OWASP

OWASP

- Introducción
- Metodología
- Para qué y para quién son
- Comprobación
- Metodología de pruebas
- Top 10

OWASP

Introducción

- Vamos a conocer la manera de explotar las vulnerabilidades de diferentes formas.
- Siempre se separa o se distingue de forma diferente al programador web, que será el que realice la página web, del que gestiona dicha página web bajo un sistema, el cuál suele ser el administrador de sistemas. Esto tiene siempre una consecuencia, y es que si el administrador de sistemas no está al día sobre las herramientas o vulnerabilidades de los sistemas web, no podrá aplicar las contramedidas necesarias.

OWASP

Metodología

"Hace referencia al conjunto de procedimientos basados en principios lógicos, utilizados para alcanzar una gama de objetivos que rigen en una investigación científica o en una exposición doctrinal."

- Metodología OWASP: *"OWASP (acrónimo de Open Web Application Security Project, en inglés 'Proyecto de seguridad de aplicaciones web abiertas') es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.*

OWASP

Metodología

- OWASP es un nuevo tipo de entidad en el mercado de seguridad informática. Estar **libre de presiones corporativas** facilita que OWASP proporcione información imparcial, práctica y reeditable sobre seguridad de aplicaciones informáticas.

OWASP

Para quién

- **Desarrolladores de Software.** Necesita usar la guía para tener la certeza de que el código que se entrega no es vulnerable.
- **Testers de Software.** Se debe usar esta guía para mejorar sus habilidades.
- **Especialistas de Seguridad.** Tienen una responsabilidad especial, cerciorarse de que las aplicaciones no se publiquen con vulnerabilidades

OWASP

Guías de pruebas

- La guía de pruebas de OWASP **cubre los procedimientos y herramientas para probar la seguridad de las aplicaciones**. La mejor manera de usar esta guía es como parte de una verificación exhaustiva de la seguridad en las mismas.

OWASP

Herramientas automatizadas

- Existen varias compañías que **comercializan herramientas de análisis y comprobación de seguridad automatizadas**.
- Debemos tener presente las **limitaciones de estas herramientas**, de modo que se pueda emplear para aquello en lo que son más útiles. **¡Las herramientas no hacen al software seguro!**, ayudan a reducir el proceso e imponer la política de seguridad.
- Lo más importante y destacable es que **estas herramientas son genéricas** - es decir, que no están diseñadas para un código específico, sino para aplicaciones en general. Lo que significa que aunque **pueden encontrar algunos problemas genéricos, no tienen el conocimiento suficiente** sobre la aplicación como para permitirles detectar la mayoría de los fallos.

OWASP

Comprobación

- Durante el ciclo de vida del desarrollo de una aplicación web, **muchas cosas han de ser probadas**.
- La comprobación o testing **es un proceso de comparación de un estado ante un conjunto de criterios basados**, en nuestro caso, en las **vulnerabilidades**.
- La mayor parte de la gente no comprueba el software **hasta que ya ha sido creado y se encuentra en la fase de desarrollo de su ciclo de vida**. Esta es generalmente una práctica muy ineficaz y costosa.

OWASP

Comprobación - Principios

- **No existe la bala de plata:** aunque es tentador pensar que un scanner de seguridad o un cortafuegos de aplicación nos proporcionará una multitud de defensas o identificará una multitud de problemas, en realidad **no existen balas de plata para el problema del software inseguro**.
- El **software de evaluación de seguridad de aplicaciones**, aunque útil como un primer paso, para el descubrimiento de vulnerabilidades, **generalmente es inefectivo para las evaluaciones en profundidad**.

OWASP

Comprobación - Principios

- **Piensa estratégicamente, no tácticamente:** En los últimos años, los profesionales de la seguridad han llegado a darse cuenta de la falacia que representa **el modelo de parchear y penetrar**, generalizado en seguridad de la información durante los 90. El modelo de parchear y penetrar comprende corregir un bug reportado, pero **sin la investigación adecuada de la causa origen**.

OWASP

Comprobación - Técnicas

- **Inspecciones y Revisiones Manuales:** las inspecciones manuales son revisiones realizadas por personas.
- **Modelado de Amenazas:** el modelado de amenazas es esencialmente la evaluación del riesgo en aplicaciones.
- **Revisión de Código:** la revisión de código fuente es el proceso de comprobar manualmente el código fuente de una aplicación web en busca de incidencias de seguridad. También se puede automatizar
- **Pruebas de Intrusión:** conocidos comúnmente como **pruebas de pentest o hacking ético**. Las pruebas de intrusión son esencialmente el "arte" de comprobar una aplicación en ejecución remota, sin saber el funcionamiento interno de la aplicación, para encontrar vulnerabilidades de seguridad.

OWASP

Qué es una prueba de pentest o intrusión

- Una prueba de intrusión **es un método de evaluación de la seguridad de un sistema de ordenadores o una red mediante la simulación de un ataque**. Una prueba de intrusión de aplicación web está enfocada solamente a **evaluar la seguridad de una aplicación web**.
- El proceso conlleva un **análisis pasivo y activo de la aplicación** en busca de cualquier debilidad, fallos técnicos o vulnerabilidades.
- Cualquier incidencia de seguridad que sea encontrada **será presentada al propietario del sistema**, junto con una evaluación de su impacto, y a menudo con una propuesta para su mitigación o una solución técnica.
- **Dos modos** de realizer las pruebas: **activo y pasivo**.

OWASP

Pentest web – Pasivo

- La persona a cargo de la realización de las pruebas **intenta comprender la lógica de la aplicación**, juega con la aplicación; puede usarse una utilidad para la recopilación de información, como un **proxy HTTP**, para observar todas **las peticiones y respuestas HTTP**. Al final de esta fase esta persona **debería comprender cuales son todos los puntos de acceso**: puertas de la aplicación, cabeceras HTTP, parámetros y cookies.

OWASP

Pentest web - Activo

- **Modo activo:** en esta fase la persona a cargo de la comprobación empieza a realizar las pruebas usando la metodología descrita posteriormente. Se divide el conjunto de pruebas en 8 subcategorías:
  - Pruebas de gestión de la configuración
  - Pruebas de la lógica de negocio
  - Pruebas de Autenticación
  - Pruebas de Autorización
  - Pruebas de gestión de sesiones
  - Pruebas de validación de datos
  - Pruebas de denegación de Servicio
  - Pruebas de Servicios Web

OWASP TOP 10

Top 10 OWASP

https://owasp.org/www-project-top-ten/

Recomiendan el uso del **Top 10 para obtener seguridad** en las aplicaciones antes de su implementación en producción.

OWASP TOP 10

Top 10 OWASP

- A1 - Inyección
- A2 - Pérdida de Autenticación y Gestión de Sesiones
- A3 - Secuencia de Comandos en Sitios Cruzados {XSS)
- A4 - Referencia Directa Insegura a Objetos
- A5 - Configuración de Seguridad Incorrecta
- A6 - Exposición de Datos Sensibles
- A7 - Ausencia de Control de Acceso a las Funciones
- A8 - Falsificación de Peticiones en Sitios Cruzados {CSRF)
- A9 - Uso de Componentes con Vulnerabilidades Conocidas
- A10 - Redirecciones y reenvíos no validos

OWASP TOP 10

A1 – Inyección

Los fallos de inyección, tales como SQL, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos maliciosos del atacante **pueden engañar al intérprete en ejecutar comandos no intencionados o acceder a datos no autorizados**.

OWASP TOP 10

A1 – Inyección

¿Soy vulnerable? La mejor manera de averiguar si una aplicación es vulnerable a una inyección es verificar que en todo uso de intérpretes se separa la información no confiable del comando o consulta.

¿Cómo prevenirlo?

1. **Usando una API** la cuál evite el uso de intérpretes por completo o provea una interfaz parametrizada.
2. Si una API parametrizada no está disponible, **debe codificar cuidadosamente los caracteres especiales**, usando la sintaxis de escape específica del intérprete
3. La evaluación de **entradas positivas o de "listas blancas"** también se recomienda, pero no es una defensa integral dado que muchas aplicaciones requieren caracteres especiales en sus entradas.

OWASP TOP 10

A2 - Pérdida de Autenticación y Gestión de Sesiones

Las funciones de la aplicación relacionadas con autenticación y gestión de sesiones **son frecuentemente implementadas de forma incorrecta**, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otros fallos de implementación para asumir la identidad de otros usuarios.

OWASP TOP 10

A2 - Pérdida de Autenticación y Gestión de Sesiones

¿Soy vulnerable? Si…

1. Las credenciales de los usuarios **no están protegidas cuando se almacenan**, utilizando un hash o cifrado.
2. **Se pueden adivinar o sobrescribir las credenciales** a través de funciones débiles de gestión de la sesión.
3. Los **ID de sesión son expuestos en la URL**.
4. Los **ID de sesión son vulnerables a ataques de fijación de la sesión**.
5. Los **ID de sesiones no expiran**, ni las sesiones de usuario ni los tokens de autenticación. En particular, los tokens de inicio de sesión única {SSO), no son invalidados durante el cierre de sesión.
6. Las contraseñas, ID de sesión y otras **credenciales son transmitidas a través de canales no cifrados.**

OWASP TOP 10

A2 - Pérdida de Autenticación y Gestión de Sesiones

¿Cómo prevenirlo?

La recomendación principal de una organización es facilitar a los desarrolladores un **único conjunto de controles de autenticación y gestión de sesiones** fuertes. Dichos controles deberán conseguir:

1. **Cumplir con todos los requisitos de autenticación y gestión de sesiones** definidos en el Application Security Verification Standard (ASVS) de OWASP, secciones V2 (Autenticación) y V3(Gestión de sesiones).
2. **Tener un interfaz simple para los desarrolladores**. Considerar el uso de ESAPI Authenticator y las APIs de usuario como buenos ejemplos a seguir, utilizar o sobre los que construir.

- Se debe realizar un gran esfuerzo en **evitar vulnerabilidades de XSS** que podrían ser utilizadas para robar ID de sesión.

OWASP TOP 10

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

- Es un tipo vulnerabilidad típica de las aplicaciones Web, que permite a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar (ej: VBScript), evitando medidas de control como la Política del mismo origen.

OWASP TOP 10

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

¿Soy Vulnerable?

- Es vulnerable **si no asegura que todas las entradas de datos ingresados por los usuarios son codificadas adecuadamente**; o si no se verifica en el momento de ingreso que los datos sean seguros antes de ser incluidos en la página de salida. Sin la codificación o validación debida, dicha entrada será tratada como contenido activo del navegador.

OWASP TOP 10

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

¿Cómo prevenirlo?

- La opción preferida es codificar los datos no confiables basados en el contexto HTML donde serán ubicados.
- Para contenido en formato enriquecido, se recomienda utilizar bibliotecas de auto sanitización como AntiSamv de OWASP o el proyecto sanitizador de HTML en Java.
- Se debe considerar utilizar políticas de seguridad de contenido para defender contra XSS la totalidad de su sitio.

OWASP TOP 10

A4 - Referencia Directa Insegura a Objetos

Una referencia directa a objetos ocurre **cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos**. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder a datos no autorizados.

OWASP TOP 10

A4 - Referencia Directa Insegura a Objetos

¿Soy Vulnerable?

- La mejor manera de poder comprobar si una aplicación es vulnerable a las referencias inseguras a objetos, es **verificar que todas las referencias a objetos tienen las protecciones apropiadas**. Para conseguir esto, debemos considerar:
  - Para referencias **directas** a recursos **restringidos**, la aplicación necesitaría verificar si el usuario está autorizado a acceder al recurso en concreto que solicita.
  - Si la referencia es una referencia **indirecta**, la correspondencia con la directa debe ser limitada a valores autorizados para el usuario en concreto.

OWASP TOP 10

A4 - Referencia Directa Insegura a Objetos

¿Cómo prevenirlo?

- Requiere seleccionar una forma de proteger los objetos accesibles por cada usuario.
- **Utilizar referencias indirectas por usuario o sesión.** La aplicación tendría que realizar la correlación entre la referencia indirecta con la clave de la base de datos correspondiente en el servidor. ESAPI de OWASP incluye relaciones tanto secuenciales como aleatorias de referencias de acceso que los desarrolladores pueden utilizar para eliminar las referencias directas a objetos.
- **Comprobar el acceso**. Cada uso de una referencia directa a un objeto de una fuente que no es de confianza debe incluir una comprobación de control de acceso para asegurar que el usuario está autorizado a acceder al objeto solicitado.

OWASP TOP 10

A5 - Configuración de Seguridad Incorrecta

- Una buena seguridad requiere tener **definida e implementada una configuración segura para la aplicación**, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataformas. Todas estas configuraciones deben ser definidas, como implementadas y mantenidas, ya que por lo general **no son seguras por defecto**. Esto incluye mantener todo el software actualizado, incluida las librerías de código utilizadas por la aplicación.

OWASP TOP 10

A5 - Configuración de Seguridad Incorrecta

¿Soy Vulnerable?

- ¿Tiene algún **software sin actualizar**? Esto incluye el SO, Servidor Web/Aplicaciones, SGBD, aplicaciones, y todas las librerías de código.
- ¿Están habilitadas o **instaladas alguna característica innecesaria**?
- ¿Están las **cuentas por defecto y sus contraseñas** aún habilitadas y sin cambiar?
- ¿El **control de de errores revela rastros de las capas de aplicación** u otros mensajes de error demasiado informativos para los usuarios?
- ¿Están las **configuraciones de seguridad en su framework de desarrollo y librerías sin configurar a valores seguros**?

OWASP TOP 10

A5 - Configuración de Seguridad Incorrecta

¿Cómo prevenirlo?

- Crear un proceso para mantener y desplegar las nuevas actualizaciones y parches de software de una manera oportuna para cada entorno. Debe incluir todas las librerías de código.
- Se debe considerar ejecutar escaneos y realizar auditorías periódicamente para ayudar a detectar fallos de configuración o parches omitidos.

OWASP TOP 10

A6 - Exposición de Datos Sensibles

- **Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjeta de crédito o credenciales de autenticación**. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el **cifrado de datos**.

OWASP TOP 10

A6 - Exposición de Datos Sensibles

¿Soy vulnerable?

- Lo primero que se debe determinar es **el conjunto de datos sensibles que requieran protección extra**. Por ejemplo, contraseñas, números de tarjetas de crédito, registros médicos, e información personal. Para estos datos:
- ¿Se almacenan en **texto claro a largo plazo**, incluyendo sus respaldos?
- ¿Se **transmite en texto claro, interna o externamente**? El tráfico por Internet es especialmente peligroso.
- ¿Se utiliza algún **algoritmo criptográfico débil/antiguo**?
- ¿Se **generan claves criptográficas débiles**, o falta una adecuada rotación o gestión de claves?

OWASP TOP 10

A6 - Exposición de Datos Sensibles

¿Cómo prevenirlo?

- Cifrar los datos sensibles almacenados o en tráfico.
- No deben almacenarse **datos sensibles** innecesariamente. **Datos que no se poseen no pueden ser robados.**
- Asegurarse de **aplicar algoritmos de cifrado fuertes y estándar**, así como claves fuertes y gestionarlas de forma segura.

OWASP TOP 10

A7 - Ausencia de Control de Acceso a las Funciones

- La mayoría de aplicaciones Web **verifican los derechos de acceso a nivel de función antes de hacer visible** la interfaz de usuario. A pesar de esto, las aplicaciones **necesitan verificar el control de acceso en el servidor** cuando se accede a cada función. Si las solicitudes de acceso **no se verifican**, **los atacantes podrán realizar peticiones** sin la autorización apropiada.

OWASP TOP 10

A7 - Ausencia de Control de Acceso a las Funciones

¿Soy Vulnerable?

- La mejor manera de determinar si una aplicación falla en restringir adecuadamente el acceso a nivel de funcionalidades es verificar cada funcionalidad de la aplicación:
- ¿La interfaz de usuario **muestra la navegación hacia funcionalidades no autorizadas**?
- ¿Existe **autenticación del lado del servidor**, o se han perdido las comprobaciones de autorización?
- ¿Los controles del lado del servidor se basan exclusivamente en la información proporcionada por el atacante?

OWASP TOP 10

A7 - Ausencia de Control de Acceso a las Funciones

¿Cómo prevenirlo?

- La aplicación debería tener un módulo de autorización consistente y fácil de analizar, apelando desde todas las funciones de negocio. Frecuentemente, esa protección es provista por uno o más componentes externos al código de la aplicación.

OWASP TOP 10

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

- Un ataque CSRF **obliga al navegador de una víctima autenticada a enviar una petición HTTP manipulada**, incluyendo la **sesión del usuario y cualquier otra información de autenticación** incluida automáticamente, a una aplicación web vulnerable. Esto **permite al atacante forzar al navegador de la víctima para generar** peticiones que la aplicación vulnerable considera como legítimas provenientes de la víctima.
- Deben **verificarse las operaciones de múltiples pasos**, ya que no son inmunes a este tipo de ataque. Los atacantes pueden falsificar fácilmente una serie de solicitudes mediante el uso de etiquetas o incluso de código javascript.

OWASP TOP 10

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

OWASP TOP 10

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

¿Soy Vulnerable?

- Token impredecible en los enlaces del formulario o entre los pasos de estos.
- Una defensa interesante puede ser la de requerir que el usuario demuestre su intención de enviar la solicitud, ya sea a través de la re-autenticación, o mediante cualquier otra prueba que demuestre que se trata de un usuario real (Por ejemplo con CAPTCHA).

OWASP TOP 10

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

¿Cómo prevenirlo?

- La prevención CSRF por lo general requiere la inclusión de un token no predecible en cada solicitud HTTP. Estos tokens deben ser, como mínimo, únicos por cada sesión del usuario.
- La opción preferida es incluir **el token único en un campo oculto**. Esto hace que el valor de dicho campo se envíe en el cuerpo de la solicitud HTTP, evitando su inclusión en la URL, sujeta a mayor exposición.
- CSRF Guard de OWASP puede incluir automáticamente los tokens secretos en JAVA EE,.NET, aplicaciones PHP. Por otro lado, ESAPI de OWASP incluye también métodos para que los desarrolladores puedan utilizar con tal de evitar este tipo de vulnerabilidades.

OWASP TOP 10

A9 - Uso de Componentes con Vulnerabilidades Conocidas

- Algunos componentes tales como los sistemas operativos, aplicaciones, librerías, los frameworks y otros módulos de software **casi siempre funcionan con todos los privilegios**. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida seria de datos.

OWASP TOP 10

A9 - Uso de Componentes con Vulnerabilidades Conocidas

¿Soy vulnerable?

- Para determinar si es vulnerable necesita buscar en **bases de datos, así como también mantenerse al tanto de la listas de correos del proyecto y anuncios** de cualquier tema que pueda ser relativo una vulnerabilidad.
- Si uno de sus componentes tiene una vulnerabilidad, **debe evaluar cuidadosamente si es o no vulnerable, revisando si su código utiliza la parte del componente vulnerable** y si el fallo puede resultar en un impacto del cual cuidarse.

OWASP TOP 10

A9 - Uso de Componentes con Vulnerabilidades Conocidas

¿Cómo prevenirlo?

- Identificar todos los componentes y la versión, incluyendo dependencias.
- Revisar la seguridad del componente en bases de datos públicas, listas de correos del proyecto, y listas de de seguridad y mantenerlos actualizados.
- Establecer políticas de seguridad que regulen el uso de componentes, como requerir ciertas prácticas en el desarrollo de software y pasar test de seguridad (Fortify).

OWASP TOP 10

A10 - Redirecciones y reenvíos no validos

Las aplicaciones web **frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web**, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder a páginas no autorizadas

OWASP TOP 10

A10 - Redirecciones y reenvíos no validos

¿Soy vulnerable?

- **Revisar el código para detectar el uso de redirecciones** o reenvíos. Para cada uso, identificar si la URL objetivo se incluye en el valor de algún parámetro. Si es así, la URL objetivo no es validada con una lista blanca, usted es vulnerable.
- Además, recorrer la aplicación para observar si genera cualquier redirección (códigos de respuesta HTTP 300-308, **típicamente 302**). **Analizar los parámetros facilitados antes de la redirección para ver si parecen ser una URL** de destino o un recurso de dicha URL. Si es así, modificar la URL de destino y observar si la aplicación redirige al nuevo destino.
- Si el código no se encuentra disponible, **se deben analizar todos los parámetros para ver si forman parte de una redirección** o reenvío de una URL de destino y probar lo que hacen estos.

OWASP TOP 10

A10 - Redirecciones y reenvíos no validos

¿Cómo evitarlo?

- Simplemente **evitando el uso de redirecciones y reenvíos**.
- Si se utiliza, no involucrar parámetros manipulables por el usuario para definir el destino.
- Si los parámetros de destino no pueden ser evitados, **asegúrese que el valor suministrado sea válido y autorizado para los usuarios**.