Vulnerabilidades de las aplicaciones web

Gran parte de los problemas de la seguridad en las aplicaciones web son causados por tener un buen control de las entradas y salidas de la aplicación.

1. Balancear riesgo y usabilidad

Es recomendable reducir la usabilidad si con ello se consigue una mejor seguridad de la aplicación web, siempre que las medidas de seguridad sean eficientes y no resulten engorrosas al usuario. Por ejemplo, el uso de *login* que solicita el nombre de usuario y contraseña permite controlar el acceso a las secciones restringidas.

2. Rastrear el paso de los datos

Se debe tener un uso adecuado de las variables globales, por ejemplo en PHP serían **$_GET, $_POST, $_COOKIE y $_SESSION** entre otros, que sirven para identificar de forma clara las entradas proporcionadas por el usuario.

3. Filtrar entradas

Si nos aseguramos que los datos son filtrados apropiadamente al entrar, podemos eliminar el riesgo de que datos contaminados sean usados para provocar funcionamientos no deseados en la aplicación. Los propios lenguajes de programación proveen una buena cantidad de filtros aunque puede ser necesario que nosotros mismos creemos filtros que éstos no ofrezcan.

4. Escapados de salidas

Es preciso identificar el destinatario de las salidas (cliente, base de datos, etc.) y adecuarlas a dicho destinatario. Existen funciones nativas en los lenguajes de programación para esta finalidad, por ejemplo en PHP existen ***htmlspecialchars(), strip_tags(),*** etc.

5. Ataques URL de tipo semántico

Al tratar con datos sensibles es aconsejable no usar el paso de datos por URL (usar el método **GET**), ya que pueden ser modificados y leídos fácilmente. En estos casos, se recomienda usar **POST**.

6. Ataques de Cross-Site Scripting(XSS)

Es un tipo de vulnerabilidad de seguridad informática típicamente encontrada en aplicaciones web que permiten la inyección de código por usuarios maliciosos en páginas web. Los atacantes típicamente se valen de código HTML y de scripts ejecutados en el cliente.

7. Ataques de Cross-Site Scripting Request Forgery

Este tipo de ataque permite al atacante enviar peticiones HTTP a voluntad desde la máquina de la víctima. Cuando un atacante conoce el formato que debe tener una URL para lograr la ejecución de una acción en el sistema, ha logrado encontrar la posibilidad de explotar este tipo de ataques.

8. Peticiones HTTP falsificadas

Un atacante puede crear sus propias peticiones HTTP a través de herramientas especiales, por ello es importante ser capaces de detectar que peticiones se deben escuchar y cuáles no.

9. Exposiciones de credenciales de Acceso

Se debe tener especial cuidado con los archivos de configuración que contengan credenciales de acceso y no permitir la localización de dichos archivos. Es necesario configurar el servidor web para rechazar las peticiones de recursos que no deben ser accesibles.

10. SQL Injection

Esta es la principal vulnerabilidad y la más peligrosa, consiste en inyectar código en peticiones a la base de datos para poder obtener toda la información posible. Para evitar esto será necesario filtrar los datos y controlar las salidas.

11. Exposición de datos

Una de las preocupaciones más comunes relacionadas con las bases de datos es la exposición de datos sensibles. Hay que procurar que estén codificados para que si el atacante consigue acceder a ellos no obtenga fácilmente los datos. Se pueden realizar procedimientos de ***hash*** a las cadenas almacenadas para que no sea entendible la información a simple vista.

12. Ataques de fuerza bruta

El atacante a base de prueba y error, intenta obtener los datos deseados. En algunos casos el atacante puede conocer nombres de usuario válidos y la contraseña es la única parte que se trata de adivinar.

13. Espionaje de contraseña (Password Sniffing)

Cuando un atacante tiene los medios para analizar el tráfico entre los usuarios y el servidor de la aplicación, debemos preocuparnos por la exposición que pueden tener los datos en el trayecto. Para ello se puede hacer uso del protocolo **HTTPS**.

14. Cookies o variables de sesión persistentes

Es necesario tener buen control de las cookies y sesiones y su tiempo de expiración, para evitar vulnerabilidades.

BIBLIOGRAFIA

https://www.seguridad.unam.mx/historico/documento/index.html-id=17

https://www.nerion.es/blog/la-importancia-de-la-seguridad-en-las-aplicaciones-web-como-crear-aplicaciones-web-seguras/

https://www.gb-advisors.com/es/vulnerabilidades-aplicaciones-web/