Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Licenciatura en Desarrollo y Gestión de Software
Asignatura: Desarrollo de Software VII

Laboratorio #2

Facilitadora:
Irina Fong

Pertenece a:
Analía Solís
Joseph Guerrero

Cedula:
2-754-1867

Correo:
analia.solis@utp.ac.pa
joseph.guerrero2@utp.ac.pa

Unidad I

Grupo:
1GS131

Fecha:
30 de Mayo del 2025

I Semestre
2025

Indice

Introducción

En la era digital actual, donde la mayoría de los servicios e interacciones se realizan a través de plataformas web, la seguridad de las aplicaciones se ha convertido en un aspecto crítico e ineludible. Cada día, miles de aplicaciones manejan datos personales, financieros y empresariales que, si no están adecuadamente protegidos, pueden caer fácilmente en manos equivocadas.

Para ayudar a los equipos de desarrollo, operaciones y seguridad a entender y mitigar los riesgos más comunes, la organización OWASP (Open Web Application Security Project) elabora y actualiza periódicamente una lista conocida como el OWASP Top 10. Esta lista recopila los diez riesgos de seguridad más críticos que afectan a las aplicaciones web, basada en datos reales recopilados a nivel global y en el análisis de expertos en ciberseguridad.

El objetivo de esta charla es presentar cada uno de estos riesgos, entender por qué son peligrosos, cómo se manifiestan en escenarios reales y qué medidas se pueden tomar para prevenirlos o mitigarlos. No se trata solo de conocer vulnerabilidades, sino de construir una cultura de seguridad desde la raíz: el diseño, el desarrollo y el despliegue de software seguro.

1. Broken Access Control (Control de Acceso Roto)

A. ¿Qué es?

Este riesgo se refiere a fallos en la aplicación que permiten a los usuarios acceder a recursos o realizar acciones fuera de su nivel de autorización. Muchas veces, esto se debe a que el control de acceso se aplica solo en el frontend o no se implementa de manera centralizada.

B. ¿Cuál es su importancia?

Si no se controla adecuadamente quién puede hacer qué dentro de una aplicación, cualquier usuario podría obtener acceso a información o funcionalidades críticas, lo que puede llevar al robo de datos, manipulación de cuentas u operaciones maliciosas.

C. Ejemplos

- Un usuario modifica la URL de un endpoint (/usuario/123) para acceder a la cuenta de otro (/usuario/124).
- Un cliente realiza una petición directamente al backend para realizar funciones administrativas, como eliminar usuarios, sin permisos adecuados.

2. Cryptographic Failures (Fallos Criptográficos)

A. ¿Qué es?

Este riesgo implica una implementación deficiente de la criptografía: ya sea por no cifrar los datos sensibles, usar algoritmos inseguros o manejar mal las claves de cifrado.

B. ¿Cuál es su importancia?

La criptografía protege datos sensibles como contraseñas, tarjetas de crédito y tokens de sesión. Su ausencia o mal uso puede permitir que un atacante intercepte, lea o modifique información crítica.

C. Ejemplos

- Enviar credenciales por HTTP en lugar de HTTPS, permitiendo que sean capturadas por un atacante con un sniffer.
- Almacenar contraseñas en texto plano o con algoritmos débiles como MD5.

3. Injection (Inyecciones)

A. ¿Qué es?

Ocurre cuando los datos introducidos por el usuario se interpretan como código en una consulta o comando. Esto puede alterar la lógica de una aplicación o permitir la ejecución de comandos arbitrarios.

B. ¿Cuál es su importancia?

Las inyecciones pueden llevar al acceso no autorizado a datos, pérdida de integridad de los sistemas e incluso control total del servidor afectado.

C. Ejemplos

- Un formulario de inicio de sesión que permite una inyección SQL como admin' OR '1'='1.
- Comandos del sistema operativo inyectados a través de formularios que se ejecutan en shell.

4. Insecure Design (Diseño Inseguro)

A. ¿Qué es?

Este riesgo se refiere a decisiones de diseño que no contemplan adecuadamente la seguridad.

B. ¿Cuál es su importancia?

Cuando la seguridad no es parte del diseño, los sistemas quedan vulnerables a ataques incluso si el código está bien implementado. Corregir estos problemas requiere cambios estructurales costosos.

C. Ejemplos

- No definir límites de uso por usuario (rate limiting), permitiendo ataques de denegación de servicio.
- Ausencia de modelos de amenazas durante el diseño de arquitecturas.

5. Security Misconfiguration (Configuración Insegura)

A. ¿Qué es?

Este riesgo ocurre cuando el software se instala, configura o despliega con valores por defecto, servicios innecesarios activos, errores de configuración o falta de parches.

B. ¿Cuál es su importancia?

Muchas brechas de seguridad ocurren no por fallos del código, sino por errores en la configuración del entorno de ejecución.

C. Ejemplos

- Servidores que exponen consolas de administración sin autenticación.
- Aplicaciones que muestran mensajes de error detallados con información interna del sistema.

6. Vulnerable and Outdated Components (Componentes Vulnerables y Obsoletos)

A. ¿Qué es?

Consiste en utilizar librerías, frameworks o dependencias que tienen vulnerabilidades conocidas y no han sido actualizadas.

B. ¿Cuál es su importancia?

Las aplicaciones modernas dependen fuertemente de componentes de terceros. Si estos no se mantienen al día, introducen vectores de ataque sin necesidad de que el código propio tenga errores.

C. Ejemplos

- Uso de versiones antiguas de Log4j con vulnerabilidades explotables (Log4Shell).
- Un frontend con jQuery obsoleto vulnerable a XSS.

7. Identification and Authentication Failures (Fallos de Identificación y Autenticación)

A. ¿Qué es?

Este riesgo agrupa todos los problemas relacionados con la gestión de identidades: autenticación débil, sesiones mal manejadas y ausencia de protección contra ataques de fuerza bruta.

B. ¿Cuál es su importancia?

Si un atacante puede hacerse pasar por otro usuario, especialmente por un administrador, puede comprometer completamente la aplicación.

C. Ejemplos

- Permitir múltiples intentos de inicio de sesión sin bloqueo de cuenta o CAPTCHA.
- Usar tokens de sesión predecibles o sin expiración.

8. Software and Data Integrity Failures (Fallos de Integridad de Software y Datos)

A. ¿Qué es?

Se refiere a situaciones donde no se verifica la integridad de software o datos críticos, permitiendo que estos sean manipulados por atacantes.

B. ¿Cuál es su importancia?

La integridad asegura que lo que se ejecuta o almacena no ha sido alterado maliciosamente. Si no se controla, se puede introducir código malicioso incluso en procesos automáticos como CI/CD.

C. Ejemplos

- Implementar una actualización automática de software desde una fuente no verificada.
- Inyectar código malicioso en un repositorio interno sin controles.

9. Security Logging and Monitoring Failures (Fallos de Registro y Monitoreo de Seguridad)

A. ¿Qué es?

Este riesgo representa la ausencia de registros adecuados y la falta de monitoreo para detectar e investigar actividades maliciosas.

B. ¿Cuál es su importancia?

Sin registros, los incidentes pueden pasar desapercibidos. Sin monitoreo, la respuesta es lenta o inexistente, y se dificulta el análisis forense tras un ataque.

C. Ejemplos

- No registrar intentos fallidos de acceso o cambios críticos en cuentas.
- No alertar cuando hay comportamientos anómalos como múltiples inicios de sesión simultáneos.

10. Server-Side Request Forgery (SSRF)

A. ¿Qué es?

Este riesgo ocurre cuando una aplicación permite que el servidor realice solicitudes HTTP arbitrarias a otras direcciones, muchas veces internas, a petición del usuario.

B. ¿Cuál es su importancia?

Puede ser explotado para escanear redes internas, acceder a servicios protegidos o extraer información confidencial.

C. Ejemplos

- Un formulario que permite subir una URL, y el servidor intenta descargar el archivo sin validar el destino.
- Uso de SSRF para acceder a la metadata de una instancia en la nube (http://169.254.169.254 en AWS).

Conclusión

Como hemos visto a lo largo de esta presentación, los riesgos identificados por el OWASP Top 10 no son amenazas abstractas ni improbables; son vulnerabilidades reales, recurrentes y con consecuencias graves cuando no se abordan adecuadamente. Desde fallos en el control de acceso hasta errores de configuración, pasando por la falta de criptografía o el uso de componentes desactualizados, todos estos problemas pueden ser explotados por atacantes con habilidades básicas o automatizadas.

La seguridad en aplicaciones web no es responsabilidad exclusiva del equipo de seguridad; es un esfuerzo colaborativo que comienza desde el diseño y continúa hasta el mantenimiento en producción. El OWASP Top 10 debe entenderse no solo como una lista de errores a evitar, sino como una guía práctica para fomentar buenas prácticas, promover la formación continua y reforzar la seguridad como un valor central en todo el ciclo de vida del software.

Adoptar una mentalidad de desarrollo seguro —desde el primer boceto hasta la última línea de código desplegada— es la mejor defensa ante un panorama de amenazas cada vez más complejo. Conocer y aplicar los principios del OWASP Top 10 es, sin duda, un paso esencial en esa dirección.

Bibliografía

1. OWASP Foundation. (2021). *OWASP Top Ten: The ten most critical web application security risks*. https://owasp.org/Top10/
2. OWASP Foundation. (n.d.). *OWASP Cheat Sheet Series*. https://cheatsheetseries.owasp.org/
3. Stuttard, D., & Pinto, M. (2011). *The web application hacker's handbook: Finding and exploiting security flaws* (2nd ed.). Wiley.
4. Howard, M., & LeBlanc, D. (2002). *Writing secure code* (2nd ed.). Microsoft Press.
5. Mozilla Foundation. (n.d.). *Web security guidelines*. https://infosec.mozilla.org/guidelines/web_security.html
6. National Institute of Standards and Technology. (2021). *Secure software development framework (SSDF) version 1.1 (SP 800-218)*. U.S. Department of Commerce. https://csrc.nist.gov/publications/detail/sp/800-218/final