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

14/11/2025

Ingeniería Informática de Gestión y Sistemas de Información

ÍNDICE

1. INTRODUCCIÓN

En esta cuarta entrega, se nos ha pedido realizar un ataque al sistema web desarrollado por otro grupo, con el objetivo de poner a prueba su seguridad y comprobar si era posible explotar alguna vulnerabilidad. Para ello, se ha trabajado sobre la primera versión del proyecto del grupo con nombre *"Podcast"* (sobre la rama entrega_1), que todavía no contaba con las correcciones de seguridad aplicadas en entregas posteriores.

El trabajo ha consistido en analizar el código y el funcionamiento del sistema para encontrar un fallo aprovechable y ejecutar un ataque controlado. Paralelamente, se registraron con detalle los pasos seguidos y las evidencias obtenidas para garantizar la reproducibilidad y facilitar su análisis.

Durante el proceso, se han utilizado herramientas como OWASP ZAP y otras aplicaciones de las cuales se nombraran más adelante, con el fin de analizar la web, encontrar posibles fallos y comprobar si podían aprovecharse para llevar a cabo un ataque con éxito.

Con este trabajo se busca ponerse en la piel de un atacante para así comprender mejor cómo actúa y qué medidas deberían tomarse para evitar este tipo de situaciones en un entorno real. A continuación se explicará detalladamente la vulnerabilidad explotada, el procedimiento seguido para llevar a cabo el ataque y los resultados obtenidos durante su ejecución.

2. ATAQUES EJECUTADOS

2.1 Broken Access Control

Estado: Confirmado.

Al introducir en la barra de direcciones `http://localhost:81/show_user.php?user=tata` la aplicación devuelve la información del usuario con nombre `tata`.

`http://localhost:81/modify_user.php?user=tata` se accede a la página de modificación de datos del usuario con nombre `tata`.

`http://localhost:81/add_item.php` se accede a la página de añadir elementos.

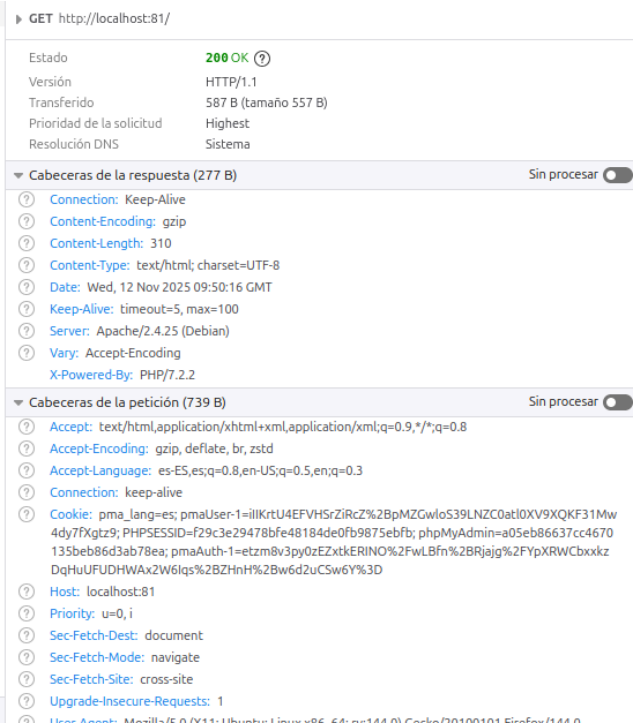
Si la página no realiza ninguna comprobación adicional en el servidor para verificar que la persona que hace la petición es efectivamente el titular de ese perfil (por ejemplo, comprobando la cookie de sesión, un JWT o cualquier otro mecanismo de autenticación y autorización), cualquiera que conozca o adivine ese identificador podrá ver y, en su caso, modificar los datos de ese usuario como si fuera el propietario legítimo incluso añadir elementos a la página sin las necesidad de estar registrado o identificado.

Además se puede acceder a los detalles de los ítems de la página al igual que modificarlos o eliminarlos de la siguiente manera probando con marcas y modelos de coche:

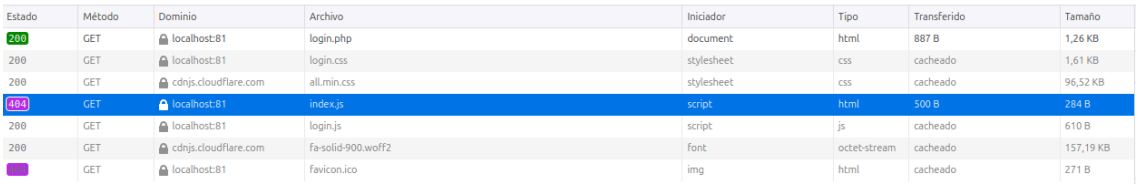
`http://localhost:81/show_item.php?item=Toyota+Corolla`
`http://localhost:81/modify_item.php?item=Toyota%20Corolla`
`http://localhost:81/delete_item.php?item=Toyota+Corolla`

2.2 Chrome DevTools

Accediendo al Chrome DevTools mediante F12 podemos analizar y sacar datos sensibles del sistemas, por ejemplo, podemos obtener datos como la versión de HTTP que se está utilizando, qué servidor web y su versión se está utilizando, la imagen de docker que se está empleando y cuáles son las peticiones que realiza la página web en cada caso.



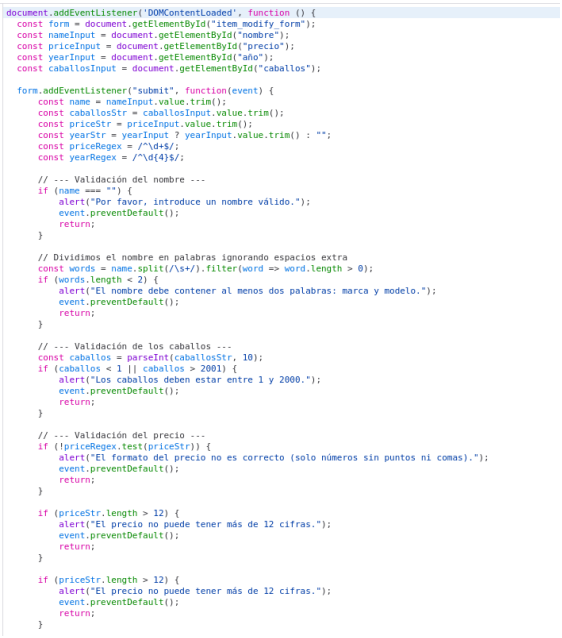
También nos podemos hacer una idea general de la estructura de los archivos que se están utilizando:



Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido	Tamaño
200	GET	localhost:81	login.php	document	html	887 B	1,26 KB
200	GET	localhost:81	login.css	stylesheet	css	cacheado	1,61 KB
200	GET	cdnjs.cloudflare.com	all.min.css	stylesheet	css	cacheado	96,52 KB
404	GET	localhost:81	index.js	script	html	500 B	284 B
200	GET	localhost:81	login.js	script	js	cacheado	610 B
200	GET	cdnjs.cloudflare.com	fa-solid-900.woff2	font	octet-stream	cacheado	157,19 KB
404	GET	localhost:81	favicon.ico	img	html	cacheado	271 B

Si observamos un archivo PHP, también se puede ver la versión PHP que están utilizando y otros datos de la cabecera:

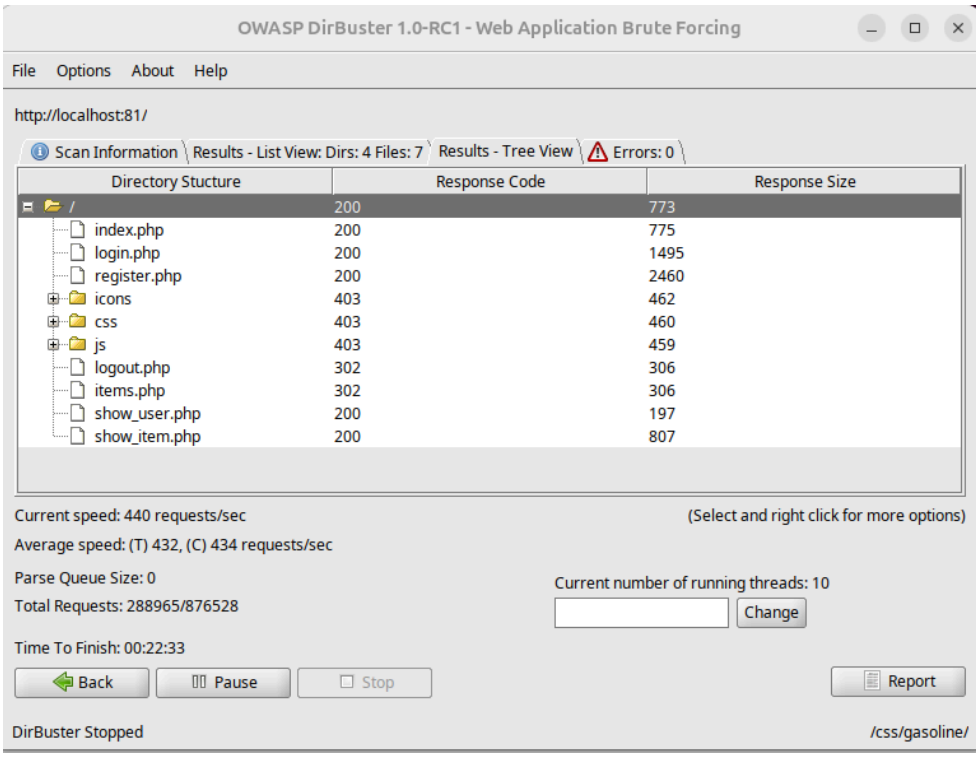
Mirando los archivos .js, en el apartado de respuesta, podemos encontrar gran parte del código javascript que se ha utilizado, el código JavaScript se encuentra sin ocultar ni ofuscar. De está manera, las reglas de validación son visibles, lo que podría ayudar a un atacante a adaptar entradas maliciosas. No es crítico, pero aumenta la superficie de información pública.



```
document.addEventListener('DOMContentLoaded', function () {
  const form = document.getElementById("item_modify_form");
  const nameInput = document.getElementById("nombre");
  const priceInput = document.getElementById("precio");
  const yearInput = document.getElementById("año");
  const caballosInput = document.getElementById("caballos");

  form.addEventListener("submit", function(event) {
      const name = nameInput.value.trim();
      const caballosStr = caballosInput.value.trim();
      const priceStr = priceInput.value.trim();
      const yearStr = yearInput ? yearInput.value.trim() : "";
      const priceRegex = /^\d+$/;
      const yearRegex = /^\d{4}$/;

      // --- Validación del nombre ---
      if (name === "") {
          alert("Por favor, introduce un nombre válido.");
          event.preventDefault();
          return;
      }

      // Dividimos el nombre en palabras ignorando espacios extra
      const words = name.split(/\s+/).filter(word => word.length > 0);
      if (words.length < 2) {
          alert("El nombre debe contener al menos dos palabras: marca y modelo.");
          event.preventDefault();
          return;
      }

      // --- Validación de los caballos ---
      const caballos = parseInt(caballosStr, 10);
      if (caballos < 1 || caballos > 2001) {
          alert("Los caballos deben estar entre 1 y 2000.");
          event.preventDefault();
          return;
      }

      // --- Validación del precio ---
      if (!priceRegex.test(priceStr)) {
          alert("El formato del precio no es correcto (solo números sin puntos ni comas).");
          event.preventDefault();
          return;
      }

      if (priceStr.length > 12) {
          alert("El precio no puede tener más de 12 cifras.");
          event.preventDefault();
          return;
      }

      if (priceStr.length > 12) {
          alert("El precio no puede tener más de 12 cifras.");
          event.preventDefault();
          return;
      }
```

2.3 Ataque con DirBuster

Además de las fallas de seguridad detalladas anteriormente, también es posible obtener los nombres y la jerarquía de los archivos del sistema. Haciendo uso del software DirBuster, es posible obtener todo lo mencionado tan solo con un análisis al url de localhost:81. Al poner en marcha el programa, DirBuster realiza un ataque por fuerza bruta para obtener los archivos y directorios de la web.



Como se puede observar en la imagen, todos los archivos del sistema web han quedado al descubierto al hacer un análisis, lo cual deja en evidencia la seguridad del código en sí.

La instalación de este programa puede variar según la distribución de Linux que el usuario esté usando. Normalmente, DirBuster suele usarse en distribuciones de Kali Linux, que se especializan en ámbitos como ciberseguridad. Si el usuario utiliza esta distribución, tan solo hace falta escribir en la terminal “sudo apt install dirbuster” para poder utilizar el programa.

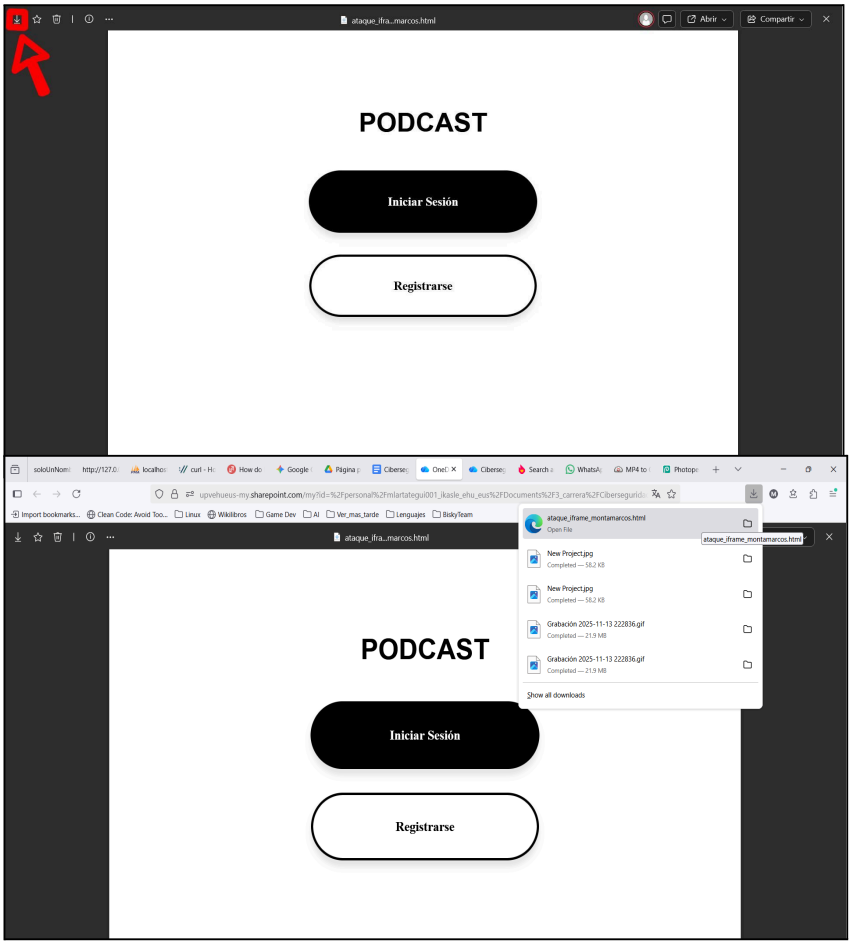
Sin embargo, en otras distribuciones como Ubuntu, el programa no está disponible en el gestor de paquetes apt, por lo que su instalación se complica un poco más. El usuario deberá de clonar el repositorio del siguiente enlace: https://gitlab.com/kalilinux/packages/dirbuster
Una vez dentro, el usuario copia el enlace para clonar el repositorio y en la terminal escribe: git clone https://gitlab.com/kalilinux/packages/dirbuster.git
Al clonar el repositorio, el usuario dispondrá de un directorio llamado “dirbuster” en su carpeta personal o donde haya decidido clonar el repositorio. Dentro del directorio, el usuario tendrá disponible para usar un archivo sh llamado DirBuster-1.0-RC1.sh, el cual abrirá el programa dirbuster. De esta forma, el usuario tendrá listo el programa para su uso cuando quiera.

Cabe mencionar que, para poder usar DirBuster, es necesario tener instalado el paquete openjdk, el cual puede instalarlo con “sudo apt install default-jdk”. Para comprobar que el usuario ya dispone de openjdk, puede revisarlo con “java -version”.

2.4 Ataque clickjacking mediante iframe

Tal y como se ha detallado en las entregas anteriores, hace falta una cabecera anti clickjacking para poder evitar este tipo de ataques. Sin embargo, la página web no cuenta con esta medida de seguridad, por lo tanto nos es posible introducir la página web en un iframe, engañando al usuario mediante una página web de la que se proporcionará la url a continuación: [web_de_imitación].

Para poder probarlo primero se debe descargar el archivo html y después abrirlo desde un navegador, tal y como aparece en la foto.



Como se ha visto al entrar al link, se muestra lo que parece ser la página principal de la página web sin ninguna alteración. Sin embargo, al pulsar el botón de iniciar sesión se envía al usuario a un video de youtube. Además, para que pueda ser más eficaz, puede dirigir el usuario a una página web que sea igual a la de inicio de sesión de la web que se está atacando, pero que sea un servidor del atacante. Así, la gente que inicie sesión pensando que es la página web original registrará sus credenciales en la base de datos del atacante. Esto es un ejemplo de clickjacking, ya que se le engaña al usuario haciéndole pensar que está dentro de la página web real, para luego al pulsar un botón o un enlace enviarle a otro sitio no deseado.

2.5 Inyección SQL

Aunque las inyecciones SQL sean una debilidad muy típica en este tipo de páginas web, no hemos conseguido encontrar ninguna inyección exitosa. El programa ZAP, a veces, marca algunas posibles inyecciones SQL, sin embargo, cuando las probamos, se puede observar que las consultas están preparadas o no son accesibles desde la web.

Hemos utilizado un programa llamado Sqlmap, instalado mediante apt install, para probar todas las posibilidades de inyecciones SQL que se nos han ocurrido pero no ha podido ejecutar ninguna con éxito.

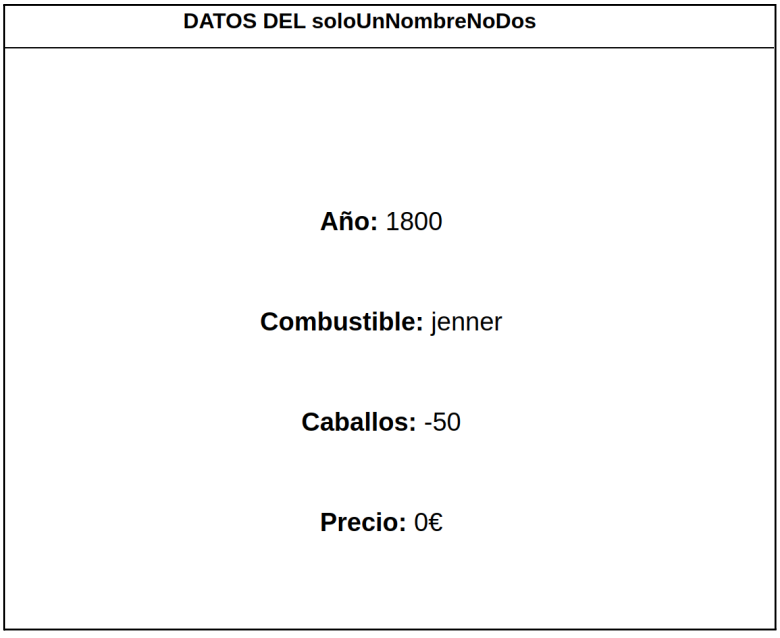
2.6 Input Validation Vulnerability:

La web ofrece la posibilidad de añadir coches; en esta página hay que indicar el nombre del coche, con la condición de que este esté separado con un espacio por Marca y Modelo, el año del vehículo, con la condición de que sea un número mayor o igual a 1886, el número del combustible que tiene el coche, cuantos caballos tiene el coche, con la condición de que tiene que estar entre 1-2000 y, por último, el precio, siendo este como máximo de 12 cifras.

Teniendo estas condiciones, si quieres poner un dato erróneo en cada apartado la página avisa con un mensaje diciendo que no puedes. Aún así, haciendo uso de “curl” en la terminal de linux se ha conseguido añadir coches rompiendo con las reglas de la web:

```
curl http://127.0.0.1:81/add_item.php \
-F item_name=soloUnNombreNoDos \
-F item_year=1800 \
-F item_combustible=jenner \
-F item_caballos=-50 \
-F precio=0
```

Consiguiendo así que el coche tenga las siguientes características:



Esto demuestra que el servidor acepta cualquier valor que se le mande aunque no cumpla las condiciones impuestas por el formulario de la web, pudiendo así saltarse la validación fácilmente y meter datos que no deberían pasar.

Otra cosa importante es que esto afecta a la integridad de los datos, porque en la base pueden acabar guardados valores imposibles o basura (como coches del año 1800 o con -50 caballos). Eso puede romper el funcionamiento de la web o de otros apartados que usen esa información.

En resumen, el problema está en que se confía solo en la validación del lado del cliente, y eso es algo que cualquiera puede saltarse con herramientas como curl. La solución es hacer la validación también en el PHP y asegurarse de que los datos cumplen las reglas antes de insertarlos.

2.7 Cross Site Scripting (XSS):

El Cross Site Scripting consiste en la inyección de código HTML o JavaScript escribiendo dicho script en los campos de los formularios. Así, cuando se muestran los datos registrados se ejecutará el código que se haya guardado mediante formularios. Es un ataque eficaz, ya que permite robar tokens críticos (como los de sesión o los tokens Anti-CSRF). Esto le da al atacante acceso a la sesión del usuario para suplantar su identidad. Para ello, el script inyectado debe leer las cookies registradas y después mediante la función `fetch()` o el objeto `XMLHttpRequest`, se enviaría el token a un servidor que controla el atacante. Como consecuencia, el atacante tiene acceso a la sesión del usuario, pudiendo manipular la cuenta para que el usuario original la pierda.

Sin embargo, al inyectar el código de prueba XSS (<script>alert(1)</script>) y revisar la página resultante, se comprueba que el ataque no funciona. Al inspeccionar el código fuente de la página (la salida HTML que llega al navegador), se puede ver que el servidor no ha guardado el script, sino que ha cambiado el formato de la información al momento de mostrarla (por ejemplo, reemplazando < por <). Esto demuestra que el servidor está aplicando codificación de salida (Output Encoding), neutralizando la vulnerabilidad XSS.

```
<title>soloUnNombreNoDos</title>
<link rel="stylesheet" href="css/show_item.css">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.6.0/css/all.
min.css">
<div class="bar">
  <div class="volver_button">
    <a href="items.php" title="Volver al inicio">
```

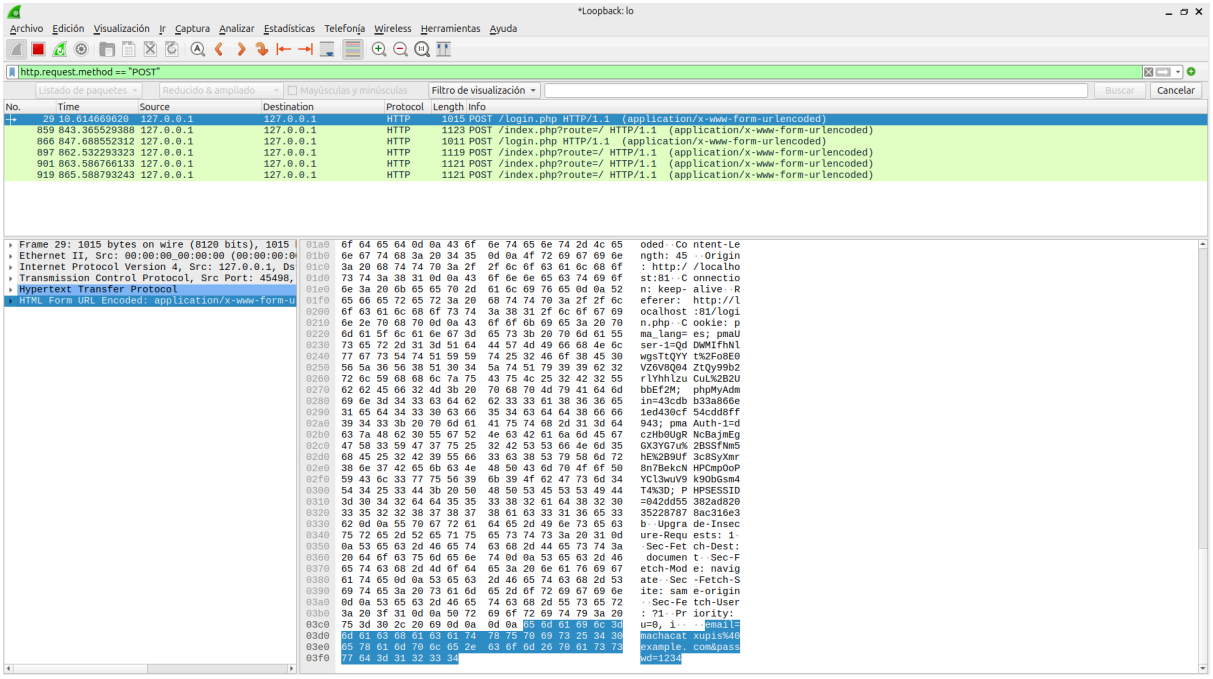
```
      <i class="fa-solid fa-house"></i>
    </a>
  </div>
  <h1>DATOS DEL soloUnNombreNoDos</h1>
  <div class="modificar_button">
      <a
href="modify_item.php?item=soloUnNombreNoDos"><button>Modificar</button>
</a>
  </div>
</div>
<div class="container">
    <div class="content">
                    <p><strong>Año:</strong> 5</p>
            <p><strong>Combustible:</strong>
&lt;script&gt;alert()&lt;/script&gt;</p>
            <p><strong>Caballos:</strong> -50</p>
            <p><strong>Precio:</strong> 0€</p>
            </div>
</div>
```

2.8 Sniffing (Packet Sniffing):

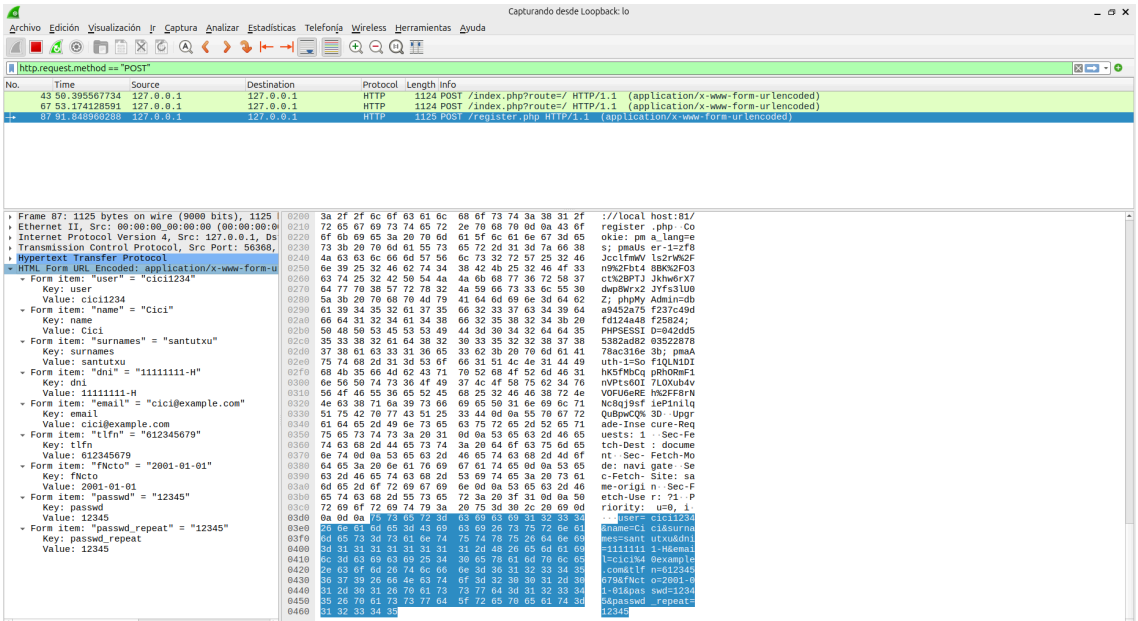
El Sniffing consiste en capturar y analizar el tráfico que circula por una red, con el fin de observar los datos que envían y reciben los dispositivos conectados. Esta técnica puede permitir obtener información sensible (contraseñas, mensajes…) en caso de que no esté cifrada.

En este caso, se ha utilizado Wireshark para poder capturar paquetes. Las contraseñas, al estar en texto plano, se pueden encontrar al capturar el tráfico de la red y robar dichas contraseñas. Para comprobarlo, se ha abierto Wireshark y se ha filtrado la captura por la red local (Loopback:lo) y filtrar el tráfico con http:request.method == “POST”, y se ha abierto la página web en el navegador.

Luego, se ha iniciado sesión con un correo electrónico (machacatxupis@example.com) y una contraseña (1234). En consecuencia, en Wireshark aparece dichos datos (los marcados en azul):



Asimismo, al registrarse, también se puede obtener toda la información que inserta el usuario (Usuario, Nombre, Apellidos, DNI, Correo, Teléfono, Fecha de Nacimiento y Contraseña):



- Cómo instalar Wireshark:
  $ sudo apt install wireshark

Para poder capturar tráfico del local host hace falta agregar nuestro usuario al grupo wireshark:

$ sudo usermod -a -G wireshark $USER

2.9 Ataque de fuerza bruta:

Para acceder a la aplicación hace falta un inicio de sesión mediante correo y contraseña. El problema es que este formulario no cuenta con mecanismos de seguridad básicos, como limitación de intentos, bloqueos temporales o filtros para que las contraseñas sean suficientemente seguras, entre otros. Debido a esto, cualquier atacante podría probar combinaciones de usuario y contraseña sin restricciones.

Para comprobar esta debilidad, hemos utilizado diferentes fuentes para obtener los usuarios y contraseñas más comunes:

Los correos electrónicos se han obtenido descargando un archivo con los usuarios más típicos y creando otro archivo con los dominios que nos han parecido más importantes (example.com, test.com, mail.com, gmail.com, hotmail.com e ikasle.ehu.eus). Luego seha creado otro archivo combinando todos los usuarios y contraseñas (`while read user; do while read domain; do echo "$user@$domain" >> emails.txt; done < domains.txt; done < users.txt`).

Para las contraseñas se han utilizado las 47 primeras del diccionario rockyou.txt, conocido por contener claves muy comunes y débiles :

```
curl -L https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt -o ~/Escritorio/rockyou.txt
```

Usando la herramienta Hydra, desde la terminal de Linux, se lanzaron intentos automatizados contra el formulario de login. Hydra prueba cada combinación posible de correo y contraseña, simulando un ataque de fuerza bruta:

```
hydra -L ~/Escritorio/emails.txt -P ~/Escritorio/rockyou-47.txt \
      localhost http-post-form \
                "/login.php:email=^USER^&passwd=^PASS^:Correo no registrado." \
      -s 81 -t 4 -f -V
```

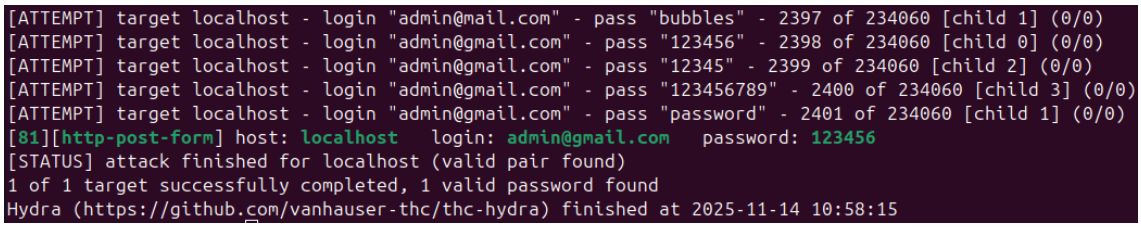
-L se utiliza para poner una lista de usuarios, -P para poner una lista de contraseñas. Localhost es la dirección IP de la web y con -s se indica el puerto en el que se ubica. `http-post-form` indica el tipo de ataque se va a realizar, en este caso un formulario con HTTP POST.

`"/login.php:email=^USER^&passwd=^PASS^:Correo no registrado."` Esta es la parte más importante, la primera parte (`login.php`) indica la ruta del formulario. La segunda parte (`email=^USER^&passwd=^PASS^`) indica los nombres de los campos (email y USER, obtenidos haciendo un curl del formulario) y le dice a Hydra que son. La

última parte indica cual es el mensaje que devuelve el formulario cuando un inicio de sesión no es exitoso (también obtenido haciendo un curl del formulario).

-t 4 es el número de hilos paralelos (conexiones simultáneas) que se utilizan en el ataque, 4 es un estándar para que no sea muy lento pero tampoco sature el Docker. -f es para que pare cuando encuentre un usuario y contraseña correctos y -V es para que vaya mostrando los intentos en tiempo real.

Como se puede ver en la foto, el ataque consiguió encontrar un usuario y contraseña, mostrando que el servidor acepta repetidos intentos sin activar ningún mecanismo de defensa:



Esto demuestra que la aplicación permite iniciar sesión utilizando contraseñas extremadamente débiles, como “123456”, y que no existe protección efectiva frente a ataques automatizados. Cualquier atacante podría explotar esto para acceder a cuentas ajenas simplemente probando contraseñas comunes.

Además, este fallo afecta directamente a la confidencialidad del sistema, ya que un atacante podría obtener acceso no autorizado a información interna si alguna cuenta usa credenciales poco seguras.

Lógicamente se ha tratado de un ataque completamente preparado en el que hemos creado un usuario que sabíamos que era vulnerable para este ataque, es decir sabíamos que el usuario y la contraseña estaban en las listas, además las hemos puesto al principio para que no tardará demasiado tiempo. En la realidad se usarían muchas más contraseñas y usuarios de las que hemos utilizado y el ataque posiblemente tardaría mucho más tiempo en ejecutarse.

Está simulación nos ha servido para demostrar que un ataque así sería posible y que si existiera alguna combinación de usuario y contraseña incluida en nuestras listas podríamos conseguirlas y entrar a la página de usuario de alguien sin permiso.

3. CONCLUSIÓN

Después de realizar el ataque sobre el sistema web, se ha comprobado que incluso un sitio aparentemente normal puede tener fallos que permitan acceder o modificar información sin autorización. El resultado demuestra la importancia de revisar a fondo la seguridad de una aplicación antes de ponerla en producción.

Este trabajo nos ha permitido ver la seguridad desde la perspectiva del atacante: qué busca y qué pasos sigue para aprovechar un fallo en el código o en la configuración. Al probar y reproducir el ataque hemos podido medir su impacto real y saber qué correcciones son más urgentes. Esto facilita priorizar cambios prácticos (validar entradas, proteger accesos, ajustar configuración) y entender por qué conviene aplicarlos desde el diseño.

Como conclusión, esta entrega, y el trabajo en general, nos ha servido para aplicar de forma práctica los conocimientos sobre seguridad web y comprobar la importancia de revisar cada detalle del código. También nos ha ayudado a entender que pequeños descuidos pueden tener consecuencias graves y que la prevención es la mejor defensa. Gracias a este trabajo, queda más claro cómo abordar futuros desarrollos con una mentalidad más segura y consciente frente a posibles ataques.

5. BIBLIOGRAFÍA

- **Arun Testing.** (2023, 21 mayo). Authenticated Scan Using OWASP ZAP Form based authentication [Video]. Youtube. https://www.youtube.com/watch?v=e-ry_Poct2o=191s&t
- **Stack Overflow.** (2020, 13 febrero). Docker compose gives invalid environment type error. https://stackoverflow.com/questions/60200966/docker-compose-gives-invalid-environment-type-error
- **OWASP Foundation.** (s.f.). OWASP Zed Attack Proxy (ZAP) Project. https://www.zaproxy.org/
- **OpenAI.** (2025). ChatGPT (GPT-5) [Modelo de lenguaje de IA]. https://chat.openai.com/
- **Qwen.** (2025). Qwen [Modelo de lenguaje de IA]. https://qwen.ai/
- **The curl project**. (s.f.). curl man page. curl.se. https://curl.se/docs/manpage.html
- **ReqBin**. (2023, 18 septiembre). How do I post form data using Curl? https://reqbin.com/req/c-sma2qrvp/curl-post-form-example
- **Super User**. (2012, 16 noviembre). Linux Bash script: single command but multiple lines. https://superuser.com/questions/508507/linux-bash-script-single-command-but-multiple-lines
- **Grupo Podcast**. (2025). Código fuente de la Web "Podcast" (Rama entrega_1). [Repositorio GitHub]. https://github.com/ikerfernandezmolano/PodcastSGSSI/tree/entrega_1
- **Google**. (2025). Gemini [Modelo de lenguaje de IA]. https://gemini.google.com/
- **Omar Palomino**. (2025, 4 octubre). Clickjacking explicado desde cero | Cómo funciona este ataque y cómo protegerse [Video]. Youtube. http://www.youtube.com/watch?v=G0__NiPznM8
- **bl3ssedc0de.** (2025, 11 mayo). Como instalar dirbuster en Ubuntu. Youtube. [Video] https://www.youtube.com/watch?v=tAWx8UApffQ
- **Cyber Pioneers.** (2022, 17 marzo). What is DirBuster and How to Use it?. Youtube. [Video] https://www.youtube.com/watch?v=TjyM_7JuTHY