Auditoría hecha por Diego Esteban

Problemas encontrados

Problemas encontrados por ZAP

- Cross site scripting (XSS)
- SQL Injection
- Application Error Disclosure
- Content Security Policy (CSP) Header Not Set
- Missing Anti-clickjacking Header
- X-Frame-Options Header Not Set
- Absence of Anti-CSRF Tokens
- Server Leaks Information
- X-Content-Type-Options Header Missing
- Information Disclosure

Cross site scripting (XSS)

Esta vulnerabilidad consiste en que un usuario malicioso puede conseguir ejecutar código aribitrariamente desde nuestra página web, como puede ser insertando el elemento `<script></script>` y conseguiendo que el navegador de otro usuario ejecute ese código.

Archivos afectados:

- /PHP/leer_prologo.php (parámetro titulo)
- /PHP/registro_usuario.php (en todos los parámetros menos contraseña y contraseña2. apellido, dni, email, nombre, pswd, tlf, usuario)
- /PHP/leer_libro.php (parámetros titulo y capitulo)
- /PHP/libro.php (parámetro título)
- /PHP/registro_usuario.php (parámetro fnacimiento)

Esto se debe a que no tratamos los datos que los usuarios nos otorgan (por ejemplo al publicar un libro). Simplemente los guardamos en la base de datos y los enviamos tal y como son.

Además, para navegar entre páginas, utilizamos links que no se tratan, y parte de esos links los insertamos la página. Por este motivo, alguien puede crear un link malicioso que le redireccione a nuestra página web, pero ejecutando el código que aparece en el link.

SQL Injection

La vulnerabilidad es parecida a la anterior, en este caso el usuario malicioso puede manipular la forma en la que accedemos a la base de datos, para que ejecute el comando que desee, o cambiar los parámetros del comando que debería ejecutarse.

Archivos afectados:

- /PHP/leer_libro.php (parámetros titulo y capitulo)

- /PHP/leer_prologo.php (parámetro titulo)
- /PHP/registro_usuario (todos los parámetros)

La causa, al igual que en el caso anterior, es no tratar los datos que nos envía el cliente. Por ejemplo, no parametrizamos los comandos, (símplemente los interpolamos en un string).

Content Security Policy (CSP) Header Not Set

CSP es una manera de decirle al navegador que políticas de seguridad debe tomar. Como por ejemplo, desde qué sitios web se permite la inyección. De esta manera se pueden evitar multitud de problemas, por ejemplo el XSS mencionado anteriormente

Archivos afectados:

- prácticamente todos

Missing Anti-clickjacking Header

En nuestro sistema, un atacante puede crear una página web fantasma sobre la nuestra, la cual puede engañar al usuario para hacer click en un sitio, cuando en realidad está haciendo click en otro. Esto se podría arreglar añadiendo una cabecera para alertar al navegador de que no permita hacer este tipo de cosas.

Archivos afectados:

- prácticamente todos

Application Error Disclosure

Nuestra página web a veces falla. Cuando estos fallos ocurren, le enviamos la información muy específica del fallo al usuario, que no necesita tener tanta información. Un atacante puede hacer fallar la página web a propósito para obtener información sobre qué sistemas estamos utilizando, lo cual le facilita para saber qué vulnerabilidades puede tener.

Archivos afectados:

- /PHP/leer_libro.php (el capítulo 2 del libro "Cinnamon Bun", este capítulo contiene caracteres especiales sin escapar)

Cuando ocurre un error deberíamos enviar un código de error genérico.

Absence of Anti-CSRF Tokens

Una página maliciosa podría hacer referencia a uno de nuestros forms y hacer que el usuario haga click en él, mientras que la página maliciosa puede controlar los campos del form. Por ejemplo, podría redirigir a modificar_datos.php cambiando la contraseña a "1234", pero como el usuario real ha hecho click, se validaría con la cookie del usuario.

Esto se solucionaría usando tokens "anti-csrf".

Archivos afectado:

- Prácticamente todos.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field

Nuestras respuestas HTTP contienen metadatos que informan al cliente de la tecnología que estamos utilizando. Esta información no es necesaria para el cliente, pero un atacante puede utilizarla para saber que vulnerabilidades puede tener nuestro sistema.

Archivos afectado:

- Prácticamente todos.

X-Content-Type-Options Header Missing

Por defecto, los navegadores intentan deducir el tipo de archivo (html, imágen, text, vídeo, etc.) que se recibe. Esto puede ser un vector de ataque puesto que un atacante puede hacer que el navegador piense que hemos enviado un tipo de archivo en concreto, dando pie a multitud de vulneravilidades. Esto se puede evitar con una cabecera HTTP.

Archivos afectado:

- Prácticamente todos.

Information Disclosure - Suspicious Comments

En un fichero javascript, le damos el valor a la cookie "username". Esto puede ser un problema porque un atacante puede saber para qué es la cookie.

Por otro lado, el uso de la cookie es trivial de entender, puesto que se ve a simple vista que en la cookie se guarda el nombre de usuario. Por lo tanto no es un gran problema.

Archivos afectados:

- /js/inicio_sesion.js

Problemas encontrados manualmente

Rotura de control de acceso

- Para mirar si la contraseña introducida es la correcta, le enviamos al cliente la contraseña correcta, y el cliente hace la comparación.
- Para saber si alguien tiene la sesión iniciada solo miramos la cookie de "username", no verificamos que tenga una contraseña.
- No logueamos los intentos de inicio de sesión.
- No generamos tokens de sesión
- A la hora de modificar los datos, se modifican los del usuario que se especifica en un campo (oculto) en el form.

Fallos criptográficos

- Usamos una conexión no cifrada (HTTP)
- No utilizamos encryption at rest.
- Almacenamos las contraseñas en plaintext.

Inyección

- No hacemos ninguna verificación sobre el archivo que se supone que es la portada del libro.
    - Ni siquiera miramos si es una imagen.
    - No hay límite de lo grande que puede ser la imagen.
    - Si ya existe una imágen con ese nombre, se sobreescribe.
- No parametrizamos los comandos SQL.
- No validamos los datos desde el servidor, sólo desde el cliente.
- No escapamos caracteres especiales. Al publicar un libro cualquiera puede meter elementos HTML, incluyendo <script></script>.
- Esto incluye los nombres de usuario. Al publicar un comentario, inserta su nombre de usuario en él, lo cual puede ser cualquier string.

Diseño inseguro

- No hay límites de accesos por segundo/minuto.
    - Podrían registrar cientos de libros con imágenes enormes, llenando así el disco duro del servidor.
    - Podrían hacerse ataques de fuerza bruta para conseguir la contraseña de un usuario.
    - Podrían registrarse miles de cuentas falsas o "bot"s
- Damos al cliente más información de la que necesita (por ejemplo le enviamos la contraseña del usuario que se intenta registrar para ver si el usuario ya existe)
- Podría ser un problema que solo miremos que no haya 2 usernames repetidos (tlfs, emails y dnis pueden repetirse).
- No se existe una configuración mínima de contraseña (más allá de el mínimo de 3 caracteres)

Configuración de seguridad insuficiente

- Usamos las credenciales por defecto
    - La contraseña del usuario admin para acceder a la base de datos es *test*
    - La contraseña del usuario admin, dentro de la página en sí, también es *test*

Componentes vulnerables y obsoletos

- Usamos la versión "latest" de phpmyadmin, en vez de una versión concreta