Sistemas de Gestión de Seguridad de Sistemas de Información

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORME DE LAS AUDITORÍAS

Lucas Abaitua
Estefanía Oñate

ÍNDICE DE CONTENIDOS Y FIGURAS.

¿Cómo se han realizado las auditorías?

Para la realización de ambas autorías hemos utilizado una de las herramientas de pentesting[1] más conocidas, ZAP. Esta herramienta actúa como si fuera un proxy y trabaja con todas las peticiones que se hacen al sitio web, de esta manera analiza tanto las peticiones como las respuestas en busca de posibles agujeros de seguridad.

Paso 1: Elegir sesión

Supondremos que el programa está instalado. Al ejecutar el programa, antes de que se abra la página principal de ZAP (Figura 2), se abrirá una ventana para elegir la sesión (Figura 1) que se desee utilizar, se selecciona la que más nos convenga y se inicia.

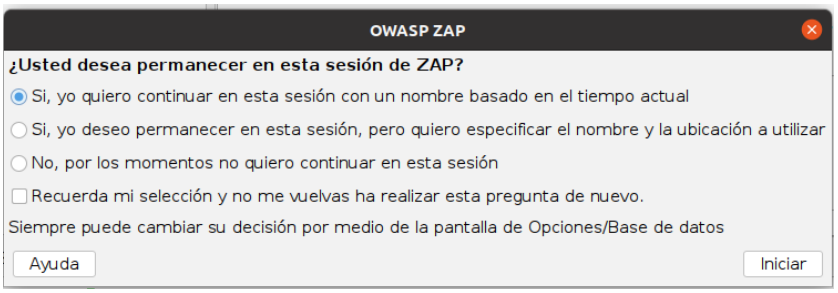

Figura 1: Elección de las sesiones de ZAP

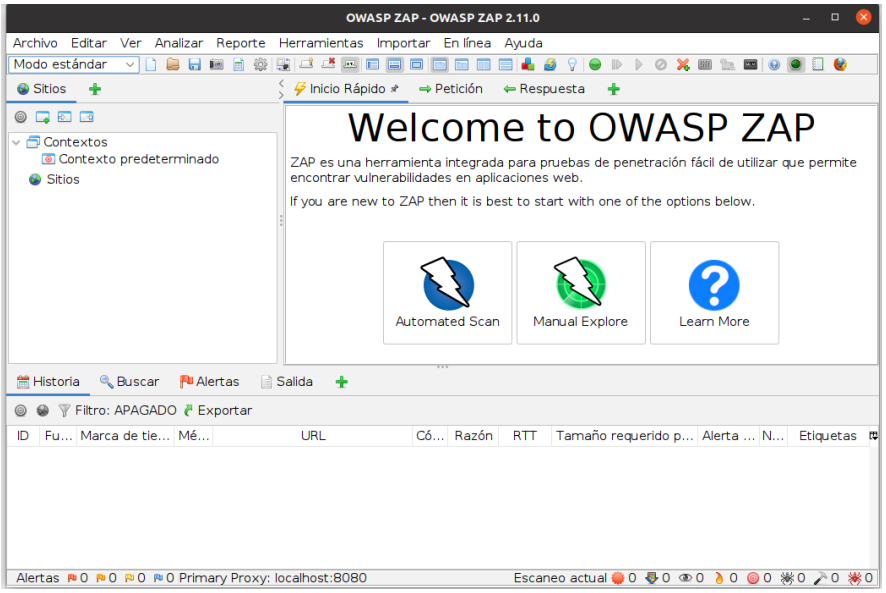

Figura 2: Página principal de ZAP

[1] Consiste en atacar diferentes entornos o sistemas con el objetivo de detectar y prevenir posibles fallos.

Paso 2: Elegir una opción para encontrar las vulnerabilidades

En ZAP tenemos dos opciones para atacar a la página, la forma de escaneado automático (Figura 3) que será la que explicaremos en el siguiente paso y la forma manual de escaneado.
La forma manual de escaneado: Consiste en hacer las peticiones y las respuestas a ZAP manualmente, el inconveniente que vemos de seleccionar esta opción es que podemos dejarnos casos sin probar.

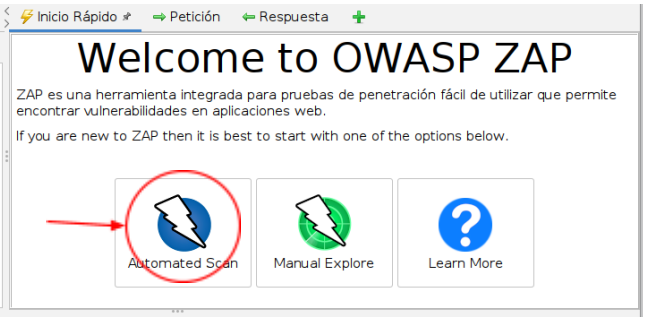

Figura 3: Pantalla principal donde está marcado el escaneo automático

Paso 3: Escaneado automático

Es la forma más fácil de escaneado. Para realizar este escaneado hay que configurar las siguientes opciones (Figura 4)

- Tendremos un campo donde habrá que poner la URL de donde queremos atacar, en nuestro caso http://localhost:81
- Lo siguiente que tenemos para seleccionar es el uso de la “araña tradicional”, aunque estará seleccionada por defecto. La “araña tradicional” descubre enlaces examinando el HTML en las respuestas de la aplicación web.
- También hay otra araña llamada ajax, que será lo siguiente que se pueda seleccionar, esta araña explora la aplicación web invocando navegadores que luego siguen los enlaces que se han generado. La araña AJAX es más lenta que la araña tradicional y requiere más configuración. Esta opción no viene seleccionada por defecto y nosotros no la seleccionaremos.
- Seleccionamos el botón “Atacar” para comenzar el escaneado

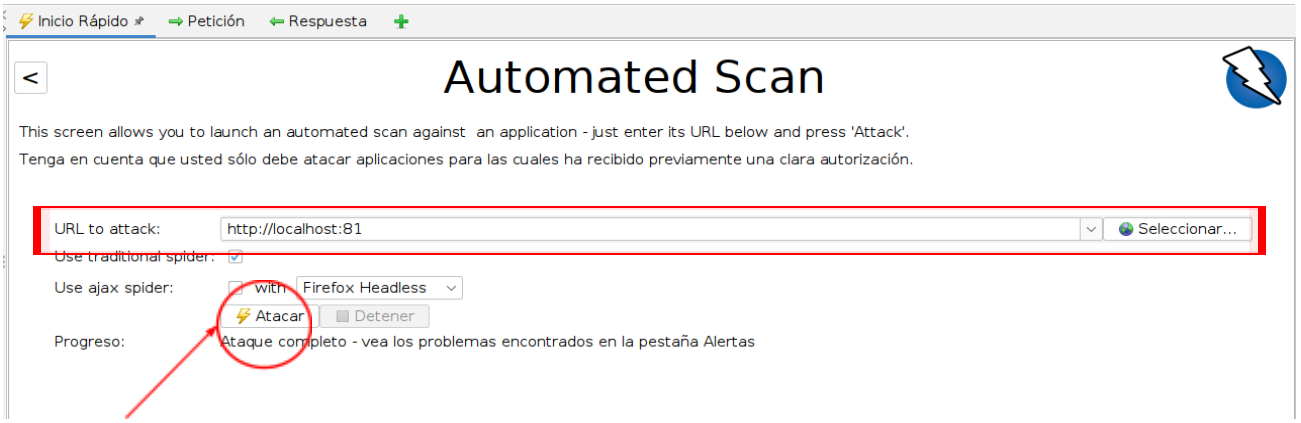

Figura 4: Opciones a seleccionar para configurar el escaneo automático con lo importante resaltado.

Paso 4: Analizar datos

Los mensajes empezarán a enviarse en la parte inferior de la página, en la pestaña de “Escaneo activo” (Figura 5), tras completarse el escaneo, se abrirá la pestaña de “Alertas” (Figura 6) donde observaremos las alertas que se han encontrado, cada alerta se clasificará por banderas, no es necesario que aparezcan todas las banderas ya que puede suceder que no haya banderas de ese tipo al realizar el ataque como es el caso de la Figura 6. Las banderas tendrán colores diferentes y estarán ordenados por niveles de riesgo:

- Bandera roja: Nivel de riesgo alto.
- Bandera naranja: Nivel de riesgo medio.
- Bandera amarilla: Nivel de riesgo bajo.
- Bandera azul: Informativo.
- Bandera verde: Falso positivo

Para poder interpretar estas alertas tenemos diferentes formas de hacerlo, por una parte podemos seleccionar la alerta (Figura 7) que queremos analizar y de esta manera nos aparecerán diferentes apartados entre los que destacamos los siguientes ya que son los que más nos van a resultar útiles como “Evidencia” que será la línea de código donde se origina la alerta, la “Descripción” y la “Solución”. Por otra parte, también podemos analizar las alertas mediante las pestañas en la parte superior derecha de la página que son “Petición” y “Respuesta”, en la segunda de estas podemos observar también una línea resaltada que será la misma de la primera forma de interpretación, es decir, la línea donde se origina la alerta.

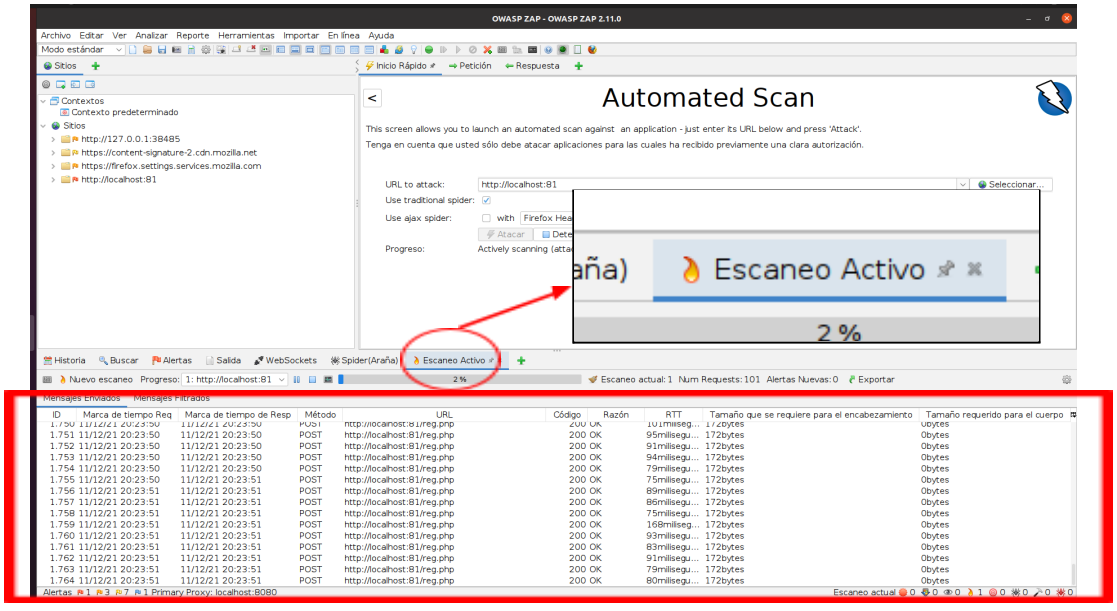

Figura 5: Pestaña de “Escaneo activo” con los mensajes enviandose en la parte inferior

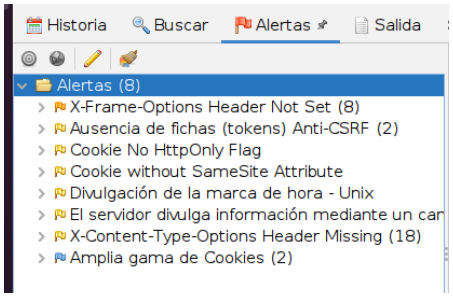

Figura 6: Pestaña de “Alertas”

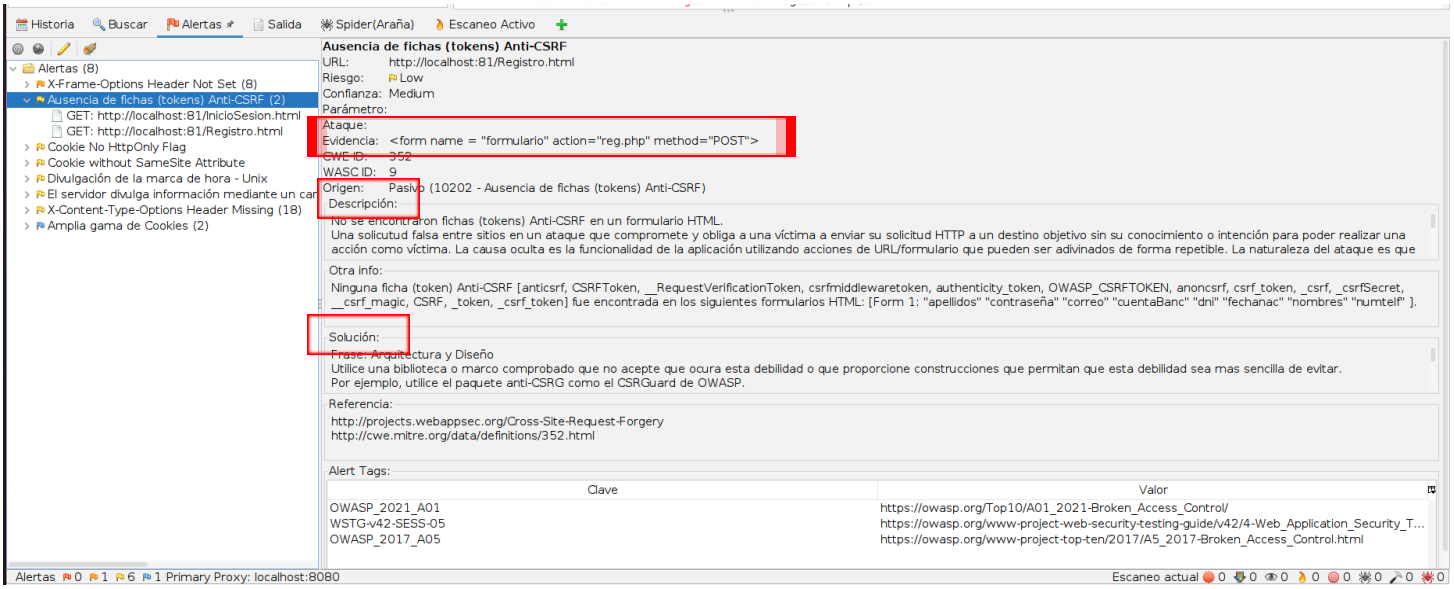

Figura 7: Descripción de una alerta

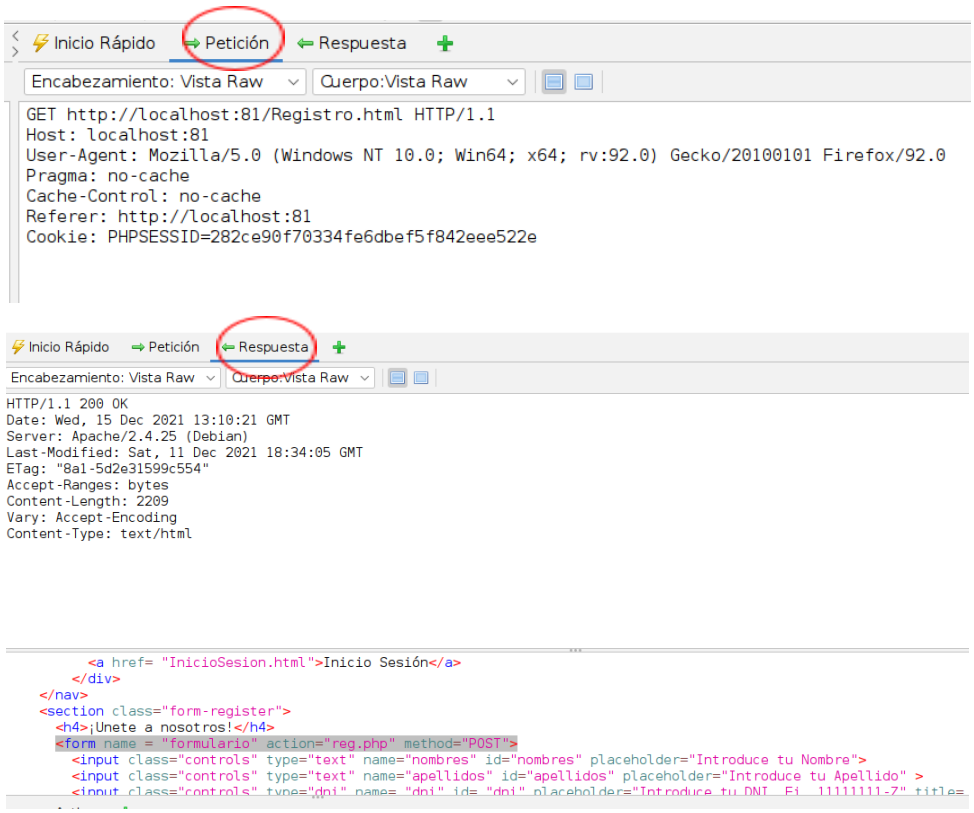

Figura 8: Pestañas de “Petición” y “Respuestas”

Análisis sobre el resultado de ambas auditorías.

AUDITORÍA 1:

En la primera auditoría realizada a la entrega original, tuvimos 11 alertas que recopilaron un total de 113 instancias. El resumen de las alertas e instancias fue la siguiente:

Nombre	Nivel de riesgo	Number of Instances
Cross Site Scripting (Reflected)	Alto	1
Falla por Inyección SQL	Alto	2
Application Error Disclosure	Medio	2
X-Frame-Options Header Not Set	Medio	27
Ausencia de fichas (tokens) Anti-CSRF	Bajo	9
Cookie No HttpOnly Flag	Bajo	1
Cookie without SameSite Attribute	Bajo	1
Divulgación de la marca de hora - Unix	Bajo	2
El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By""	Bajo	25
X-Content-Type-Options Header Missing	Bajo	37
Amplia gama de Cookies	Informativo	6

Figura 9: Resumen de las alertas de nuestra primera entrega

Analizaremos cada alerta en orden de nivel alto de riesgo al más bajo/informativo tal y como aparece en la imagen (Figura 9).

1. Secuencias de comandos en sitios cruzados: Permite ejecutar secuencias de comandos en el navegador de la víctima, para robar credenciales, secuestrar sesiones, o la instalación de software malicioso en el equipo de la víctima. (n4, 2018) En nuestro caso pasaba en el archivo editarAct.php (Figura 10).

URL	http://localhost:81/editarAct.php?+id=%22%3E%3Cscrlpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Método	GET
Atacar	"><scrlpt>alert(1);</scRipt>
Evidence	"><scrlpt>alert(1);</scRipt>

Figura 10: Detalle de la alerta “Secuencias de comandos en sitios cruzados”

2. Falla por inyección SQL: En nuestro caso, teníamos dos instancias que permitían la inyección por SQL, en el archivo inicioSes.php que es el archivo mediante el que validamos la información para que se pueda iniciar sesión (Figura 11).

URL	http://localhost:81/InicioSes.php
Método	POST
Atacar	ZAP' OR '1'='1' --
Evidence	

Figura 11: Detalle de una de las instancias de la alerta “Falla por inyección SQL”

3. Divulgación de errores de la aplicación: Nos advierte de que nuestra página contiene un mensaje de error / advertencia que puede revelar información confidencial (Figura 12).Tenemos dos instancias de esta alerta.

URL	http://localhost:81/a%C3%B1adirAct.php
Método	GET
Atacar	
Evidence	Warning: mysqli_num_rows() expects parameter 1 to be mysqli_result, boolean given in /var/www/html/añadirAct.php on line 11

Figura 12: Detalle de una de las instancias de la alerta “Divulgación de errores de la aplicación”.

4. Encabezado X-Frame-Options no establecido: El X-Frame options sirve para prevenir aperturas de la página en frames o iframes. Luchando así contra el clickjacking[2]. Tenemos 27 instancias de esta alerta (Figura 13).

URL	http://localhost:81/a%C3%B1adirAct.php
Método	GET
Atacar	
Evidence	

Figura 13: Detalle de una de las instancias de la alerta “Encabezado X-Frame-Options no establecido”.

5. Ausencia de Fichas (tokens) Anti-CSRF: No se encontraron fichas (tokens) Anti-CSRF en nuestro formulario HTML. Los ataques de CSRG[3] son muy efectivos en varias situaciones por ejemplo en nuestro caso ya que la “víctima” tiene una sesión activa en el sitio de destino. Tenemos 9 instancias de esta alerta (Figura 14).

[2] Es un ataque que engaña a un usuario para que haga clic en un elemento de la página web que es invisible o está disfrazado de otro elemento.
[3] Cross Site Request Forgery o falsificación de petición en sitios cruzados

URL	http://localhost:81/Actividades.php
Método	GET
Atacar	
Evidence	<form name = "anadirAct" action="añadirAct.php" method="POST">

Figura 14: Detalle de una de las instancias de la alerta “Ausencia de Fichas (tokens) Anti-CSRF”.

6. Cookie No HttpOnly Flag: Se ha establecido una cookie sin la marca HttpOnly, lo que significa que la cookie puede ser accedido por JavaScript. Si se puede ejecutar un script malicioso en esta página, la cookie es accesible y se puede transmitir a otro sitio. Únicamente tenemos una instancia de esta alerta (Figura 15).

URL	http://localhost:81
Método	GET
Atacar	
Evidence	Set-Cookie: PHPSESSID

Figura 15: Detalle de la alerta “Cookie No HttpOnly Flag”.

7. Cookie sin atributo SameSite: Nos informa que las cookies no tienen el atributo SameSite que es una manera segura de identificar las cookies que no sean de petición ‘Cross-site’. El atributo SameSite es una medida efectiva contra solicitudes de falsificación de cross-site, inclusión de secuencias cross-site, y ataques sincronizados. Tenemos una única instancia de esta alerta (Figura 16).

URL	http://localhost:81
Método	GET
Atacar	
Evidence	Set-Cookie: PHPSESSID

Figura 16: Detalle de la alerta “Cookie sin atributo SameSite”.

8. Divulgación de la marca de hora - Unix: Una marca de tiempo ha sido divulgada por el servidor de la aplicación/ navegador - Unix. Tenemos dos instancias de esta alerta (Figura 17).

URL	http://localhost:81/Registro.html?apellidos=ZAP&contrasena=ZAP&correo=ZAP&dni&fechanac=2021-12-01&nombres=ZAP&numtelf=9999999999
Método	GET
Atacar	
Evidence	11111111

Figura 17: Detalle de una de las instancias de la alerta “Divulgación de la marca de hora - Unix”.

9. El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP “X-Powered By”: El acceso a tal información podría facilitarle a los atacantes la identificación de otros componentes de los que su aplicación web depende y las vulnerabilidades a las que pueden estar sujetos. Tenemos 25 instancias de esta alerta (Figura 18).

URL	http://localhost:81/a%C3%B1adirAct.php
Método	GET
Atacar	
Evidence	X-Powered-By: PHP/7.2.2

Figura 18: Detalle de una de las instancias de la alerta “El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP “X-Powered By””.

10. X-Content-Type-Header-Missing: El encabezado Anti-MIME-Sniffing X-Content-Type-Options no se estableció en 'nosniff'. Esto permite versiones anteriores de Internet Explorer y Chrome para realizar un rastreo de MIME en la respuesta cuerpo, lo que puede hacer que el cuerpo de la respuesta se interprete y muestre como un contenido tipo distinto del tipo de contenido declarado. Tenemos 37 instancias de esta alerta (Figura 19).

URL	http://localhost:81/Registro.html?apellidos=ZAP&contrasena=ZAP&correo=ZAP&dni&fechanac=2021-12-01&nombres=ZAP&numtelf=9999999999
Método	GET
Atacar	
Evidence	

Figura 19: Detalle de una de las instancias de la alerta “X-Content-Type-Header-Missing”.

AUDITORÍA 2

Esta auditoría se ha realizado tras solucionar las 7 vulnerabilidades que se nos ha indicado en el enunciado. Entre las alertas que nos muestra esta auditoría se encuentran también alertas de la primera auditoría por lo que únicamente explicaremos de manera teórica cómo se solucionan las de esta última auditoría (Figura 20).

Alertas

Nombre	Nivel de riesgo	Number of Instances
Ausencia de fichas (tokens) Anti-CSRF	Bajo	2
Cookie No HttpOnly Flag	Bajo	1
Cookie without SameSite Attribute	Bajo	1
Divulgación de la marca de hora - Unix	Bajo	1
El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP ""X-Powered-By""	Bajo	9
X-Content-Type-Options Header Missing	Bajo	10
Amplia gama de Cookies	Informativo	2

Figura 20: Resumen de las alertas de nuestra entrega final

1. Ausencia de Fichas (tokens) Anti-CSRF: Esta alerta se podría solucionar mediante el uso de bibliotecas o marcos que no permitan que ocurra el *Cross Site Request Forgery* en la implementación de la página donde suceda, un paquete a utilizar puede ser el “anti-CSRF” de OWASP mismamente.

2. Cookie No HttpOnly Flag: Esta alerta se solucionaría asegurando que este flag está activado para todas las cookies. Se podrían inicializar los parámetros de las cookies en un php y hacer la llamada a su activación antes del session_start()

3. Cookie Without SameSite Attribute: Para evitar esta alerta tendríamos que incorporar el atributo SameSite a todas las cookies, en este caso a las de la página principal. Esto se podría realizar para un servidor HTTPS y para un explorador como Microsoft Edge.

4. Divulgación de la marca de hora - Unix: La solución a esta alerta reside simplemente en la confirmación manual de que los datos de marca de hora no son sensibles, y que los datos no pueden ser agregados a patrones explotables de divulgación. Lo cual sería nuestro caso ya que lo que supuestamente está divulgando nuestra página es un ejemplo del formato a introducir del DNI del usuario en el campo de registro.

5. El servidor divulga información mediante un campo(s) de encabezado de respuesta HTTP “”X-Powered-By””: Para evitar este tipo de alertas deberíamos utilizar una

función que impida a nuestro servidor devolver como respuesta de protocolo HTTP el indicar que se trata de una respuesta HTTP “”X-Powered-By” o que suprima los encabezados ”X-Powered-By” de la respuesta HTTP.

6. X-Content-Type-Options Header Missing: Solucionaremos esta alestra si introducimos una función “header” de tipo “X-Content-Type-Options: nosniff” en nuestros archivos CSS y JAVASCRIPT, y no sólo en los PHP como ya hicimos para solucionar la pérdida de autenticación.

7. Amplia gama de cookies: Una amplia gama de cookies no sería utilizada si utilizaramos “Nombres de Dominio Completamente Calificados”, que son nombres de dominio que identifica todos los niveles del dominio, incluido un subdominio además de la raíz habitual, dominio de nivel superior (TLD), y dominio de segundo nivel. Ejemplo: “mail.google.com” sería uno de ellos.

Explicación de los cambios realizados en el código para solucionar las vulnerabilidades indicadas.

Para este trabajo se nos pidió solucionar 8 vulnerabilidades que mostraremos a continuación con sus respectivas soluciones.

A1.- Inyección El sistema no debe permitir “SQL Injection”.

Para que nuestro sistema no permitiera inyección SQL tuvimos problemas ya que no bastó con tratar los datos filtrandolos mediante una expresión regular. En la primera entrega enviamos la información con el método REQUEST por lo que para evitar modificar el método y arreglar la vulnerabilidad tratamos de utilizar las funciones prepare() y execute() pero estas no llegaron a funcionar. Finalmente tuvimos que modificar el método a POST en vez de REQUEST y utilizamos la función mysqli_real_escape_string() de esta manera solucionamos el problema.

A2.- Pérdida de autenticación

Nuestro sistema ya tenía la opción de cerrar sesión creada en la anterior entrega, por lo tanto simplemente hemos tenido que añadir la función para detectar la inactividad de 60 segundos. Para ello, tras investigar diferentes maneras decidimos crear una función que una vez que se cargue la página, mediante las funciones incorporadas en php “document.onmousemove” y “document.onkeydown” además de un pequeño algoritmo, se puedan contar los segundos en los que el ratón no se mueve o que aunque se mueva estemos en otra página o aplicación que no sea la nuestra. Por lo tanto, simplemente tuvimos que añadir que a los 60000 milisegundos(60 segundos) se hiciera la llamada a otro archivo PHP que, pese a tener la misma funcionalidad que el de cerrar sesión, quisimos crearlo para diferenciar la acción por la cual se produce una desconexión de la sesión, así que de esa manera acabamos con la sesión enviándonos a la página principal de nuestro sistema web.

Para evitar que se acceda a nuestras cookies mediante scripts, primero de todo, hemos tenido que pasar todos nuestros archivos HTML a PHP para poder introducir la función necesaria, ya que en los HTML sucedía también el problema. Tras tener todos los archivos en PHP únicamente hemos tenido que introducir un par de funciones de tipo “header”, la “X-Content-Type-Options: nosniff” y la "X-Frame-Options: DENY", también son útiles para evitar el clickjacking. De esta manera se impide el acceso a las cookies y viajan en protocolos más seguros.

A3.- Exposición de datos sensibles

En esta vulnerabilidad se nos pedía solucionar dos cosas, por una parte teníamos que cifrar la contraseña del usuario en la base de datos para eso nos ayudamos de una página (*Cómo Usar Password_hash De PHP Para Hash Y Verificar Contraseñas*, n.d.) y modificamos el archivo mediante el que nos registramos (reg.php) para que al poner una contraseña válida se inserte cifrada en la BD con la función password_hash(), que implementa una sal aleatoria, donde meteremos como parámetros la contraseña que queremos cifrar y el algoritmo PASSWORD_DEFAULT. También modificamos el archivo para iniciar sesión

(inicioSes.php) de manera que aunque la contraseña está guardada cifrada podamos iniciar sesión con la contraseña con la que nos hayamos registrado, para eso utilizamos la función password_verify() que tendrá como parámetros nuestra contraseña sin cifrar y la contraseña cifrada. Para que todo funcione correctamente tuvimos que modificar el tamaño del campo "contraseña" en la BD por un tamaño de 255.
Por otra parte se nos pedía añadir un nuevo campo al usuario que fuera su cuenta bancaria, que se guarde en la BD cifrada pero que al mostrar al usuario sus datos se muestre descifrada y que pueda modificarlo. Al tener que descifrar la cuenta bancaria en algún punto del código, las funciones utilizadas anteriormente no nos servían por lo que investigamos acerca del cifrado simétrico estudiado en la asignatura. Para implementar esto nos ayudamos de un video (Jesús Domínguez Gutú, 2021), añadimos un nuevo campo en el usuario para la cuenta bancaria con tamaño 255 y editamos el archivo "reg.php" donde definimos una "llave" que será una de las herramientas con las que se cifrará y creamos las funciones de cifrar() y descifrar() como nos explica el video mencionado anteriormente. Para poder visualizar en el perfil del usuario editaremos el archivo "perfil.php" donde llamaremos a la función descifrar(). Para poder modificarlo cogeremos el valor que escribe el usuario y cuando se pulse el botón "Cambiar cuenta" volveremos a llamar a la función cifrar() para que se guarde nuevamente cifrar en la BD.

A5.-Rotura de control de acceso

En este caso teníamos que solucionar 3 cosas:

- Para que el sistema no permita a nadie que no esté logueado el acceso a consultar el listado de actividades ofertadas hemos modificado el archivo donde las listamos hemos utilizado la misma función que a la hora de entrar a "Mi Perfil" no se pudiera visualizar ningún tipo de información, la función de PHP "isset()"(función para ver si una variable está definida). Esta función acompañada del atributo "$_SESSION["user"] (que es el que nos devuelve el usuario que la ha iniciado sesión) se puede utilizar como comprobación de que existe una sesión iniciada. Pusimos una condición, si la variable mencionada anteriormente está definida(lo cual quiere decir que hay una sesión iniciada) que se muestran los datos de la tabla, si no, mostrar un aviso de que para visualizarla se debe loguear.
- Para que el sistema no permita que un usuario modifique los datos de otro usuario, como pasaba en nuestra página al poder eliminar una actividad cualquiera eliminandose también del listado de actividades en el que está apuntado un usuario, hemos tenido que eliminar esa opción de borrado.
- Para que el sistema cierre automáticamente la sesión tras un minuto de inactividad, como hemos mencionado anteriormente, realizamos la llamada al PHP creado para esta ocasión (desconectar.php) cuando se detecte que el ratón no se mueve o cuando se detecte que no nos encontremos visualizando la página pese a mover el ratón durante 60 segundos. Cuando el usuario quiera retomarla tiene la opción de volver a iniciar sesión.

A6.- Configuración de seguridad incorrecta

Para configurar la seguridad correctamente obligamos al usuario a registrar una contraseña segura, es decir, que su longitud sea de al menos 8 caracteres y contenga letras mayúsculas, minúsculas, números y caracteres "extraños" como el punto, la admiración, el

dólar, etc. para esto utilizamos una expresión regular (/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@$!%*?&.])[A-Za-z\d$@$!%*?&.]{8,}$/) que se verificará en un archivo JavaScript con lo que el usuario introduzca como contraseña de manera que si la contraseña no es segura le pedirá que ponga una nueva.

A7.- Secuencia de comandos en sitios cruzados

Debido a que el sistema no debía permitir ataques de tipo XSS (Cross-Site Scripting) hemos tenido que suprimir la acción de poder eliminar una actividad de la lista de actividades (ya que se permitiría la “inyección” de código JavaScript en nuestra página) además de tener que cambiar el acceso a las mismas debido a la rotura de control de acceso. Esto es debido a que cualquier usuario era capaz de poder eliminar una actividad que un usuario guardado en la base de datos tuviera asignada, lo cual permitiría realizar cambios en la base de datos sin poder tener acceso a ello(ya que para añadir una actividad hacemos uso del $_REQUEST y pasamos información por la barra de búsqueda y para eliminar actividades hacíamos uso del $_GET)

A10.- Logueo y monitorización insuficientes

Para almacenar los intentos de acceso, tanto los exitosos como los fallidos, creamos una nueva tabla en la base de datos “intentos_usuario” donde también almacenaremos el correo del usuario y fecha y hora del acceso. En el archivo “inicSes.php” es donde gestionaremos esta tabla, si el intento de inicio sesión es exitoso se añadirá a la tabla su correo, fecha y hora y la palabra ‘exitosa’, de lo contrario aparecerán los mismos datos pero con la palabra ‘fallida’.

Conclusiones sobre las diferencias entre ambas auditorías

Si realizamos una comparativa de lo obtenido en ambas auditorías podemos llegar a varias conclusiones. La primera de ellas es la evidente diferencia de alertas o avisos existentes, en la primera observamos que hay una excesiva cantidad de alertas que hacen de nuestro sistema web uno de muy poca fiabilidad para la estabilidad de ella misma y para la identificación, el procesamiento y el guardado de los datos tanto en la base de datos como los que se mandan de una página a otra para identificar sucesos, esto se ve reflejado en las alertas más graves como la posibilidad de la inyección SQL o la posibilidad de producirse secuencias de comandos en sitios cruzados. Por ello decimos que pese a tener la idea de un diseño con una buena estructura y orientación no era suficientemente funcional para lo que se pedía.

En la segunda auditoría se observa una gran mejora respecto a la primera en el aspecto funcional ya que no permiten (aunque no de manera total) la posibilidad de sucesos tan graves como los que podían ocurrir en la primera. Hemos observado que arreglando ocho de los diez riesgos más críticos en Aplicaciones Web según OWASP en la versión de 2017, las alertas más graves como las clasificadas en ZAP como rojas y naranjas, han desaparecido.

Como conclusión final hemos aprendido la importancia de conocer las principales vulnerabilidades de los sistemas web, como por ejemplo las que nos dan OWASP, para tenerlas en cuenta de cara al diseño, estructura e implementación de la página ya que es muy importante proteger la información del usuario en un sistema. Por otro lado también es conveniente conocer herramientas como ZAP para conocer las debilidades de nuestra página y para comprobar si finalmente las hemos solucionado.

Bibliografía

Cómo usar password_hash de PHP para hash y verificar contraseñas. (n.d.). QA Stack. Retrieved December 18, 2021, from https://qastack.mx/programming/30279321/how-to-use-phps-password-hash-to-hash-and-verify-passwords

Jesús Domínguez Gutú, J. (2021, September 18). *Cifrado Simétrico con algoritmo AES en PHP*. YouTube. Retrieved December 18, 2021, from https://www.youtube.com/watch?v=D-d10rXX4nE

Missing 'X-Content-Type-Options' Header. (2018, November 28). Tenable. Retrieved December 18, 2021, from https://www.tenable.com/plugins/was/112529

n4. (2018, October 25). *Buenas prácticas de desarrollo seguro: A7 – Secuencia de comandos en sitios cruzados (XSS) – NIVEL4 Labs*. NIVEL4 Labs. Retrieved December 19, 2021, from https://blog.nivel4.com/hacking/buenas-practicas-de-desarrollo-seguro-a7-secuencia-de-comandos-en-sitios-cruzados-xss/

OWASP ZAP – Cookie without SameSite Attribute. (n.d.). OWASP ZAP. Retrieved December 18, 2021, from https://www.zaproxy.org/docs/alerts/10054/