Auditoría hecha por Diego Esteban y Francisco gonzalez Diego Esteban ha explicado las partes arregladas por él y por ambos, y las no arregladas. Francisco Gonzalez ha explicado las partes arregladas por él.

Problemas arreglados desde la auditoría 1

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)

Arreglado por Diego Esteban

La solución ha sido escapar los caracteres especiales de cualquier cosa que venga desde una petición HTTP. Ya sea los atributos de una petición GET, POST o una cookie. De esta manera, cualquier cosa sobra la que no tenemos control directo se trata como texto plano, no se puede tratar como código. esto lo hace la función htmlspecialchars() de PHP.

SQL Injection

Arreglado por Diego Esteban y por Francisco González

Las inyecciones SQL se han arreglado utilizando queries paramterizadas. De esta manera, se separa la estructura de las consultas SQL de sus parámetros. Así se consigue que no se pueda modificar la estructura de la consulta mediante parámteros maliciosos que controla el usuario.

Content Security Policy (CSP) Header Not Set

Arreglado por Diego Esteban

Hemos añadido el header en default-ssl.conf, por lo que las conexiones https están protegidas de este ataque. Que debería ser la única forma de conectarse como explicamos aqui.

Estas son las políticas que hemos puesto:

- frame-ancestors 'none'
- script-src 'self' https://www.google.com/recaptcha/ https://www.gstatic.com/recaptcha/ 'unsafe-inline'
- style-src 'self' https://fonts.googleapis.com
- img-src 'self'

- connect-src 'self'
- frame-src 'self' https://www.google.com/recaptcha/ https://recaptcha.google.com/recaptcha/
- font-src 'self' https://fonts.gstatic.com
- media-src 'self'
- object-src 'self'
- manifest-src 'self'
- worker-src 'self'
- prefetch-src 'self'
- form-action 'self'

De esta manera nos aseguramos que ese tipo de datos solo vienen de nuestro servidor, el navegador descartaría todo aquello que provenga desde fuera.

Además también permitimos a google varias cosas para poder usar el recaptcha y las fuentes.

por último, permitimos "unsafe-inline" de los scripts porque es algo que utilizamos, hemos mitigado los posibles ataques en la sección de XSS, pero estaría bien cambiar la forma en el que hacemos las cosas para eliminar el "unsafe-inline" y que sea más seguro.

Missing Anti-clickjacking Header

Arreglado por Diego Esteban

Hemos añadido el header en default-ssl.conf, por lo que las conexiones https están protegidas de este ataque. Que debería ser la única forma de conectarse como explicamos aqui.

Application Error Disclosure

Arreglado por Diego Esteban

Hemos eliminado todas las llamadas a mysqli_error() y las hemos reemplazado por un error genérico "Error interno E890".

Aunque no nos podemos asegurar de que todos los errores se hayan suprimido, se ha reducido drásticamente la cantidad de información que otorgamos mediante errores.

X-Frame-Options Header Not Set

Arreglado por Diego Esteban

Hemos añadido el header en default-ssl.conf, por lo que las conexiones https están protegidas de este ataque. Que debería ser la única forma de conectarse como explicamos aqui.

Absence of Anti-CSRF Tokens

Arreglado por Diego Esteban

Hemos implantado un sistema de tokens, en el que se guarda un token de sesión en las cookies del usuario. Gracias a esto, cuando le enviamos un form a un usuario, generamos aleatoriamente otro token vinculado a su token de sesión. Ambos tokens son válidos solo durante un periodo de tiempo, los de sesión 12 horas y los anti-csrf 1 hora. Ambos tienen una longitud de 32 bytes.

Cuando recibimos una respuesta de un form, verificamos que su token anti-csrf está vinculado a la cookie de sesión del usuario.

Con este sistema, no puede una tercera persona usar un form que se le ha servido para que lo rellene otro. Solo el usuario que ha pedido el form puede rellenarlo.

Al usar tokens de sesión no es necesario que un usuario haya iniciado sesión, si no tiene un token de sesión, se le otorga uno cuando lo necesite. Las cookies de sesión tienen los atributos "httponly" y "samesite" para prevenir que otras páginas web accedan a ellas.

Cada 30 minutos se eliminan de la base de datos los tokens inválidos, aun así, se verifica cada vez que el token no haya expirado.

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

Arreglado por Diego Esteban

Esto tenía fácil solución. Basta con añadir `expose_php = Off` a php.ini

X-Content-Type-Options Header Missing

Arreglado por Diego Esteban

Hemos añadido el header en default-ssl.conf, por lo que las conexiones https están protegidas de este ataque. Que debería ser la única forma de conectarse como explicamos aqui.

Information Disclosure - Suspicious Comments

Como mencionamos en la auditoría 1, este problema no es relevante, se trata de un falso positivo.

Problemas encontrados manualmente

- Rotura del control de acceso
- Fallos criptográficos
- Inyección
- Diseño inseguro
- Configuración de seguridad insuficiente
- Componentes vulnerables y obsoletos

Rotura de control de acceso

No logueamos los intentos de inicio de sesión.

Arreglado por Francisco González. Tabla de la base de datos hecha por Ibai Mendivil. Cuando un usuario intenta iniciar sesión, el servidor guarda ese intento junto con un mensaje de si lo ha conseguido o no y la fecha/hora del intento.

No generamos tokens de sesión

Arreglado por Diego Esteban

Como hemos mencionado anteriormente, ahora usamos tokenes de sesión, por lo que todo el sistema de login se hace desde el lado del servidor. El usuario envía su contraseña (encriptada por HTTPS), y el servidor la hashea. Si la validación es correcta se le asigna su usuario a su token de sesión.

Nos hemos asegurado de que aunque el usuario cambie su nombre de usuario, el token lo tendrá en cuenta. De este modo no puede usar su token para conectare a una cuenta con su antiguo nombre.

Para mirar si la contraseña introducida es la correcta, le enviamos al cliente la contraseña correcta, y el cliente hace la comparación.

Arreglado por Diego Esteban

Una vez más, el sistema de tokens de sesión soluciona este problema.

Para saber si alguien tiene la sesión iniciada solo miramos la cookie de "username", no verificamos que tenga una contraseña.

Arreglado por diego Esteban

Una vez más, el sistema de tokens de sesión soluciona este problema.

A la hora de modificar los datos, se modifican los del usuario que se especifica en un campo (oculto) en el form.

Arreglado por Diego Esteban

Ahora se modifican los datos del usuario obtenido por el token de sesión. No puedes modificar los datos de otro usuario sin haber iniciado sesión en ese usuario.

Fallos criptográficos

Usamos una conexión no cifrada (HTTP)

Arreglado por Diego Esteban

Para acceder a la página web con https basta con entrar en https://localhost:444

Es posible entrar con http mediante http://localhost:81 ya que nuestro certificado no está firmado por un CA y es incómodo. Pero si fuésemos a usar esta página web de verdad, no deberíamos permitir que se use una conexión http, ya que la contraseña se envía en texto plano.

No utilizamos encryption at rest.

Arreglado por Francisco González

Parcialmente. Solo se encuentra encriptada mediante AES la información sensible del usuario.

Almacenamos las contraseñas en plaintext.

Arreglado por Diego Esteban. Ibai Mendivil ha cambiado el campo contraseña de varchar(50) a varchar(60)

Utilizamos las funciones de PHP password_verify() y password_hash(). De este modo almacenamos tanto el hash en sí como la sal en un solo string, que en total ocupa 60 caracteres, por eso hemos tenido que ampliar el tamaño del campo de la base de datos.

Usamos el algoritmo CRYPT_BLOWFISH, que está pensado para hashear contraseñas y es seguro.

Inyección

No hacemos ninguna verificación sobre el archivo que se supone que es la portada del libro. (Arreglado, Francisco González)

Arreglado por Francisco González

Se comprueba que las portadas subidas sean imágenes.

Ni siquiera miramos si es una imagen.

Arreglado por Francisco González

Se comprueba que las portadas subidas sean imágenes.

No hay límite de lo grande que puede ser la imagen.

Arreglado por Francisco González

El tamaño del archivo no puede ser mayor a 500000 bytes.

Si ya existe una imagen con ese nombre, se sobreescribe.

No arreglado Si ocurre esto se le asignara a la imagen el path de la que ya está subida.

No parametrizamos los comandos SQL.

Arreglado, ver inyección SQL.

No validamos los datos desde el servidor, sólo desde el cliente.

Arreglado Diego Esteban y Francisco Gonzalez

Mientras que hemos conservado la validación del lado del cliente, tambien lo validamos desde el servidor, ya que es posible saltarse las validaciones de cliente.

Aunque dejarlo en los dos lados supone un coste de mantenimiento mayor, la experiencia de usuario es mucho mejor si javascript alerta de los problemas, sin tener que preguntarle al servidor.

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.

Arreglado, ver Cross site scripting

Diseño inseguro

No hay límites de intentos de inicio de sesión.

Arreglado por Francisco González

Ahora el servidor bloquea a a cualquier usuario que falle al iniciar sesión 3 veces seguidas. Si un usuario acierta antes de llegar a esos 3 intentos, su contador se reseteará a 0. Si por el contrario no lo consigue, se le pondrá a 0 automáticamente desde el servidor en un tiempo no superior a 30 minutos.

No hay límites de accesos por segundo/minuto.

No arreglado

Podrían registrar cientos de libros con imágenes enormes, llenando así el disco duro del servidor.

Arreglado por Francisco González. Se ha añadido un captcha además de que se ha limitado el tamaño de las imágenes a 500000 bytes

Podrían incluir archivos maliciosos en vez de imágenes al subir la portada de un libro.

Arreglado por Francisco González. Ahora el servidor comprueba que el tipo de archivo sea una imagen. Esto no quita que pueda encontrarse código malicioso dentro de ella.

Podrían hacerse ataques de fuerza bruta para conseguir la contraseña de un usuario.

Arreglado por Francisco Gonzalez

Ahora el servidor bloquea a a cualquier usuario que falle al iniciar sesión 3 veces seguidas. Si un usuario acierta antes de llegar a esos 3 intentos, su contador se reseteará a 0. Si por el contrario no lo consigue, se le pondrá a 0 automáticamente desde el servidor en un tiempo no superior a 30 minutos.Por otro lado, el servidor registra todos los intentos de inicio de sesión reiterados, lo que puede ayudar a detectar este tipo de ataque.

Podrían registrarse miles de cuentas falsas o "bots"

Arreglado por Francisco Gonzalez Se ha añadido un captcha

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)

Arreglado por Diego Esteban

Ahora el cliente le envía al servidor el usuario. Si el servidor le responde con el mismo usuario, es que existe. Si el servidor no responde nada es que no existe.

Este sistema se puede explotar para obtener una lista de usuarios que existen, ya que no hay límites de acceso a la API por tiempo. Aun así, es mucho más seguro que el sistema anterior.

Podría ser un problema que solo miremos que no haya 2 usernames repetidos (tlfs, emails y dnis pueden repetirse).

No arreglado

Ya que no utilizamos ninguno de estos datos, no es de gran importancia que tengan sentido, por lo que hemos priorizado fallos de seguridad más importantes.

No se existe una configuración mínima de contraseña (más allá de el mínimo de 3 caracteres)

Arreglado por Francisco González La configuración mínima establecida es la siguiente: longitud>=6, una mayúsculas, una minúsculas, un número y un caracter especial.

Configuración de seguridad insuficiente

La contraseña del usuario admin para acceder a la base de datos es *test*

Arreglado por Diego Esteban.

No sólo hemos utilizado una contraseña mucho más segura, si no que además hemos automatizado el proceso de cambiarla. El procedimiento para cambiar la contraseña se ha documentado en `/documentacion/cambio de contraseña.md`

Consiste en básicamente cambiar la contraseña en `db_pass.txt` y ejecutar el script de python `update_password.py`.

Hay que mencionar lo obvio, db_pass.txt se encuentra en un repositorio público de GitHub, por lo que este sistema es mejorable.

Por ejemplo podríamos establecer un entorno de pruebas y un entorno de producción, y no cualquier persona podría acceder al db_pass.txt de producción.

La contraseña del usuario admin, dentro de la página en sí, también es *test*

No arreglado.

Dado que el usuario admin de la página web no tiene privilegios especiales, ni libros escritos por él, no tiene mucha importancia. No es más que un usuario más con un nombre distinto.

Componentes vulnerables y obsoletos

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

Arreglado por Diego Esteban

Ahora usamos la versión 5.2.0, que es la más reciente en este momento. De esta manera la única diferencia es que si llega a haber una actualización maliciosa no nos afectaría. Pero tenemos que estar más atentos si hay nuevas versiones que arreglan fallos de seguridad, para actualizar manualmente.