

# Seguridad en la Red Informática Mundial

## Repaso Temas 1 y 2

Semana 16 clases 31 y 32

Mtra. María Noemí Araiza Ramírez

---

# **Top-Ten Vulnerabilidades según OWASP**

## **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **Seguridad en la Red informática**

El mantra de todo buen ingeniero de seguridad es: "La seguridad no es un producto, sino un proceso." implica algo mas que la implantación de criptografía robusta en un sistema: se trata de diseñar el sistema entero de manera que todas las medidas de seguridad, incluyendo la criptografía, trabajen conjuntamente.

Bruce Schneier

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Seguridad de la información

Es el conjunto de medidas y procedimientos, tanto humanos como técnicos, que permiten proteger la integridad, confidencialidad y disponibilidad de la información.

- Integridad: asegurando que la información y sus métodos de proceso son exactos y completos.
  - Confidencialidad: hacer constar que sólo pueden acceder a la información y modificarla los usuarios autorizados.
  - Disponibilidad: dejando que la información pueda estar disponible cuando los usuarios la requieran.
-

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **La seguridad en informática**

Es una rama de la seguridad de la información que intenta proteger la información que utiliza una infraestructura informática y de telecomunicaciones para ser almacenada o transmitida.

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

**En la seguridad informática se desea proteger:**

En función de lo que se quiere proteger:

- Seguridad física: que se asocia a la protección física del sistema ante amenazas como inundaciones, incendios, robos, etc.
  - Seguridad lógica: mecanismos que protegen la parte lógica de un sistema informático (datos, aplicaciones y sistemas operativos). Uno de los medios más utilizados es la criptografía.
-

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

**En la seguridad informática se desea proteger:**

En función del momento en que tiene lugar la protección:

- Seguridad activa: se encarga de prevenir, detectar y evitar cualquier incidente en los sistemas informáticos antes de que se produzca (medidas preventivas).

Por ejemplo, utilización de contraseñas.

- Seguridad pasiva: comprende todas aquellas técnicas o procedimientos necesarios para minimizar las consecuencias de un incidente de seguridad (medidas correctoras).

Por ejemplo, las copias de seguridad.

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Activo

Es el recurso del sistema (informático o no) que involucra a los trabajadores, el software, los datos, los archivos, el hardware, las comunicaciones, etc. que una organización necesita para lograr sus objetivos, como el proteger debidamente aquello que se enfrente a un eventual suceso o percance intencionado o no.

---



# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

En la informática se consideran activos principales de una empresa:

- Información: datos almacenados en cualquier tipo de soporte.  
Por ejemplo, documentos, libros, patentes, correspondencia, manuales de usuario, etc.
  - Software: programas o aplicaciones que utiliza la organización para su buen funcionamiento.  
Por ejemplo: las aplicaciones comerciales, los sistemas operativos, etc.
  - Físicos: la infraestructura tecnológica empleada para almacenar, procesar, gestionar o transmitir toda la información para el buen funcionamiento de la organización.  
Por ejemplo: servidores, computadoras de escritorio, etc.
  - Personal de la organización que utilice la estructura tecnológica y de comunicación para el manejo de la información.
-

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Vulnerabilidad

Se refiere a la debilidad “agujeros de seguridad” de un activo que pueda afectar de alguna manera el correcto funcionamiento del sistema informático.

Hablamos de fallos en la implementación de las aplicaciones o en la configuración del sistema operativo, etc.

Como ejemplo tenemos el no utilizar algún tipo de protección frente a fallos eléctricos o no contar con mecanismos de protección frente a ataques informáticos, como antivirus o cortafuegos.

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Amenaza

Es cualquier entidad o circunstancia que atenta contra el buen funcionamiento de un sistema informático. Sin embargo hay amenazas que afectan a los sistemas de forma involuntaria, como un desastre natural.

Se tienen dos tipos de amenazas, las pasivas y las activas.

- Pasivas, conocidas como “escuchas”. Su objetivo es obtener información relativa a una comunicación.

Por ejemplo, los equipos informáticos portátiles que utilizan programas especializados para monitorizar el tráfico de una red WiFi.

- Activas, que intentan realizar algún cambio no autorizado en el estado del sistema, son más peligrosas que las anteriores.

Como ejemplos se encuentran la inserción de mensajes ilegítimos, la usurpación de identidad, etc.

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **Ataque**

Es una acción que trata de aprovechar una vulnerabilidad de un sistema informático para provocar un impacto sobre él e incluso tomar el control del mismo. Son acciones intencionadas o fortuitas que pueden llegar a poner en riesgo un sistema.

Como ejemplos de ataques tenemos la utilización de programas para conseguir acceso al servidor de forma ilegítima o la realización de ataques de denegación de servicio para colapsar el servidor.

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

Un ataque informático pasa por las siguientes fases:

- Reconocimiento. Consiste en obtener toda la información necesaria de la víctima, que puede ser una persona o una organización.
  - Exploración. Se intenta conseguir información sobre el sistema a atacar, como por ejemplo, direcciones IP, nombres de host, datos de autenticación, etc.
  - Obtención de acceso. A partir de la información descubierta en la fase anterior, se intenta explotar alguna vulnerabilidad detectada en la víctima para llevar a cabo el ataque.
  - Mantener el acceso. Después de acceder al sistema, se buscará la forma de implantar herramientas que permitan el acceso de nuevo al sistema en futuras ocasiones.
  - Borrar las huellas que se hayan podido dejar durante la intrusión para evitar ser detectado.
-

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Riesgo

Según la UNE-71504:2008, un riesgo “es la estimación del grado de exposición a que una amenaza se materialice sobre uno o más activos causando daños o perjuicios a la organización”.

El Centro Criptológico Nacional define el riesgo como “la probabilidad de que una amenaza se materialice aprovechando una vulnerabilidad y causando daño (impacto) en un proceso o sistema”.

El riesgo es, una medida de la probabilidad de que se materialice una amenaza.

Por ejemplo, si la instalación eléctrica del edificio es antigua, existirá un riesgo elevado de sufrir una interrupción del servicio en caso de producirse una subida de tensión.

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## ¿Qué es un Método?

Es una forma ordenada y sistemática que nos permitirá conseguir una meta o lograr un objetivo.

Procedimiento para alcanzar algo

Por ejemplo:

Definir los pasos necesarios para la automatización de un sistema

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **¿Qué es una Metodología?**

Conjunto de métodos empleados por una disciplina

Una metodología le da énfasis al entorno en el cuál se analiza y estructura el desarrollo de un sistema o una aplicación.

Hay distintas metodologías, aunque no todas son compatibles con nuestros desarrollos, debido a que el ciclo de vida del software puede variar.

Por esta razón, es importante que dependiendo del tipo de la aplicación que se vaya a desarrollar, se identifique la metodología idónea

---



# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **Metodología de Desarrollo**

- Es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de SI.
  - Es un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar nuevo software.
  - Es una metodología para desarrollar software de manera sistemática.
-

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

- Están disponibles sistemas de seguridad que se utilizan para proteger a los sistemas operativos sobre los cuales se realizan pruebas durante el desarrollo de software
  - La seguridad en una aplicación generalmente evita que se produzcan errores en los sistemas operativos en donde se realiza el desarrollo, así como en las pruebas del funcionamiento de la aplicación
  - Cuando se hacen pruebas, es común que se produzcan fallas permanentes, las cuales afecten al sistema; pero realmente es necesario que se ejecuten para corregirlas, ya que de otro modo no habría forma de detectarlas
-

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

¿Qué metodologías de desarrollo de software seguro son las más conocidas en la actualidad?

- Microsoft (Iniciativa Trustworthy Computing)
  - OSSTMM (Open Source Security Testing Methodology Manual )
  - OWASP (Open Web Application Security Project)
  - OASIS Web Application Security (WAS) project
-

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **Microsoft (Iniciativa Trustworthy Computing)**

Diseñada por Microsoft en el año 2001 como una necesidad del mercado para contar con soluciones que consideren los aspectos de seguridad y aspectos de operación y funcionalidad, con un enfoque de trabajo colaborativo a largo plazo y tiene como objetivo crear y disponer para la comunidad en general una computación basada en mejores prácticas (best practices) que sea más segura, privada y confiable.

---

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **Microsoft (Iniciativa Trustworthy Computing)**

Este modelo también abarca conceptos como la estabilidad, la confianza y la seguridad en la plataforma y este ultimo concepto se sustenta de 5 pilares primordiales que no debemos dejar de tomar en cuenta.

- Aislamiento y flexibilidad
  - Calidad
  - Autenticación
  - Autorización y control de accesos
  - Orientación y formación
-

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

Microsoft (Iniciativa Trustworthy Computing)

El modelo TwC se basa en 4 pilares: Seguridad, Privacidad, Confiabilidad y Mejores prácticas.

**Seguridad** Se busca garantizar la seguridad de los sistemas y la información que manejan, contra ataques (considerando a la tecnología, las personas y los procesos como base de la Seguridad).

Esto es:

- Son resistentes a ataques
  - Protegen la confidencialidad, integridad y disponibilidad de los datos y sistemas.
  - Ningún virus atentará contra nuestros sistemas o los volverá inutilizables.
-

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Microsoft (Iniciativa Trustworthy Computing)

**Privacidad** Se requiere mantener segura la información personal, financiera e identidad de los usuario mientras su control y uso quedan a disposición de su dueño.

Es decir, Los individuos controlan sus datos personales, la información personal no será expuesta de ninguna forma, ni utilizada de manera que no sea la explícitamente indicada, los productos y servicios en línea se adhieren a principios de información básicos.

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Microsoft (Iniciativa Trustworthy Computing)

**Confiabilidad** Se busca desarrollar software confiable para atender las necesidades tanto de usuarios y negocios.

Esto es que cuando instalemos un programa, no provocará efectos colaterales sobre otro software instalado ni sobre el hardware, además cuentan con excelencia en Ingeniería, son confiables y tienen un desempeño acorde a las expectativas y están disponible cuando se necesita.

---



# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*

Surge el **Modelo de Amenazas (Threat Model)** en él vemos que la seguridad debe abarcar todas las fases. Por lo tanto, el software debe ser:

### **Seguro por Defecto**

La aplicación recién instalada tiene un comportamiento suficientemente seguro.

Un ejemplo, es el gestor de áreas de exposición de SQL Server

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

Microsoft (Iniciativa Trustworthy Computing)  
*Modelo de Amenazas (Threat Model)*

## **Seguro por Diseño**

Ninguna parte de la aplicación queda fuera del control de seguridad

## **Seguro en la Distribución**

Informar al usuario sobre la seguridad de la aplicación, mecanismos de modificación de las características de seguridad, la creación de parches de seguridad tan pronto como se detecte una nueva vulnerabilidad.

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

Microsoft (Iniciativa Trustworthy Computing)  
*Modelo de Amenazas (Threat Model)*

## **Seguro en las Comunicaciones**

Se debe tener asegurada la transferencia de los datos y los medios de transmisión para que no se pueda afectar el paso de datos por una red.

---

# Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*

La metodología Microsoft establece cinco fases

**1**

Identificar activos de la aplicación

**2**

Crear información general sobre la arquitectura

**3**

Descomponer la aplicación

**4**

Identificar, documentar y clasificar las amenazas

**5**

Identificar las vulnerabilidades

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

Microsoft (Iniciativa Trustworthy Computing)  
*Modelo de Amenazas (Threat Model)*

Debemos considerar las amenazas que nos muestra el modelo STRIDE

**S**poofing (Suplantación)

**T**ampering (Manipulación)

**R**epudiation (Repudio)

**I**nformation Disclosure (Divulgación de Información)

**D**enial of Service (Denegación de Servicio)

**E**levation of privileges (Elevación de privilegios)

## Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*

### **Spoofing (Suplantación)**

Se adopta la personalidad de otro usuario en un equipo.  
Ejemplos:

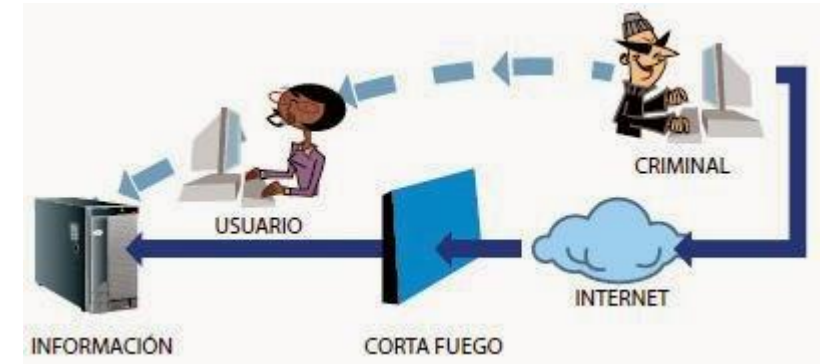
- Acceso no autorizado a la información de autenticación de otro usuario y su uso, como nombre de usuario y contraseña.
- Enviar mensajes de correo electrónico con una dirección falsa o de otro usuario.
- Crear una cuenta de una red social con información de otro usuario.



# Microsoft (Iniciativa Trustworthy Computing)

## *Modelo de Amenazas (Threat Model)*

### **Tampering (Manipulación)**



Es la modificación malintencionada de los datos por parte de algún usuario.  
Ejemplos:

- Alterar datos durante su transferencia en una red.
- Cambiar datos en archivos almacenados en un equipo.
- Cambiar datos en una base de datos.

## Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*



### **Repudiation (Repudio)**

Son acciones en las que los usuarios niegan su autoría sin que se pueda probar lo contrario. El no rechazo se refiere a la posibilidad de un sistema de contrarrestar las amenazas de repudio.

Ejemplos:

- Un usuario que realiza una operación ilegal sin que existan bitácoras o logs.
  - Cuando a un usuario le llega un paquete por mensajería debe firmar un recibo contra entrega, la cual es la prueba de entrega del paquete que tiene el proveedor.
-



# Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*

## Information Disclosure (Divulgación de Información)

Las amenazas de divulgación de información revelan información a personas no autorizadas.

Ejemplos:

- Divulgar información en mensajes de error.
- Lectura de un archivo al que no se debería tener acceso.
- Exponer el código de un sitio web.
- Intruso leyendo datos que están en transferencia entre dos equipos.



## Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*

### Denial of Service (Denegación de servicio)

**Denial-of-service  
Attack**



Los ataques por denegación de servicio (DoS) ocasionan la pérdida de servicio a los usuarios válidos, por ejemplo, deshabilitando temporalmente un servidor Web, que ponen en riesgo la disponibilidad y fiabilidad del sistema.

Ejemplos:

- Inundar la red con paquetes ICMP falsificados.
- Inundar la red con paquetes de sincronización.

## Microsoft (Iniciativa Trustworthy Computing) *Modelo de Amenazas (Threat Model)*

### **Elevation of privileges (Elevación de privilegios)**



En este tipo de amenaza, un usuario sin privilegios obtiene acceso con privilegios, por lo que se pone en peligro a todo el sistema, tanto en su acceso como en la confiabilidad de sus datos.

Ejemplos:

- Explotar la saturación de un búfer para obtener privilegios en el sistema.
- Un usuario que burla las defensas de un sistema y obtiene privilegios de administrador de forma no autorizada.

# **Tema 1: Metodologías de desarrollo de aplicaciones web seguras**

## **Metodología OSSTMM**

Manual de Metodología de Pruebas de Seguridad de Código Abierto o bien se conoce como Manual de la Metodología Abierta del Testeo de Seguridad.

Fue creado por Pete Herzog y desarrollado por ISECOM (Institute for Security and Open Methodologies) <http://www.isecom.org>

ISECOM es una organización sin fin de lucro, dedicada al desarrollo de metodologías para la verificación de la seguridad, programación segura, verificación de software y concientización en seguridad.

---

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras



OSSTMM es el proyecto más destacado de ISECOM, sin embargo también tiene otros:

## **SCARE**

The Source Code Analysis and Risk Evaluation.  
Análisis de código fuente y evaluación de riesgos.

## **Child Safety and Security Methodology**

Una metodología para enseñar seguridad a los niños a través de juegos e historias

## **Home Security Methodology**

Metodología para mantener seguros los hogares y mantenerlos a salvo de posibles amenazas

## **National Security Methodology**

Definición de políticas y metodologías para mejorar la seguridad nacional.

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## OSSTMM

Es uno de los estándares más usado en auditorías de seguridad para revisar la seguridad de los sistemas desde Internet.

Incluye un marco de trabajo que describe las fases que hay que realizar para la ejecución de la auditoría.

# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## OSSTMM



# Tema 1: Metodologías de desarrollo de aplicaciones web seguras

## Seguridad de la Información

Recolección de documentos

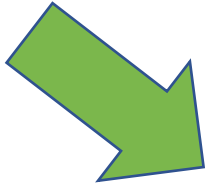
Revisión de privacidad

Revisión de la inteligencia  
competitiva



## Seguridad de los procesos

Testeo de solicitud



Testeo de sugerencia dirigida



Testeo de las personas confiables

## Seguridad de las tecnologías de internet

Logística y controles



Exploración de red



Identificación de los servicios del sistema



Búsqueda de información competitiva y revisión de privacidad



Obtención de documentos

## Seguridad de las tecnologías de internet

Búsqueda y verificación de vulnerabilidades



Testeo de aplicaciones de internet



Enrutamiento



Testeo de sistemas confiados



Testeo de control de acceso

## Seguridad de las tecnologías de internet

Testeo de sistema de detección de intrusos



Testeo de medidas de contingencia



Descifrado de contraseñas



Testeo de denegación de servicios



Evaluación de políticas de seguridad

## Seguridad en las Comunicaciones

Testeo de PBX (Private Branch Exchange)

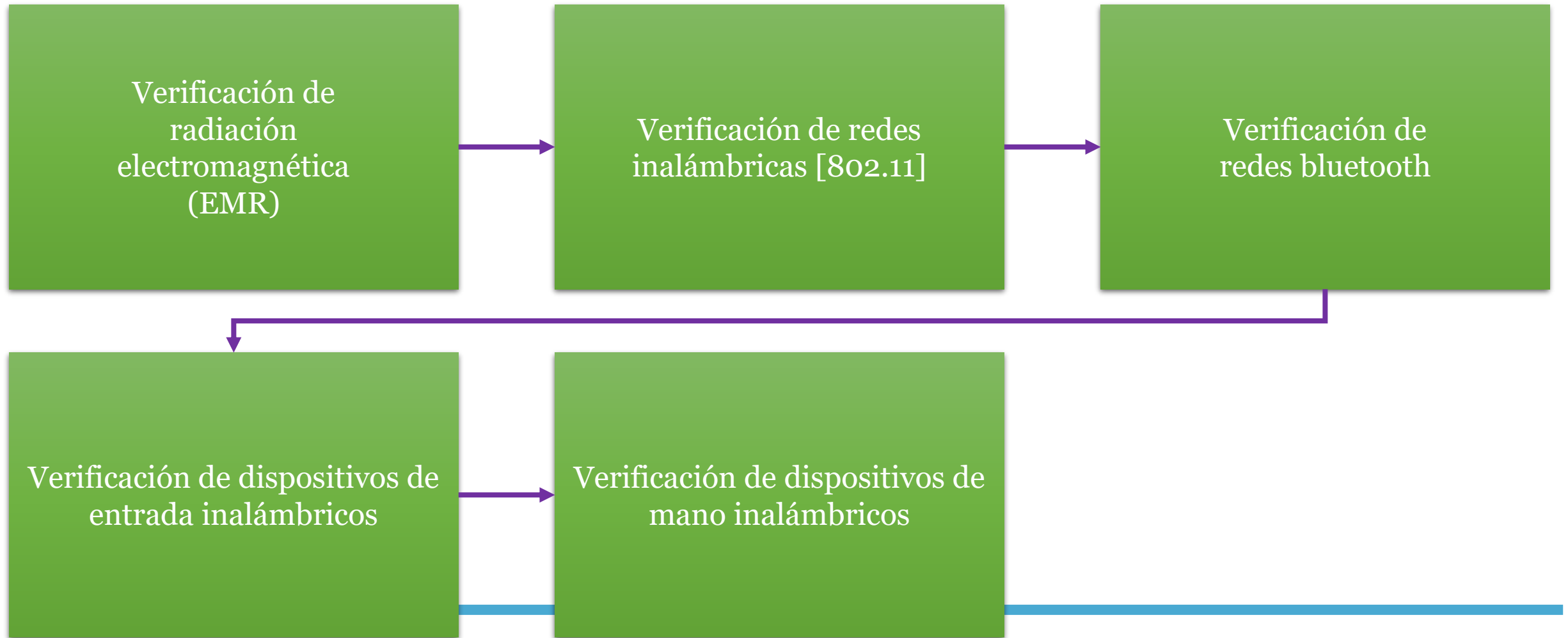
Testeo del correo de voz

Revisión del fax

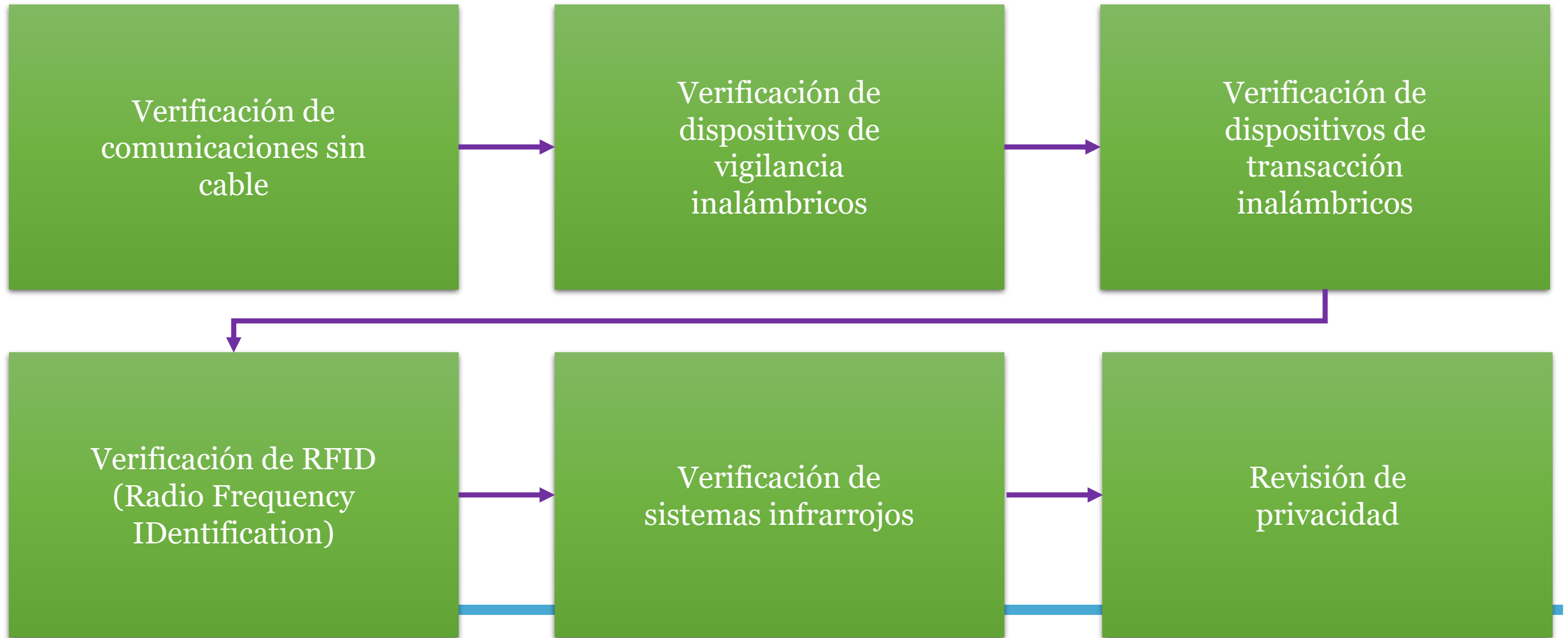
Testeo del módem

---

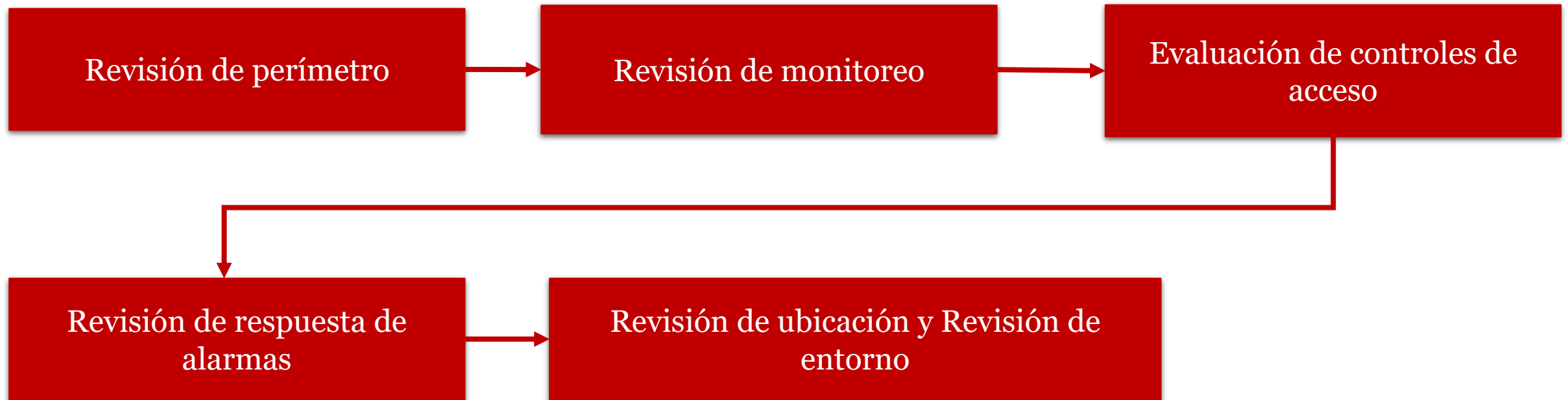
## Seguridad Inalámbrica



## Seguridad Inalámbrica



## Seguridad Física





## Metodología OWASP

- Tiene como objetivo ofrecer una metodología:

De libre acceso y utilización.

Que pueda ser utilizada como material de referencia por parte de los arquitectos de software, desarrolladores, fabricantes y profesionales de la seguridad.

Todos ellos involucrados en el diseño, desarrollo, despliegue y verificación de la seguridad de las aplicaciones y servicios web.

# Metodología OWASP

## ● Proyectos más destacados:

**OWASP Top Ten Project:** Representa un consenso a nivel global sobre las 10 vulnerabilidades web más importantes.

**WebGoat:** Herramienta destinada a la educación y que permite practicar y explotar las vulnerabilidades más frecuentes de un sitio Web, con el fin de poner en práctica una metodología de desarrollo seguro.

**WebScarab:** Es un framework para el análisis de aplicaciones que utilizan como base los protocolos HTTP y HTTPS. Están escrito en Java y es multiplataforma.

## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

Validación de la entrada y salida de información:

- La entrada y salida de información es el principal mecanismo que dispone un atacante para enviar o recibir código malicioso contra el sistema.
  - Siempre debe verificarse que cualquier dato entrante o saliente es apropiado y en el formato que se espera.
  - Las características de estos datos deben estar predefinidas y debe verificarse en todas las ocasiones.
-

## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

Diseños simples:

- Los mecanismos de seguridad deben diseñarse para que sean los más sencillos posibles, huyendo de sofisticaciones que compliquen excesivamente la vida a los usuarios.
  - Si los pasos necesarios para proteger de forma adecuada una función o módulo son muy complejos, la probabilidad de que estos pasos no se ejecuten de forma adecuada es muy elevada.
-

## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

Utilización y reutilización de componentes de confianza:

- Debe evitarse “reinventar la rueda” constantemente.
  - Cuando exista un componente que resuelva un problema de forma correcta, lo más inteligente es utilizarlo.
-

## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

### Defensa en profundidad

- Nunca confiar en que un componente realizará su función de forma permanente y ante cualquier situación.
  - Hemos de disponer de los mecanismos de seguridad suficientes para que cuando un componente del sistema fallen ante un determinado evento, otros sean capaces de detectarlo.
-

## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

### Verificación de privilegios

- Los sistemas deben diseñarse para que funcionen con los menos privilegios posibles.
  - Igualmente, es importante que los procesos únicamente dispongan de los privilegios necesarios para desarrollar su función, de forma que queden compartimentados.
-

## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

Ofrecer la mínima información

- Ante una situación de error o una validación negativa, los mecanismos de seguridad deben diseñarse para que faciliten la mínima información posible.
  - De la misma forma, estos mecanismos deben estar diseñados para que una vez denegada una operación, cualquier operación posterior sea igualmente denegada.
-



## **Principios básicos de la seguridad de cualquier aplicación o servicio Web**

### Otros consideraciones

- De arquitectura.
  - Mecanismos de autenticación.
  - Gestión de sesiones de usuario.
  - Control de acceso.
  - Registro de actividad.
  - Consideraciones de privacidad y criptografía.
-

# Metodología OWASP

## Elementos de la Guía de desarrollo

Para garantizar el desarrollo seguro de software, la Guía de Desarrollo OWASP proporciona una pautas para salvaguardar de amenazas en las aplicaciones Web en los siguientes elementos:

- Manejo de Pagos en el Comercio Electrónico
  - Phishing
  - Servicios Web
  - Autenticación
  - Autorización
-

# Metodología OWASP

## Elementos de la Guía de desarrollo

### Phishing

Pautas para evitar el problema del Phishing en el desarrollo de aplicaciones Web:

- No sea la fuente de robos de identidad.
  - Implemente protecciones dentro de su aplicación.
  - Monitorice actividad inusual en las cuentas.
  - Tome control de los nombres de dominio fraudulentos.
  - Trabaje con las autoridades competentes
  - Sea amable con su cliente cuando ocurre un ataque – él es la víctima.
-

# Metodología OWASP

## Elementos de la Guía de desarrollo

### Manejo de Sesiones (cookies)

El uso de cookies es para reconocer al usuario en el momento en el que se conecta al servidor.

Una de las páginas que recoge la petición del usuario puede comprobar si existe una cookie que ha dejado anteriormente, si la encuentra sabe que ese usuario ya ha visitado ese sitio web y por lo tanto puede leer valores que le identifiquen.



# Metodología OWASP

## Elementos de la Guía de desarrollo

### Manejo de Sesiones (cookies)

Otro uso de las cookies es ofrecer personalización del usuario, es decir, en muchos sitios web es posible elegir color de fondo, tipo de letra, entre otros recursos.

Estos valores pueden almacenarse en cookies, de forma que cuando acceda de nuevo al sitio web, se compruebe la existencia de esos valores y recuperarlos para utilizarlos en la personalización de la página tal y como el usuario estableció en su momento.



# Metodología OWASP

## Framework de Pruebas

Describe un marco de pruebas típico que puede ser desarrollado en una Organización

Hay que verla como un marco de referencia que comprende tanto técnicas como tareas que es apropiado realizar en varias fases del ciclo de vida de desarrollo del software (SDLC)

---

# Metodología OWASP

## Framework de Pruebas OWASP

Está estructurado de la siguiente forma:

### Fase 1 Antes de Empezar el Desarrollo

- 1a Revisión de Estándares y Políticas
- 1b Desarrollo de Métricas y Criterios de Medición (Asegurar la Trazabilidad)

### Fase 2 Durante el Diseño y Definición

- 2a Revisión de los Requisitos de Seguridad
- 2b Diseño de una Arquitectura de Revisión
- 2c Creación y Revisión de Modelos UML
- 2d Creación y Revisión de Modelos de Amenaza

# Metodología OWASP

## Framework de Pruebas OWASP

### Fase 3 Durante el Desarrollo

- 3a Inspección de Código por Fases
- 3b Revisiones de Código

### Fase 4 Durante la Implementación

- 4a Testing de Penetración de Aplicaciones
- 4b Testing de Gestión de Configuraciones

### Fase 5 Mantenimiento y Operación

- 5a Ejecución de Revisiones de la Gestión Operativa
- 5b Ejecución de Comprobaciones Periódicas de Mantenimiento
- 5c Asegurar la Verificación de Cambios

Flujo de Pruebas típico en un SDLC (Software Development Life Cycle)



## **Tema 2: Top 10 de vulnerabilidades según OWASP**

### **Objetivos del Top 10:**

Educar a desarrolladores, diseñadores, arquitectos, gerentes, y organizaciones sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web.

Pretende crear conciencia sobre la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las Organizaciones.

Top10 no es un programa de seguridad en aplicaciones.

---

## **Tema 2: Top 10 de vulnerabilidades según OWASP**

### **Objetivos del Top 10:**

Referenciado por numerosos estándares, libros, y organizaciones, incluyendo MITRE, PCI DSS, DISA, FTC, y muchos más.

El OWASP Top 10 fue lanzado por primera vez en 2003, se han realizado actualizaciones posteriores en 2004, 2007, 2010, 2013 y 2017.

OWASP recomienda que las organizaciones establezcan una base sólida de formación, estándares y herramientas que hagan posible la codificación segura.

---

## **Tema 2: Top 10 de vulnerabilidades según OWASP**

### **Objetivos del Top 10:**

Por encima de esa base, las organizaciones deben integrar la seguridad en su desarrollo, verificación y procesos de mantenimiento.

La gerencia puede utilizar los datos generados por estas actividades para gestionar los costos y riesgos asociados a la seguridad en aplicaciones.

---

## **Tema 2: Top 10 de vulnerabilidades según OWASP**

### **Objetivos del Top 10:**

Se basa en el envío de datos de más de 40 empresas que se especializan en seguridad de aplicaciones y una encuesta de la industria que fue completada por más de 500 personas.

Esta información abarca vulnerabilidades recopiladas de cientos de organizaciones y más de 100.000 aplicaciones y APIs del mundo real.

Las 10 principales categorías fueron seleccionadas y priorizadas de acuerdo con estos datos de prevalencia, en combinación con estimaciones consensuadas de detectabilidad e impacto.

---

## Tema 2: Top 10 de vulnerabilidades según OWASP



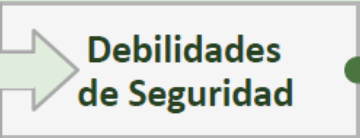


**A1**

### Inyección

Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un interprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder datos no autorizados.

# Top-Ten Vulnerabilidades según OWASP

## A1 Inyección

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección PROMEDIO	Impacto SEVERO	Específico de la aplicación/negocio
<p>Considere a cualquiera que pueda enviar información no confiable al sistema, incluyendo usuarios externos, usuarios internos y administradores.</p>	<p>El atacante envía ataques con cadenas simples de texto, los cuales explotan la sintaxis del interprete a vulnerar. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo las fuentes internas.</p>	<p>Las <u>fallas de inyección</u> ocurren cuando una aplicación envía información no confiable a un interprete. Estas fallas son muy comunes, particularmente en el código antiguo. Se encuentran, frecuentemente, en las consultas SQL, LDAP, Xpath o NoSQL; los comandos de SO, intérpretes de XML, encabezados de SMTP, argumentos de programas, etc. Estas fallas son fáciles de descubrir al examinar el código, pero difíciles de descubrir por medio de pruebas. Los analizadores y «fuzzers» pueden ayudar a los atacantes a encontrar fallas de inyección.</p>		<p>Una inyección puede causar pérdida o corrupción de datos, pérdida de responsabilidad, o negación de acceso. Algunas veces, una inyección puede llevar a el compromiso total de el servidor.</p>	<p>Considere el valor de negocio de los datos afectados y la plataforma sobre la que corre el intérprete. Todos los datos pueden ser robados, modificados o eliminados. ¿Podría ser dañada su reputación?</p>

## Tema 2: Top 10 de vulnerabilidades según OWASP

**A2**

### Pérdida de Autenticación y Gestión de Sesiones



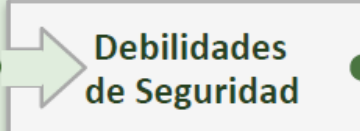

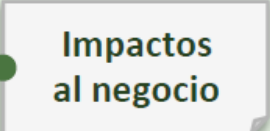
Deficiencias en estas áreas, con mucha frecuencia implican fallas en la protección de credenciales y testigos (tokens) de sesión a través de su ciclo de vida.

Estos defectos pueden conducir a un robo de usuarios o cuentas de administración, sabotaje de los controles de autorización y registro, causando violaciones a la privacidad

# Top-Ten Vulnerabilidades según OWASP

**A2**

**Pérdida de Autenticación y  
Gestión de Sesiones**

 <p>Agentes de Amenaza</p>	 <p>Vectores de Ataque</p>	 <p>Debilidades de Seguridad</p>		 <p>Impactos Técnicos</p>	 <p>Impactos al negocio</p>
Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia DIFUNDIDO	Detección PROMEDIO	Impacto SEVERO	Específico de la aplicación/negocio
<p>Considere atacantes anónimos externos, así como a usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros. Considere también a trabajadores que quieran enmascarar sus acciones.</p>	<p>El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (ej. cuentas expuestas, contraseñas, identificadores de sesión) para suplantar otros usuarios.</p>	<p>Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero construirlos en forma correcta es difícil. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en el cierre de sesión, gestión de contraseñas, tiempo de desconexión (expiración), función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil ya que cada implementación es única.</p>		<p>Estas vulnerabilidades pueden permitir que algunas o <u>todas</u> las cuentas sean atacadas. Una vez que el ataque resulte exitoso, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son objetivos prioritarios.</p>	<p>Considere el valor de negocio de los datos afectados o las funciones de la aplicación expuestas.</p> <p>También considere el impacto en el negocio de la exposición pública de la vulnerabilidad.</p>



# Top-Ten Vulnerabilidades según OWASP

**A3**

## Secuencia de Comandos en Sitios Cruzados (XSS)

Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada.

XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

---

# Top-Ten Vulnerabilidades según OWASP

**A3**

Secuencia de Comandos en Sitios  
Cruzados (XSS)

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia MUY DIFUNDIDA	Detección FACIL	Impacto MODERADO	Específico de la aplicación / negocio
<p>Considere cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.</p>	<p>El atacante envía cadenas de texto que son secuencias de comandos de ataque que explotan el intérprete del navegador. Casi cualquier fuente de datos puede ser un vector de ataque, incluyendo fuentes internas tales como datos de la base de datos.</p>	<p><u>XSS</u> es la falla de seguridad predominante en aplicaciones web. Ocurren cuando una aplicación, en una página enviada a un navegador incluye datos suministrados por un usuario sin ser validados o codificados apropiadamente. Existen tres tipos de fallas conocidas XSS: 1) <u>Almacenadas</u>, 2) <u>Reflejadas</u>, y 3) <u>basadas en DOM</u>.</p> <p>La mayoría de las fallas XSS son detectadas de forma relativamente fácil a través de pruebas o por medio del análisis del código.</p>		<p>El atacante puede ejecutar secuencias de comandos en el navegador de la víctima para secuestrar las sesiones de usuario, alterar la apariencia del sitio web, insertar código hostil, redirigir usuarios, secuestrar el navegador de la víctima utilizando malware, etc.</p>	<p>Considere el valor para el negocio del sistema afectado y de los datos que éste procesa.</p> <p>También considere el impacto en el negocio la exposición pública de la vulnerabilidad.</p>

# Top-Ten Vulnerabilidades según OWASP

**A4**

## Referencia directa insegura a objetos




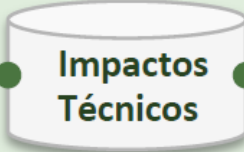

Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio o base de datos.

Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.

# Top-Ten Vulnerabilidades según OWASP

**A4**

Referencia directa insegura a objetos

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la aplicación/negocio
<p>Considere los tipos de usuarios en su sistema. ¿Existen usuarios que tengan únicamente acceso parcial a determinados tipos de datos del sistema?</p>	<p>Un atacante, como usuario autorizado en el sistema, simplemente modifica el valor de un parámetro que se refiere directamente a un objeto del sistema por otro objeto para el que el usuario no se encuentra autorizado. ¿Se concede el acceso?</p>	<p>Normalmente, las aplicaciones utilizan el nombre o clave actual de un objeto cuando se generan las páginas web. Las aplicaciones no siempre verifican que el usuario tiene autorización sobre el objetivo. Esto resulta en una vulnerabilidad de referencia de objetos directos inseguros. Los auditores pueden manipular fácilmente los valores del parámetro para detectar estas vulnerabilidades. Un análisis de código muestra rápidamente si la autorización se verifica correctamente.</p>		<p>Dichas vulnerabilidades pueden comprometer toda la información que pueda ser referida por parámetros. A menos que el espacio de nombres resulte escaso, para un atacante resulta sencillo acceder a todos los datos disponibles de ese tipo.</p>	<p>Considere el valor de negocio de los datos afectados o las funciones de la aplicación expuestas.</p> <p>También considere el impacto en el negocio de la exposición pública de la vulnerabilidad.</p>

# Top-Ten Vulnerabilidades según OWASP

**A5**

## Configuración de Seguridad Incorrecta


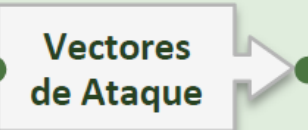



Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto.

Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.

# Top-Ten Vulnerabilidades según OWASP

**A5**

**Configuración de Seguridad  
Incorrecta**

 <p>Agentes de Amenaza</p>	 <p>Vectores de Ataque</p>	 <p>Debilidades de Seguridad</p>		 <p>Impactos Técnicos</p>	 <p>Impactos al negocio</p>
Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la aplicación / negocio
<p>Considere atacantes anónimos externos así como usuarios con sus propias cuentas que pueden intentar comprometer el sistema. También considere personal interno buscando enmascarar sus acciones.</p>	<p>Un atacante accede a cuentas por defecto, páginas sin uso, fallas sin parchear, archivos y directorios sin protección, etc. para obtener acceso no autorizado o conocimiento del sistema.</p>	<p>Las configuraciones de seguridad incorrectas pueden ocurrir a cualquier nivel de la aplicación, incluyendo la plataforma, servidor web, servidor de aplicación, base de datos, framework, y código personalizado. Los desarrolladores y administradores de sistema necesitan trabajar juntos para asegurar que las distintas capas están configuradas apropiadamente. Las herramientas de detección automatizadas son útiles para detectar parches omitidos, fallos de configuración, uso de cuentas por defecto, servicios innecesarios, etc.</p>		<p>Estas vulnerabilidades frecuentemente dan a los atacantes acceso no autorizado a algunas funcionalidades o datos del sistema. Ocasionalmente provocan que el sistema se comprometa totalmente.</p>	<p>El sistema podría ser completamente comprometido sin su conocimiento. Todos sus datos podrían ser robados o modificados lentamente en el tiempo.</p> <p>Los costes de recuperación podrían ser altos.</p>

# Top-Ten Vulnerabilidades según OWASP

**A6**

## Exposición de datos sensibles

Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas 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.



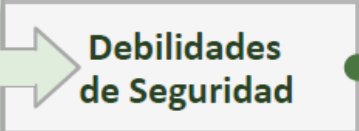


Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.



# Top-Ten Vulnerabilidades según OWASP

**A6**

Exposición de datos sensibles

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad DIFÍCIL	Prevalencia NO COMÚN	Detección PROMEDIO	Impacto SEVERO	Específico de la Aplicación/Negocio
<p>Considere quién puede obtener acceso a sus datos sensibles y cualquier respaldo de éstos. Esto incluye los datos almacenados, en tránsito, e inclusive en el navegador del cliente. Incluye tanto amenazas internas y externas.</p>	<p>Los atacantes típicamente no quiebran la criptografía de forma directa, sino algo más como robar claves, realizar ataques “man in the middle”, robar datos en texto claro del servidor, mientras se encuentran en tránsito, o del navegador del usuario.</p>	<p>La debilidad más común es simplemente no cifrar datos sensibles. Cuando se emplea cifrado, es común detectar generación y gestión débiles de claves, el uso de algoritmos débiles, y particularmente técnicas débiles de hashing de contraseñas. Las debilidades a nivel del navegador son muy comunes y fáciles de detectar, pero difíciles de explotar a gran escala. Atacantes externos encuentran dificultades detectando debilidades en a nivel de servidor dado el acceso limitado y que son usualmente difíciles de explotar.</p>		<p>Los fallos frecuentemente comprometen todos los datos que deberían estar protegidos. Típicamente, esta información incluye datos sensibles como ser registros médicos, credenciales, datos personales, tarjetas de crédito, etc.</p>	<p>Considere el valor de negocio de la pérdida de datos y el impacto a su reputación. ¿Cuál su responsabilidad legal si estos datos son expuestos? También considere el daño a la reputación.</p>



# Top-Ten Vulnerabilidades según OWASP

**A7**

## Inexistente Control de Acceso a nivel de funcionalidades

La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma 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.

# Top-Ten Vulnerabilidades según OWASP

**A7**

**Inexistente Control de Acceso  
a nivel de funcionalidades**

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad FÁCIL	Prevalencia COMÚN	Detección PROMEDIO	Impacto MODERADO	Específico de la aplicación/negocio
<p>Cualquiera con acceso a la red puede enviar una petición a su aplicación. ¿Un usuario anónimo podría acceder a una funcionalidad privada o un usuario normal acceder a una función que requiere privilegios?</p>	<p>El atacante, que es un usuario legítimo en el sistema, simplemente cambia la URL o un parámetro a una función con privilegios. ¿Se le concede acceso? Usuarios anónimos podrían acceder a funcionalidades privadas que no estén protegidas.</p>	<p>Las aplicaciones no siempre protegen las funcionalidades adecuadamente. En ocasiones la protección a nivel de funcionalidad se administra por medio de una configuración, y el sistema está mal configurado. Otras veces los programadores deben incluir un adecuado chequeo por código, y se olvidan.</p> <p>La detección de este tipo de vulnerabilidad es sencillo. La parte más compleja es identificar qué páginas (URLs) o funcionalidades atacables existen.</p>		<p>Estas vulnerabilidades permiten el acceso no autorizado de los atacantes a funciones del sistema.</p> <p>Las funciones administrativas son un objetivo clave de este tipo de ataques.</p>	<p>Considere el valor para su negocio de las funciones expuestas y los datos que éstas procesan. Además, considere el impacto a su reputación si esta vulnerabilidad se hiciera pública.</p>

# Top-Ten Vulnerabilidades según OWASP

**A8**



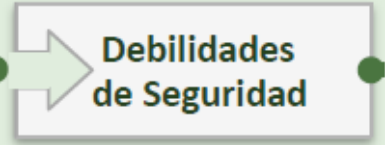
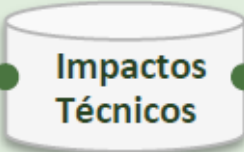
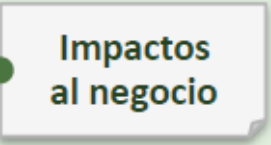
## Falsificación de Peticiones en Sitios Cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificado, 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 pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la víctima.

# Top-Ten Vulnerabilidades según OWASP

**A8**

Falsificación de Peticiones en Sitios  
Cruzados (CSRF)

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la aplicación/negocio
<p>Considere cualquier persona que pueda cargar contenido en los navegadores de los usuarios, y así obligarlos a presentar una solicitud para su sitio web. Cualquier sitio web o canal HTML que el usuario acceda puede realizar este tipo de ataque.</p>	<p>El atacante crea peticiones HTTP falsificadas y engaña a la víctima mediante el envío de etiquetas de imágenes, XSS u otras técnicas. <u>Si el usuario está autenticado</u>, el ataque tiene éxito.</p>	<p>CSRF aprovecha el hecho que la mayoría de las aplicaciones web permiten a los atacantes predecir todos los detalles de una acción en particular. Dado que los navegadores envían credenciales como cookies de sesión de forma automática, los atacantes pueden crear páginas web maliciosas que generan peticiones falsificadas que son indistinguibles de las legítimas. La detección de fallos de tipo CSRF es bastante fácil a través de pruebas de penetración o de análisis de código.</p>		<p>Los atacantes pueden cambiar cualquier dato que la víctima esté autorizada a cambiar, o a acceder a cualquier funcionalidad donde esté autorizada, incluyendo registro, cambios de estado o cierre de sesión.</p>	<p>Considerar el valor de negocio asociado a los datos o funciones afectados. Tener en cuenta lo que representa no estar seguro si los usuarios en realidad desean realizar dichas acciones. Considerar el impacto que tiene en la reputación de su negocio.</p>

# Top-Ten Vulnerabilidades según OWASP

**A9**

## Uso de Componentes con Vulnerabilidades Conocidas

Algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre funcionan con todos los privilegios.




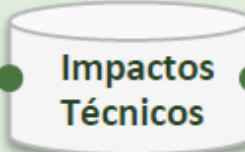

Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida seria de datos.

Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.

# Top-Ten Vulnerabilidades según OWASP

**A9**

Uso de Componentes con  
Vulnerabilidades Conocidas

 Agentes de Amenaza	 Vectores de Ataque	 Debilidades de Seguridad		 Impactos Técnicos	 Impactos al negocio
Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia DIFUNDIDO	Detectabilidad DIFÍCIL	Impacto MODERADO	Específico de la aplicación / negocio
<p>Algunos componentes vulnerables (por ejemplo frameworks) pueden ser identificados y explotados con herramientas automatizadas, aumentando las opciones de la amenaza más allá del objetivo atacado.</p>	<p>El atacante identifica un componente débil a través de escaneos automáticos o análisis manuales. Ajusta el exploit como lo necesita y ejecuta el ataque. Se hace más difícil si el componente es ampliamente utilizado en la aplicación.</p>	<p>Virtualmente cualquier aplicación tiene este tipo de problema debido a que la mayoría de los equipos de desarrollo no se enfocan en asegurar que sus componentes / bibliotecas se encuentren actualizadas. En muchos casos, los desarrolladores no conocen todos los componentes que utilizan, y menos sus versiones. Dependencias entre componentes dificultan incluso más el problema.</p>		<p>El rango completo de debilidades incluye inyección, control de acceso roto, XSS, etc. El impacto puede ser desde mínimo hasta apoderamiento completo del equipo y compromiso de los datos.</p>	<p>Considere qué puede significar cada vulnerabilidad para el negocio controlado por la aplicación afectada. Puede ser trivial o puede significar compromiso completo.</p>



# Top-Ten Vulnerabilidades según OWASP

**A10**






## Redirecciones y reenvíos no válidos

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 páginas no autorizadas.

# Top-Ten Vulnerabilidades según OWASP

## A10 Redirecciones y reenvíos no válidos

 <p>Agentes de Amenaza</p>	 <p>Vectores de Ataque</p>	 <p>Debilidades de Seguridad</p>		 <p>Impactos Técnicos</p>	 <p>Impactos al negocio</p>
Específico de la Aplicación	Explotabilidad PROMEDIO	Prevalencia POCO COMÚN	Detección FÁCIL	Impacto MODERADO	Específico de la Aplicación / Negocio
<p>Considere la probabilidad de que alguien pueda engañar a los usuarios a enviar una petición a su aplicación web. Cualquier aplicación o código HTML al que acceden sus usuarios podría realizar este engaño</p>	<p>Un atacante crea enlaces a redirecciones no validadas y engaña a las víctimas para que hagan clic en dichos enlaces. Las víctimas son más propensas a hacer clic sobre ellos ya que el enlace lleva a una aplicación de confianza. El atacante tiene como objetivo los destinos inseguros para evadir los controles de seguridad.</p>	<p>Con frecuencia, las aplicaciones redirigen a los usuarios a otras páginas, o utilizan destinos internos de forma similar. Algunas veces la página de destino se especifica en un parámetro no validado, permitiendo a los atacantes elegir dicha página.</p> <p>Detectar redirecciones sin validar es fácil. Se trata de buscar redirecciones donde el usuario puede establecer la dirección URL completa. Verificar reenvíos sin validar resulta más complicado ya que apuntan a páginas internas.</p>		<p>Estas redirecciones pueden intentar instalar código malicioso o engañar a las víctimas para que revelen contraseñas u otra información sensible. El uso de reenvíos inseguros puede permitir evadir el control de acceso.</p>	<p>Considere el valor de negocio de conservar la confianza de sus usuarios.</p> <p>¿Qué pasaría si sus usuarios son infectados con código malicioso?</p> <p>¿Qué ocurriría si los atacantes pudieran acceder a funciones que sólo debieran estar disponibles de forma interna?</p>