcp    open  http    Apache httpd
|_http-title: Did not follow redirect to https://localhost/
|_http-server-header: Apache
443/tcp    open  ssl/http Apache httpd
|_ssl-date: TLS randomness does not represent time
|_http-title: Cine N\xC3\xB3mada
|_ssl-cert: Subject: organizationName=UPV/stateOrProvinceName=BASK COUNTRY/countryName=ES
|_Not valid before: 2023-11-09T12:25:23
|_Not valid after:  2024-11-08T12:25:23
|_tls-alpn:
|_ http/1.1
|_http-server-header: Apache

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.90 seconds
```

Figura 4.4: Output de nmap

Como podemos ver, nmap ha dejado de devolvernos información sensible como la versión de PHP o Apache y ahora nos informa de que el servidor hace una redirección al 443 con certificado SSL.

4.4. Metaexploit

Por último, vamos a comprobar que nuestro servidor web ya no es vulnerable a ataques de fuerza bruta.

```
msf6 > search phpMyAdmin

Matching Modules
=====

#  Name                                     Disclosure Date  Rank    Check  Description
--  ---                                     -
0  exploit/unix/webapp/phpmyadmin_config    2009-03-24      excellent No      PhpMyAdmin Config File Code Injection
1  auxiliary/scanner/http/phpmyadmin_login  normal          No      PhpMyAdmin Login Scanner
2  post/linux/gather/phpmyadmin_credsteal   normal          No      PhpMyAdmin credential stealer
3  auxiliary/admin/http/telpho10_credential_dump 2016-09-02      normal  No      Telpho10 Backup Credentials Dumper
4  exploit/multi/http/zpanel_information_disclosure_rce 2014-01-30      excellent No      Zpanel Remote Unauthenticated RCE
5  exploit/multi/http/phpmyadmin_3522_backdoor 2012-09-25      normal  No      PhpMyAdmin 3.5.2.2 server_sync.php Backdoor
6  exploit/multi/http/phpmyadmin_lfi_rce      2018-06-19      good    Yes     PhpMyAdmin Authenticated Remote Code Execution
7  exploit/multi/http/phpmyadmin_null_termination_exec 2016-06-23      excellent Yes     PhpMyAdmin Authenticated Remote Code Execution
8  exploit/multi/http/phpmyadmin_preg_replace 2013-04-25      excellent Yes     PhpMyAdmin Authenticated Remote Code Execution via preg_replace()

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/http/phpmyadmin_preg_replace

msf6 > use 1
msf6 auxiliary(scanner/http/phpmyadmin_login) > set RHOSTS localhost
RHOSTS => localhost
msf6 auxiliary(scanner/http/phpmyadmin_login) > set RPORT 8890
RPORT => 8890
msf6 auxiliary(scanner/http/phpmyadmin_login) > set PASS_FILE /usr/share/wordlists/metasploit/http_default_pass.txt
PASS_FILE => /usr/share/wordlists/metasploit/http_default_pass.txt
msf6 auxiliary(scanner/http/phpmyadmin_login) > set USER_FILE /usr/share/wordlists/metasploit/http_default_users.txt
USER_FILE => /usr/share/wordlists/metasploit/http_default_users.txt
msf6 auxiliary(scanner/http/phpmyadmin_login) > run

[*] PhpMyAdmin Version: Not Detected
[-] 127.0.0.1:8890 - Failed: 'root:admin'
[!] No active DB -- Credential data will not be saved!
[-] 127.0.0.1:8890 - Failed: 'root:password'
[-] 127.0.0.1:8890 - Failed: 'root:manager'
[-] 127.0.0.1:8890 - Failed: 'root:letmein'
[-] 127.0.0.1:8890 - Failed: 'root:cisco'
[-] 127.0.0.1:8890 - Failed: 'root:default'
[-] 127.0.0.1:8890 - Failed: 'root:root'
[-] 127.0.0.1:8890 - Failed: 'root:apc'
[-] 127.0.0.1:8890 - Failed: 'root:pass'
[-] 127.0.0.1:8890 - Failed: 'root:security'
[-] 127.0.0.1:8890 - Failed: 'root:user'
[-] 127.0.0.1:8890 - Failed: 'root:system'
[-] 127.0.0.1:8890 - Failed: 'root:sys'
[-] 127.0.0.1:8890 - Failed: 'root:none'
[-] 127.0.0.1:8890 - Failed: 'root:xampp'
[-] 127.0.0.1:8890 - Failed: 'root:wampp'
[-] 127.0.0.1:8890 - Failed: 'root:ppmax2011'
[-] 127.0.0.1:8890 - Failed: 'root:turnkey'
[-] 127.0.0.1:8890 - Failed: 'root:vagrant'
[-] 127.0.0.1:8890 - Failed: 'root:test'
[-] 127.0.0.1:8890 - Failed: 'admin:admin'
[-] 127.0.0.1:8890 - Failed: 'admin:password'
```

Figura 4.5: Output de metasploit

Dado que hemos cambiado las contraseñas de nuestros servicios, metasploit ya no puede acceder a ellos. Esta no es la mejor solución, ya que teóricamente phpMyAdmin sigue siendo vulnerable a ataques de fuerza bruta, pero dado que phpMyAdmin solo es accesible desde localhost, no es un problema de seguridad. Lo mismo ocurre con la base de datos MariaDB la cual solo es accesible mediante la web y no desde el exterior.

Conclusiones

Este proyecto ha constituido una valiosa oportunidad para constatar la vulnerabilidad intrínseca de las páginas web y servicios en línea cuando no se configuran adecuadamente. La experiencia adquirida ha demostrado ser esencial, destacando la necesidad crítica de abordar las deficiencias de seguridad que pueden comprometer la integridad y confidencialidad de los datos. En el transcurso de este proceso, hemos puesto a prueba no solo nuestras habilidades de programación, sino también nuestras capacidades de investigación, al enfrentarnos a la tarea de identificar no solo las vulnerabilidades en sí, sino de concebir métodos efectivos para su mitigación y resolución.

La comparativa entre las pruebas de penetración realizadas y presentadas en este informe revela una transformación significativa en la seguridad de la plataforma. Los ataques previamente exitosos, que tradicionalmente se emplean contra las páginas web, han perdido su eficacia. Este resultado tangible subraya el éxito de nuestras estrategias de fortificación, evidenciando el impacto positivo de las medidas implementadas. La labor llevada a cabo no solo ha consistido en cerrar brechas de seguridad, sino también en desarrollar un entorno digital robusto y resistente, capaz de hacer frente los desafíos cada vez más sofisticados que plantea el panorama actual de amenazas cibernéticas. Este proyecto, en última instancia, ha fortalecido nuestra comprensión integral de la seguridad web y ha ampliado nuestras habilidades prácticas y teóricas en este campo de estudio crucial.

Bibliografía

- OWASP. (2021). Informe de Vulnerabilidades. OWASP. <https://owasp.org/www-project-top-ten/>
- GPT-3.5. (2023). Respuestas a preguntas varias. OpenAI. <https://www.openai.com/>
- GitHub Copilot. (2022). Autocompletado. GitHub. <https://github.com/features/copilot>
- PHP. (2021). Manual de PHP. PHP. <https://www.php.net/manual/es/>
- reCAPTCHA. (2018). Documentación de reCAPTCHA. Google. <https://developers.google.com/recaptcha/docs/v3>
- sqlmap. (2017). Documentación de sqlmap. sqlmap. <https://github.com/sqlmapproject/sqlmap/wiki/>
- ZAP. (2023). Documentación de ZAP. OWASP. <https://www.zaproxy.org/docs/>
- metasploit. (2023). Documentación de metasploit. Rapid7. <https://docs.rapid7.com/metasploit/>
- nmap. (2020). Documentación de nmap. nmap. <https://nmap.org/man/es/>