

# SGSSI

## ENTREGA 2



Xabier Badiola  
Ander Martín

# VULNERABILIDADES A SOLUCIONAR

## Rotura de control de acceso:

Primer pentesting

Cabecera Content Security Policy (CSP) no configurada:

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 declarar fuentes de contenido aprobadas que los navegadores deberían poder cargar en esa página; los tipos cubiertos son JavaScript, CSS, marcos HTML, fuentes, imágenes y objetos incrustados como applets de Java, ActiveX, archivos de audio y video.

Cambios a realizar

En header.php:

```
<meta http-equiv="Content-Security-Policy" content="directives">  
<meta http-equiv="Content-Security-Policy" content="script-src 'self' https://apis.google.com">
```

## Fallos criptográficos:

Primer pentesting

Nuestra web ya 'hashea' con un hash robusto las contraseñas de los usuarios. Además, ya usa conexiones seguras con SSL/TLS.

Cambios a realizar

Ninguno.

# Inyección:

## Primer pentesting

Hemos encontrado vulnerabilidades de inyección SQL tanto en el registro como en el login. En lugar de preparar la consulta con parámetros, estábamos interpolando directamente el valor de \$email en la cadena de consulta, lo que podría permitir ataques de inyección de SQL. Para solucionar este problema, debemos utilizar consultas preparadas y parámetros vinculados.

## Cambios a realizar

### Login:

```
$statement = $conn->prepare("SELECT * FROM usuarios WHERE email =  
'$email' LIMIT 1 ");  
$statement->bindParam(':email', $email, PDO::PARAM_STR);  
$statement->execute();
```

En este código, :email es un marcador de posición que será vinculado al valor de \$email utilizando bindParam. Esto asegura que la consulta se ejecute de manera segura y evita la posibilidad de inyección de SQL.

### Register:

```
$statement = $conn->prepare("SELECT * FROM usuarios WHERE email =  
'$email'");  
$statement->bindParam(':email', $email, PDO::PARAM_STR);  
$statement->execute();  
$conn->prepare("INSERT INTO usuarios  
(name,apellidos,DNI,phone_number,fecha_de_nacimiento,email,password)  
VALUES ('$name','$apellidos', '$DNI', '$phoneNumber',  
'$fechaDeNacimiento', '$email', '$hashedPassword')");  
$conn->bindParam(':name', $name, PDO::PARAM_STR);  
$conn->bindParam(':apellidos', $apellidos, PDO::PARAM_STR);  
$conn->bindParam(':DNI', $DNI, PDO::PARAM_STR);  
$conn->bindParam(':phoneNumber', $phoneNumber, PDO::PARAM_STR);  
$conn->bindParam(':fechaDeNacimiento',$fechaDeNacimiento,PDO::PARAM_STR  
);  
$conn->bindParam(':email', $email, PDO::PARAM_STR);  
$conn->bindParam(':hashedPassword', $hashedPassword, PDO::PARAM_STR);  
$conn->execute();
```

## Diseño inseguro:

### Primer pentesting

Nuestra página web ya tiene un diseño seguro en el que las solicitudes se hacen por HTTPS.

### Cambios a realizar

Ninguno.

## Configuración de seguridad insuficiente:

### Primer pentesting

En esta primera prueba nos encontramos con numerosas vulnerabilidades en cuanto a la configuración de seguridad insuficiente se refiere, como por ejemplo; falta de cabecera Anti-Clickjacking, Hidden File Found (Archivo Oculto Encontrado), HTTP 'X-Powered-By', X-Content-Type-Options Header Missing...

Esto lo hemos arreglado añadiendo código a nuestro fichero 'sistemasweb.conf'.

### Cambios a realizar

Falta de cabecera Anti-Clickjacking:

Añadimos en el fichero sistemasweb.conf:

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="X-Frame-Options" value="SAMEORIGIN" />
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

Estas instrucciones te permitirán agregar la cabecera X-Frame-Options a tu sitio web en IIS. Esta configuración ayuda a mitigar el riesgo de ataques de clickjacking al indicar al navegador que limite la visualización de tu sitio web (<frame> o <iframe>) según la política que hayas establecido (en este caso, SAMEORIGIN).

### Hidden File Found (Archivo Oculto Encontrado):

```
<Directory /var/www/html/>
    #AllowOverride all
    Options -Indexes
    AllowOverride None
    Require all granted
</Directory>
```

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

```
<IfModule mod_headers.c>
    Header unset X-Powered-By
</IfModule>
Header unset X-Powered-By
```

### X-Content-Type-Options Header Missing:

```
<IfModule mod_headers.c>
    Header unset X-Powered-By
    Header set X-Content-Type-Options "nosniff"
</IfModule>
```

### Amplia gama de Cookies:

```
<IfModule mod_headers.c>
    Header unset X-Powered-By
    Header set X-Content-Type-Options "nosniff"
    Header always set Strict-Transport-Security
"max-age=31536000; includeSubDomains; preload"
</IfModule>
```

### Authentication Request Identified:

```
<Directory /var/www/html/>
    #AllowOverride all
    Options -Indexes
    AllowOverride None
    Require all granted
    AuthType Basic
    AuthName "Área Restringida"
    AuthUserFile /etc/apache2/apache2.conf/.htpasswd
    Require valid-user
</Directory>
```

## **Componentes vulnerables y obsoletos:**

### **Primer pentesting**

Al realizar la construcción del docker, tanto mariadb como phpmyadmin usaban una versión preestablecida por el usuario. Esto puede derivar en varios problemas, como que la versión usada se quede obsoleta y vulnerable a fallos de seguridad.

### **Cambios a realizar**

Para solucionar el problema, en el constructor se ha cambiado el número de la versión por 'latest'. Esto asegura que siempre se utilice la última versión.

## **Fallos de identificación y autenticación:**

### **Primer pentesting**

En el primer pentesting se han descubierto varios fallos críticos en el proceso de identificación y autenticación. Uno de ellos es que el usuario puede ver un archivo oculto, concretamente /.git/config.

Otro problema encontrado ha sido el no haber implementado la bandera 'HttpOnly' en las cookies, permitiendo de esta forma que un script pueda manipular las distintas cookies.

Por último, este sistema web no cancelaba la sesión del usuario al pasar un tiempo inactivo. Además, la contraseña del administrador es una contraseña muy común. Las contraseñas de los usuarios deben de tener un mínimo de 8-12 caracteres, lo cual es algo insuficiente. Tampoco hay un mínimo de números, letras mayúsculas/minúsculas o caracteres especiales a la hora de crear la contraseña.

### **Cambios a realizar**

Se ha añadido en el archivo de configuración de apache una regla de acceso para que deniega a los usuarios la entrada a /.git/config. También se ha añadido la flag HttpOnly a todas las cookies:

```
<Directory "./.git">  
    Order deny,allow  
    Deny from all  
</Directory>
```

Además la sesión del usuario se destruirá si pasa inactivo más de 10 minutos:

```

if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity'] > 600)) {
    // Si han pasado más de 10 minutos, destruye la sesión
    session_unset(); // Elimina todas las variables de sesión
    session_destroy(); // Destruye la sesión
}

```

Respecto a las contraseñas, se le ha añadido complejidad a la contraseña del administrador. Además, el rango de las contraseñas ha aumentado a un mínimo de 12-25 caracteres y se han añadido restricciones a las contraseñas (una letra mayúscula, una letra minúscula, un número y un carácter especial):

```

// checking password character pattern
if
(!password.value.match(/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[^\a-zA-Z0-9])(?!.*\s){12,25}$/) {
    alert("Password must contain at least one lowercase letter, one uppercase letter, one
numeric digit, and one special character, and must be between 12 and 25 characters
long.");
    password.focus();
    return false;
}
if (password.value.indexOf('@') !== -1 || password.value.indexOf('#') !== -1 ||
password.value.indexOf('"') !== -1 || password.value.indexOf("'") !== -1 ||
password.value.indexOf(';') !== -1 || password.value.indexOf('&') !== -1 ||
password.value.indexOf('\') !== -1) {
    alert("La contraseña no puede contener los caracteres @, #, \", ', \\", , &")
    password.focus();
    return false;
}

```

## Fallos en la integridad de datos y software:

### Primer pentesting

No se ha detectado el uso de ninguna librería no oficial.

### Cambios a realizar

Ninguno.

## Fallos en la monitorización de la seguridad:





### Primer pentesting

El servidor web ya monitoriza y almacena correctamente los diferentes errores en un servidor web.

### Cambios a realizar

Ninguno.

## SEGUNDO PENTESTING

- >  Ausencia de fichas (tokens) Anti-CSRF (4)
- >  Cabecera Content Security Policy (CSP) no configurada (7)
- >  Cookie sin el atributo SameSite
- >  Server Leaks Version Information via "Server" HTTP Response Header Field (14)

Es este segundo pentesting se puede observar que se han arreglado la mayoría de problemas, aunque todavía la aplicación ZAP nos marca que hay problemas. Estos problemas los hemos catalogado como falsos positivos.