

# **Documentación fallos** **de seguridad**

**Por: Diego Nido, Unax Zardoya, Mikel Rivera, Imanol Martínez**

# Índice

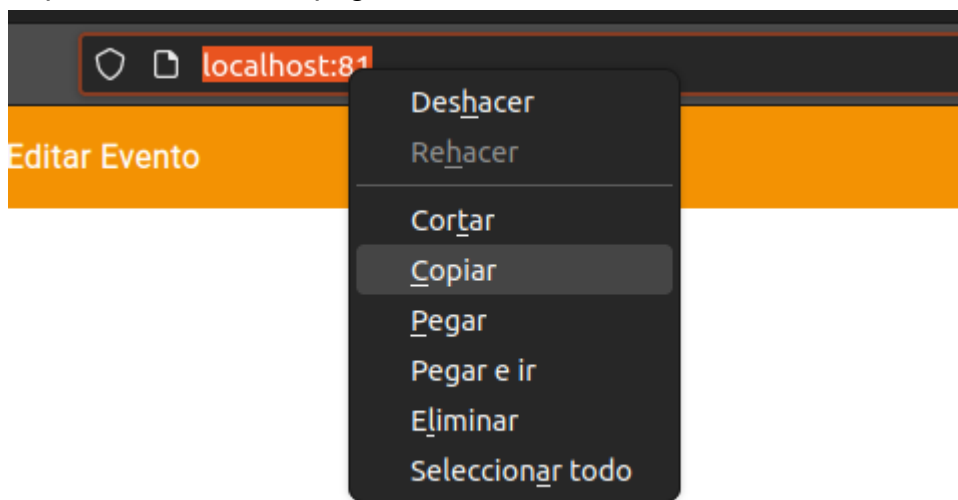
<b>Documentación fallos de seguridad.....</b>	<b>1</b>
<b>Índice.....</b>	<b>3</b>
<b>1. Auditoría inicial.....</b>	<b>5</b>
1.1. Tutorial zap.....	5
1.2. Rotura de control de acceso.....	6
1.3. Fallos criptográficos.....	17
1.4. Inyección.....	17
1.5. Diseño inseguro.....	20
1.6. Configuración de seguridad insuficiente.....	20
1.7. Componentes vulnerables y obsoletos.....	23
1.8. Fallos de identificación y autenticación.....	23
1.9. Fallos en la integridad de datos y software.....	24
1.10. Fallos en la monitorización de seguridad.....	25
<b>2. Cambios.....</b>	<b>25</b>
2.1. Filtros del lado del servidor.....	25
2.2. Consultas parametrizadas.....	26
2.3. Sanitización del output.....	27
2.4. CSRF token.....	28
2.5. Hashes y sal en la contraseña.....	28
2.6. Intentos de inicio de sesión.....	29
2.7. Uso de cabeceras de seguridad.....	30
2.8. Cambio de versiones en el container de docker.....	30
2.9. JWT token.....	31
<b>3. Auditoría tras los cambios.....</b>	<b>32</b>
3.1. Anti-CSRF.....	32
3.2. Fallos criptográficos.....	33
3.3. Inyección.....	33
3.4. Diseño inseguro.....	33
3.5. Configuración de seguridad insuficiente.....	33
3.6. Componentes vulnerables y obsoletos.....	33
3.7. Fallos de identificación y autenticación.....	33
3.8. Fallos en la integridad de datos y software.....	33
3.9. Fallos en la monitorización de seguridad.....	33
<b>Bibliografía:.....</b>	<b>33</b>



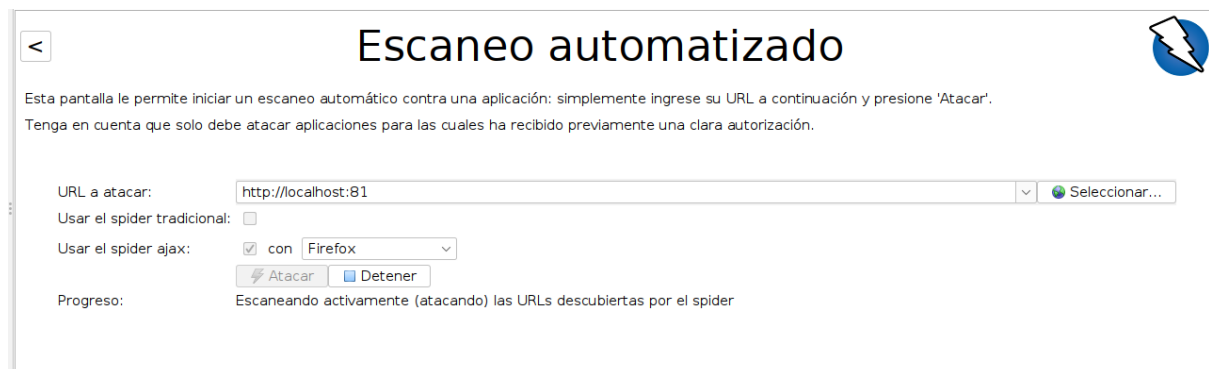
# 1. Auditoría inicial

## 1.1. Tutorial zap

Copiamos el url de la página



Vamos a Zap y en escaneo automatizado lo configuramos de la siguiente manera pegando la url de la pagina.



Nos irán saliendo por pantalla los métodos http que está probando en nuestra página

URLs vulnerables	Nodos ingresados	Mensajes
Procesado	Método	URI
●	GET	http://localhost:8080/submit_eventos.php
●	GET	https://fonts.googleapis.com/icon?family=Material+Icons
●	GET	https://stackoverflow.com/questions/3665115/how-to-create-a-file-in-memory-...
●	POST	http://localhost:81/submit.css
●	POST	http://localhost:81/submit.php
●	POST	http://localhost:81/editar.php
●	POST	http://localhost:81/editar.php
●	GET	https://stackoverflow.com/q/8935632
●	GET	https://stackoverflow.com/q/9862761
●	GET	https://stackoverflow.com/questions/71678250/how-to-post-body-data-using-f...
●	POST	http://localhost:81/perfil.css
●	POST	http://localhost:81/submit_eventos.php
●	POST	http://localhost:81/submit_eventos.php

Cuando la barra de progreso termine podemos ver las alertas

Alertas	Buscar	Alertas	Salida	Spider(Araña)	Escaneo Activo	AJAX Spider
Alertas (13)						
<ul style="list-style-type: none"> <li>Cross Site Scripting (Reflected) (2)</li> <li>Cross Site Scripting XSS (Persistente)</li> <li>Inyección SQL - MySQL (4)</li> <li><b>Ausencia de fichas (tokens) Anti-CSRF (5)</b></li> <li>Cabecera Content Security Policy (CSP) no con</li> <li>Falta de cabecera Anti-Clickjacking (8)</li> <li>El servidor divulga información mediante un ca</li> <li>Server Leaks Version Information via "Server"</li> <li>X-Content-Type-Options Header Missing (19)</li> <li>Authentication Request Identified</li> <li>Divulgación de información - Comentarios sosp</li> <li>User Agent Fuzzer (48)</li> <li>User Controllable HTML Element Attribute (Pot</li> </ul>						

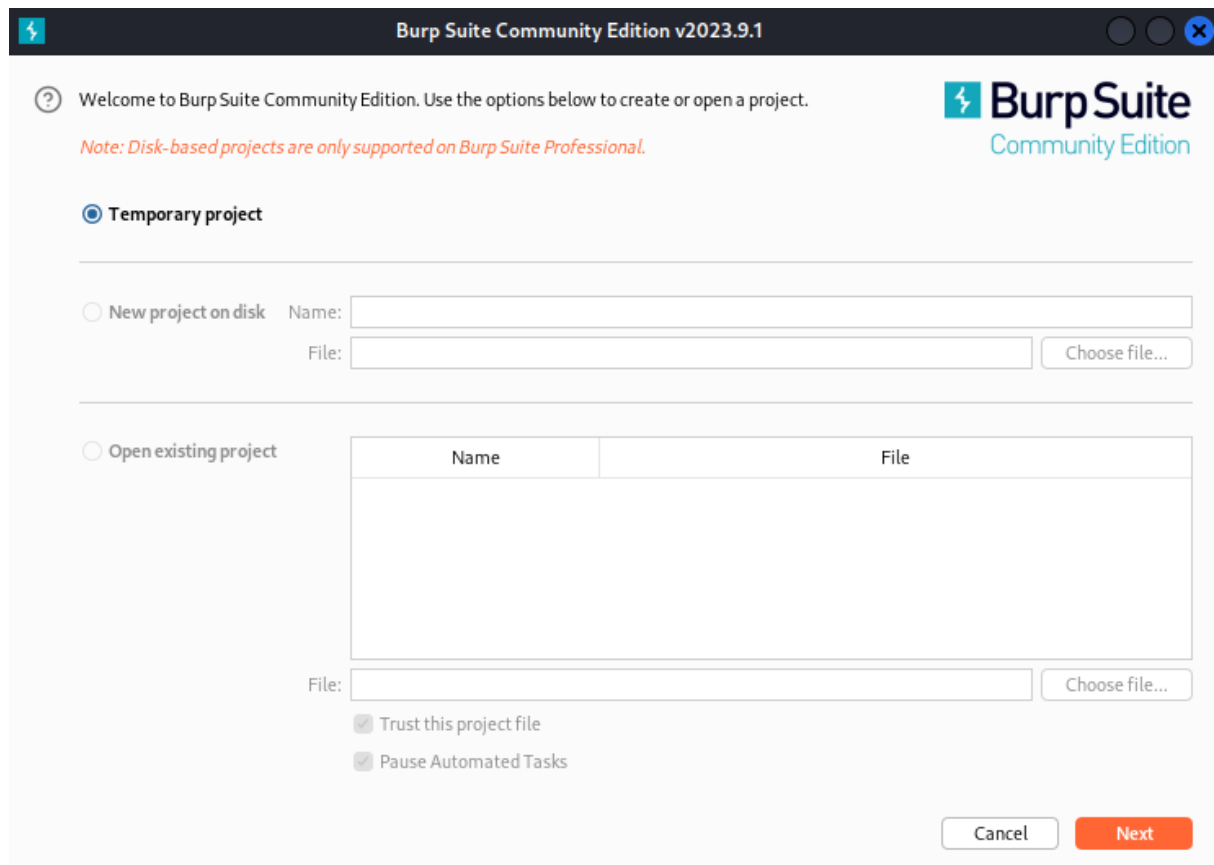
## 1.2. Rotura de control de acceso

Un usuario puede actuar con permisos que no se le han asignado. En la página web de OWASP podemos ver algunos ejemplos:

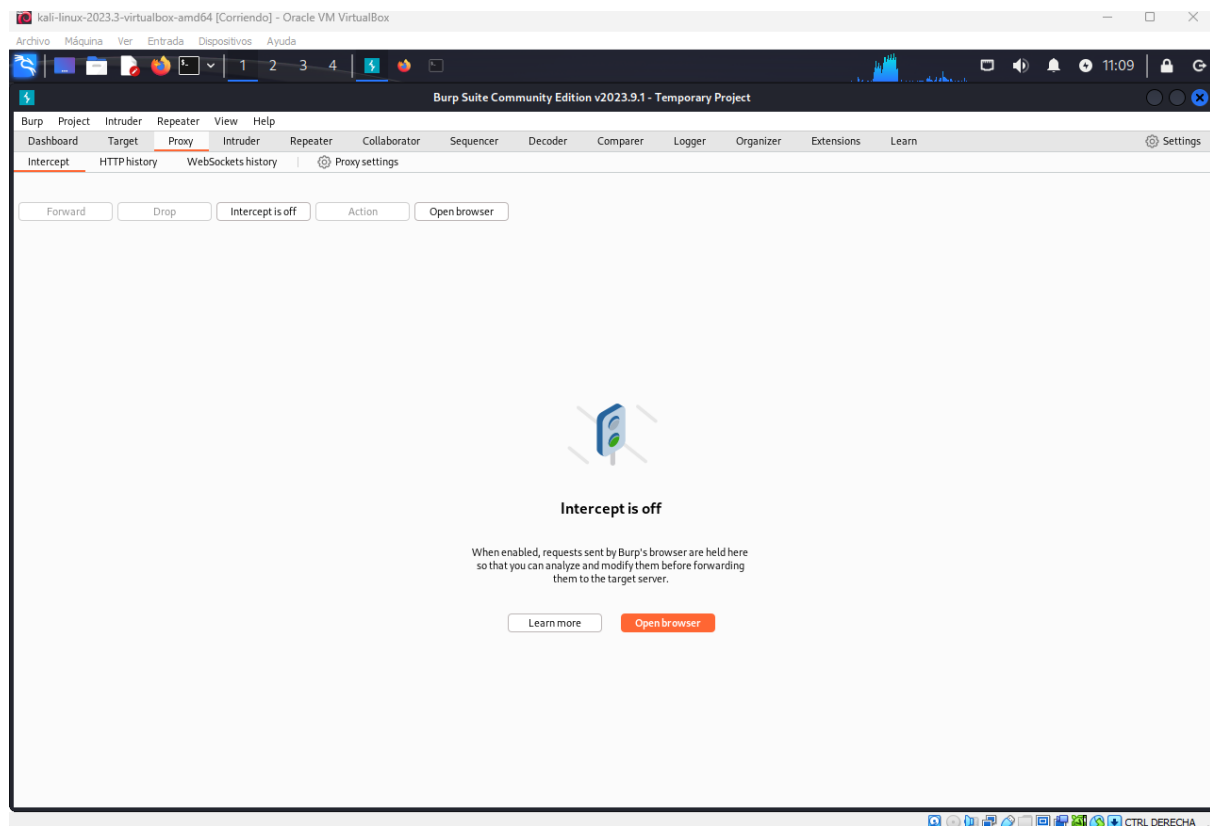
- Modificar una cookie o campo oculto
- Configuración incorrecta de CORS
- Acceder a APIs con controles de acceso inexistentes para los métodos POST, PUT y DELETE
- Permitir ver o editar la cuenta de otra persona, con tan solo conocer su identificador único (referencia directa insegura a objetos)
- Forzar la navegación a páginas autenticadas siendo usuario no autenticado o a páginas privilegiadas siendo usuario regular

Primero para comprobar si existen cookies o campos ocultos que podemos modificar utilizaremos Burpsuite, un proxy HTTP.

Ejecutamos Burpsuite y creamos un proyecto temporal:



Seleccionamos la configuración por defecto, creamos el proyecto y vamos a la pestaña de proxy:



Le damos a “Intercept is off” para activar el proxy. Después le damos a Open browser y vamos a nuestra web.

Para comprobar si existen cookies de sesión o campos ocultos interceptamos los distintos formularios que están en nuestra web.

Primero en iniciar sesión:

Eventify

localhost:81/login.php

Inicio Crear Evento Editar Evento Iniciar sesión

Nombre y Apellidos: Jon Tom

asfdasfb

Teléfono: 123456789

123456789

DNI: 11111111-Z

16087799-N

Email: jontom@gmail.com

fsadfasdasdf@dfvas

Fecha de nacimiento

11 / 23 / 2023

Nombre de usuario: JonTom123

Asdf1234

Contraseña: asd\$27

.....

Realizar

Cambiar a Iniciar sesión

Le damos a realizar y vemos en Burpsuite nuestra petición HTTP:

Burp Suite Community Edition v2023.9.1 - Temporary Project

Dashboard

Target

Proxy

Intruder

Repeater

Collaborator

Sequencer

Decoder

Comparer

Logger

Organizer

Extensions

Learn

Intercept HTTP history WebSockets history Proxy settings

Request to http://localhost:81 [127.0.0.1]

Forward

Drop

Intercept is on

Action

Open browser

Comment this item HTTP/1

Pretty

Raw

Hex

1 POST /submit.php HTTP/1.1

2 Host: localhost:81

3 Content-Length: 151

4 Cache-Control: max-age=0

5 sec-ch-ua:

6 sec-ch-ua-mobile: ?0

7 sec-ch-ua-platform: ""

8 Upgrade-Insecure-Requests: 1

9 Origin: http://localhost:81

10 Content-Type: application/x-www-form-urlencoded

11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36

12 Accept:

13 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

14 Sec-Fetch-Site: same-origin

15 Sec-Fetch-Mode: navigate

16 Sec-Fetch-User: ?1

17 Referer: http://localhost:81/login.php

18 Accept-Encoding: gzip, deflate

19 Accept-Language: en-US,en;q=0.9

20 Connection: close

21

22 nombre=asfdasfb&telefono=123456789&dni=16087799-N&email=fsadfasdasdf%40dfvas&nacimiento=2023-11-23&usuario=Asdf1234&passwd=Asdf1234&tiporegistro=signup

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 8

Request cookies 0

Request headers 19

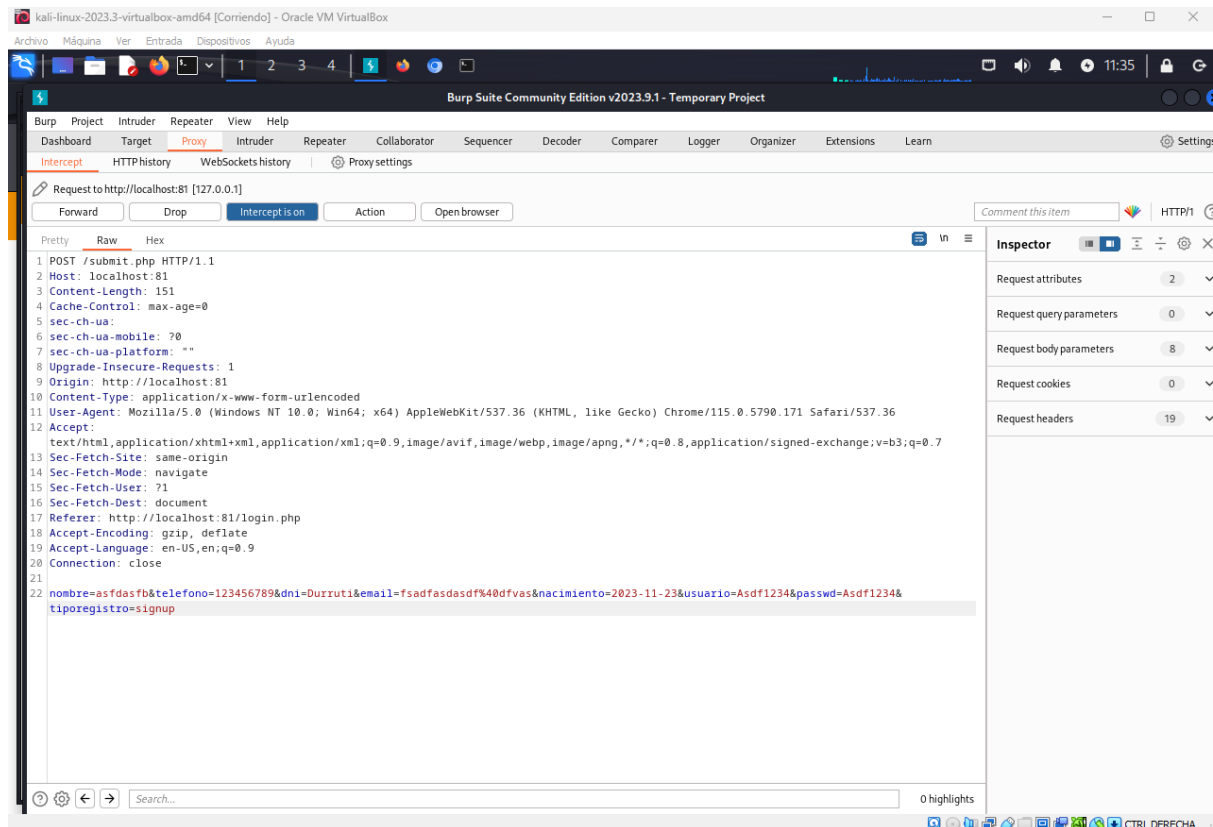
0 highlights



Podemos modificar cualquier campo y saltarnos los filtros del lado del cliente (OWASP:M4: Insufficient Input/Output Validation).

Modificamos el DNI y pondremos cualquier cosa, por ejemplo: Durruti

También se puede observar el campo “tiporegistro”, que no era visible al usuario, por lo que parece sirve para identificar si es un registro (signup) u otra cosa.



Le damos a Forward y podemos ver que nos ha asignado la cookie, el valor es el nombre de usuario. Probaremos a modificarlo por otro al ir a editar perfil.

## Response

Pretty

Raw

Hex

Render

```
1 HTTP/1.1 302 Found
2 Date: Sat, 04 Nov 2023 15:36:26 GMT
3 Server: Apache/2.4.25 (Debian)
4 X-Powered-By: PHP/7.2.2
5 Set-Cookie: user=Asdf1234; expires=Mon, 04-Dec-2023 15:36:26 GMT;
  Max-Age=2592000; path=/
6 Location: /
7 Content-Length: 0
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10
11
```



0 highlights

# Mi Perfil

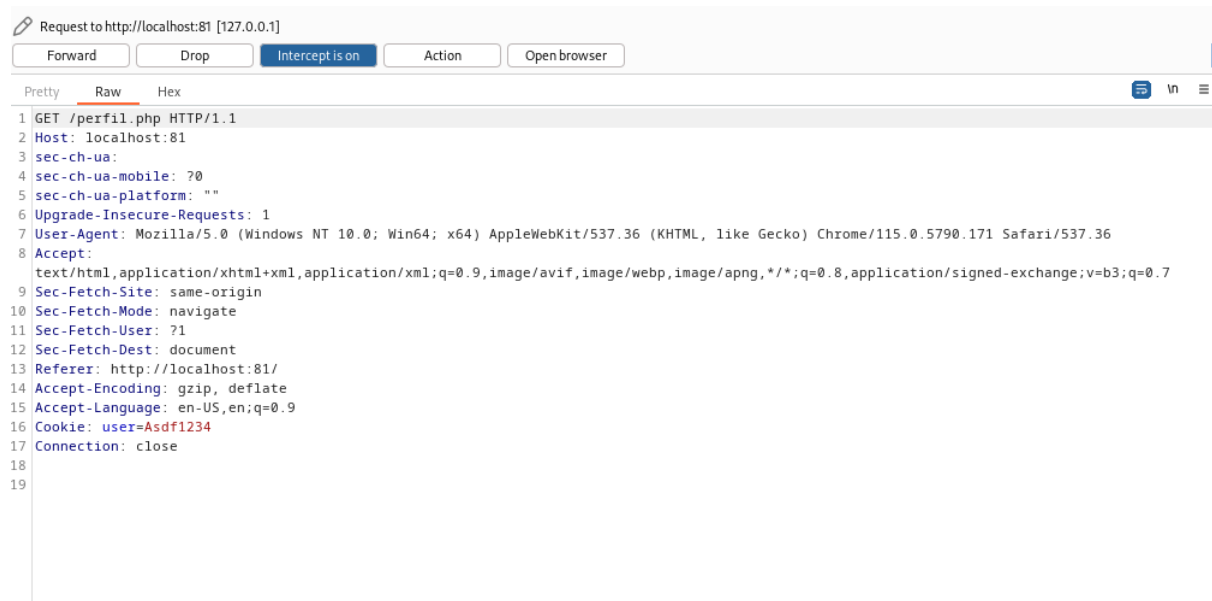
Nombre de usuario: Asdf1234

Nombre y Apellidos: asfdasfb

Teléfono: 123456789

DNI: Durruti

Se puede ver como el filtro del DNI no se ha aplicado en el lado del servidor.



Sustituiré Asdf1234 por el usuario ImanolMM, podemos saber que existe este usuario, ya que en el inicio salen los autores de diferentes “eventos”.

Podemos actuar como si fuéramos ImanolMM si utilizamos esa Cookie:

# Mi Perfil

Nombre de usuario: ImanolMM

Nombre y Apellidos: Imanol  
Martinez

Imanol Martinez

Teléfono: 684399392

684399392

DNI: 46368446-D

46368446-D

Email: imanolm.upv@gmail.com

imanolm.upv@gmail.com

Nacimiento: 2003-08-08

08/08/2003



Contraseña: imanolMM

.....

Editar

Podemos ver que las contraseñas se almacenan en texto claro e incluso podemos cambiarla sin necesidad de conocerla, puesto que nos dejan modificar cualquiera de sus datos sin haber utilizado la contraseña. Vamos a editar sus eventos, en cada petición podemos modificar la cookie para actuar como ImanolMM:



## Editar Eventos



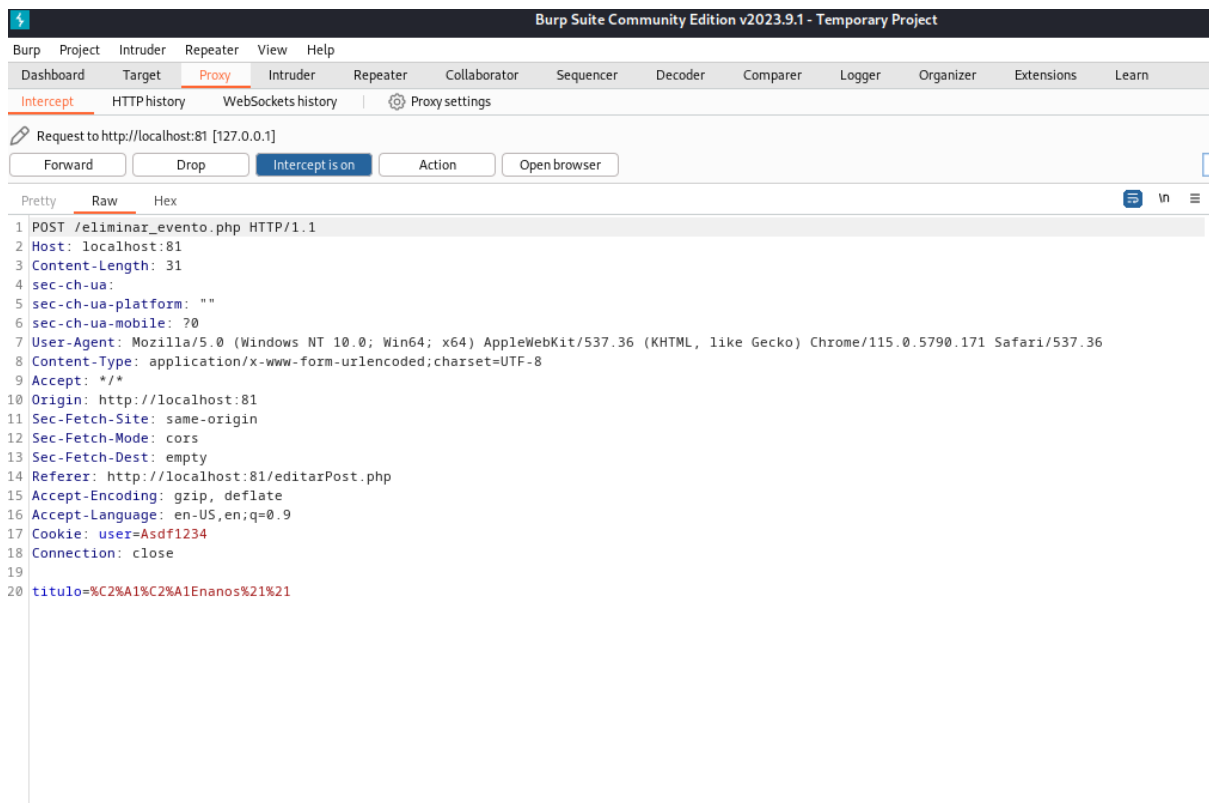
Editar

Eliminar evento

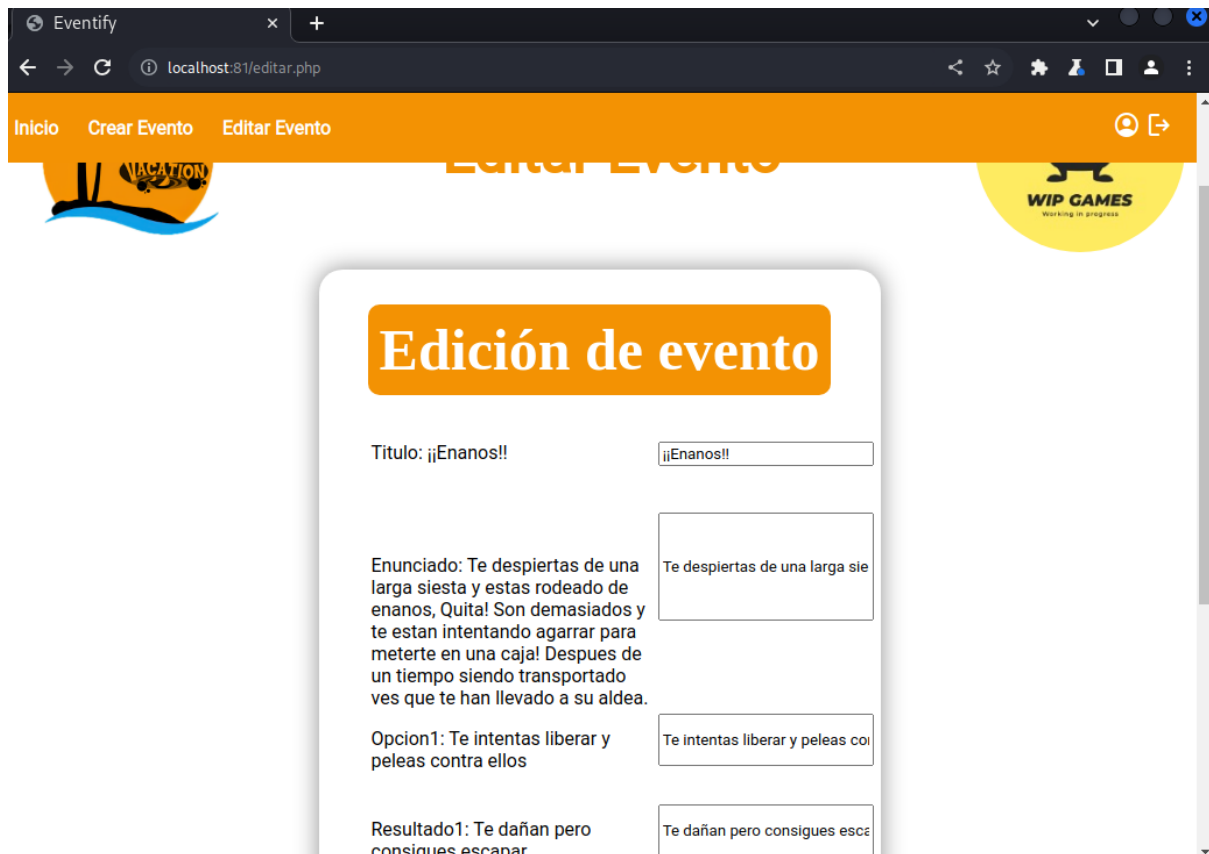
¡¡Enanos!!

Te despiertas de una larga siesta y estas rodeado de enanos, Quitá! Son demasiados y te estan intentando aga...

Le doy a eliminar:



Podemos ver que se está enviando el título junto con la Cookie. Podríamos modificar ambos para eliminar cualquier “evento” de cualquier usuario, puesto que la cookie de cualquier usuario es su nombre de usuario y el título es visible para cualquier persona. Le daré a Drop para probar la funcionalidad de editar con ese mismo “evento”:



Podemos ver como hay otro campo oculto (flagedit), posiblemente indicará si crear o editar un evento:

```

Pretty Raw Hex
1 POST /submit_eventos.php HTTP/1.1
2 Host: localhost:81
3 Content-Length: 595
4 Cache-Control: max-age=0
5 sec-ch-ua:
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: ""
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost:81
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
12 Accept:
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://localhost:81/editar.php
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Cookie: user=Asdf1234
22 Connection: close
23
24 titulo=%C2%A1%C2%A1Enanos%21%21&viejoTitulo=%C2%A1%C2%A1Enanos%21%21&enunciado=
25 Te+despiertas+de+una+larga+siesta+y+estas+rodeado+de+enanos%2C+Quita%21+Son+demasiados+y+te+estan+intentando+agarrar+para+meterte+en+una+ca
26 ja%21+Despues+de+un+tiempo+siendo+transportado+ves+que+te+han+llevado+a+su+aldea.&opcion1=Te+intentas+liberar+y+peleas+contra+ellos&
27 resultado1=Te+da%C3%B1an+pero+consigues+escapar&opcion2=Usas+tu+linterna+para+intentar+sorprenderles&resultado2=
28 Est%C3%A1n+sorprendidos.+Nunca+antes+hab%C3%ADan+visto+algo+as%C3%AD%2C++te+toman+por+su+dios+y+te+dan+de+comer+y+beber&flagedit=edit

```

De la misma forma podemos crear eventos como otro usuario.

Para solucionar estos problemas no debe poderse adivinar la Cookie de otros usuarios, por ejemplo, combinando cierta información del usuario y utilizando un algoritmo resumen/hash.

Ahora comprobaré si podemos utilizar métodos GET en formularios de tipo POST, por ejemplo, al editar el perfil cambiamos el nombre y sustituimos POST por GET:

```
1 GET /submit.php HTTP/1.1
2 Host: localhost:81
3 Content-Length: 134
4 Cache-Control: max-age=0
5 sec-ch-ua:
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: ""
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost:81
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
12 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
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost:81/perfil.php
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: user=Asdf1234
21 Connection: close
22
23 nombre=hola&telefono=123456789&dni=Durruti&email=fsadfasdasdf%40dfvas&nacimiento=2023-11-23&passwd=Asdf1234&tiporegistro=edit
```

La respuesta no indica ningún error:

```
1 GET / HTTP/1.1
2 Host: localhost:81
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
6 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
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 sec-ch-ua:
12 sec-ch-ua-mobile: ?0
13 sec-ch-ua-platform: ""
14 Referer: http://localhost:81/perfil.php
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: user=Asdf1234
18 Connection: close
19
20
```

Pero al editar el perfil no ha cambiado el usuario. Por lo que están comprobando el método HTTP:

```
1 GET / HTTP/1.1
2 Host: localhost:81
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
6 Accept:
7 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
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 sec-ch-ua:
13 sec-ch-ua-mobile: ?0
14 sec-ch-ua-platform: ""
15 Referer: http://localhost:81/perfil.php
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: user=Asdf1234
19 Connection: close
20
```

## 1.3. Fallos criptográficos


No utilizamos ningún algoritmo de encriptación (HTTPS TLS)


## 1.4. Inyección

La inyección ya sea sql o nosql se basa en insertar datos o comandos maliciosos en una aplicación o sistema con el fin de explotar debilidades y obtener acceso no autorizado o realizar acciones no deseadas.


Para la detección de estos fallos hemos empleado ZAP. Mediante un ataque con spider(como el enseñado al principio en el tutorial de zap). Al usarlo obtenemos los siguientes problemas de inyección:



SQL Injection - MySQL	
URL:	http://localhost:81/submit.php
Risk:	 High
Confidence:	Medium
Parameter:	usuario
Attack:	teyxrKvU' / sleep(15) / '
Evidence:	
CWE ID:	89
WASC ID:	19
Source:	Active (40019 - SQL Injection - MySQL)
Input Vector:	Form Query
Description:	SQL injection may be possible.
80	

SQL Injection - MySQL	
URL:	http://localhost:81/editar.php
Risk:	 High
Confidence:	Medium
Parameter:	titulo
Attack:	dHCpUfhyzedlzXX' / sleep(15) / '
Evidence:	
CWE ID:	89
WASC ID:	19
Source:	Active (40019 - SQL Injection - MySQL)
Input Vector:	Form Query
Description:	SQL injection may be possible.
80	

En estos dos casos podemos ver cómo tanto en submit.php como en editar.php se podría realizar inyección sql en las variables usuario y título respectivamente.


Cross Site Scripting (Reflected)	
URL:	http://localhost:81/editar.php
Risk:	 High
Confidence:	Medium
Parameter:	titulo
Attack:	""<script>alert(1);</script>
Evidence:	""<script>alert(1);</script>
CWE ID:	79
WASC ID:	8
Source:	Active (40012 - Cross Site Scripting (Reflected))
Input Vector:	Form Query

### Cross Site Scripting (Reflected)


URL: http://localhost:81/submit.php  
Risk:  High  
Confidence: Medium  
Parameter: usuario  
Attack: "'<script>alert(1);</scRipt>  
Evidence: "'<script>alert(1);</scRipt>  
CWE ID: 79  
WASC ID: 8  
Source: Active (40012 - Cross Site Scripting (Reflected))  
Input Vector: Form Query

En estas imágenes podemos ver otro tipo de inyección en este caso html y el concepto sería el mismo y se estarían empleando las dos mismas variables usuario y título.

### Cross Site Scripting (DOM Based)

URL: http://localhost:81/editarPost.php#jaVaScRipt:/\*-/\*`/\*\`/\*'/\*"/\*\*/(/\* \*/oNcliCk=alert(5397) )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e  
Risk:  High  
Confidence: High  
Parameter:  
Attack: #jaVaScRipt:/\*-/\*`/\*\`/\*'/\*"/\*\*/(/\* \*/oNcliCk=alert(5397) )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--!>\x3csVg/<sVg/oNloAd=alert(5397)//>\x3e  
Evidence:  
CWE ID: 79  
WASC ID: 8  
Source: Active (40026 - Cross Site Scripting (DOM Based))

### Cross Site Scripting (DOM Based)


URL: http://localhost:81/?name=abc#  
Risk:  High  
Confidence: High  
Parameter:  
Attack: ?name=abc#  
Evidence:  
CWE ID: 79  
WASC ID: 8  
Source: Active (40026 - Cross Site Scripting (DOM Based))

Por último ZAP también ha detectado inyección en la url http://localhost:81, es decir, donde se encuentra nuestra web.

## 1.5. Diseño inseguro

Se permite la creación de cuentas con datos inventados. La solución es verificar que los correos existen, son del que los ha puesto y no están en uso de otra cuenta. No utilizamos ningún SALT. (más detallado en fallos de identificación)


## 1.6. Configuración de seguridad insuficiente

**X-Content-Type-Options Header Missing**  
URL: http://localhost:81/login.css  
Riesgo:  Low  
Confianza: Medium  
Parámetro: x-content-type-options  
Ataque:  
Evidencia:  
CWE ID: 693  
WASC ID: 15  
Origen: Pasivo (10021 - X-Content-Type-Options Header Missing)  
Vector de Entrada:  
Descripción:  
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than  
Otra información:  
This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.  
At "High" threshold this scan rule will not alert on client or server error responses.  
Solución:  
Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.  
If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can  
Referencias:  
<http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx>  
<https://owasp.org/www-community/Security-Headers>

El encabezado X-Content-Type-Options, que previene la detección de tipos MIME no se configuró adecuadamente como 'nosniff'. Esto podría permitir que versiones antiguas de Internet Explorer y Chrome interpreten el contenido de una manera no deseada. Firefox actual y versiones antiguas seguirán el tipo de contenido declarado, si está presente.

**Server Leaks Version Information via "Server" HTTP Response Header Field**

URL: <http://localhost:81/login.css>

Riesgo:  Low

Confianza: High

Parámetro:

Ataque:

Evidencia: Apache/2.4.25 (Debian)

CWE ID: 200

WASC ID: 13

Origen: Pasivo (10036 - HTTP Server Response Header)

Vector de Entrada:

Descripción:  
The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Otra información:


Solución:  
Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Referencias:  
<http://httpd.apache.org/docs/current/mod/core.html#servertokens>  
[http://msdn.microsoft.com/en-us/library/ff648552.aspx#ht\\_urlscan\\_007](http://msdn.microsoft.com/en-us/library/ff648552.aspx#ht_urlscan_007)  
<http://blogs.msdn.com/b/varunm/archive/2013/04/23/remove-unwanted-http-response-headers.aspx>

El servidor web o de aplicaciones está filtrando información de la versión a través del encabezado HTTP de respuesta "Server". Permitir el acceso a esta información podría ayudar a los atacantes a identificar posibles vulnerabilidades a las que su servidor web o de aplicaciones podría estar expuesto.

**El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By""**

URL: <http://localhost:81/login.php>

Riesgo:  Low

Confianza: Medium

Parámetro:

Ataque:

Evidencia: X-Powered-By: PHP/7.2.2

CWE ID: 200

WASC ID: 13

Origen: Pasivo (10037 - El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By"")

Vector de Entrada:

Descripción:  
El servidor de la web/aplicación está divulgando información mediante uno o más encabezados de respuesta HTTP ""X-Powered-By"". El acceso a tal información podría facilitar a los atacantes la identificación de otros marcos/componentes de los que su aplicación web depende y las vulnerabilidades a las que pueden estar sujetos tales componentes.

Otra información:

Solución:  
Asegúrese que su servidor web, servidor de aplicación, equilibrador de carga, etc. está configurado para suprimir encabezados ""X-Powered-By".

Referencias:  
<http://blogs.msdn.com/b/varunm/Archive/2013/04/23/Remove-Unwanted-http-Response-headers.aspx>  
[http://www.troyhunt.com/2012/02/shhh-don ' t-deje-la-respuesta-headers.html](http://www.troyhunt.com/2012/02/shhh-don-t-deje-la-respuesta-headers.html)

El servidor de la web/aplicación está revelando información a través de los encabezados de respuesta HTTP 'X-Powered-By'. Esto podría permitir que los atacantes identifiquen los componentes y vulnerabilidades de su aplicación web.

**Falta de cabecera Anti-Clickjacking**

URL: http://localhost:81/  
 Riesgo:  Medium  
 Confianza: Medium  
 Parámetro: x-frame-options  
 Ataque:  
 Evidencia:  
 CWE ID: 1021  
 WASC ID: 15  
 Origen: Pasivo (10020 - Cabecera Anti-Clickjacking)  
 Referencia de Alerta: 10020-1  
 Vector de Entrada:

**Descripción:**  
 La respuesta no incluye Content-Security-Policy con la directiva 'frame-ancestors' ni X-Frame-Options para proteger contra ataques de 'Clickjacking'.

**Otra información:**

**Solución:**  
 Los navegadores web modernos admiten los encabezados HTTP Content-Security-Policy y X-Frame-Options. Asegúrese de que uno de ellos esté configurado en todas las páginas web devueltas por su sitio/aplicación.  
 Si espera que la página esté enmarcada solo por páginas en su servidor (por ejemplo, es parte de un FRAMESET), querrá usar SAMEORIGIN; de lo

**Referencias:**  
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>

La respuesta no incluye Content-Security-Policy con la directiva 'frame-ancestors' ni X-Frame-Options para proteger contra ataques de 'ClickJacking'.

**Cabecera Content Security Policy (CSP) no configurada**

URL: http://localhost:81/robots.txt  
 Riesgo:  Medium  
 Confianza: High  
 Parámetro:  
 Ataque:  
 Evidencia:  
 CWE ID: 693  
 WASC ID: 15  
 Origen: Pasivo (10038 - Cabecera Content Security Policy (CSP) no configurada)  
 Referencia de Alerta: 10038-1  
 Vector de Entrada:

**Descripción:**  
 La Política de seguridad de contenido (CSP) es una capa adicional de seguridad que ayuda a detectar y mitigar ciertos tipos de ataques, incluidos Cross Site Scripting (XSS) y ataques de inyección de datos. Estos ataques se utilizan para todo, desde el robo de datos hasta la desfiguración del sitio o la distribución de malware. CSP proporciona un conjunto de encabezados HTTP estándar que permiten a los propietarios de sitios web

**Otra información:**

**Solución:**  
 Asegúrese de que su servidor web, servidor de aplicaciones, balanceador de carga, etc. esté configurado para establecer la cabecera Content-Security-Policy.

**Referencias:**  
[https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing\\_Content\\_Security\\_Policy](https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy)  
[https://cheatsheetseries.owasp.org/cheatsheets/Content\\_Security\\_Policy\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html)

CSP (Política de Seguridad de Contenido) es una medida de seguridad que protege contra ataques como XSS e inyección de datos. Permite a los propietarios de sitios web especificar las fuentes de contenido permitidas para que los navegadores carguen en una página, abarcando JavaScript, CSS, marcos HTML, imágenes y más. Esto reduce el riesgo de robo de datos y otros ataques en el sitio web.

**Ausencia de fichas (tokens) Anti-CSRF**

URL: <http://localhost:81/login.php>

Riesgo:  Medium

Confianza: Low

Parámetro:

Ataque:

Evidencia: `<form class="form" action="/submit.php" id="form-registro" method="POST">`

CWE ID: 352

WASC ID: 9

Origen: Pasivo (10202 - Ausencia de fichas (tokens) Anti-CSRF)

Vector de Entrada:

Descripción:

No se encontraron fichas (tokens) Anti-CSRF en un formulario HTML.  
Una solicitud falsa entre sitios en un ataque que compromete y obliga a una víctima a enviar su solicitud HTTP a un destino objetivo sin su conocimiento o intención para poder realizar una acción como víctima. La causa oculta es la funcionalidad de la aplicación utilizando acciones de

Otra información:

Ninguna ficha (token) Anti-CSRF [anticsrf, CSRFToken, \_\_RequestVerificationToken, csrfmiddlewaretoken, authenticity\_token, OWASP\_CSRFTOKEN, anoncsrf, csrf\_token, \_\_csrf, \_\_csrfSecret, \_\_csrf\_magic, CSRF, \_token, \_csrf\_token] fue encontrada en los siguientes formularios HTML: [Form 1: "dni" "email" "nacimiento" "nombre" "passwd" "telefono" "tiporegistro" "usuario" ].

Solución:

Fase: Arquitectura y Diseño  
Utilizar una biblioteca o framework verificado y confiable que evite esta vulnerabilidad o proporcione elementos que faciliten evitarla.  
Por ejemplo, utilice el paquete anti-CSRG como el CSRGuard de OWASP.

Referencias:

<http://projects.webappsec.org/Cross-Site-Request-Forgery>  
<http://cwe.mitre.org/data/definitions/352.html>

El error advierte sobre la falta de tokens Anti-CSRF en un formulario HTML y explica el concepto de ataque CSRF, que involucra forzar a una víctima a realizar acciones no deseadas en un sitio web sin su conocimiento. Estos ataques son efectivos en ciertas condiciones, como cuando la víctima está autenticada en el sitio objetivo o está en la misma red. También se menciona que el riesgo aumenta cuando se combina con ataques XSS.

También se nos indica que no tenemos HttpOnly ni SameSite en nuestras cookies. La configuración de SameSite y HttpOnly en las cookies proporciona capas adicionales de seguridad y control sobre cómo se manejan las cookies en el navegador. Aquí hay algunas implicaciones de no establecer estas configuraciones:

#### SameSite:

- Si no se establece SameSite: La cookie se enviará en todas las solicitudes, incluso si provienen de sitios externos (terceros). Esto podría aumentar el riesgo de ataques CSRF (Cross-Site Request Forgery), ya que las cookies también se enviarían en solicitudes desde otros sitios.
- Con SameSite=Lax: La cookie se enviará solo en solicitudes que provengan del mismo sitio (o de una navegación principal desde un sitio externo). Esto ayuda a mitigar el riesgo de CSRF al limitar el alcance de las cookies.
- Con SameSite=Strict: La cookie se enviará solo en solicitudes que provengan del mismo sitio. Esto proporciona la máxima protección contra ataques CSRF.

#### HttpOnly:

- Si no se establece HttpOnly: La cookie es accesible desde scripts del lado del cliente (JavaScript). Esto podría aumentar el riesgo de ataques

XSS (Cross-Site Scripting), ya que un atacante podría intentar robar la cookie utilizando scripts maliciosos.

- Con HttpOnly: La cookie solo es accesible desde el lado del servidor y no puede ser manipulada por scripts del lado del cliente, lo que mejora la seguridad contra ataques XSS.

## 1.7. Componentes vulnerables y obsoletos

Buscando en <https://www.exploit-db.com/> los componentes de la aplicación, phpmyadmin 5.10.2 y mariadb 10.8.2, no tienen exploits (al menos de momento)

## 1.8. Fallos de identificación y autenticación

En la base de datos las contraseñas no están resumidas en un hash por lo que una rotura en la seguridad de la base de datos podría resultar en una filtración de todas las contraseñas de los usuarios. Podemos descubrir que no están resumidas cuando editamos nuestro perfil. La tabla de la base de datos se ve de la siguiente forma:

nombre	telef	dni	email	nacimiento	usuario	passwd
Imanol Martinez	684399392	46368446-D	imanolm.upv@gmail.com	2003-08-08	ImanolMM	imanolMM

Tampoco se establece un límite de intentos fallidos al iniciar sesión por lo que cualquier persona puede realizar un ataque de fuerza bruta hasta obtener todos los usuarios y sus contraseñas.

## 1.9. Fallos en la integridad de datos y software

A la hora de crear el container de docker tenemos puesto que use la última versión de phpMyAdmin lo que resulta en un fallo de seguridad ya que en una versión nueva podría haber una apertura en la seguridad o bien un actor malicioso puede haberse hecho pasar por proveedor y, en consecuencia, podría introducir vulnerabilidades o software malicioso en la actualización:

```
22     MYSQL_DATABASE: database
23     ports:
24     - "8889:3306"
25
26     phpmyadmin:
27     image: phpmyadmin/phpmyadmin:latest
28     links:
29     - db
30     ports:
31     - 8890:80
32     environment:
33     MYSQL_USER: admin
34     MYSQL_PASSWORD: test
35     MYSQL_DATABASE: database
```

## 1.10. Fallos en la monitorización de seguridad

No se guardan los intentos de autenticación fallidos, por esto podemos suponer que en ningún momento, durante el uso de la aplicación, se crean logs. Para solucionarlo habrá que crear un archivo en el que desde php guardaremos tanto los intentos fallidos como posibles ataques a la web (nos llega información a la API que no podría haber pasado el filtro de Javascript, por tanto, se trata de un atacante)



## 2. Cambios

### 2.1. Filtros del lado del servidor

Para solucionar que el usuario se pueda saltar los filtros a la hora de enviar una petición al servidor hemos puesto los filtros del lado del servidor para las funciones sign in, sign up y editar evento:

```
if($tipo == "signup" && !$existeUsuario){
    $error = false;
    $motivo = "";
    if (!comprobarEmail($email)){
        $error = true;
        $motivo = "Email no válido";
    }elseif (!comprobarNombre($nombre)){
        $error = true;
        $motivo = "Nombre no válido";
    }elseif (!comprobarNacimiento($nacimiento)){
        $error = true;
        $motivo = "Fecha de nacimiento no válida";
    }elseif (!comprobarUsuario($usuario)){
        $error = true;
        $motivo = "Usuario no válido";
    }elseif (!validarDNI($dni)){
        $error = true;
        $motivo = "DNI no válido";
    }elseif (!comprobarPasswd($passwd)){
        $error = true;
        $motivo = "Contraseña no válida";
    }
}
```

Las funciones son muy parecidas a las de java script, aquí un ejemplo:

```
function comprobarNombre($nombre) {
    // Solo letras y espacios
    if (preg_match('/^[A-Za-z\sñÑáéíóúÁÉÍÓÚçÇ]+$/ ', $nombre)) {
        return true;
    } else {
        return false;
    }
}
```

### 2.2. Consultas parametrizadas

Para arreglar la inyección sobre consultas sql empleamos consultas parametrizadas como la que se ve en la imagen:

```
$consulta = "DELETE FROM eventos WHERE titulo = ? AND usuario = ?";
$tipos = "ss";
$parametros = array($titulo, $usuario);
if($stmt = mysqli_prepare($conn, $consulta)){
    $stmt->bind_param($tipos, ...$parametros);
    $stmt->execute();
    $stmt->close();
    $mensaje = "Evento eliminado"; // respuesta que recibirá el fetch de eliminarEvento.j
}else{
    $mensaje = "error";
}
$mensaje = $mensaje . " , titulo: " . $titulo;
echo $mensaje;
```

Aquí como se puede observar la consulta que realizamos comprueba que los datos son válidos antes de ejecutar la consulta en sí (utilizando marcadores de posición `?`), por tanto no se podría inyectar código y que se tomase como una consulta válida.

Esto mismo lo hemos repetido con todos los accesos a la base de datos susceptibles a poder recibir inyecciones

## 2.3. Sanitización del output

Para evitar ataques de XSS utilizaremos la función `htmlspecialchars()` de PHP, esta transformará los caracteres HTML que ha introducido el usuario para que no puedan inyectar código Javascript/HTML/CSS. Por ejemplo en `editar.php`:

```

while ($row = mysqli_fetch_array($query)) {
echo '<div class="formbox">
    <div class="form-title">
        Edición de evento
    </div>
    <!-- Alinear inputs https://stackoverflow.com/questions/4309950/how-to-align-input-forms-in-html -->
    <form class="form" action="/submit_eventos.php" id="form-registro" method="POST">
        <div class="linea-form">
            <p>Titulo: '.$row['titulo'].'</p>
            <input type="text" name="titulo" value="'.htmlspecialchars($row['titulo'], ENT_QUOTES).'">
            <input type="hidden" name="viejoTitulo" value="'.htmlspecialchars($row['titulo'], ENT_QUOTES).'">
        </div>
        <div class="linea-form">
            <p>Enunciado: '.$row['enunciado'].'</p>
            <input type="text" name="enunciado" value="'.htmlspecialchars($row['enunciado'], ENT_QUOTES).'">
        </div>
        <div class="linea-form">
            <p>Opcion1: '.$row['opcion1'].'</p>
            <input type="text" name="opcion1" value="'.htmlspecialchars($row['opcion1'], ENT_QUOTES).'">
        </div>
        <div class="linea-form">
            <p>Resultado1: '.$row['resultado1'].'</p>
            <input type="text" name="resultado1" value="'.htmlspecialchars($row['resultado1'], ENT_QUOTES).'">
        </div>
        <div class="linea-form">
            <p>Opcion2: '.$row['opcion2'].'</p>
            <input type="text" name="opcion2" value="'.htmlspecialchars($row['opcion2'], ENT_QUOTES).'">
        </div>
        <div class="linea-form">
            <p>Resultado2: '.$row['resultado2'].'</p>
            <input type="text" name="resultado2" value="'.htmlspecialchars($row['resultado2'], ENT_QUOTES).'">
        </div>
        <div class="linea-form">
            <input type="hidden" value="edit" name="flagedit">
            <p>
                <button type="submit" class="boton" id="botonRegistro">Editar</button>
            </p>
        </div>
    </form>
</div>';
}

```

Este proceso lo hemos repetido en todos los outputs.

## 2.4. CSRF token

Para evitar ataques CSRF implementaremos esto:

<https://www.phptutorial.net/php-tutorial/php-csrf/>

Además los desarrolladores que decidan utilizar Eventify deben cambiar el usuario y la contraseña de la base de datos en el docker-compose.yml (también se podrían utilizar secretos de Docker pero, de la misma forma, el usuario deberá elegir una contraseña):

```

db:
  image: mariadb@sha256:490f01279be1452f12f497a592112cb960cf0500938dbf0ea3f0135cb6728d3d
  restart: always
  volumes:
    - ./database.sql:/docker-entrypoint-initdb.d/setup.sql
    - ./mysql:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_USER: admin
    MYSQL_PASSWORD: test
    MYSQL_DATABASE: database
  ports:
    - "8889:3306"

phpmyadmin:
  image: phpmyadmin/phpmyadmin@sha256:67ba2550fd004399ab0b95b64021a88ea544011e566a9a1995180a3decb6410d
  links:
    - db
  ports:
    - 8890:80
  environment:
    MYSQL_USER: admin
    MYSQL_PASSWORD: test
    MYSQL_DATABASE: database

```

Puesto que el usuario y la contraseña por defecto son inseguras. Por esto, le indicaremos en el readme al usuario que clone el repositorio que cambie el usuario y contraseña.

## 2.5. Hashes y sal en la contraseña

Para solucionar el fallo en la identificación y autenticación vamos a crear un sistema en el que lo que se guarde en la base de datos sea un hash de la contraseña para cuando el usuario se identifique comprobemos el hash que se crea con esa contraseña con la de la base de datos. Además para asegurar mayor seguridad vamos a almacenar una SAL. Esta SAL se genera al crear la cuenta y al cambiar cualquier dato de la cuenta. Código referente a resumir contraseñas:

```

if($tipo == "signup" && !$existeUsuario){

    $consulta = "INSERT INTO usuarios(nombre,telef,dni,email,nacimiento,usuario,passwd,sal) VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
    $sal = bin2hex(random_bytes(16));
    $contraseñaSal = $sal . $passwd;
    $contraseña = password_hash($contraseñaSal, PASSWORD_BCRYPT);
    $tipos = "sissssss";
    $parametros = array($nombre, (int) $telef, $dni, $email, $nacimiento, $usuario, $contraseña, $sal);

    if($stmt = mysqli_prepare($conn, $consulta)){
        $stmt->bind_param($tipos, ...$parametros);
        $stmt->execute();
        $stmt->close();
        $mensaje = "Usuario creado";
        // https://www.w3schools.com/php/func_network_setcookie.asp
        $cookie_name = "user";
        $cookie_value = $usuario;
        setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 1 día de duración
    }
}

```

Así queda la base de datos:

nombre	telef	dni	email	nacimiento	usuario	passwd	sal
aa	688851580	79113393-v	aa@gmail.com	2003-04-28	aa	\$2y\$10\$sDdSOWniZlaNYIGH1aYQBu52KkTCFPTdOFHadxAr/Gm...	5df8b35fe

También tenemos que hacer el código para la comprobación en el sign in:

```
$consulta_usuario = "SELECT * FROM usuarios WHERE usuario = ?";
$stmt = mysqli_prepare($conn, $consulta_usuario);
mysqli_stmt_bind_param($stmt, "s", $usuario);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);

$passCorrecta = false;
$ip_usuario = $_SERVER['REMOTE_ADDR'];

while ($row = mysqli_fetch_array($result)) {
    $sal = $row['sal'];
    $con = $sal . $passwd;
    $passCorrecta = password_verify($con, $row['passwd']);
}
```

## 2.6. Intentos de inicio de sesión

Para solucionar el problema de hacer demasiados inicios de sesión habrá que implementar un límite de intentos basado en la dirección IP, es posible que un atacante realice este ataque desde muchas direcciones ips y, por ello, un captcha sería la mejor opción, pero no queremos utilizarlo para este proyecto.

El límite de intentos lo hemos establecido de esta manera:

```
$consulta_accesos = "SELECT intentos FROM accesos WHERE usuario = ? AND ip = ? AND fecha = CURDATE()";
$stmt = mysqli_prepare($conn, $consulta_accesos);
mysqli_stmt_bind_param($stmt, "ss", $usuario, $ip_usuario);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
$intentos=0;
if ($row = mysqli_fetch_array($result)) {
    $intentos = $row['intentos'];
}
if ($passCorrecta && $intentos < 5) {
    $mensaje = "Inicio de sesión correcto";
    setCookieUsuarioSegura($usuario);
    header("Location: /");
    exit();
}
else if ($intentos >= 5) {
    $mensaje = "Demasiados intentos, espere 24 horas";
}
```

## 2.7. Uso de cabeceras de seguridad

Para solucionar los problemas planteados sobre la configuración de seguridad insuficiente hemos realizado un archivo .htaccess. Este es un archivo de

configuración utilizado en servidores web que ejecutan el software Apache. El nombre "htaccess" significa "hipertexto de acceso", y el archivo se utiliza para controlar la configuración del servidor web y realizar diversas tareas, como la reescritura de URL, la autenticación de usuarios, la redirección, entre otras.

```
1 <IfModule mod_headers.c>
2   Header set Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline'
3   Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
4   Header always set X-Content-Type-Options "nosniff"
5   Header always set X-Frame-Options "DENY"
6   Header unset Server
7   Header unset X-Powered-By
8 </IfModule>
9 <Files "composer.json">
10  Deny from all
11 </Files>
12 <Files "composer.lock">
13  Deny from all
14 </Files>
15 RedirectMatch 403 ^/vendor/.*$
```

También deniega el acceso a archivos que se han añadido durante el proyecto y que proporcionarían información útil al atacante.

## 2.8. Cambio de versiones en el container de docker

Para solucionar el fallo de integridad de datos lo que hacemos es poner una versión fija que sabemos que no tiene fallos de seguridad, además utilizaremos sus hashes para verificar su integridad:

```
1  web:
2    image: web
3    environment:
4      - ALLOW_OVERRIDE=true
5    ports:
6      - "81:80"
7    links:
8      - db
9    volumes:
10     - ./app:/var/www/html/
11
12  db:
13    image: mariadb@sha256:490f01279be1452f12f497a592112cb960cf0500938dbf0ea3f0135cb6728d3d
14    restart: always
15    volumes:
16     - ./database.sql:/docker-entrypoint-initdb.d/setup.sql
17     - ./mysql:/var/lib/mysql
18    environment:
19      MYSQL_ROOT_PASSWORD: root
20      MYSQL_USER: admin
21      MYSQL_PASSWORD: test
22      MYSQL_DATABASE: database
23    ports:
24     - "8889:3306"
25
26  phpmyadmin:
27    image: phpmyadmin/phpmyadmin@sha256:67ba2550fd004399ab0b95b64021a88ea544011e566a9a1995180a3decb6410d
28    links:
29     - db
30    ports:
31     - 8890:80
32    environment:
33      MYSQL_USER: admin
34      MYSQL_PASSWORD: test
35      MYSQL_DATABASE: database
36
37  composer:
38    image: composer/composer@sha256:d79e1a62bfecc274f6eddd9f64be151813e863fdd8572d184209eca8dd21e9fc
39    ports:
40     - "9001:9000"
41    volumes:
42     - ./app:/var/www/html/
43    container_name: composer
44    working_dir: /var/www/html/
45    command: install
```

## 2.9. Ajustes de las cookies

```
Set-Cookie: user=invitado; expires=Fri, 24-Nov-2023 19:50:23 GMT;  
Max-Age=300; path=/; SameSite=Strict; HttpOnly
```

```
Set-Cookie: PHPSESSID=68851083b9935f9f476fa1ad73fa4ff6; path=/  
samesite=Restrict; domain=localhost:81; secure; HttpOnly
```

```
$secure = false; // solo https  
$httponly = true; // no se puede acceder a la cookie con javascript  
$samesite = 'Restrict';  
  
if(PHP_VERSION_ID < 70300) {  
    session_set_cookie_params($maxlifetime, '/; samesite='.$samesite, $_SERVER['HTTP_HOST'])  
} else {  
    session_set_cookie_params([  
        'lifetime' => $maxlifetime,  
        'path' => '/',  
        'domain' => $_SERVER['HTTP_HOST'],  
        'secure' => $secure,  
        'httponly' => $httponly,  
        'samesite' => $samesite  
    ]);  
}  
session_start();  
  
setcookie("user", $val, time() + (5 * 60), "/; SameSite=Strict; HttpOnly");
```

La configuración de SameSite y HttpOnly en las cookies proporciona capas adicionales de seguridad y control sobre cómo se manejan las cookies en el navegador. Aquí hay algunas implicaciones de no establecer estas configuraciones:

### SameSite:

- Si no se establece SameSite: La cookie se enviará en todas las solicitudes, incluso si provienen de sitios externos (terceros). Esto podría aumentar el riesgo de ataques CSRF (Cross-Site Request Forgery), ya que las cookies también se enviarían en solicitudes desde otros sitios.
- Con SameSite=Lax: La cookie se enviará solo en solicitudes que provengan del mismo sitio (o de una navegación principal desde un sitio externo). Esto ayuda a mitigar el riesgo de CSRF al limitar el alcance de las cookies.
- Con SameSite=Strict: La cookie se enviará solo en solicitudes que provengan del mismo sitio. Esto proporciona la máxima protección contra ataques CSRF.

### HttpOnly:

- Si no se establece HttpOnly: La cookie es accesible desde scripts del lado del cliente (JavaScript). Esto podría aumentar el riesgo de ataques XSS



(Cross-Site Scripting), ya que un atacante podría intentar robar la cookie utilizando scripts maliciosos.

- Con HttpOnly: La cookie solo es accesible desde el lado del servidor y no puede ser manipulada por scripts del lado del cliente, lo que mejora la seguridad contra ataques XSS.

## 2.10. JWT token

JWT significa JSON Web Token, y es un estándar abierto (RFC 7519) que define una manera compacta y autónoma de representar información entre dos partes de una manera que es segura y se puede verificar. Los tokens JWT son comúnmente utilizados para autenticación y autorización en aplicaciones web y servicios.

Un token JWT consiste en tres partes:

Encabezado (Header): Contiene información sobre cómo se debe procesar el token y qué tipo de firma se utiliza. Este se codifica en Base64 y típicamente contiene dos partes: el tipo de token (JWT) y el algoritmo de firma utilizado, por ejemplo:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Cuerpo (Payload): Contiene la información que se quiere transmitir. También se codifica en Base64 y puede contener cualquier cantidad de pares clave-valor. Existen algunas claves predefinidas, como "iss" (emisor), "sub" (sujeto), "exp" (tiempo de expiración), "iat" (tiempo de emisión), entre otros. Por ejemplo:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

Firma (Signature): Se utiliza para verificar que el remitente del token es quien dice ser y para garantizar que la parte del token no ha sido modificada en el camino. La firma se calcula mediante la combinación del encabezado codificado en Base64, el cuerpo codificado en Base64 y una clave secreta, utilizando el algoritmo especificado en el encabezado.

Cuando un usuario se autentica en una aplicación, el servidor genera un token JWT y lo envía al cliente. El cliente almacena el token y lo incluye en cada solicitud subsiguiente al servidor. El servidor verifica la validez del token para autorizar las solicitudes.

## 2.11. Logs

Hemos añadido logs para intentos fallidos de registro y saltos de filtros del lado del cliente. De esta manera podemos identificar si hay alguien intentando crear una cuenta con objetivos maliciosos. En este log guardamos la ip para al revisarlo poder saber quien lo está haciendo y si se considera necesario banear esa ip. Por otro lado también guardamos el momento del intento fallido del registro. Código:

```
function logFailedSignUpAttempt($username, $ipAddress, $message, $type) {  
    $logDirectory = "logs";  
    $logFile = "failed_signin_attempts.log";  
    $timestamp = date("Y-m-d H:i:s");  
  
    // Mensaje de registro  
    $logMessage = "$message at $timestamp from IP $ipAddress for user $username on $type\n";  
  
    // Guardar el registro en el archivo  
    file_put_contents($logFile, $logMessage, FILE_APPEND | LOCK_EX);  
}
```

## 3. Auditoría tras los cambios

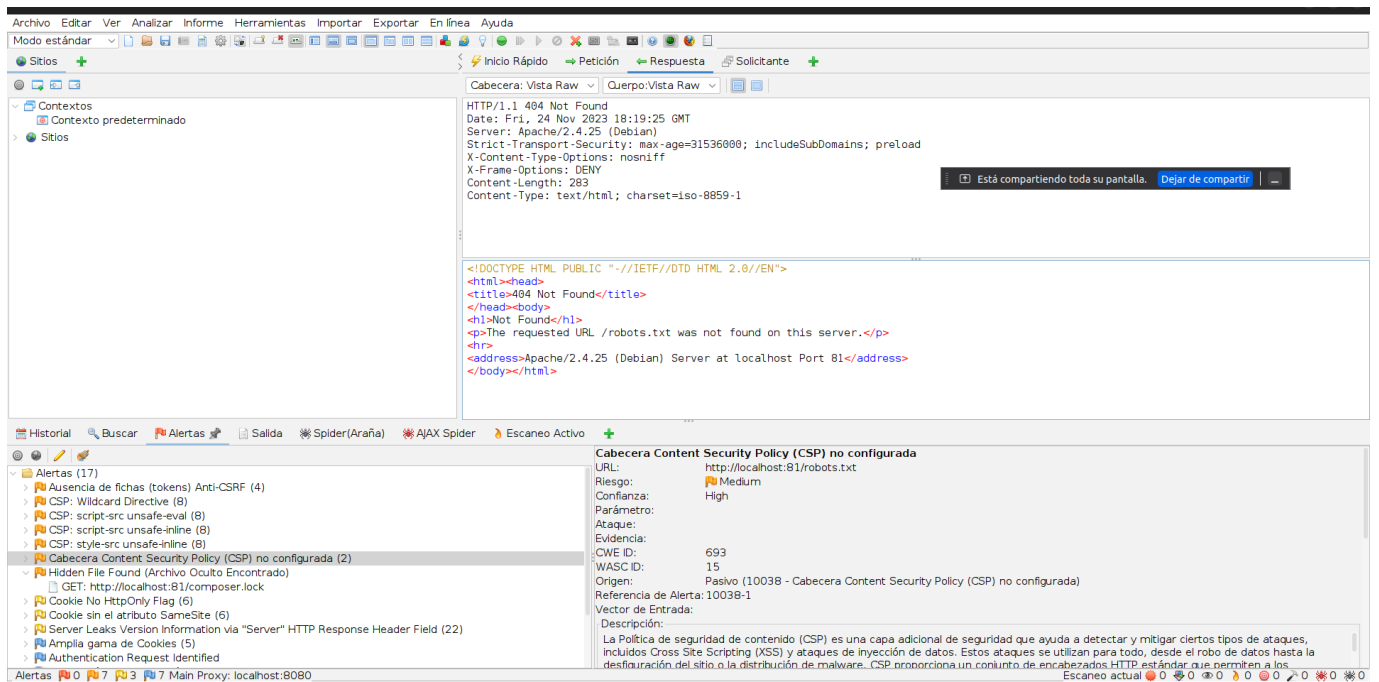
- Alerts (12)
  - Absence of Anti-CSRF Tokens (4)
  - CSP: Wildcard Directive (8)
  - CSP: script-src unsafe-eval (8)
  - CSP: script-src unsafe-inline (8)
  - CSP: style-src unsafe-inline (8)
  - Content Security Policy (CSP) Header Not Set (2)
  - Server Leaks Version Information via "Server" HTTP Response Header Field (21)
  - Authentication Request Identified
  - Information Disclosure - Suspicious Comments
  - Loosely Scoped Cookie (7)
  - Session Management Response Identified (7)
  - User Controllable HTML Element Attribute (Potential XSS)

### 3.1. Anti-CSRF

Origen: Pasivo (10202 - Ausencia de fichas (tokens) Anti-CSRF)	
Vector de Entrada:	
Descripción:	
No se encontraron fichas (tokens) Anti-CSRF en un formulario HTML. Una solicitud falsa entre sitios en un ataque que compromete y obliga a una víctima a enviar su solicitud HTTP a un destino objetivo sin su conocimiento o intención para poder realizar una acción como víctima. La causa oculta es la funcionalidad de la aplicación utilizando acciones	
Otra información:	
Ninguna ficha (token) Anti-CSRF [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] fue encontrada en los siguientes formularios HTML: [Form 1: "dni" "email" "nacimiento" "nombre" "passwd" "telefono" "tiporegistro" "token" "usuario" ].	
Solución:	
Fase: Arquitectura y Diseño Utilizar una biblioteca o framework verificado y confiable que evite esta vulnerabilidad o proporcione elementos que faciliten evitarla. Por ejemplo, utilice el paquete anti-CSRG como el CSRGuard de OWASP.	
Referencias:	
<a href="http://projects.webappsec.org/Cross-Site-Request-Forgery">http://projects.webappsec.org/Cross-Site-Request-Forgery</a> <a href="http://cwe.mitre.org/data/definitions/352.html">http://cwe.mitre.org/data/definitions/352.html</a>	
Etiquetas de Alerta:	
Clave	Valor
OWASP_2021_A01	<a href="https://owasp.org/Top10/A01_2021-Broken_Access_Control/">https://owasp.org/Top10/A01_2021-Broken_Access_Control/</a>
WSTG-v42-SESS-05	<a href="https://owasp.org/www-project-web-security-testing-guide/v42/4-We...">https://owasp.org/www-project-web-security-testing-guide/v42/4-We...</a>
OWASP_2017_A05	<a href="https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Acce...">https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Acce...</a>

Esto es una falsa alerta porque hemos creado el token anti CSRF pero como no lo hemos llamado con uno de los nombres que zap identifica y piensa que no lo tenemos, cuando realmente si lo tenemos solo que lo hemos llamado token.

## 3.2. CSP



Esta es una alerta falsa puesto que podemos ver que está solicitando un recurso que no existe. En el resto de casos, podemos ver que existen las cabeceras adecuadas para paliar esta vulnerabilidad como en la siguiente imagen:

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Fri, 24 Nov 2023 17:51:16 GMT
3 Server: Apache/2.4.25 (Debian)
4 Strict-Transport-Security: max-age=31536000; includeSubDomains;
  preload
5 X-Content-Type-Options: nosniff
6 X-Frame-Options: DENY
7 Expires: Thu, 19 Nov 1981 08:52:00 GMT
8 Cache-Control: no-store, no-cache, must-revalidate
9 Pragma: no-cache
10 Set-Cookie: user=
  eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpYXQiOiE3MDA4NDgyNzYsImp0aSI6I6I1wvZkVjNHY5OEZCaW1CYjZpQ0VrUU1RPT0iLCJuYmYiOiE3MDA4NDgyNzYsImV4cCI6MTcwMDg0ODU3NiwiZGF0YSI6eyJ1c2VyTmFtZSI6Im9vIn19.rQ_zedt08ziMWR9jTM0PnI4Kp9Gx2un8JAEn5HA-znTscP9u_CLjhRp5yxaWuwRdZ0aHwBOPqRw649DcSKqQsw; expires=Fri, 24-Nov-2023 17:56:16 GMT; Max-Age=300; path=/
11 Vary: Accept-Encoding
12 Content-Security-Policy: default-src 'self'; script-src 'self'
  'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline';
13 Content-Length: 3378
14 Connection: close
15 Content-Type: text/html; charset=UTF-8
16
17
18 <link rel="stylesheet" href="styles.css">
19
20 <!-- Fuente de letra roboto de Google
  https://fonts.google.com/specimen/Roboto -->
21 <link href="
  https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="
  stylesheet">
22
```

### 3.3. Server leaks

No es un problema de seguridad muy importante ya que solo se muestra la versión que utilizamos de apache.

# Conclusión:

En conclusión vemos que la seguridad es un proceso laborioso pero de vital importancia en un proyecto de software. También hemos visto que dependiendo de la medida de seguridad que utilicemos podemos generar efectos adversos en cuanto a la experiencia del usuario como, por ejemplo, poner un captcha.

Después de realizar dos auditorías detalladas en la página web, pudimos identificar y abordar de manera efectiva vulnerabilidades críticas. A través de la implementación de varias medidas y ajustes en el código, conseguimos reforzar considerablemente su seguridad. Aunque hemos alcanzado un importante logro en la protección de la página web, es fundamental mantener una vigilancia constante, ya que las amenazas evolucionan y requieren una respuesta continua para preservar la integridad y la seguridad de la información.

# Bibliografía:

Prevenir XSS:

- <https://stackoverflow.com/questions/1996122/how-can-i-prevent-xss-with-html-php>

Token JWT:

- <https://www.sitepoint.com/php-authorization-jwt-json-web-tokens/>
- <https://github.com/sitepoint-editors/basic-php-jwt-auth-example/tree/main>

Ajustar cookies:

- <https://www.php.net/manual/en/function.session-set-cookie-params.php>
- <https://stackoverflow.com/questions/39750906/php-setcookie-samesite-strict>

ChatGPT:

- <https://chat.openai.com/>

Imágenes:

- <https://fonts.google.com/icons?hl=es-419>

Docker-compose:

- <https://stackoverflow.com/questions/48127851/how-to-use-composer-with-docker-compose>

docker ver hashes:

- <https://stackoverflow.com/questions/32046334/where-can-i-find-the-sha256-code-of-a-docker-image>

Información general:

- <https://cwe.mitre.org/data/definitions/1344.html>