**Universidad Mariano Gálvez de Guatemala**

**Ingeniería en Sistemas**

**Curso: Aseguramiento De La Calidad De Software**

**Ing. Carmelo Estuardo Mayén Monterroso**

# Contenido:

**Guía OWASP, Investigación Planes y Casos de prueba**

Estudiante:

Jeremy Saul Rodriguez Garcia

Carnet:

1790-20-20725

01 de Agosto de

**Guía Básica para evitar los errores de seguridad más comunes en sitios web**

**Basada en OWASP Top 10 – 2021**

**Dirigida a personas sin conocimientos técnicos**

Esta guía tiene como objetivo explicar, de forma fácil de entender, cuáles son los problemas de seguridad más frecuentes en sitios web o aplicaciones en línea. También se explica cómo se pueden evitar de forma práctica, sin necesidad de ser un experto en computadoras.

## Introducción

Hoy en día, muchas personas usan aplicaciones web todos los días, como redes sociales, tiendas en línea, bancos, sistemas escolares o plataformas de trabajo. Aunque parezcan seguras, estos sitios pueden tener errores que permiten que alguien haga cosas indebidas, como robar información, entrar sin permiso o dañar el sistema.

A estos errores se les llama vulnerabilidades, y son muy importantes porque pueden afectar tanto a los dueños del sitio como a las personas que lo usan. Por eso, una organización llamada OWASP (Proyecto Abierto de Seguridad en Aplicaciones Web, por sus siglas en inglés) creó una lista con los 10 problemas más comunes y peligrosos que se deben evitar.

Esta guía explica de forma clara y sencilla cuáles son esas vulnerabilidades, qué significan y cómo se pueden evitar, sin necesidad de tener conocimientos técnicos.

## 1. Accesos sin permiso (Broken Access Control)

### ¿Qué es?

Es cuando alguien puede entrar a zonas del sistema que no debería. Por ejemplo, si un estudiante entra al sistema de notas y puede ver o cambiar las calificaciones de otros, eso es un acceso sin permiso.

### ¿Cómo prevenirlo?

- A cada persona se le debe dar acceso solo a lo que necesita.
- Es importante ocultar las funciones o botones que no todos pueden usar.
- Se deben hacer pruebas con diferentes tipos de usuarios para asegurarse de que no puedan ver ni hacer lo que no les corresponde.

## 2. Protección débil de los datos (Cryptographic Failures)

### ¿Qué es?

Sucede cuando la información importante, como contraseñas, direcciones o números de tarjeta, no está bien protegida y puede ser vista o robada.

### ¿Cómo prevenirlo?

- Proteger los datos importantes para que nadie los pueda leer fácilmente.
- Usar sitios con conexión segura (los que empiezan con "https").

- No compartir información personal en páginas que no sean confiables.

**3. Datos maliciosos (Injection)**

**¿Qué es?**
Ocurre cuando alguien escribe algo extraño o peligroso (como comandos o códigos) en un formulario del sitio para tratar de dañarlo o hacer que funcione de forma incorrecta.

**¿Cómo prevenirlo?**

- Revisar lo que las personas escriben antes de que el sistema lo use.
- No dejar que lo escrito se ejecute automáticamente.
- Evitar que se puedan ingresar símbolos raros que puedan causar problemas.

**4. Falta de seguridad en el diseño (Insecure Design)**

**¿Qué es?**
Es cuando el sistema fue creado sin pensar bien en la seguridad desde el principio, lo que lo hace más fácil de atacar o fallar.

**¿Cómo prevenirlo?**

- Planear la seguridad desde el momento en que se empieza a crear el sistema.
- Imaginar posibles problemas y pensar en cómo evitarlos antes de construir.
- Revisar el diseño con personas que sepan de riesgos para prevenir errores.

**5. Mala configuración del sistema (Security Misconfiguration)**

**¿Qué es?**
Pasa cuando el sistema tiene ajustes equivocados, como dejar funciones activas que no se usan, no cambiar contraseñas por defecto o mostrar información que debería estar oculta.

**¿Cómo prevenirlo?**

- Desactivar lo que no se necesita.
- Cambiar las contraseñas por otras seguras al instalar cualquier sistema.
- Hacer revisiones frecuentes para asegurarse de que todo está bien configurado.

**6. Uso de herramientas viejas o inseguras (Vulnerable and Outdated Components)**

**¿Qué es?**
Sucede cuando el sistema usa programas antiguos que ya tienen fallas conocidas y no han sido corregidas.

**¿Cómo prevenirlo?**

- Mantener todos los programas y sistemas actualizados.
- No usar herramientas que ya no tengan soporte o mejoras.
- Revisar si hay nuevas versiones disponibles y aplicarlas.

**7. Problemas con el inicio de sesión (Identification and Authentication Failures)**

**¿Qué es?**
Pasa cuando alguien puede entrar al sistema con contraseñas fáciles o incluso sin tener permiso, porque el inicio de sesión no es seguro.

**¿Cómo prevenirlo?**

- Pedir contraseñas fuertes, que combinen letras, números y símbolos.
- Activar un segundo paso para iniciar sesión (por ejemplo, un código al celular).
- Cerrar la sesión automáticamente si no se usa durante un tiempo.

**8. Cambios sin revisar (Software and Data Integrity Failures)**

**¿Qué es?**
Es cuando se hacen cambios en el sistema, como instalar programas o actualizar archivos, sin asegurarse de que vienen de fuentes seguras.

**¿Cómo prevenirlo?**

- Descargar programas o actualizaciones solo desde lugares confiables.
- Verificar que los archivos no hayan sido modificados por personas no autorizadas.
- Controlar quién puede hacer cambios importantes en el sistema.

**9. Falta de vigilancia o registros (Security Logging and Monitoring Failures)**

**¿Qué es?**
Cuando el sistema no guarda lo que pasa o no avisa si algo extraño ocurre, es difícil saber si hubo un ataque o un error.

**¿Cómo prevenirlo?**

- Activar registros que guarden las acciones del sistema.
- Revisar esos registros con regularidad.
- Configurar alertas que avisen si pasa algo raro.

**10. Conexiones no controladas (Server-Side Request Forgery – SSRF)**

**¿Qué es?**
Es cuando alguien engaña al sistema para que se conecte a lugares internos o prohibidos, lo que puede ponerlo en peligro.

**¿Cómo prevenirlo?**

- Controlar a qué sitios o direcciones se puede conectar el sistema.
- No permitir que los usuarios escriban direcciones libremente sin revisión.
- Solo permitir conexiones a sitios previamente aprobados.

- **Glosario de términos clave**

| **Término** | **Explicación sencilla** |
|---|---|
| **Aplicación web / Sitio web** | Programa que se usa desde un navegador de internet, como una tienda en línea, un sistema escolar o una red social. |
| **Vulnerabilidad** | Es un error o debilidad en un sistema que puede ser aprovechado por alguien para hacer daño o acceder sin permiso. |
| **Formulario** | Parte del sitio donde se escriben datos, como cuando uno llena su nombre y contraseña. |
| **Contraseña segura** | Clave difícil de adivinar, que tiene letras, números y símbolos. |
| **Inicio de sesión** | Proceso para entrar a una cuenta, escribiendo un usuario y una contraseña. |
| **Verificación en dos pasos** | Seguridad extra que pide un código (por ejemplo, enviado al celular) además de la contraseña. |
| **Actualización** | Mejora o corrección que se hace a un programa para que funcione mejor o sea más seguro. |
| **Permisos de acceso** | Lo que cada persona puede o no puede hacer dentro del sistema. |
| **Configuración** | Los ajustes o formas en que está preparado un sistema para funcionar. |
| **Cifrado / Protección de datos** | Forma de esconder información para que nadie más la pueda leer, incluso si logra verla. |

| Término | Explicación sencilla |
|---|---|
| **Registro / Log** | Lista de las actividades o acciones que han ocurrido dentro del sistema. |
| **Conexión segura (https)** | Tipo de página web que protege los datos mientras se usan, indicada por un candado al lado de la dirección. |
| **Fuente confiable** | Lugar o persona conocida y segura desde donde se obtiene información o archivos. |
| **Ataque informático** | Cuando alguien intenta dañar un sistema, robar información o usarlo sin permiso. |

**¿Qué es el plan de pruebas en las pruebas de software?**

La planificación de pruebas es un aspecto fundamental del ciclo de vida de las pruebas de software. Independientemente de su producto, requisitos o alcance de trabajo, necesita un plan de pruebas integral para diseñar y ejecutar pruebas de software exitosas.

1. **¿Qué es un plan de pruebas?**


Un plan de pruebas es un documento completo que describe las estrategias, los objetivos, el cronograma, el equipo, la tecnología, las estimaciones, las fechas límite y el personal necesario para realizar las pruebas de los productos de software. Sirve de guía para que los responsables de pruebas garanticen que el proyecto en cuestión cumpla con todos los requisitos.

- Un documento de plan de pruebas bien redactado se adapta a la nueva información y etapas del proyecto.
- Es el centro alrededor del cual giran los esfuerzos de un equipo de pruebas.

**2. ¿Cuál es la importancia de un plan de pruebas?**

Debe crear y utilizar un plan de pruebas en su proceso de pruebas de software porque ofrece numerosos beneficios que incluyen, entre otros:

- Funciona como una guía detallada: Usar un plan de pruebas es como seguir un mapa para llegar a tu destino. Abarca cada etapa del proceso con detalles sobre las responsabilidades individuales, los métodos de prueba y las herramientas a utilizar.
- Mitigación de riesgos: un plan de pruebas bien definido ayuda a identificar riesgos potenciales en todo el proceso de pruebas y describe varios planes de contingencia para mitigarlos.
- Soporte de control de calidad mejorado: Un plan de pruebas garantiza que se cubra cada aspecto del software y se aborden todos los requisitos de prueba. Su objetivo es detectar y corregir todos los defectos antes del lanzamiento del software.
- Optimización de recursos: Contar con un plan de pruebas adecuado facilita una gestión eficiente de los recursos. Un plan de pruebas incluye las pruebas necesarias, lo que también ayuda a identificar los recursos necesarios, como personal, entorno, herramientas, etc. Esto permite planificar y asignar estos recursos eficazmente, evitando sobrecargas innecesarias.

### 3. Componentes de un plan de pruebas


- **Alcance**: Especifica los casos de usuario que se pueden probar. El alcance del proyecto puede excluir los casos o problemas.
- **Programación**: específica cuándo comenzarán las pruebas y cuándo se esperan los resultados.
- **Asignación de recursos**: indica qué evaluador trabajo en qué plan de pruebas.
- Entorno de pruebas: Explica en profundidad las características, capacidades y limitaciones del entretenimiento de pruebas.
- **Herramientas de prueba**: las herramientas de prueba de software específicos los recursos que se utilizan para pasar, seguimiento de problemas y tareas similares.
- **Gestión de defectos**: definir a quién se debe notificar sobre los defectos, cómo enviarlos y cómo informar debe acompañar a cada informe durante el proceso de gestión de pruebas, como imágenes, registros de texto o clips de problemas de código.

- **Gestión de riesgos:** enumeración los peligros potenciales que podrían surgir durante las pruebas de software, así como los peligros que el propio programa enfrentaría si se implementa sin pruebas adecuadas.

**¿Qué es un caso de prueba en pruebas de software?**

Un caso de prueba en prueba de software es un documento que consiste en un conjunto de condiciones de pasos enfermos para verificar si una característica de funcionalidad específica de una aplicación de software función según lo anterior. Defina las entradas, las acciones y los resultados esperados para un escenario de prueba específico, garantizando así el correcto funcionamiento del software en diversas condiciones. Un caso de prueba suele incluye información como el ID del caso, la descripción, los pasos, las precondiciones, los datos de entrada, los pasos de ejecución, el resultado esperado y cualquier postcondición o corrección necesaria.

Los casos de prueba son esenciales para validar la funcionalidad, el rendimiento y la confianza de las aplicaciones de software.

Se establecerán variables específicas que los equipos de control de calidad deben comparar para determinar si la función funciona corregamente. Los componentes del caso de prueba mencionan la entrada, la ejecución y la salida/respuesta esperada. Indican a los ingenieros que hacen, cómo hacen y que resultan son aceptables.

**El objetivo de escribir casos de prueba en pruebas de software**

El objetivo de escribir casos de prueba es garantizar que cada característica funciona según lo anterior y detectar problemas en las primeras etapas del ciclo de desarrollo. Para ello, es fundamental comprender los objetivos de la escritura de casos de prueba. Una continuación, se enumeran algunos de ellos:

- Para validar características y funciones específicas del software.
- Guiar a los evaluadores a viajes de su actividad práctica diaria.
- Para registrador un catálogo de los pasos realizados, que puede revisar en caso de que surja un error.
- Proporcionar un modelo para futuros proyectos y evaluadores para que no tengan que empezar a trabajar desde cero.

**Formato de caso de prueba estándar**

A continuación, se muestra un formato estándar para escribir casos de prueba:

**Formato del caso de prueba:**

- ID de caso de prueba
- Escenario de prueba
- pasos de prueba
- Prerrequisitos

**Ejemplo de Plan de Pruebas**

- datos de prueba
- Resultados esperados/previstas
- Resultados reales
- Estado de la prueba: aprobado/reprobado

**Introducción**

El Plan de pruebas describe el alcance, el enfoque, los recursos y el cronograma de todas las actividades de prueba. Identifica los elementos y características que se van a probar, es decir, los tipos de pruebas. Contiene una estrategia detallada y ejecutable para la realización. Define el objetivo detallado de la prueba, específico de un sistema en particular, el enfoque de prueba, el entorno de prueba, las condiciones de prueba y el plan de prueba.

**Alcance**

El alcance de este plan de prueba es garantizar que se cumplan todos los requisitos técnicos, funcionales y comerciales. El propósito de este documento es describir el plan de prueba general y la estrategia para evaluar un sitio web. El enfoque descrito en este documento proporciona el marco para todas las pruebas relacionadas con los sitios web.

Este documento evolucionará según sea necesario con las actualizaciones de los requisitos. También debemos asegurarnos de que se obtengan todos los resultados esperados.

**Objetivo de las pruebas**

El objetivo general de las pruebas es validar la exactitud de la generación del archivo de datos de la interfaz, su contenido y cualquier condición de error. Los objetivos de calidad de las pruebas del sitio web son garantizar la validación completa de los requisitos comerciales y de software:

1. Verificar que los requisitos de software estén cubiertos y sean precisos.
2. Realizar una planificación detallada de las pruebas.
3. Identificar los estándares y procedimientos de prueba que se utilizarán.
4. Preparar y documentar escenarios de prueba y casos de prueba.
5. Gestionar el proceso de seguimiento de defectos.
6. Finalizar el proyecto para su lanzamiento.

A continuación, se mencionan las fases y metodologías de prueba detalladas. Siguiendo los diferentes protocolos de cada fase conseguiremos los mejores resultados.

Análisis de requisitos

- Pruebas de diseño
- Pruebas de funcionalidad
- Pruebas de integración
- Pruebas API
- Pruebas de usabilidad
- Pruebas de compatibilidad
- Pruebas de rendimiento
- Pruebas de seguridad
- Pruebas de automatización
- Pruebas de humo
- Pruebas beta

**Plan de pruebas – Catálogo (Crear, Editar y Eliminar)**

**¿Para qué es este plan?**

Este plan sirve para revisar que todo funcione bien en una parte del sistema donde se guarda información, como una lista de productos, personas, tareas, etc. Se va a probar que se puedan:

- Crear nuevos elementos (agregar algo nuevo a la lista)
- Cambiar elementos que ya existen.
- Borrar elementos que ya no se necesitan.
- Ver la lista completa correctamente.

**¿Qué se va a revisar?**

- Que se puedan agregar datos nuevos sin errores.
- Que se puedan cambiar datos ya guardados.
- Que se puedan borrar datos.
- Que todo lo anterior se vea reflejado correctamente en la lista.

**¿Cómo sabremos si todo está bien?**

- Si el sistema deja agregar datos cuando todo está bien escrito.
- Si no deja guardar cuando hay errores (por ejemplo, un campo vacío).
- Si avisa cuando algo salió mal.

**Lista de pruebas a realizar**

| Nº | Qué se va a probar | Qué se hace | Qué se espera que pase |
|---|---|---|---|
| 1 | Agregar un nuevo dato | Se llena todo el formulario con datos correctos y se guarda | El dato aparece en la lista |
| 2 | Agregar con campos vacíos | Se deja uno o más campos vacíos y se intenta guardar | El sistema muestra un mensaje diciendo que falta completar |
| 3 | Agregar con datos incorrectos | Se escribe texto donde debería ir un número (por ejemplo) | El sistema no deja guardar y avisa del error |

| | | | |
|---|---|---|---|
| 4 | Editar un dato existente | Se cambia algo de un dato ya guardado y se guarda | El cambio aparece en la lista |
| 5 | Borrar un dato | Se selecciona un dato y se elige eliminarlo | El dato desaparece de la lista |
| 6 | Cancelar borrado | Se intenta borrar un dato pero se cancela | El dato sigue en la lista |
| 7 | Ver la lista | Se entra al catálogo para ver todos los datos | Se muestra correctamente la lista actual |
| 8 | Evitar duplicados | Se intenta agregar dos datos iguales (como el mismo código) | El sistema no deja guardar y avisa que ya existe |

**¿Cuándo se termina de probar?**

- Cuando se hayan hecho todas las pruebas.
- Cuando todos los errores importantes hayan sido corregidos y se haya vuelto a probar.
- Cuando todo funcione como se espera.

**¿Quién hace qué?**

- **Persona que prueba:** Revisa si todo funciona como debe.
- **Persona que desarrolla:** Arregla los errores si algo falla.
- **Persona que revisa:** Se asegura de que todo esté en orden antes de usarlo oficialmente.