Seguridad en Sistemas Web

Mikel Egaña Aranguren

mikel-egana-aranguren.github.io

mikel.egana@ehu.eus

Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Seguridad en Sistemas Web

https://doi.org/10.5281/zenodo.4302267

https://github.com/mikel-egana-aranguren/EHU-SGSSI-01

Principales vulnerabilidades en Sistemas Web

Las aplicaciones deben ser seguras

La Open Web Application Security Project (OWASP) analiza las vulnerabilidades más comunes

Informe periódico: OWASP Top Ten (~~2017~~, 2021)

A1 - Inyección

Datos no confiables son enviados como parte de una consulta a un intérprete

Se puede engañar al intérprete y lograr acceso a datos no autorizados

El más conocido SQL Injection

A1:2017-Injection

Testing for SQL injection

A1 - Inyección

Vector	Incidents	Description
1 and 1=2 union select password from qianbo_admin	634,566	Trying to query passwords
1'A=0	125,365	Probing
N;O=D and 1=1 nessus= or 1=1– CONCAT('whs(',')SQLi')	76,155	Probing by vulnerability scanners: Veracode, Nessus and WhiteHat Security, respectively
' union select unhex(hex(version())) —	848,096	Attempting to discover database version
;WAITFOR DELAY '00:00:28';	1,226,955	Blind probing — testing for delay in response

SQL Injection Attacks: So Old, but Still So Relevant. Here's Why (Charts)

A1 - Inyección

¿Como evitarla?

- Validación de las entradas evitando caracteres "peligrosos"
- No usar cuentas con privilegios de administrador
- No proporcionar mayor información de la necesaria (evitar la información de los errores hacia el usuario)
- No construir las sentencias SQL directamente con los valores recogidos, usar sentencias parametrizadas

A1 - Inyección

¿Como evitarla? Filtrar los datos

- Si tenemos en nuestro formulario el campo username, y sabemos que los usuarios sólo pueden estar compuestos por letras y números, no se deben permitir caracteres como " ' " o " = "
- Si se trata del campo e-mail, podemos utilizar expresiones regulares para validarlo, como preg_match('/^.+@.+\..{2,3}$/',$_POST['email'])

A1 - Inyección

¿Como evitarla? Filtrar los datos

- Usar funciones que escapan caracteres especiales de una cadena para su uso en una sentencia SQL: mysql_real_escape_string() coloca barras invertidas antes de los siguientes caracteres: \x00, \n, \r, \, ', " y \x1a
- addslashes(), (la directiva de PHP magic_quotes_gpc está activada por defecto, y ejecuta la función addslashes() en todos los datos GET, POST, y COOKIE).

A2 - Pérdida de autenticación

Aprovecharse de una sesión establecida por un usuario legítimo

Si se usan identificadores de sesión visibles se puede copiar y usarlo a posteriori

Si se almacena el identificador de sesión en una cookie, se puede acceder a él

- A través de un ataque XSS (ver vulnerabilidad A7)
- A través de una escucha en la red

A2 - Pérdida de autenticación

Si la sesión no se cierra se puede acceder a la misma url cuando el usuario legítimo haya abandonado su máquina

A2 - Pérdida de autenticación

Posibles causas

- Uso de passwords sencillos y conocidos (1234, admin, etc.)
- Uso de preguntas de seguridad sencillas o poco configurables
- Las contraseñas no se encriptan con un algoritmo seguro
- Se muestra la contraseña en la url
- Incorrecta gestión de las sesiones

A2 - Pérdida de autenticación

Problemas de las cookies

- Si almacenan los datos de la sesión de manera permanente, cualquiera puede acceder a ellos
- Con un lenguaje de script se puede acceder al fichero y a sus valores mientras se navega por el sitio web
- Si los datos de la cookie se envían sin cifrar, cualquiera puede escucharlos por la red

A2 - Pérdida de autenticación

¿Cómo evitarla?

- Cerrar la sesión cuando el usuario se desloguea
- Cerrar la sesión cuando el usuario pasa cierto tiempo sin hacer nada (timeout)
- Para evitar el acceso mediante lenguajes de Script, usar el atributo **httponly**

A2 - Pérdida de autenticación

';--have i been pwned?

A3 - Exposición de datos sensibles

No tomar medidas de protección para aquellos datos que se consideren sensibles en la aplicación

- Almacenamiento de datos sensibles sin cifrar
- Transmisión de datos sensibles sin cifrar
- Uso de algoritmos de cifrado débiles

A3 - Exposición de datos sensibles

¿Cómo evitarla?

- Almacenando cifrada la información sensible
- Asegurando que la información sensible se envíe a través de protocolos seguros
- Usando algoritmos de cifrado robustos

A4 - Entidades XML externas

Aprovecharse de vulnerabilidades en los procesadores de XML que incluyen muchos servicios web

- Subiendo ficheros XML con código embebido
- Haciendo uso de las dependencias de los ficheros XML

A4 - Entidades XML externas

¿Cómo evitarla?

- Evitar el uso del formato XML (Usar JSON)
- Validar todo fichero XML antes de procesarlo
- Actualizar los procesadores de XML para evitar vulnerabilidades conocidas
- Limitar los orígenes desde los que se pueden enviar ficheros XML

A5 - Rotura de control de acceso

Acceder sin autentificación o con una autentificación no adecuada a objetos

www.sitio.com/consultar_datos.php?dni=45 (Sólo el usuario con DNI= 45 debería poder ver sus datos. No es suficiente con saber si el usuario que quiere acceder está logueado, hay que saber si es el de DNI=45)

A5 - Rotura de control de acceso

¿Cómo evitarla?

- Usando sesiones o cookies

A6 - Configuración de seguridad incorrecta

No tener una buena configuración de la aplicación por desconocimiento/despiste/dejadez

- ¿Se mantienen las cuentas/servicios que se crean por defecto?
- ¿Están habilitadas funcionalidades que no deberían?
- ¿Se trabaja con usuarios con permisos de administrador?
- ¿Se trabaja con software sin actualizar?
- ¿Se permite el uso de malas contraseñas?

A6 - Configuración de seguridad incorrecta

¿Cómo evitarla?

- Eliminar aquellas cuentas de servicios que no se usen
- Modificar las contraseñas que traen por defecto muchos programas
- Deshabilitar puertos/servicios que no se usen
- Actualizar las aplicaciones
- Obligar a los usuarios a usar contraseñas "seguras"

A6 - Configuración de seguridad incorrecta

¿Cómo evitarla?

- Definir usuarios específicos para cada aplicación con los permisos correspondientes a esa aplicación (En MySQL, no usar el usuario root para las conexiones. Crear uno o varios usuarios para la aplicación con los permisos que correspondan sólo sobre las tablas que correspondan)

A7 - Secuencia de comandos en sitios cruzados (XSS)

Ejecutar código en lenguaje Script que pueda introducir el propio usuario

El método más sencillo, a través de campos de formularios

Puede ser persistentes si se almacena el script en una base de datos:

- Se introduce como un campo que se almacena
- Se ejecuta cada vez que se muestra el valor de ese campo

A7 - Secuencia de comandos en sitios cruzados (XSS)

¿Cómo evitarla?

- Limpiando (escapando) el texto que introduzca el usuario eliminando los caracteres “peligrosos”
- Limpiando los valores que obtengamos de la URL eliminando los caracteres “peligrosos”
- Existen funciones que realizan el trabajo de escapado devolviendo un texto “limpio”

A8 - Deserialización insegura

Aprovecharse del proceso de desempaquetado de datos introduciendo datos falsos /maliciosos

Cuando los datos vienen en formatos que los procesan para construir objetos a partir de ellos se pueden introducir datos que generen objetos no deseados

A8 - Deserialización insegura

¿Cómo evitarla?

- Usando formatos que sólo admitan datos en tipos de datos primitivos (string, int, etc.), por ejemplo JSON
- Desempaquetando sólo datos que vengan firmados
- Deserializando únicamente los datos que vengan de orígenes fiables
- Ejecutando el proceso de deserialización de manera externa al resto de la aplicación y en un entorno controlado

A9 - Uso de componentes con vulerabilidades conocidas

¿Cómo evitarla?

- Estando al día de la publicación de vulnerabilidades
- Aplicando los parches correspondientes en cuanto salgan
- Buscando alternativas que no presenten vulnerabilidades

A10 - Logueo y monitorización insuficientes

No registrar los intentos de acceso tanto si tienen éxito como si no

La información de los intentos de acceso puede ser muy valiosa

- Para detectar conductas sospechosas (empleados accediendo al sistema cuando no deben)
- Para detectar ataques (múltiples intentos fallidos con el mismo usuario)
- En algunos casos es obligatorio almacenar esta información debido a la LOPD

A10 - Logueo y monitorización insuficientes

¿Cómo se soluciona?

- Almacenando la información referente a las conexiones exitosas y a las fallidas
- Analizando la información sobre las conexiones
- Bloqueando al usuario si tiene X intentos fallidos consecutivos