
# Informe de auditorías

# "VIAJES GALÁCTICOS"

Autores:

Unai Durán Arce

Marcos Martínez Gonzalo

ENTREGA : 19 - 12 -2021

# ÍNDICE

# REALIZACIÓN DE LAS AUDITORÍAS

Herramienta ZAP :

Para la realización de las auditorías de las entregas 1 y 3, se ha utilizado la herramienta creada por OWASP (*Open Web Application Security Project*). En este primer punto se representará por medio de pasos a modo de tutorial de como se han realizado ambas auditorías.

1º Paso : *Cargar y ejecutar el proyecto mediante Docker :*

Como primer paso debemos cargar y ejecutar el proyecto al que se le desea realizar la auditoría mediante la utilización de Docker (*Herramienta que ya se utilizó en la primera entrega*). Los pasos para cargar el proyecto se encuentran en el archivo denominado "*guia_docker.md*" el cual esta almacenado en el .zip de la primera entrega.

2º Paso : Abrir la herramienta ZAP :

Suponiendo que anteriormente se ha descargado e instalado la herramienta ZAP (), debemos ejecutar la propia aplicación la cual nos mostrará el siguiente menú (**Ilustración 1**).

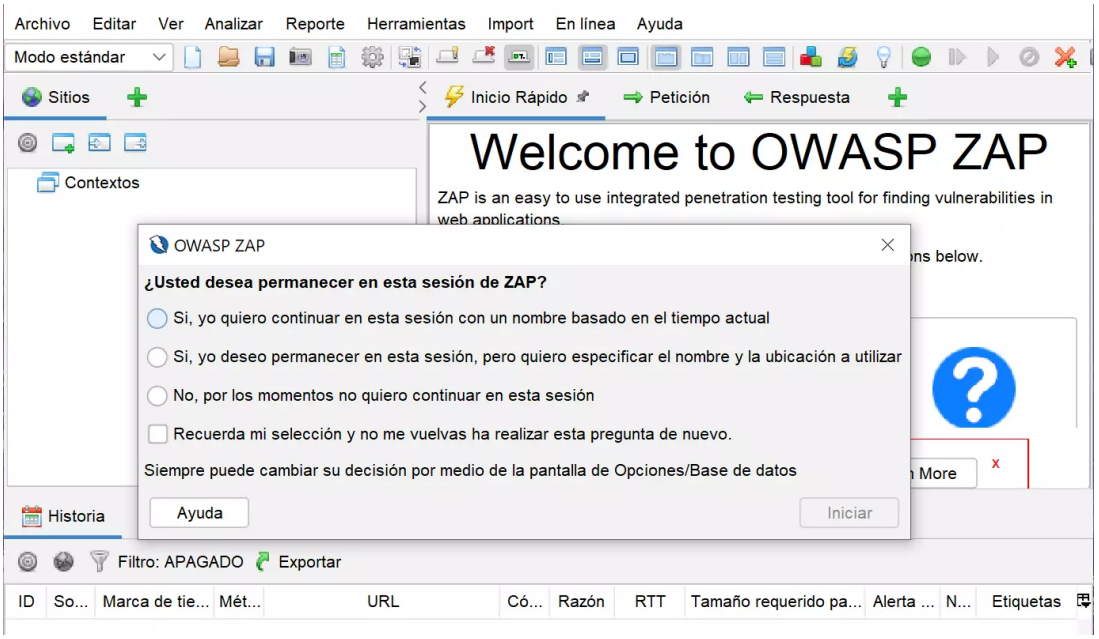


**Ilustración 1**

En el siguiente menú debemos elegir la opción de "*No, por los momentos no quiero continuar en esta sesión*" (*3º opción*). Tras la selección se debe pulsar en el botón de "*Iniciar*".

3º Paso : Comenzar la auditoría (Forma Automática) :

Una vez finalizado el apartado anterior, Veremos el siguiente menú (**Ilustración 2**)

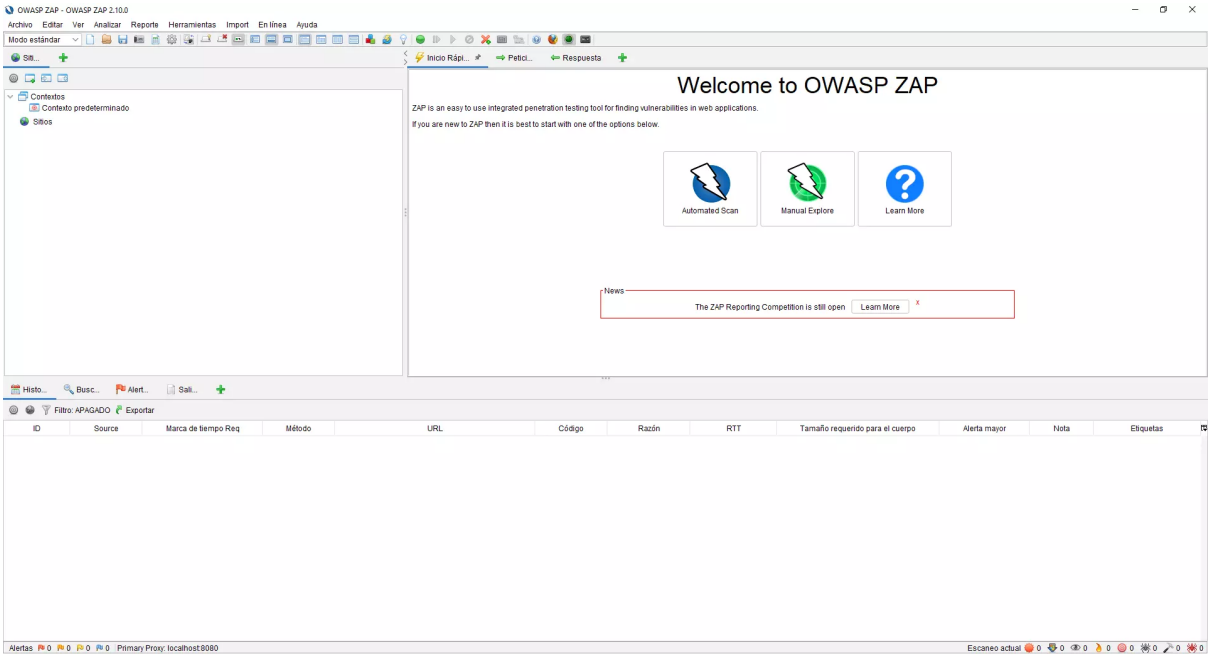


**Ilustración 2**

En este menú debemos hacer click en el icono que contiene un rayo blanco y un fondo azul en el cual está escrito "*Automated scan*" (**Ilustración 3**)

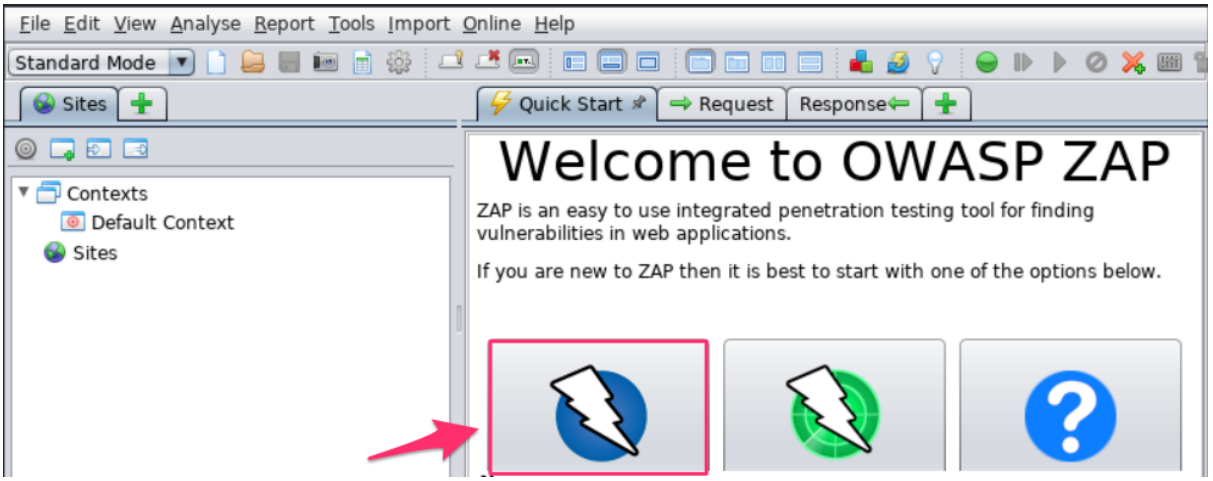


**Ilustración 3**

Una vez clickado aparecerá una nueva pantalla en la cual debemos introducir la información correspondiente para que pueda realizar la auditoría (**Ilustración 4**)

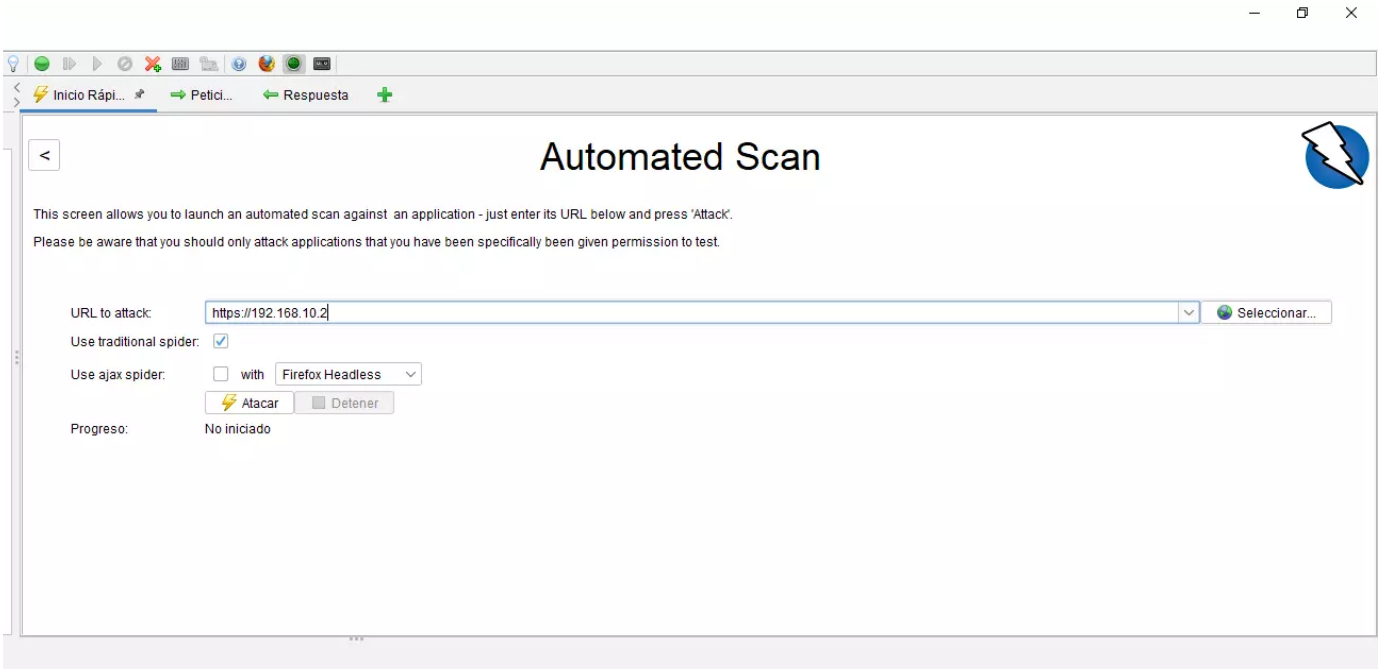


**Ilustración 4**

En este menú debemos tenemos varios apartados :

- URL to attack → En este apartado se debe introducir la dirección de nuestra página web, en nuestro caso se trata de la siguiente "*http://localhost:81/*"
- Use traditional spider → En este apartado nos da la oportunidad de utilizar un spider tradicional de Ajax (*Herramienta que permite descubrir nuevos recursos en una URL indicada*). Debe estar activada.
- Use ajax spider → En este apartado podemos seleccionar el perfil de navegador, en nuestro caso hemos utilizado **Firefox**.

Una vez rellenado los apartados, se debe pulsar en el botón "*Atacar*", una vez presionado comenzará el ataque el cual no tardará mucho tiempo.

4º Paso : Analizar el resultado del ataque :

Una vez la herramienta finalice con el ataque a nuestra web obtendremos una serie de avisos en la consola que contiene la herramienta la cual se encuentra en la zona inferior. (**Ilustración 5**)

**Ilustración 5**

En la parte izquierda aparecen los avisos los cuales determinan las vulnerabilidades de la web. Una vez se seleccione alguno de los avisos, aparecerá su información correspondiente en el lado derecho, en el cual te informa del problema (nivel de gravedad , en que archivo se encuentra, en algunos casos como solucionarlo …)

# ANÁLISIS DE LAS AUDITORÍAS

**Auditoría 1 :**

En cuanto a la primera auditoría nos proporciona 10 alertas sobre nuestra web las cuales presentaremos en los siguientes puntos :

- **X-Frame-Options Header →**
  *Nos alerta de que la cabecera X-Frame-Options no esta incluida en la respuesta de HTTP para proteger frente a los ataques "Clickjacking".*
- **Ausencia de fichas (tokens) Anti-CSRF →**
  *Esto permite que se puedan llevar a cabo ataques XSS.*
- **Cookie No HttpOnly Flag →**
  *Nos alerta sobre que una cookie no utiliza el flag HttpOnly, esto permite que se pueda acceder a dicha cookie con JavaScript.*
- **Cookie without SameSite Attribute →**
  *Nos alerta sobre que una cookie ha sido enviada sin el atributo SameSite, esto conlleva que la cookie pueda ser enviada como resultado de una solicitud "cross-site".*
- **Divulgación de la marca de hora - Unix →**
  *Una marca de tiempo ha sido divulgada por el servidor, aplicación ... (Este problema surge por los comentarios con los ejemplos de cómo se deben rellenar ciertos campos, por lo que no es una amenaza real).*
- **El servidor divulga información mediante un campo →**
  *El servidor de la web está divulgando información mediante uno o más encabezados de respuesta HTTP "X-Powered-By".*
- **X-Content-Type-Options Header Missing →**
  *La cabecera "Anti-MME-Sniffing" llamada "X-Content-Type-Options" no ha sido declarada como "nonsniff", esto ayuda a las versiones antiguas de Chrome e Internet Explorer a realizar MIME-sniffing.*
- **Divulgación de información - Comentarios sospechosos →**
  Alerta sobre que entiende que hay comentarios sospechosos que ayudarían al atacante a forzar vulnerabilidades. (*En este caso referencia un comentario en el que habla sobre la acción que debe realizar un método, lo cual no es una vulnerabilidad*).
- **Information Disclosure - Sensitive information URL →**
  *La petición aparenta contener información sensible divulgada en el URL.*

**Auditoría 2 :**

En cuanto a esta segunda auditoría representaremos los errores sin subsanar respecto la anterior auditoría junto con su posible solución :

- **X-Frame-Options Header →**
  *Solución →*
  - *Asegúrese de que está puesto en todas las páginas devueltas por la web.*

- **Ausencia de fichas (tokens) Anti-CSRF →**
  *Solución →*
  - *Utilice una biblioteca o marco comprobado que no acepte que ocurra esta debilidad o que proporcione construcciones que permitan que esta debilidad sea más sencilla de evitar. Por ejemplo, utilice el paquete anti-CSRG como el CSRGuard de OWASP.*

- **Cookie No HttpOnly Flag →**
  *Solución →*
  - *Asegúrese de que el flag "HttpOnly" está puesto para todas las cookies.*

- **Cookie without SameSite Attribute →**
  *Solución →*
  - *Asegúrese que el atributo SameSite está establecido como 'lax' o idealmente 'strict' para todas las cookies.*

- **El servidor divulga información mediante un campo →**
  *Solución →*
  - *Asegúrese que su servidor web, servidor de aplicación, equilibrador de carga, etc. Está configurado para suprimir encabezados "X-Powered-By".*

- **X-Content-Type-Options Header Missing →**
  *Solución →*
  - *Asegúrese de que el servidor web utilice la cabecera "Content-Type" apropiadamente, y que también aplique la cabecera "X-Content-Type-Options" a "nosniff" para todas las páginas de la web.*

# CAMBIOS REALIZADOS

***A1.- Inyección :***

En cuanto a la vulnerabilidad de inyección, estaba bastante cubierta en la primera entrega, puesto que algunos datos debíamos filtrarlos (*Mediante la utilización de regex*) antes de introducirlos en la base de datos, esto evita que se puedan introducir caracteres "raros", otra manera utilizada es evitar que un usuario pueda ver y editar los datos de los demás usuarios, lo cual era una función que debía existir en la primera entrega.

***A2.- Pérdida de autenticación:***

En cuanto a la pérdida de autenticación, la manera de evitar esta vulnerabilidad debía darse con la automatización del cierre de sesión tras un minuto de inactividad. Para ello se crea un JavaScript con la función de inactividad, la cual llama a un archivo PHP para avisar que ha cerrado sesión y cierra sesión. Se llama al archivo JavaScript desde todas las páginas de la web. (Se cierra sesión automática al pasar más de un minuto en una página de la web, independientemente si mueves el ratón o tecleas).

***A3.- Exposición de datos sensibles :***

En cuanto a la exposición de datos sensibles, se ha realizado un cifrado de las contraseñas de los usuarios en el que se ha utilizado un "*salt*" por cada usuario registrado, además, se ha cifrado un nuevo atributo de la clase usuario el cual contiene el número de cuenta bancaria de cada usuario.

***A5.-Rotura de control de acceso***

En cuanto a la rotura de control de acceso, el único cambio realizado ha sido el de borrar la opción de poder eliminar los viajes reservados por usuarios diferentes al registrado, en cuanto a las modificaciones de los datos de usuario, únicamente se podían editar los datos del usuario que ha iniciado sesión (*Función ya realizada en la primera entrega*). Por otra parte, cada usuario solo puede acceder a editar sus propios viajes reservados por lo que, si un usuario no tiene viajes reservados la tabla de "*Mis Viajes*" se verá vacía, esto permite que no pueda acceder a modificar ningún viaje que no corresponda a su usuario.

***A6.- Configuración de seguridad incorrecta :***

Para evitar esa vulnerabilidad, se impone a cada usuario que su contraseña sea "*segura*"; es decir, que su contraseña contenga un mínimo de ocho caracteres de los cuales deben contener un al menos una minúscula, una mayúscula, un número y un carácter especial como un punto, para ello se a utilizado el filtraje mediante regex en el cual se han indicado los parámetros correctos.

***A7.- Secuencia de comandos en sitios cruzados :***

Para no permitir ataques de tipo XSS, se ha utilizado la misma medida que en el apartado A1 en el cual se ha explicado que en los "*textbox"* se ha utilizado un método de filtrado (*regex*) para evitar caracteres extraños que permitan ataques.

***A10.- Logueo y monitorización insuficientes :***

Para llevar a cabo un monitoreo de todos los inicios de sesión tanto correctos como incorrectos, se ha creado una nueva tabla dentro de la base de datos la cual registra el usuario que ha intentado iniciar sesión, junto a la fecha y hora del intento y su estado (T = Correcto y F = Incorrecto).

# CONCLUSIÓN

El desarrollo de esta nueva entrega nos a permitido entender cómo utilizar nuevas herramientas como *"ZAP"* y a la vez descubrir la importancia que le debemos de dar a la seguridad de nuestra página web o aplicación web, puesto que gracias a los análisis de la herramienta comentada creada por *"OWASP"* en nuestro caso nos ha abierto los ojos sobre los muchos de los agujeros de seguridad que teníamos en nuestra aplicación y además, nos ha aconsejado sobre cómo conseguir solucionarlos junto con una explicación sobre el posible problema que nos puede provocar si un usuario maligno consigue acceder a dicho déficit de seguridad.

Además, por un lado , mediante esta entrega hemos aprendido la utilización de cifrado de los datos para evitar su intercepción por agentes externos como la importancia del filtraje de la información que se introduce para conseguir que usuarios malignos no anden a sus anchas por toda nuestra aplicación web sino que sigan el camino creado por los desarrolladores y evitar que extraigan información sensible sobre otros usuarios registrados.

Por otro lado, hemos aprendido, las distintas formas en las que se puede atacar a una página y ver que no hace falta ningún tipo de software o aplicación compleja para poder robar los datos de una web que no esté bien configurada o protegida.

Finalmente, sacamos la conclusión de que es muy importante proteger bien los datos de la base de datos, para evitar ser atacado; puesto que un ataque atrae problemas de posible revelación de los datos, chantajes, secuestros de los datos… lo cual no solo nos perjudica a nosotros como desarrolladores sino que también afecta a nuestros clientes y con ello repercute en la confianza de futuros clientes en utilizar nuestras aplicaciones o páginas web o en contratarnos para desarrollar una nueva.

# BIBLIOGRAFÍA


Ilustraciones (1-5):

[1] redeszone.net
OWASP ZAP, audita la seguridad de webs y evita vulnerabilidades
https://www.redeszone.net/tutoriales/seguridad/owasp-zap-auditar-seguridad-web/
(accedido el 01/12/2021)

Líneas de código :

[2] youtube.com (Autor : Carlos Alfaro)
Encriptar datos (ha sido adaptado por nosotros)
https://www.youtube.com/watch?v=pJhwIm3YL0o
(accedido el 01/12/2021)

[3] ejemplocodigo.com
Generate random string
https://ejemplocodigo.com/ejemplo-php-generar-cadena-aleatoria-o-random-string/
(accedido el 01/12/2021)

[4] lawebdelprogramador.com
Cierre de sesion por inactividad
https://www.lawebdelprogramador.com/foros/PHP/1362548-Cierre-de-sesion-por-inactividad-en-PHP.html