Sistema Web: Los Montamarcos

Sistemas de Gestión de Seguridad de Sistemas de Información

Laura Calvo

Marco Lartategui

Iker Fuente

Tabata Morente

Iván Herrera

Gorka Bidaguren

28/10/2025

Ingeniería Informática de Gestión y Sistemas de Información

ÍNDICE

1. INTRODUCCIÓN

Después de haber desarrollado la página web “Los Montamarcos” centrada en apuestas de carreras de cerdos, se nos ha pedido hacer una auditoría centrada en la seguridad de dicha página web. Haciendo uso de ZAP, además de criterio personal, analizaremos el sistema web en busca de posibles vulnerabilidades observando las peticiones y sus respuestas; para ello se especificará el riesgo, ID 's, descripción y la posible solución encontrada de la debilidad descrita.

Como complemento a ZAP, hemos hecho uso de una extensión llamada “Access Control Testing” la cual se encarga de comprobar si una persona no identificada puede acceder a páginas de las cuales es necesaria la identificación; como por ejemplo si en nuestra página se accediese a realizar una apuesta sin que el usuario esté previamente identificado.

Hay que aclarar que el riesgo especificado en cada debilidad ha sido seleccionado manualmente por los miembros del equipo, formando así una jerarquía a la hora de elegir qué debilidades son más importantes de solucionar, y cuáles pueden dejarse en un segundo plano prescindiendo de su solución.

2. DEBILIDADES ENCONTRADAS

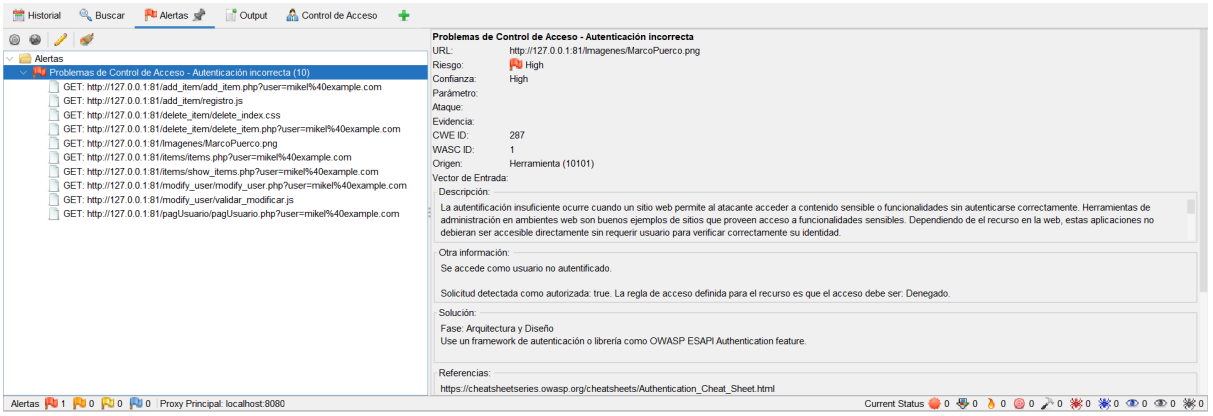
2.1 Broken Access Control (CWE: 284 y WASC: 2)

Alerta de un nivel de riesgo Alto/Crítico.

- **Descripción**: Este fallo de seguridad ocurre cuando la web no logra cumplir la autorización, y permite a atacantes ver/modificar datos que no deberían ser visibles para ellos. Es decir, si se conoce la URL de la página a la que se quiere acceder, cualquiera puede acceder a ella, en lugar de comprobar si realmente se ha iniciado sesión o si se tiene los permisos necesarios. Por ejemplo, en nuestra web de “Los Montamarcos” si se escribe la URL y el email del usuario, permite entrar a dicha página, y por motivos de seguridad no debería suceder.
- Este fallo no cumple con las propiedades de Confidencialidad (permite a otros usuarios acceder a información sensible), Integridad (un atacante puede modificar/eliminar información sin permiso) y Autenticidad (la web no valida correctamente la identidad del usuario).
- En consecuencia, algunos riesgos serían:
    - Acceso no autorizado a datos sensibles: Ver información de otros usuarios (apuestas, poder ver sus datos…)
    - Manipulación del flujo de la aplicación: Saltar pasos críticos como iniciar sesión
- Solución:
    - Verificar siempre la autorización y restringir los endpoints por el rol o por el permiso.
    - Usar control de sesión seguro como tokens o cookies protegidas. Cuando se añadan las cookies o los tokens, si no se hace con cuidado se nos podría crear otra vulnerabilidad de riesgo medio, Ausencia de Tokens Anti-CSRF:
        - Cuando un usuario mantiene una sesión activa, un atacante podría conseguir el token o la cookie de alguna manera y dado que el navegador envía automáticamente las cookies de sesión, el servidor interpretaría estas solicitudes como legítimas.
        - **Solución:** Implementación del token de seguridad:
            1. Generar token pseudoaleatorio
            2. Enviar token al cliente al iniciar sesión
            3. Incluirlo en cada solicitud de cambio de estado
            4. Verificar en el servidor el token antes de procesar la solicitud

		<< No autenticado >>		mikel	
Método	URL	¿Autorizado?	Control de Acceso	¿Autorizado?	Control de Acceso
GET	http://127.0.0.1:81/login/login.js	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/login/login.css	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/register/registro.js	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/login/index.html	Sí	Válido	Sí	Válido
POST	http://127.0.0.1:81/login/login.php	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/pagUsuario/pagUsuario.php?user=mikel%40example.com	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/items/items.php?user=mikel%40example.com	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/delete_item/delete_index.css	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/index.html	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/items/show_items.php?user=mikel%40example.com	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/modify_user/validar_modificar.js	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/add_item/add_item.php?user=mikel%40example.com	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/modify_user/modify_user.php?user=mikel%40example.com	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/register/index.html	Sí	Válido	Sí	Válido
GET	http://127.0.0.1:81/add_item/registro.js	Sí	Inválido	Sí	Válido
GET	http://127.0.0.1:81/delete_item/delete_item.php?user=mikel%40example.com	Sí	Inválido	Sí	Válido

Este informe fue generado con la extensión en ZAP, donde indica qué usuarios accedieron a la página web y quiénes debieron o no debieron entrar. Por cada control de acceso inválido se muestra una alerta de la siguiente forma:



2.2 SQL Injection - MySQL (CWE: 89 y WASC: 19)

Alerta con un nivel de riesgo muy alto porque se puede llegar a borrar o modificar la base de datos de maneras no intencionadas sin realizar ningún ataque sofisticado.

- **Descripción:** Explota las vulnerabilidades de una aplicación web al insertar comandos SQL maliciosos en los campos de entrada como nombres o contraseñas en formularios u otras peticiones.
- **Solución:** una posible solución sería verificar y limpiar todas las entradas de los usuarios antes de realizar las consultas SQL.

2.3 Cabecera Content Security Policy (CSP) no configurada (CWE: 693 WASC: 15)

Alerta con un nivel de riesgo medio

- **Descripción**: La cabecera CSP es un mecanismo de seguridad que indica al navegador qué contenido puede cargar o ejecutar. Es decir, la cabecera permite definir explícitamente qué fuentes de contenido son consideradas seguras.

  La página incluye formularios que permiten enviar datos, lo cual aumenta el riesgo de ataques como los XSS (Cross-Site Scripting). Los ataques XSS son una vulnerabilidad que permite a un atacante inyectar código malicioso dentro de las páginas web.

  Esto compromete la integridad y la seguridad de los datos introducidos en los formularios.

- **Solución:** Configurar la cabecera en el html. Si se configura la cabecera, cuando alguien inyecte código, no cargará porque no estará marcada como segura.

2.4 Falta de cabecera Anti-Clickjacking (CWE ID: 1021 WASC ID: 15)

Alerta de riesgo medio

- **Descripción**: ClickJacking es un ataque donde un atacante inserta la página legítima dentro de un iframe* en otra página maliciosa y engaña al usuario para que haga clic en botones o enlaces ocultos en la página maliciosa.

  *Un iframe es un elemento HTML que permite incrustar otra página web dentro de otra.

- **Solución:** Configurar la cabecera en el html para denegar que la página se pueda incrustar en un frame o configurarlo para que solo tu dominio pueda incrustar la página en un iframe.

2.5 El servidor filtra información de versión a través del campo "Server" del encabezado de respuesta HTTP

Alerta de riesgo bajo porque no ataca directamente a la web pero puede ayudar a encontrar una vulnerabilidad.

- **Descripción:** En esta cabecera que manda el servidor web, en este caso Apache, se suele indicar la tecnología o versión del servidor. Que se filtre esta información no permite un acceso directo a la página pero puede facilitar un ataque más preciso contra vulnerabilidades conocidas de la versión o tecnología.

- **Solución:** Eliminar o deshabilitar la cabecera en el php.

2.6 El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By"" (CWE ID: 200 WASC ID: 13)

Alerta de riesgo bajo porque no ataca directamente a la web pero puede ayudar a encontrar una vulnerabilidad.

- **Descripción:** En esta cabecera que manda el lenguaje que ejecuta mi web, en este caso PHP, se suele indicar la tecnología o versión del lenguaje. Que se filtre esta información no permite un acceso directo a la página pero puede facilitar un ataque más preciso contra vulnerabilidades conocidas de la versión o tecnología.

- **Solución:** Deshabilitar o eliminar el encabezado X-Powered-By.

2.7 Falta encabezado X-Content-Type-Options (CWE: 693 y WASC: 19)

- **Descripción**: La falta de esta cabecera puede que fuerce al navegador que interprete contenido como un tipo diferente, por ejemplo, ejecutar un script en lugar de tratarlo como texto.
  También puede facilitar ataques XSS o de content injection.

- **Solución**: Configurar la cabecera

2.8 Guardar contraseñas en texto plano

Alerta de riesgo medio, ya que aunque si el atacante consigue penetrar generaría un peligro masivo, tiene que lograr acceso a la base de datos.

- **Descripción:** Como bien es sabido ya, se deben almacenar las contraseñas para cada usuario en la base de datos. Sin embargo, en nuestra base de datos las contraseñas de los usuarios están guardadas en texto plano. De esta forma, si algún individuo no deseado de alguna forma logra acceso a la base de datos, tendrá disponible todas las contraseñas de todos los usuarios. Con lo cual, el atacante podrá iniciar sesión con la cuenta de cualquier usuario y hacer lo que le plazca con su información.

  Estas contraseñas no deberían de estar disponibles para ningún usuario ajeno a ellas, lo cual conlleva una violación a la privacidad y seguridad de la información de los usuarios.

- **Solución:** La solución más viable a esto es usar algoritmos hash para proteger las contraseñas, ya que estos algoritmos son unidireccionales y no se pueden deshacer.
  - Si los hash no se implementan bien, se pueden crear otras debilidades como el **Use of a One-Way Hash without a Salt (**CWE-759) que hace que más de un usuario pueda tener el mismo hash de la contraseña lo que ayuda a un atacante a identificar contraseñas y a poder planear un ataque.
  - **Solución:** hacer uso de sal, un código aleatorio en cada contraseña de cada usuario, para que no se puedan generar hashes iguales, evitando así colisiones.

2.9 Hardcoding de credenciales (CWE: 798)

Alerta de riesgo medio, debido a que puede romper toda la base de datos, pero requiere conocer el código.

- **Descripción**: El atacante tiene el código php que gestiona la base de datos visible. Dicho código para poder conectarse a la base de datos requiere un nombre de usuario y una contraseña, que están escritas de forma explícita en el código. Entonces, si obtiene el código, tendría acceso a toda la base de datos. Para ello, el phpmyadmin del sitio web tendría que estar disponible para cualquier usuario.

- **Solución:** Se propone utilizar librerías de php que ocultan información del usuario administrador de la base de datos. También es una posibilidad guardar las credenciales en un archivo externo, que el código abriría cada vez que necesita modificar o consultar información en la base de datos.

3. CONCLUSIÓN

Tras la auditoría de seguridad realizada sobre la página web “Los Montamarcos” mediante el uso de ZAP y la extensión Access Control Testing, se han podido identificar distintas vulnerabilidades relacionadas con el control de acceso a nuestra página web. Gracias a este proceso, hemos podido obtener una visión más clara del estado actual de la seguridad de nuestro sistema web, lo que permite priorizar las soluciones hacia las debilidades y, así, poder reforzar la página.

Aún habiendo encontrado diferentes debilidades en la página web, nos gustaría aclarar que la seguridad en una página web nunca va a ser absoluta, ya que se trata de un proceso continuo. Esto quiere decir que, a medida que la página evolucione, se seguirán encontrando nuevas vulnerabilidades, siendo imposible alcanzar un 100% de seguridad. Por ello, es fundamental mantener una supervisión constante, aplicar actualizaciones periódicas y seguir buenas prácticas de desarrollo para minimizar los riesgos y garantizar seguridad para los usuarios a la hora de navegar por nuestra página.