TAREA 3)

El pentesting o test de intrusión

Sonia Salido

Índice para la petición de la rúbrica

Índice general

Ejercicio 1 → WSTG

OWASP son las siglas de **Open Web Application Security Project (Proyecto Abierto de Seguridad de Aplicaciones Web)**. Nació en 2001 y en 2004 se constituyó como fundación.

El objetivo de la OWASP es ser una comunidad global dedicada a **dar visibilidad sobre la seguridad en el mundo del software**.

Proyectos más importantes de la OWASP:

- **OWASP Top 10**: La metodología OWASP Top 10 se basa en un TOP 10 de vulnerabilidades en aplicaciones web. Al generar este listado de las debilidades más populares permite a los profesionales de la ciberseguridad poner a prueba sus sistemas ejecutando diferentes técnicas para la explotación de las vulnerabilidades informadas por AWASP.

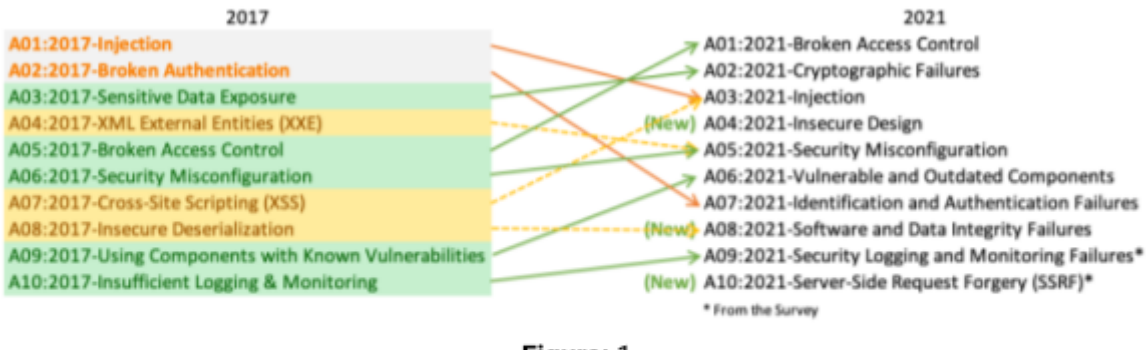


Figura: 1
Fuente: Top 10 Web Application Security Risks
url: https://owasp.org/www-project-top-ten/

Variación del ranking de riesgos para la web desde 2013 a 2017 y 2020.

En la guía del OWASP Top 10 también se tiene en cuenta el desarrollo seguro, pero desde luego su enfoque se centra en la definición de actores, amenazas, factores de riesgo, prevalencia de las vulnerabilidades y su explotabilidad. Esto significa que no especifica con demasiada profundidad las cuestiones técnicas que hay que considerar a la hora de crear software y probar sus funcionalidades desde el punto de vista de la calidad y seguridad.

- **Juice Shop**: Proyecto para favorecer el aprendizaje y la experimentación en entornos seguros.

- **WSTG (Web Security Testing Guide)**. Es un manual bastante completo que no solamente cubre cuestiones relacionadas con el pentesting en aplicaciones web, sino que además su enfoque es integrador y pretende aportar una visión global de la seguridad de una aplicación web desde sus cimientos, considerando el desarrollo seguro como uno de los procesos más importantes (o puede que el más importante) en la seguridad de cualquier aplicación web.

WSTG Características + importantes

La guía WSTG se encuentra disponible en:
https://owasp.org/www-project-web-security-testing-guide/

Nace en el año 2013 como una guía complementaria al OWASP TOP 10 y que "reemplazó" a las guías OTG (guías genéricas de prueba). Este documento provee información más específica y técnica sobre los controles de seguridad que se deben contemplar a la hora de realizar pentesting en aplicaciones web. Además, a la WSTG se le suma un complemento llamado OWASP Web Application Penetration Checklist que nos ayudará a contabilizar y categorizar los controles de seguridad que debemos ejecutar según OWASP TOP 10 y la información de la WSTG.

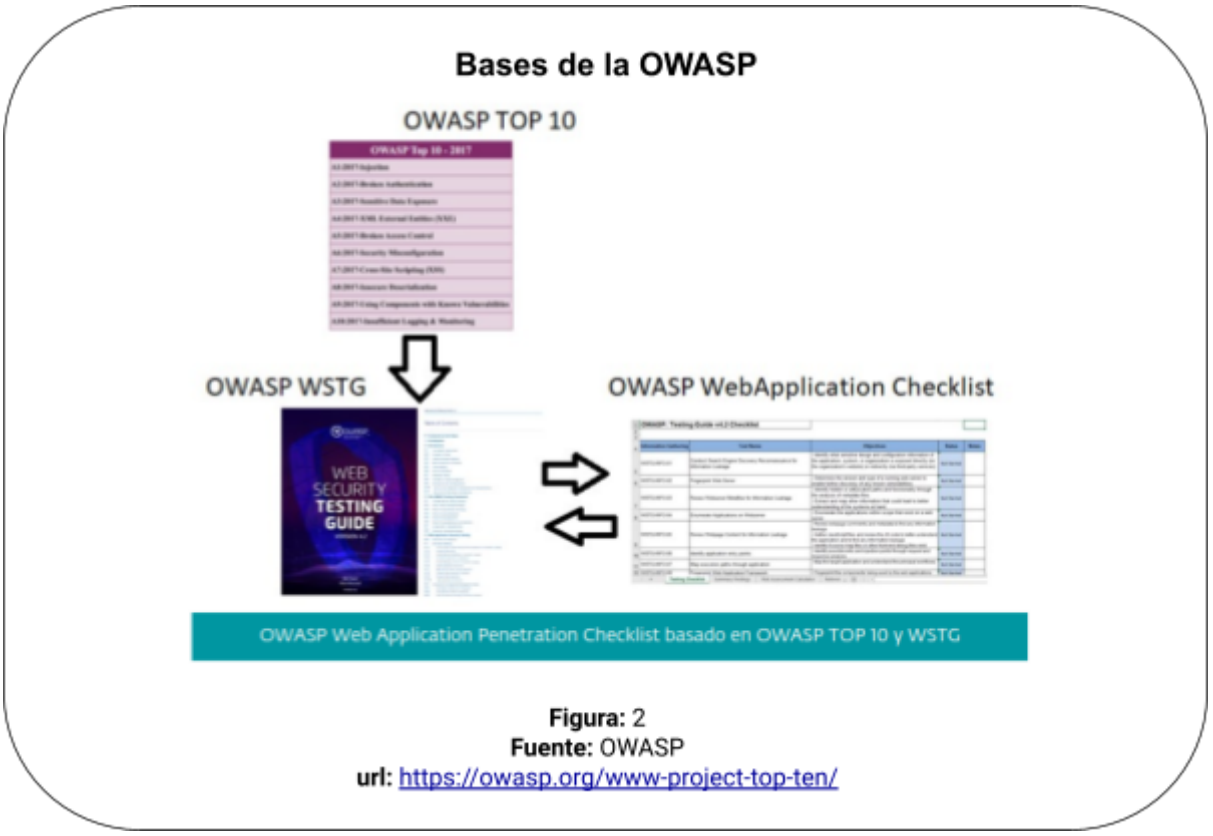


Figura: 2
Fuente: OWASP
url: https://owasp.org/www-project-top-ten/

El proyecto Web Security Testing Guide (WSTG) produce el **principal recurso de pruebas de ciberseguridad** para desarrolladores de aplicaciones web y profesionales de la seguridad.

El WSTG es una **guía completa para probar la seguridad de las aplicaciones web y los servicios web.** Creado por los esfuerzos de colaboración de profesionales de la seguridad cibernética y voluntarios dedicados, el WSTG proporciona un marco de mejores prácticas utilizado por evaluadores de penetración y organizaciones de todo el mundo.

(Volver al Índice)

Contenidos y estructura de la WSTG

En la introducción se explica que esta guía se trata de una guía completa con **buenas prácticas para desarrolladores y profesionales de la seguridad** que se deben de tener en cuenta para cumplir con su rol. Se plantean una serie de cuestiones importantes desde el punto de vista técnico y funcional como ocurre en el OWASP Top 10, pero con un nivel de detalle mucho más amplio tal como se puede ver en el índice y en cada uno de los apartados de la guía. Las secciones del documento se detallan a continuación y un breve resumen sobre qué incluyen.

0.- Prefacio.

Introducción de la importancia de la seguridad, de la Open Web Application Security Project (OWASP).

1.- Frontispiece.

Es una sección muy corta en la que detalla una pequeña introducción sobre la guía, su licencia Creative Commons 4.0 y las personas que aportan al proyecto. Tal como se puede apreciar, hay aproximadamente unas 50 personas que colaboran en los contenidos de la guía.

(Volver al Índice)

2.- Introducción.

En esta sección se detallan conceptos básicos sobre desarrollo de software y revisión de código, así como metodologías orientadas al SDLC (ciclo de vida del desarrollo del software) y el enfoque que se le debe dar a las pruebas de intrusión en entornos web.

¿Cuándo hacer los testing?

La mayoría de las personas hoy en día no prueban el software hasta que ya se ha creado y se encuentra en la fase de implementación de su ciclo de vida (es decir, el código se ha creado y se ha creado una instancia en una aplicación web funcional). Esta es generalmente una práctica muy ineficaz y de costo prohibitivo. Uno de los mejores métodos para evitar que aparezcan errores de seguridad en las aplicaciones de producción es mejorar el Ciclo de Vida de Desarrollo de Software (SDLC) al **incluir la seguridad en cada una de sus fases.**

¿Qué probar?

Un programa de pruebas efectivo debe tener componentes que prueben lo siguiente:

- **Personas:** para garantizar que haya una educación y una conciencia adecuadas.
- **Proceso:** para garantizar que existan políticas y estándares adecuados y que las personas sepan cómo seguir estas políticas.
- **Tecnología:** para asegurar que el proceso ha sido efectivo en su implementación.

A menos que se adopte un enfoque holístico, **probar sólo la implementación técnica de una aplicación no descubrirá las vulnerabilidades operativas o de administración** que podrían estar presentes. Al probar a las personas, las políticas y los procesos, una organización puede detectar problemas que luego se manifestaría en defectos en la tecnología, erradicando así los errores temprano e identificando las causas fundamentales de los defectos. Del mismo modo, probar sólo algunos de los problemas técnicos que pueden estar presentes en un sistema dará como resultado una evaluación de la postura de seguridad incompleta e inexacta.

Cómo se referencia los diferentes escenarios la WSTG :

Cada escenario tiene un identificador en el formato WSTG-<version>-<categoría>-<número>, dónde: 'categoría' es una cadena de 4 caracteres en mayúsculas que identifica el tipo de prueba o debilidad, y 'número' es un valor numérico con relleno cero de 01 a 99. Por ejemplo: WSTG-INFO-02 es la segunda prueba de recopilación de información.

(Volver al Índice)

3.- OWASP Testing Framework.

En esta sección de la guía se describe el framework de testing que se puede implementar, definiendo cada una de las etapas de construcción del software, empezando por las fases iniciales correspondientes a la recolección de requisitos y diseño pasando por las etapas de desarrollo y despliegue hasta la puesta en producción, operaciones y mantenimiento del software. Esta es una de las secciones más importantes de la guía ya que en ella se definen una serie de buenas prácticas que, cualquiera que haya estado en un equipo de desarrollo, ha visto como muchas de ellas NO se aplican o se aplican de manera incompleta/incorrecta.

Fases en las que se distribuye la WSTG y actividades más importantes que se desarrollan en cada una de esas fases.

1. **Fase 1 →ANTES de que comience el desarrollo.**
    a. **Fase 1.1 → Definir un SLC**: Antes de que comience el desarrollo de la aplicación, se debe definir un SDLC adecuado donde la seguridad sea inherente en cada etapa.

    b. **Fase 1.2 →Revisar políticas y estándares**: Debe existir políticas, estándares y documentación adecuados.

    c. **Fase 1.3 → Desarrollar criterios de medición y métricas y garantizar la trazabilidad**: Al definir los criterios que deben medirse, proporciona visibilidad de los defectos, tanto en el proceso como en el producto.

2. **Fase 2 →Durante la definición y el diseño:**
    a. **Fase 2.1 →Revisar los requisitos de seguridad**: Los requisitos de seguridad definen cómo funciona una aplicación desde una perspectiva de seguridad.
    Si buscamos brechas en los requisitos, debemos buscar mecanismos de seguridad como:
        - Gestión de usuarios
        - Autenticación
        - Autorización
        - Confidencialidad de los datos
        - Integridad
        - Responsabilidad
        - Gestión de sesiones
        - Seguridad del transporte
        - Segregación del sistema por niveles
        - Cumplimiento normativo (incluidos los estándares de privacidad, gubernamentales e industriales)

b. **Fase 2.2 → Revisar Diseño y Arquitectura**: Debemos garantizar que el diseño y la arquitectura apliquen el nivel adecuado de seguridad según lo definido en sus requisitos.

c. **Fase 2.3 → Crear y revisar modelos UML**: Una vez que el diseño y la arquitectura estén completos, cree modelos de lenguaje de modelado unificado (UML) que describan cómo funciona la aplicación.

d. **Fase 2.4 → Crear y revisar modelos de amenazas**: Desarrolle escenarios de amenazas realistas.

3. **Fase 3 → Durante el desarrollo.**
   a. **Fase 3.1 → Tutorial de código**: Permite que el equipo de revisión de código obtenga una comprensión general del código y permite a los desarrolladores explicar por qué desarrollaron ciertas cosas y por qué se hizo de cierta manera.

   b. **Fase 3.2 →Revisión de código**: Examinar el código real en busca de defectos de seguridad.

4. **Fase 4 → Durante el despliegue.**
   a. **Fase 4.1 → Pruebas de penetración de aplicaciones**.

   b. **Fase 4.2 → Pruebas de gestión de la configuración**.

5. **Fase 5 → Durante el mantenimiento y las operaciones.**
   a. **Fase 5.1 → Realizar revisiones de gestión operativa**: Es necesario que exista un proceso que detalle cómo se administra el lado operativo de la aplicación y la infraestructura.

   b. **Fase 5.2 → Realice controles de salud periódicos**: Se deben realizar verificaciones de estado mensuales o trimestrales tanto en la aplicación como en la infraestructura para garantizar que no se hayan introducido nuevos riesgos de seguridad y que el nivel de seguridad siga intacto.

   c. **Fase 5.3 → Garantizar la verificación de cambios**: Después de que cada cambio haya sido aprobado y probado en el entorno de control de calidad e implementado en el entorno de producción, es vital que el cambio se verifique para garantizar que el nivel de seguridad no se haya visto afectado por el cambio.

(Volver al Índice)

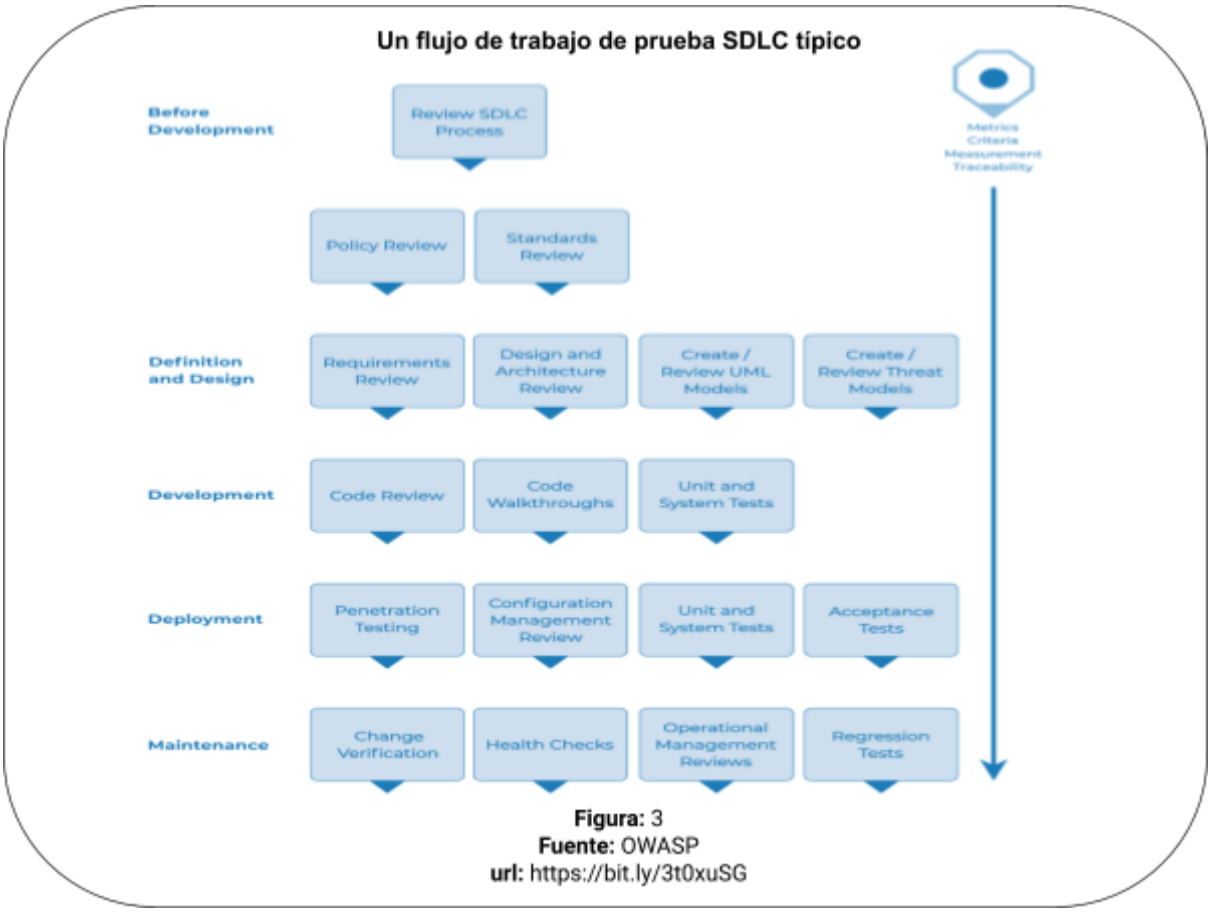


Figura: 3
Fuente: OWASP
url: https://bit.ly/3t0xuSG

(Volver al Índice)

4.- Web Application Security Testing.

En esta sección es la más extensa de todo el documento y precisamente la que más valor aporta a un técnico, ya que se detallan las pruebas que se deben realizar en 12 secciones claramente definidas y acotadas. **Las pruebas que se indican se pueden aplicar a cualquier aplicación web** independiente de su tamaño, diseño o funcionalidad y van más allá de las típicas pruebas de seguridad que se definen en el OWASP Top 10. Estas secciones describen con todo lujo de detalle, lo que hace cualquier pentester, cuándo debe ejecutar una auditoría web, con casos prácticos, herramientas y ejemplos muy bien documentados.

Las **secciones con las herramientas de monitorización para detectar vulnerabilidades y tipos de ataque** son las siguientes:

Estructura del capítulo: "Web Application Security Testing"
"

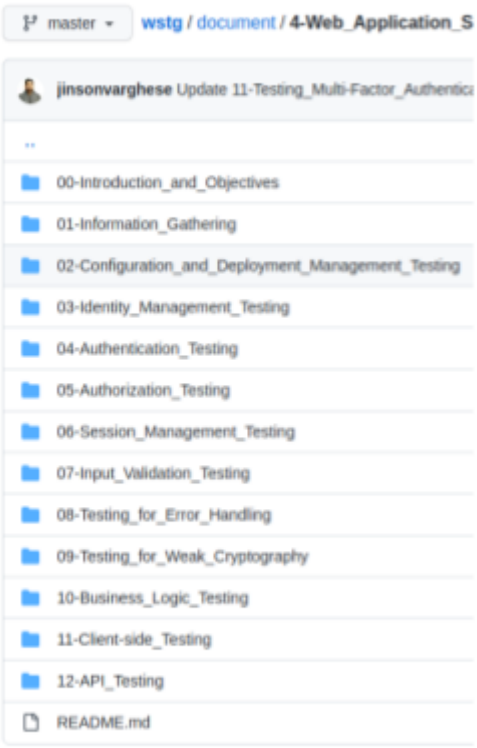


Figura: 4
Fuente: OWASP
url: https://bit.ly/3t0xuSG

4.0 Introducción y Objetivos.

4.1 Recopilación de Información.

4.1.1 Llevar a cabo un reconocimiento de descubrimiento de motores de búsqueda para detectar fugas de información

4.1.2 Servidor web de huellas digital. Técnica banner grabbing: netcat, nc. Escaneo de puertos: netcraft, nikto, nmap

4.1.3 Revisar los metarchivos del servidor web para detectar fugas de información. curl, wget, burp suite, zap, analizar robots.txt usando Google Webmaster Tools.

4.1.4 Enumerar aplicaciones en el servidor web. nslookup, Search engines (Google, Bing), Nmap, Nessus Vulnerability Scanner, Nikto.

4.1.5 Revisar el contenido de la página web para detectar fugas de información. Wget, Browser "view source" function, Eyeballs, Curl, Zaproxy, Burp Suite
Waybackurls, Google Maps API Scanner

4.1.6 Identificar los puntos de entrada de la aplicación. **OWASP Zed Attack Proxy (ZAP)**, OWASP Attack Surface Detector, Burp Suite, Fiddler.

4.1.7 Asignar rutas de ejecución a través de la aplicación. OWASP Zed Attack Proxy (ZAP), List of spreadsheet software, Diagramming software.

4.1.8 Marco de aplicaciones web de huellas digitales. Wappalyzer

4.1.9 Mapa de la arquitectura de la aplicación.

4.2 Pruebas de gestión de configuración e implementación.

4.2.1 Prueba de la configuración de la infraestructura de red

4.2.2 Prueba de la configuración de la plataforma de la aplicación

4.2.3 Prueba del manejo de extensiones de archivo para información confidencial. wget, curl, google para “web mirroring tools”.

4.2.4 Revisar copias de seguridad antiguas y archivos sin referencia en busca de información confidencial. nessus, nikto2, wget, curl, spike proxy.

(Volver al Índice)

4.2.5 Enumerar interfaces de administración de aplicaciones e infraestructura. OWASP zap, thc-hydra, netsparker dictionary.

4.2.6 Prueba de métodos HTTP. ncat, curl, nmap.

4.2.7 Prueba de la seguridad de transporte estricta de HTTP. comando curl.

4.2.8 Permiso de archivo de prueba. comando namei, Windows AccessEnum, Windows AccessChk.

4.2.9 Prueba de adquisición de subdominio. dig, recon-ng, theHarvester, Sublist3r.

4.2.10 Prueba de almacenamiento en la nube. aws cli.

4.2.11 Prueba de la política de seguridad de contenido. Google CSP Evaluator, CSP Auditor - Burp Suite Extension, CSP Generator Chrome / Firefox.

4.3 Identity Management Testing

4.3.1 Test Role Definitions

4.3.2 Test User Registration Process

4.3.3 Test Account Provisioning Process

4.3.4 Testing for Account Enumeration and Guessable User Account

4.3.5 Testing for Weak or Unenforced Username Policy

4.4 Pruebas de autenticación

4.4.1 Prueba de credenciales transportadas a través de un canal cifrado

4.4.2 Prueba de credenciales predeterminadas. Burp Intruder, THC Hydra, Nikto2.

4.4.3 Prueba de mecanismo de bloqueo débil. Intentos erróneos de login. Técnicas de capcha.

4.4.4 Pruebas para eludir el esquema de autenticación. WebGoat, OWASP Zed Attack Proxy (ZAP).

4.4.5 Prueba para recordar contraseña vulnerable. ClickJacking attacks, CSRF attacks.

4.4.6 Comprobación de las debilidades de la memoria caché del navegador. OWASP Zed Attack Proxy.

4.4.7 Prueba de política de contraseña débil. Número de caracteres permitidos. Tipo de caracteres.

4.4.8 Prueba de respuesta de pregunta de seguridad débil

4.4.9 Prueba de funcionalidades de cambio o restablecimiento de contraseña débil

4.4.10 Prueba de autenticación más débil en canal alternativo

4.4.11 Prueba de autenticación de múltiples factores

4.5 Pruebas de autorización.

4.5.1 Prueba de inclusión de archivos transversales de directorios

4.5.2 Pruebas para eludir el esquema de autorización

4.5.3 Pruebas de escalamiento de privilegios

4.5.4 Pruebas de referencias a objetos directos inseguros

4.5.5 Pruebas de debilidades de OAuth

4.5.5.1 Prueba de debilidades del servidor de autorización de OAuth

4.5.5.2 Prueba de debilidades del cliente OAuth

4.6 Pruebas de gestión de sesiones.

4.6.1 Prueba del esquema de gestión de sesiones

4.6.2 Prueba de atributos de cookies

4.6.3 Prueba de Fijación de Sesión

4.6.4 Prueba de variables de sesión expuestas

4.6.5 Prueba de falsificación de solicitudes entre sitios

4.6.6 Prueba de funcionalidad de cierre de sesión

4.6.7 Tiempo de espera de la sesión de prueba

4.6.8 Pruebas de desconcierto de sesión

4.6.9 Prueba de secuestro de sesión

4.6.10 Prueba de JSON WEB Tokens.

4.7 Input Validation Testing

4.7.1 Testing for Reflected Cross Site Scripting

4.7.2 Testing for Stored Cross Site Scripting

4.7.3 Testing for HTTP Verb Tampering

4.7.4 Testing for HTTP Parameter Pollution

4.7.5 Testing for SQL Injection

- 4.7.5.1 Testing for Oracle
- 4.7.5.2 Testing for MySQL
- 4.7.5.3 Testing for SQL Server
- 4.7.5.4 Testing PostgreSQL
- 4.7.5.5 Testing for MS Access
- 4.7.5.6 Testing for NoSQL Injection
- 4.7.5.7 Testing for ORM Injection
- 4.7.5.8 Testing for Client-side

4.7.6 Testing for LDAP Injection

4.7.7 Testing for XML Injection

4.7.8 Testing for SSI Injection

4.7.9 Testing for XPath Injection

4.7.10 Testing for IMAP SMTP Injection

4.7.11 Testing for Code Injection

- 4.7.11.1 Testing for File Inclusion

4.7.12 Testing for Command Injection

4.7.13 Testing for Format String Injection

4.7.14 Testing for Incubated Vulnerability

4.7.15 Testing for HTTP Splitting Smuggling

4.7.16 Testing for HTTP Incoming Requests

4.7.17 Testing for Host Header Injection

4.7.18 Testing for Server-side Template Injection

4.7.19 Testing for Server-Side Request Forgery

4.7.20 Testing for Mass Assignment

4.8 Testing for Error Handling

4.8.1 Testing for Improper Error Handling

4.8.2 Testing for Stack Traces

4.9 Testing for Weak Cryptography

4.9.1 Testing for Weak Transport Layer Security

4.9.2 Testing for Padding Oracle

4.9.3 Testing for Sensitive Information Sent via Unencrypted Channels

4.9.4 Testing for Weak Encryption

(Volver al Índice)

4.10 Business Logic Testing

4.10.0 Introduction to Business Logic

4.10.1 Test Business Logic Data Validation

4.10.2 Test Ability to Forge Requests

4.10.3 Test Integrity Checks

4.10.4 Test for Process Timing

4.10.5 Test Number of Times a Function Can Be Used Limits

4.10.6 Testing for the Circumvention of Work Flows

4.10.7 Test Defenses Against Application Misuse

4.10.8 Test Upload of Unexpected File Types

4.10.9 Test Upload of Malicious Files

4.10.10 Test Payment Functionality

4.11 Client-Side Testing

4.11.1 Testing for DOM-Based Cross Site Scripting

- 4.11.1.1 Testing for Self DOM Based Cross Site Scripting

4.11.2 Testing for JavaScript Execution

4.11.3 Testing for HTML Injection

4.11.4 Testing for Client-side URL Redirect

4.11.5 Testing for CSS Injection

4.11.6 Testing for Client-side Resource Manipulation

4.11.7 Testing Cross Origin Resource Sharing

4.11.8 Testing for Cross Site Flashing

4.11.9 Testing for Clickjacking

4.11.10 Testing WebSockets

4.11.11 Testing Web Messaging

4.11.12 Testing Browser Storage

4.11.13 Testing for Cross Site Script Inclusion

4.11.14 Testing for Reverse Tabnabbing

4.12 API Testing

4.12.1 Testing GraphQL

5.- Reporting

Se habla de los detalles a tener en cuenta en el reporte que se debe entregar al finalizar una auditoría. Esta sección es interesante ya que indica cada uno de los elementos que incluye un informe profesional y orientado a enseñar la calidad de los trabajos realizados. Incluye algunas referencias en donde se explica cómo estructurar adecuadamente dicho documento que pueden ser útiles.

6. Apéndices

Por último se encuentran los apéndices en donde se enumeran un conjunto de herramientas básicas para hacking web, lecturas recomendadas, referencias externas a vectores/técnicas de ataque y utilidades tanto para desarrolladores como para pentesters.

Ejercicio 2 →Informe de auditoría

Análisis del Informe elegido

COMPREHENSIVE REPORT
BEAST
HYBRID APPLICATION ASSESSMENT 2017
BishopFox
DECEMBER 8, 2017
URL:
https://bit.ly/3t3thh7

En este informe se evalúa la seguridad de Boost C++ Beast que es una Biblioteca de redes HTTP/S. El autor sigue las indicaciones de la metodología OWASP en busca de vulnerabilidades, problemas críticos y de alto riesgo (especialmente problemas de corrupción de memoria) en la biblioteca Boost C++ Best.

El equipo de evaluación identificó los **siguientes problemas** como resultado de la evaluación en intervalos de tiempo de la versión 124 de la biblioteca Beast:

- Denegación de servicio: el equipo de evaluación demostró con éxito tres ataques de denegación de servicio contra Beast mediante el envío de tramas WebSocket con formato incorrecto que contenían una carga útil comprimida. Los problemas se identificaron al descifrar el código del servidor WebSocket responsable de descomprimir los mensajes de los clientes.

- Aleatoriedad insegura: Beast utiliza una fuente insuficiente de entropía como valor inicial para un generador lineal congruente (LCG) con el fin de generar valores aleatorios que sirvan como valor de enmascaramiento cuando se envían marcos de cliente de WebSocket. En circunstancias especiales, un atacante puede aprovechar este problema para envenenar las cachés HTTP proporcionadas por intermediarios mal implementados.

Funcionalidad de la biblioteca: la biblioteca Beast proporciona un conjunto de clases, derivadas de la popular biblioteca de red Boost ASIO y coherente con ella, para ayudar a los desarrolladores a escribir aplicaciones de red que sirvan a clientes HTTP/S o consuman servicios HTTP/S. Además de admitir los protocolos HTTP 1.0 y 1.1, la biblioteca también implementa la funcionalidad de comunicación WebSocket. La biblioteca Beast se basa en gran medida en las características del estándar C++11.

(Volver al Índice)

La biblioteca Beast no implementa funciones de seguridad específicas relacionadas con aplicaciones web comunes. Es responsabilidad de los consumidores de bibliotecas comprender las vulnerabilidades de seguridad de las aplicaciones web a las que pueden estar expuestos y desarrollar sus aplicaciones para dar cuenta de esos problemas. Esto incluye vulnerabilidades como secuencias de comandos en sitios cruzados (XSS), falsificación de solicitudes en sitios cruzados (CSRF), denegación de servicio en la capa de aplicación, configuración insegura de SSL/TLS, recorrido de rutas y divulgación de archivos, inyección de comandos, reparación de UI y vulnerabilidades relacionadas con el encabezado HTTP (por ejemplo, configuraciones de cookies no seguras, HSTS, CORS).

Recomendaciones para los desarrolladores si desean escribir aplicaciones que usan esta biblioteca:

- Realizar una estricta validación de los Inputs.
- Hacer compilaciones reforzadas con indicadores de seguridad en tiempo de compilación.
- Cumplir con los estándares SEI CERT C++ que son un gran número de reglas para escribir sistemas seguros y confiables.
- Evite exponer los detalles de implementación.
- Usar Encrypted WebSocket Connections.
- Estar actualizado con los últimos parches.

(Volver al Índice)

Fases de un ataque seguidas por un atacante

Figura: 5
Fuente: Incibe
url: https://www.incibe.es/protege-tu-empresa/blog/las-7-fases-ciberataque-las-conoces

Reconocimiento:

El reconocimiento tiene como objetivo la recopilación de toda la información posible del sistema que se va a atacar. Se divide en:

- Un reconocimiento pasivo es cuando se recopila información sin interactuar con el sistema.
- Un reconocimiento activo es cuando se recopila información interactuando con el sistema, como por ejemplo escaneando la red o se enumeran cuando dispositivos están conectados.

(Volver al Índice)

Preparación:

Una vez recopilada la información necesaria, el atacante prepara el ataque contra el objetivo marcado.

Distribución:

En esta etapa se produce la transmisión del ataque. La distribución depende del método de ataque se ejerza. Por ejemplo, si se va a usar un usb infectado la distribución es distinta de si el método de ataque es obtener las credenciales de un usuario.

Explotación:

Esta fase implica la «detonación» del ataque, comprometiendo al equipo infectado y a la red que pertenezca. En esta fase el atacante obtiene el acceso en el ordenador o en el servicio objetivo.

Instalación:

Fase en la que el atacante instala el malware en la víctima. También puede darse la circunstancia de que no se requiera instalación, como en el robo de credenciales o en el fraude del CEO. En esta fase la persona ciberatacante asegura el acceso en el equipo.

Comando y control

El atacante ya tiene el control del sistema y ejecuta las acciones que tenga pensadas, como por ejemplo, puede sustraer información confidencial, alterar o eliminar datos.

Acciones sobre los objetivos:

El atacante se hace con los datos e intenta expandir su acción maliciosa hacia más objetivos.

(Volver al Índice)

Presentación Diapositivas

(Volver al Índice)

Tarea 3
El pentesting o test de intrusión

Sonia Salido

Ejercicio 1 → WSTG

- OWASP son las siglas de **Open Web Application Security Project (Proyecto Abierto de Seguridad de Aplicaciones Web)**.

- **Proyectos más importantes de la OWASP:**
  - **OWASP Top 10.**
  - **Juice Shop.**
  - **WSTG (Web Security Testing Guide).**

WSTG Características + importantes

- **Nace en el año 2013** como una guía complementaria al OWASP TOP 10 y que “reemplazó” a las guías OTG (guías genéricas de prueba). Este documento provee información más específica y técnica sobre los controles de seguridad que se deben contemplar a la hora de realizar pentesting en aplicaciones web.

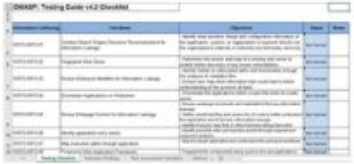
- El WSTG es una **guía completa para probar la seguridad de las aplicaciones web y los servicios web.** Creado por los esfuerzos de colaboración de profesionales de la seguridad cibernética y voluntarios dedicados, el WSTG proporciona un marco de mejores prácticas utilizado por evaluadores de penetración y organizaciones de todo el mundo.

Bases de la OWASP

OWASP TOP 10

OWASP WSTG

OWASP WebApplication Checklist



OWASP Web Application Penetration Checklist basado en OWASP TOP 10 y WSTG

Figura: 2
Fuente: OWASP
url: https://owasp.org/www-project-top-ten/

Contenidos y estructura de la WSTG

- Esta guía se trata de una guía completa con **buenas prácticas para desarrolladores y profesionales de la seguridad** que se deben de tener en cuenta para cumplir con su rol.

- Las secciones del documento se detallan a continuación y un breve resumen sobre qué incluyen
  - **Prefacio.**
  - **Frontispiece.**
  - **Introducción.**
    - **¿Cuándo hacer los testing?**
    - **¿Qué probar?**
    - **Cómo se referencia los escenarios: WSTG-v12-INFO-02**
  - **OWASP Testing Framework.**
  - **Web Application Security Testing.**
  - **Reporting.**
  - **Apéndices.**

Sección: OWASP Testing Framework

- Se describe el framework de testing que se puede implementar, definiendo cada una de las etapas de construcción del software, empezando por las fases iniciales correspondientes a la recolección de requisitos y diseño pasando por las etapas de desarrollo y despliegue hasta la puesta en producción, operaciones y mantenimiento del software

- Fases en las que se distribuye la WSTG y actividades más importantes que se desarrollan en cada una de esas fases:
  - **Fase 1 →ANTES de que comience el desarrollo:**
    - **Fase 1.1 → Definir un SLC.**
    - **Fase 1.2 →Revisar políticas y estándares.**
    - **Fase 1.3 → Desarrollar criterios de medición y métricas y garantizar la trazabilidad.**
  - **Fase 2 →Durante la definición y el diseño:**
    - **Fase 2.1 →Revisar los requisitos de seguridad.**
    - **Fase 2.2 → Revisar Diseño y Arquitectura.**
    - **Fase 2.3 → Crear y revisar modelos UML.**
    - **Fase 2.4 → Crear y revisar modelos de amenazas.**

Sección: OWASP Testing Framework

- Fases en las que se distribuye la WSTG y actividades más importantes que se desarrollan en cada una de esas fases:
  - **Fase 3 → Durante el desarrollo:**
    - **Fase 3.1 → Tutorial de código.**
    - **Fase 3.2 →Revisión de código.**
  - **Fase 4 → Durante el despliegue:**
    - **Fase 4.1 → Pruebas de penetración de aplicaciones.**
    - **Fase 4.2 → Pruebas de gestión de la configuración.**
  - **Fase 5 → Durante el mantenimiento y las operaciones:**
    - **Fase 5.1 → Realizar revisiones de gestión operativa.**
    - **Fase 5.2 → Realice controles de salud periódicos.**
    - **Fase 5.3 → Garantizar la verificación de cambios.**

Sección: OWASP Testing Framework

- Flujo de trabajo de una prueba SDLC:

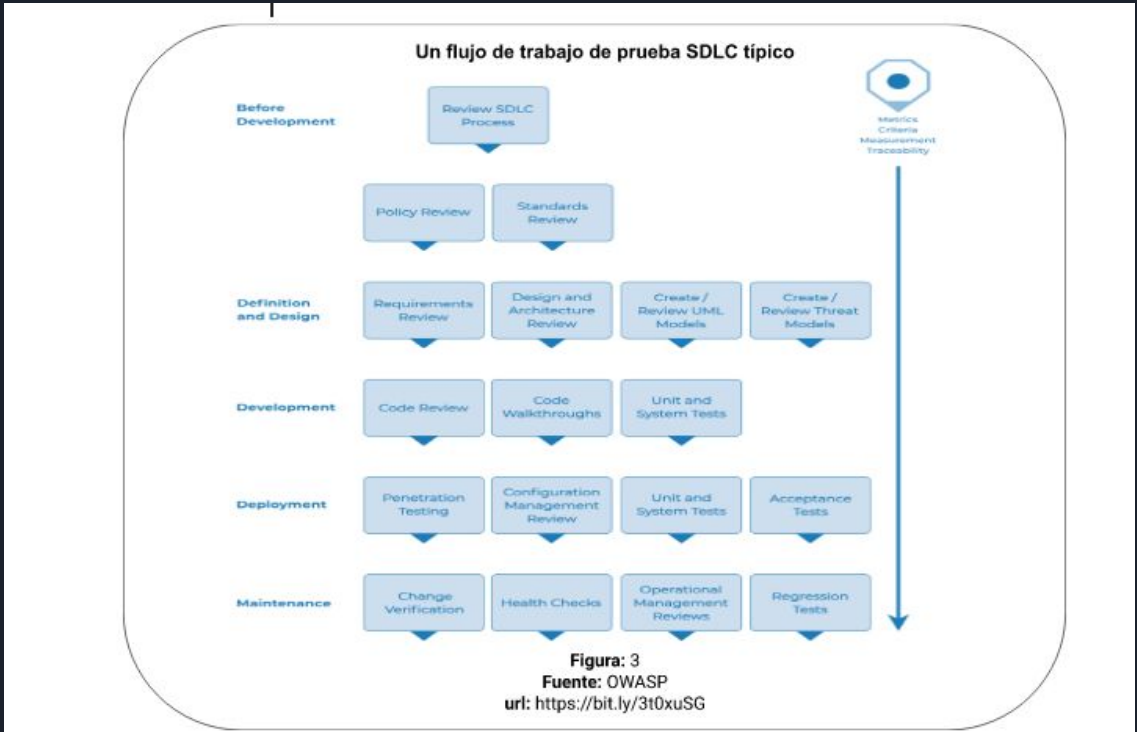


Figura: 3
Fuente: OWASP
url: https://bit.ly/3t0xuSG

Sección: Web Application Security Testing

- En esta sección se detallan las pruebas que se deben realizar en 12 secciones claramente definidas y acotadas. Las pruebas que se indican se pueden aplicar a cualquier aplicación web.
- Las secciones con las herramientas de monitorización para detectar vulnerabilidades y tipos de ataque son las siguientes:

4.0 Introducción y Objetivos.

4.1 Recopilación de Información.

4.1.1 Llevar a cabo un reconocimiento de descubrimiento de motores de búsqueda para detectar fugas de información

4.1.2 Servidor web de huellas digital. Técnica banner grabbing: netcat, nc. Escaneo de puertos: netcraft, nikto, nmap

4.1.3 Revisar los metarchivos del servidor web para detectar fugas de información. curl, wget, burp suite, zap, analizar robots.txt usando Google Webmaster Tools.

4.1.4 Enumerar aplicaciones en el servidor web. nslookup, Search engines (Google, Bing), Nmap, Nessus Vulnerability Scanner, Nikto.

Sección: Web Application Security Testing

4.1.5 Revisar el contenido de la página web para detectar fugas de información. Wget, Browser "view source" function, Eyeballs, Curl, Zaproxy, Burp Suite Waybackurls, Google Maps API Scanner

4.1.6 Identificar los puntos de entrada de la aplicación. **OWASP Zed Attack Proxy (ZAP)**, OWASP Attack Surface Detector, Burp Suite, Fiddler.

4.1.7 Asignar rutas de ejecución a través de la aplicación. OWASP Zed Attack Proxy (ZAP), List of spreadsheet software, Diagramming software.

4.1.8 Marco de aplicaciones web de huellas digitales. Wappalyzer

4.1.9 Mapa de la arquitectura de la aplicación.

4.2 Pruebas de gestión de configuración e implementación.

4.2.1 Prueba de la configuración de la infraestructura de red

4.2.2 Prueba de la configuración de la plataforma de la aplicación

4.2.3 Prueba del manejo de extensiones de archivo para información confidencial. wget, curl, google para “web mirroring tools”.

4.2.4 Revisar copias de seguridad antiguas y archivos sin referencia en busca de información confidencial. nessus, nikto2, wget, curl, spike proxy.

Sección: Web Application Security Testing

4.2.5 Enumerar interfaces de administración de aplicaciones e infraestructura. OWASP zap, thc-hydra, netsparker dictionary.

4.2.6 Prueba de métodos HTTP. ncat, curl, nmap.

4.2.7 Prueba de la seguridad de transporte estricta de HTTP. comando curl.

4.2.8 Permiso de archivo de prueba. comando namei, Windows AccessEnum, Windows AccessChk.

4.2.9 Prueba de adquisición de subdominio. dig, recon-ng, theHarvester, Sublist3r.

4.2.10 Prueba de almacenamiento en la nube. aws cli.

4.2.11 Prueba de la política de seguridad de contenido. Google CSP Evaluator, CSP Auditor - Burp Suite Extension, CSP Generator Chrome / Firefox.

4.3 Identity Management Testing

4.3.1 Test Role Definitions

4.3.2 Test User Registration Process

4.3.3 Test Account Provisioning Process

Sección: Web Application Security Testing

4.3.4 Testing for Account Enumeration and Guessable User Account

4.3.5 Testing for Weak or Unenforced Username Policy

4.4 Pruebas de autenticación

4.4.1 Prueba de credenciales transportadas a través de un canal cifrado

4.4.2 Prueba de credenciales predeterminadas. Burp Intruder, THC Hydra, Nikto2.

4.4.3 Prueba de mecanismo de bloqueo débil. Intentos erróneos de login. Técnicas de capcha.

4.4.4 Pruebas para eludir el esquema de autenticación. WebGoat, OWASP Zed Attack Proxy (ZAP).

4.4.5 Prueba para recordar contraseña vulnerable. ClickJacking attacks, CSRF attacks.

4.4.6 Comprobación de las debilidades de la memoria caché del navegador. OWASP Zed Attack Proxy.

4.4.7 Prueba de política de contraseña débil. Número de caracteres permitidos. Tipo de caracteres.

4.4.8 Prueba de respuesta de pregunta de seguridad débil

Sección: Web Application Security Testing

4.4.9 Prueba de funcionalidades de cambio o restablecimiento de contraseña débil

4.4.10 Prueba de autenticación más débil en canal alternativo

4.4.11 Prueba de autenticación de múltiples factores

4.5 Pruebas de autorización.

4.5.1 Prueba de inclusión de archivos transversales de directorios

4.5.2 Pruebas para eludir el esquema de autorización

4.5.3 Pruebas de escalamiento de privilegios

4.5.4 Pruebas de referencias a objetos directos inseguros

4.5.5 Pruebas de debilidades de OAuth

4.5.5.1 Prueba de debilidades del servidor de autorización de OAuth

4.5.5.2 Prueba de debilidades del cliente OAuth

Sección: Web Application Security Testing

4.6 Pruebas de gestión de sesiones.

4.6.1 Prueba del esquema de gestión de sesiones

4.6.2 Prueba de atributos de cookies

4.6.3 Prueba de Fijación de Sesión

4.6.4 Prueba de variables de sesión expuestas

4.6.5 Prueba de falsificación de solicitudes entre sitios

4.6.6 Prueba de funcionalidad de cierre de sesión

4.6.7 Tiempo de espera de la sesión de prueba

4.6.8 Pruebas de desconcierto de sesión

4.6.9 Prueba de secuestro de sesión

4.6.10 Prueba de JSON WEB Tokens.

Sección: Web Application Security Testing

4.7 Input Validation Testing

4.7.1 Testing for Reflected Cross Site Scripting

4.7.2 Testing for Stored Cross Site Scripting

4.7.3 Testing for HTTP Verb Tampering

4.7.4 Testing for HTTP Parameter Pollution

4.7.5 Testing for SQL Injection

- 4.7.5.1 Testing for Oracle
- 4.7.5.2 Testing for MySQL
- 4.7.5.3 Testing for SQL Server
- 4.7.5.4 Testing PostgreSQL
- 4.7.5.5 Testing for MS Access
- 4.7.5.6 Testing for NoSQL Injection
- 4.7.5.7 Testing for ORM Injection
- 4.7.5.8 Testing for Client-side

4.7.6 Testing for LDAP Injection

4.7.7 Testing for XML Injection

Sección: Web Application Security Testing

4.7.8 Testing for SSI Injection

4.7.9 Testing for XPath Injection

4.7.10 Testing for IMAP SMTP Injection

4.7.11 Testing for Code Injection

- 4.7.11.1 Testing for File Inclusion

4.7.12 Testing for Command Injection

4.7.13 Testing for Format String Injection

4.7.14 Testing for Incubated Vulnerability

4.7.15 Testing for HTTP Splitting Smuggling

4.7.16 Testing for HTTP Incoming Requests

4.7.17 Testing for Host Header Injection

4.7.18 Testing for Server-side Template Injection

Sección: Web Application Security Testing

4.7.19 Testing for Server-Side Request Forgery

4.7.20 Testing for Mass Assignment

4.8 Testing for Error Handling

4.8.1 Testing for Improper Error Handling

4.8.2 Testing for Stack Traces

4.9 Testing for Weak Cryptography

4.9.1 Testing for Weak Transport Layer Security

4.9.2 Testing for Padding Oracle

4.9.3 Testing for Sensitive Information Sent via Unencrypted Channels

4.9.4 Testing for Weak Encryption

Sección: Web Application Security Testing

4.10 Business Logic Testing

4.10.0 Introduction to Business Logic

4.10.1 Test Business Logic Data Validation

4.10.2 Test Ability to Forge Requests

4.10.3 Test Integrity Checks

4.10.4 Test for Process Timing

4.10.5 Test Number of Times a Function Can Be Used Limits

4.10.6 Testing for the Circumvention of Work Flows

4.10.7 Test Defenses Against Application Misuse

4.10.8 Test Upload of Unexpected File Types

4.10.9 Test Upload of Malicious Files

4.10.10 Test Payment Functionality

Sección: Web Application Security Testing

4.11 Client-Side Testing

4.11.1 Testing for DOM-Based Cross Site Scripting

- 4.11.1.1 Testing for Self DOM Based Cross Site Scripting

4.11.2 Testing for JavaScript Execution

4.11.3 Testing for HTML Injection

4.11.4 Testing for Client-side URL Redirect

4.11.5 Testing for CSS Injection

4.11.6 Testing for Client-side Resource Manipulation

4.11.7 Testing Cross Origin Resource Sharing

4.11.8 Testing for Cross Site Flashing

4.11.9 Testing for Clickjacking

Sección: Web Application Security Testing

4.11.10 Testing WebSockets

4.11.11 Testing Web Messaging

4.11.12 Testing Browser Storage

4.11.13 Testing for Cross Site Script Inclusion

4.11.14 Testing for Reverse Tabnabbing

4.12 API Testing

4.12.1 Testing GraphQL

Secciones: Reporting & Apéndices

- Reporting: indica cada uno de los elementos que incluye un informe profesional y orientado a enseñar la calidad de los trabajos realizados.

- Apéndices: se enumeran un conjunto de herramientas básicas para hacking web, lecturas recomendadas, referencias externas a vectores/técnicas de ataque y utilidades tanto para desarrolladores como para pentesters.

Ejercicio 2 → Informe de auditoría

- **COMPREHENSIVE REPORT**
  BEAST
  HYBRID APPLICATION ASSESSMENT 2017
  BishopFox
  DECEMBER 8, 2017
  URL:
  https://bit.ly/3t3thh7

- En este informe se evalúa la seguridad de Boost C++ Beast que es una Biblioteca de redes HTTP/S. El autor sigue las indicaciones de la metodología OWASP en busca de vulnerabilidades, problemas críticos y de alto riesgo (especialmente problemas de corrupción de memoria) en la biblioteca Boost C++ Best.

Ejercicio 2 → Informe de auditoría

- El equipo de evaluación identificó los siguientes problemas como resultado de la evaluación en intervalos de tiempo de la versión 124 de la biblioteca Beast:
  - Denegación de servicio.
  - Aleatoriedad insegura.

- Funcionalidad de la biblioteca.

- Recomendaciones para los desarrolladores si desean escribir aplicaciones que usan esta biblioteca.

Bonus Track: Fases de un ataque seguidas por un atacante

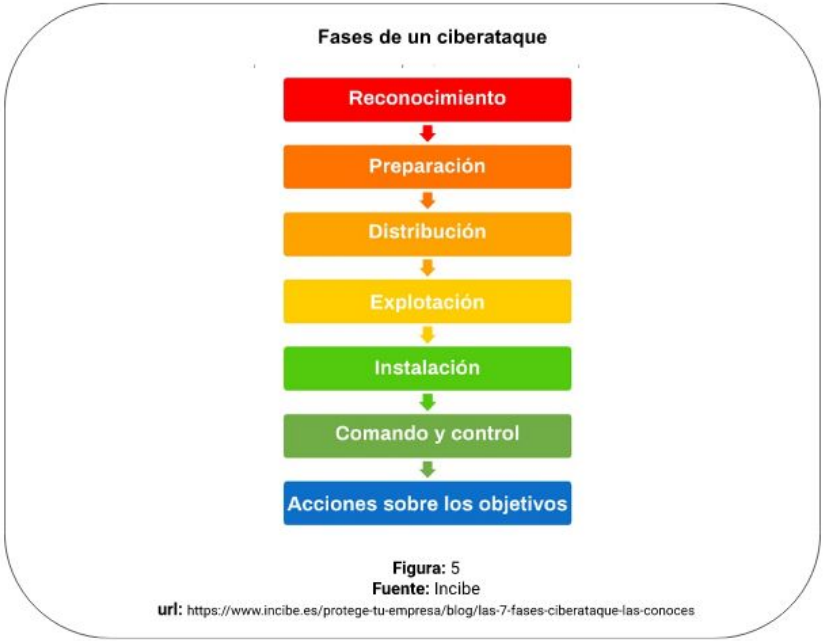


Figura: 5
Fuente: Incibe
url: https://www.incibe.es/protege-tu-empresa/blog/las-7-fases-ciberataque-las-conoces