Centro de
Investigación en
Computación

OWASP

Resumen

Background

Course

Ciberseguridad

Instructor

Acosta Bermejo Raúl

Lecture notes

2025-A
Febrero del 2025
Última actualizacioón

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

1. Introducción
2. Top Ten
3. Mobile Applications
4. Varios temas

Introducción

Generalidades

Resumen

Introducción

Generalidades

OWASP (*Open Web Application Security Project*)

- El OWASP es una fundación sin fines de lucro que trabaja para mejorar la seguridad del software. La fundación se lanzó el 1 de diciembre de 2001 y se constituyó en EU el 21 de abril de 2004.
- Su programación incluye:
  - Proyectos de código abierto liderados por la comunidad que incluyen código, documentación y estándares.
  - Más de 250 capítulos locales en todo el mundo.
  - Decenas de miles de miembros.
  - Conferencias educativas y de capacitación líderes en la industria.
- Son una comunidad abierta dedicada a permitir que las organizaciones conciban, desarrollen, adquieran, operen y mantengan aplicaciones en las que se pueda confiar.
- Todos los proyectos, herramientas, documentos, foros y capítulos son gratuitos y están abiertos a cualquier persona interesada en mejorar la seguridad de las aplicaciones.
- **Links**
  - https://owasp.org/
  - Muchos videos: https://www.youtube.com/user/OWASPGLOBAL/videos

Introducción

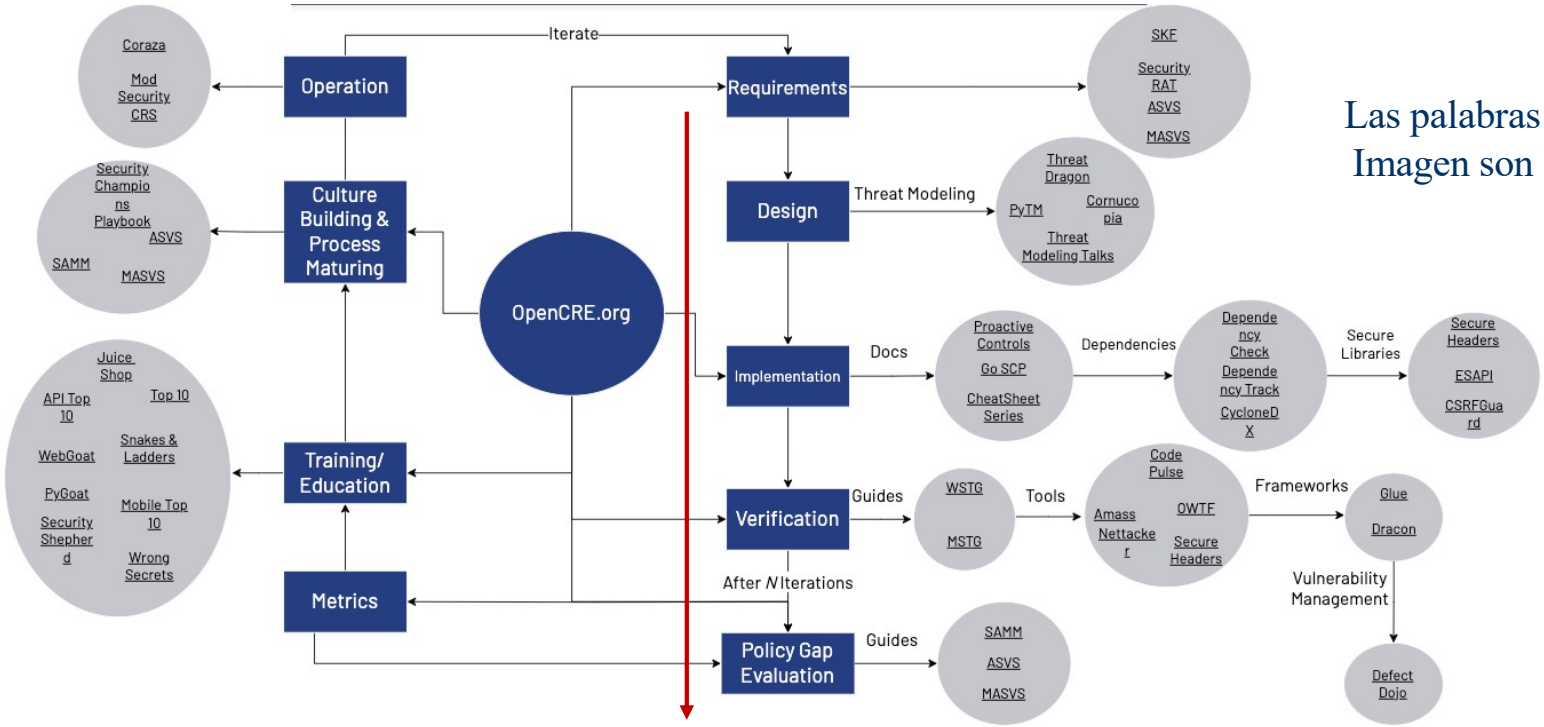
Generalidades

Proyectos

- Es un organismos con muchos proyectos de ciberseguridad: 363[Año 2025].
- Los más famosos son:
  - OWASP Top Ten 2021
    OWASP Top 10 2024
  - OWASP Mobile Application Security (MAS)
    MASVS Verification Standard
    MASWE Weakness Enumeration. MAS Security Checklist
    MASTG Testing Guide

Introducción

Generalidades



Ciclo de Vida del Desarrollo del Software

Las palabras en la Imagen son ligas

Introducción

Generalidades

Hay proyectos Maduros y algunos en Producción

Production Projects

- OWASP API Security Project
  More info soon...
- OWASP Bug Logging Tool
  OWASP BLT is a tool enabling internet users to report all kinds of issues they encounter, thereby improvi
  and allowing companies to launch their own bug hunting programs, promoting responsible disclosure an
- OWASP Coraza Web Application Firewall
  OWASP Coraza is a golang enterprise-grade WAF framework compatible with Modsecurity and OWASP
- OWASP CSRFGuard
  OWASP CSRFGuard is a library that implements a variant of the synchronizer token pattern to mitigate t
- OWASP ModSecurity
  ModSecurity is the standard open-source web application firewall (WAF) engine.
- OWASP SamuraiWTF
  SamuraiWTF (Web Training Framework) is a collection of tools and training bundled into a platform to pr
- OWASP Secure Headers Project
  Provides technical information about HTTP security headers.
- OWASP WrongSecrets
  Examples with how to not use secrets

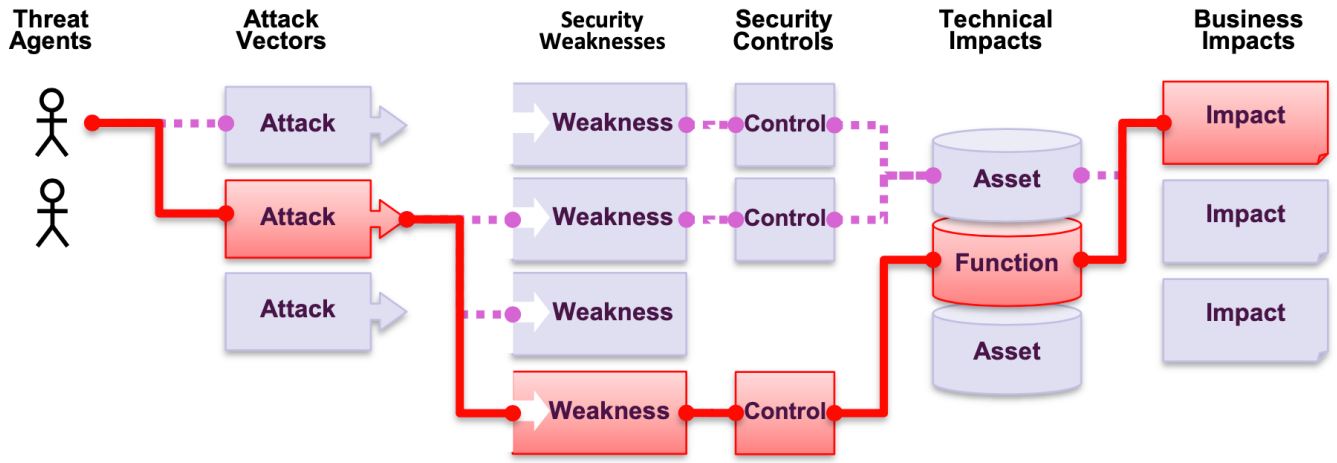
- OWASP Amass
  An open source framework that helps information security professionals perform network mapping of attack surfaces and gathering and reconnaissance techniques!
- OWASP Application Security Verification Standard (ASVS)
  The OWASP Application Security Verification Standard (ASVS) Project is a framework of security requirements that focus developing and testing modern web applications and web services.
- OWASP Cheat Sheet Series
  The OWASP Cheat Sheet Series project provides a set of concise good practice guides for application developers and de
- OWASP CycloneDX (ECMA-424)
  OWASP CycloneDX is a full-stack Bill of Materials (BOM) standard that provides advanced supply chain capabilities for cy
- OWASP Defectdojo
  The leading open source application vulnerability management tool built for DevOps and continuous security integration.
- OWASP Dependency-Check
  Dependency-Check is a Software Composition Analysis (SCA) tool suite that identifies project dependencies and checks i
- OWASP Dependency-Track
  Intelligent Component Analysis platform that allows organizations to identify and reduce risk in the software supply chain.
- OWASP Juice Shop
  Probably the most modern and sophisticated insecure web application for security trainings, awareness demos and CTFs DevSecOps pipelines!
- OWASP Mobile Application Security
  The OWASP Mobile Application Security (MAS) project consists of a series of documents that establish a security standar covers the processes, techniques, and tools used during a mobile application security assessment, as well as an exhausti and complete results.
- OWASP CRS
  The OWASP CRS is a set of generic attack detection rules for use with ModSecurity or compatible web application firewal of attacks, including the OWASP Top Ten, with a minimum of false alerts.
- OWASP OWTF
  Offensive Web Testing Framework (OWTF), is an OWASP+PTES focused try to unite great tools and make pen testing m
- OWASP SAMM
  A Software Assurance Maturity Model (SAMM) that provides an effective and measurable way for all types of organization
- OWASP Security Shepherd
  OWASP Security Shepherd is a web and mobile application security training platform. Security Shepherd has been desig skill-set demographic. The aim of this project is to take AppSec novices or experienced engineers and sharpen their pene
- OWASP Top Ten
  The OWASP Top 10 is the reference standard for the most critical web application security risks. Adopting the OWASP To your software development culture focused on producing secure code.
- OWASP Web Security Testing Guide
  The Web Security Testing Guide (WSTG) Project produces the premier cybersecurity testing resource for web application

Introducción

Definiciones

What Are Application Security Risks?

Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention.



Sometimes these paths are trivial to find and exploit, and sometimes they are extremely difficult. Similarly, the harm that is caused may be of no consequence, or it may put you out of business. To determine the risk to your organization, you can evaluate the likelihood associated with each threat agent, attack vector, and security weakness and combine it with an estimate of the technical and business impact to your organization. Together, these factors determine your overall risk.

Introducción

Definiciones

esquema de evaluación, basado en la Metodología de Evaluación de Riesgos de OWASP.

Agente de Amenaza	Explotabilidad	Prevalencia de Vulnerabilidad	Detección de Vulnerabilidad	Impacto Técnico	Impacto de Negocio
Específico de la Aplicación	Fácil 3	Difundido 3	Fácil 3	Severo 3	Específico del Negocio
	Promedio 2	Común 2	Promedio 2	Moderado 2	
	Difícil 1	Poco Común 1	Difícil 1	Mínimo 1	

OWASP Top Ten

Web

Resumen

OWASP

Top Ten

The Ten Most Critical Web Application Security Risks

Evolución

4 años

OWASP Top 10 – 2013		OWASP Top 10 – 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

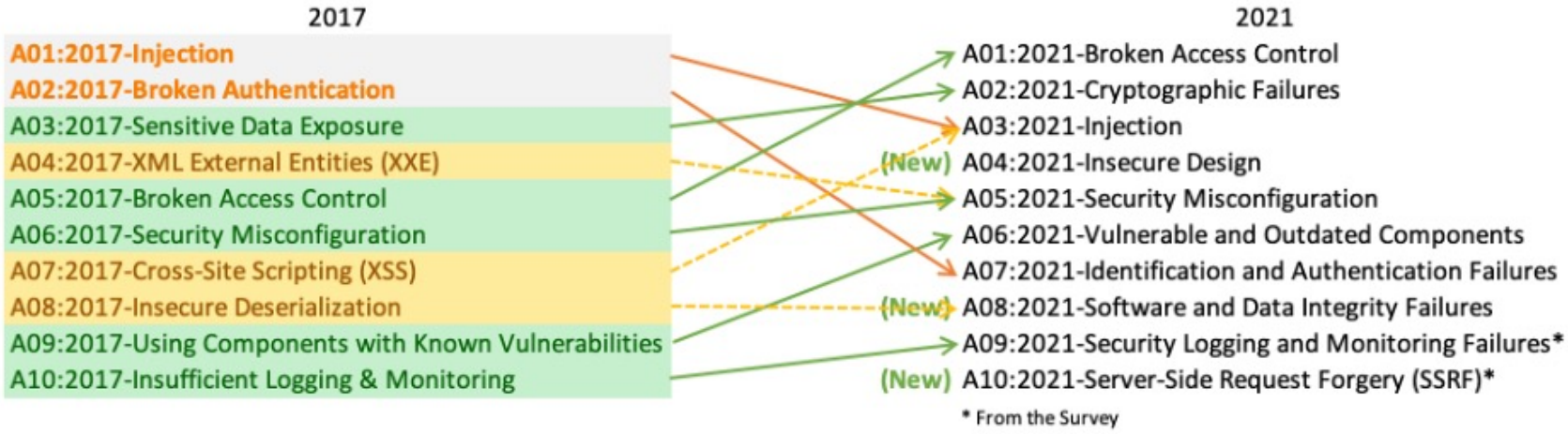
OWASP

Top Ten

Evolución

Próximo a salir el del
2025

4 años



OWASP

Top Ten

URLs

En transformación:

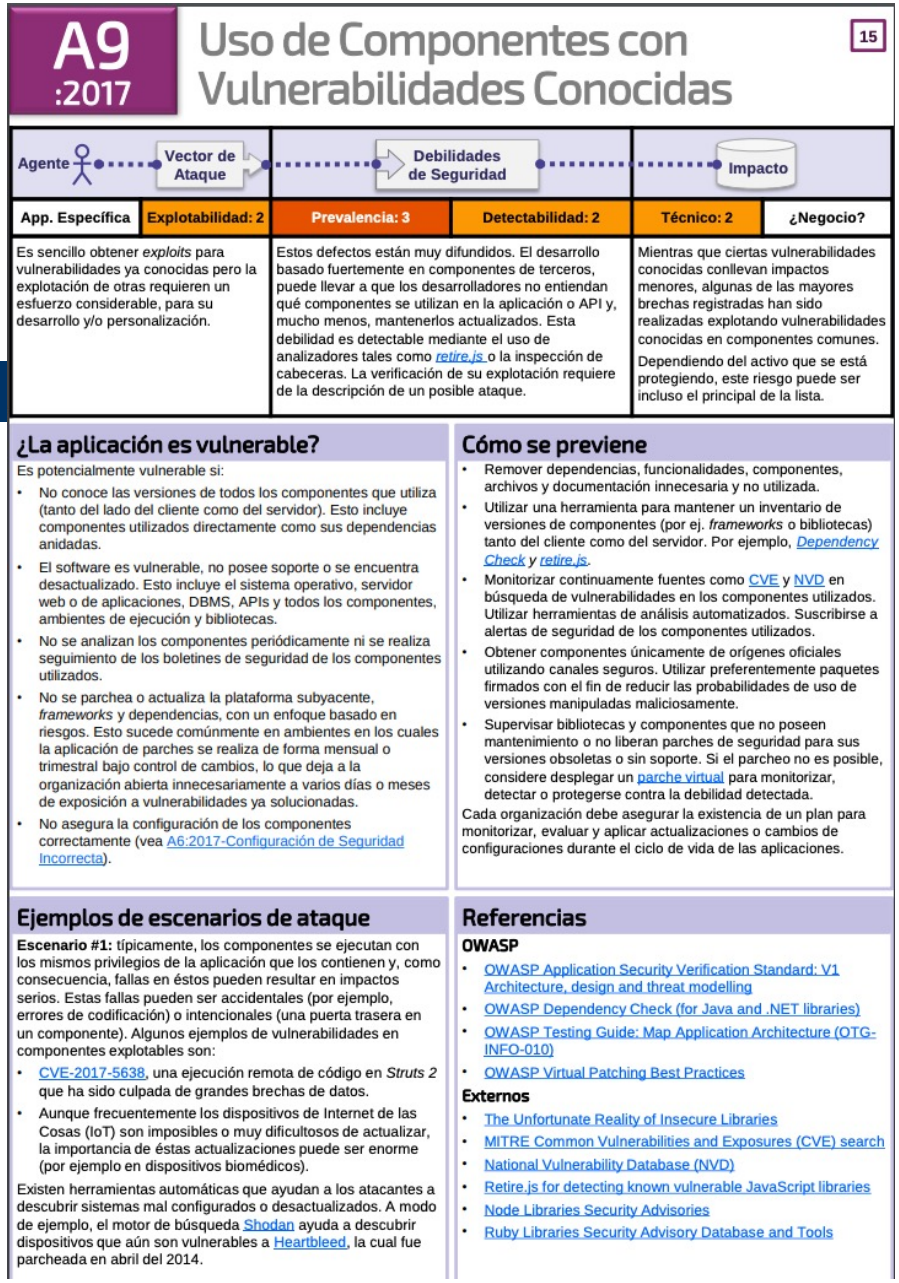
- https://owasp.org/www-project-top-ten/
- https://www.owasptopten.org/
- https://owasp.org/Top10/
- https://github.com/OWASP/Top10

OWASP

Top Ten

Ejemplo 2017

- Existe un PDF.
- https://owasp.org/www-project-top-ten/2017/



A9 :2017 Uso de Componentes con Vulnerabilidades Conocidas

15

Agente → Vector de Ataque → Debilidades de Seguridad → Impacto

App. Específica	Explotabilidad: 2	Prevalencia: 3	Detectabilidad: 2	Técnico: 2	¿Negocio?
Es sencillo obtener *exploits* para vulnerabilidades ya conocidas pero la explotación de otras requieren un esfuerzo considerable, para su desarrollo y/o personalización.		Estos defectos están muy difundidos. El desarrollo basado fuertemente en componentes de terceros, puede llevar a que los desarrolladores no entiendan qué componentes se utilizan en la aplicación o API y, mucho menos, mantenerlos actualizados. Esta debilidad es detectable mediante el uso de analizadores tales como retire.js o la inspección de cabeceras. La verificación de su explotación requiere de la descripción de un posible ataque.		Mientras que ciertas vulnerabilidades conocidas conllevan impactos menores, algunas de las mayores brechas registradas han sido realizadas explotando vulnerabilidades conocidas en componentes comunes. Dependiendo del activo que se está protegiendo, este riesgo puede ser incluso el principal de la lista.	

¿La aplicación es vulnerable?

Es potencialmente vulnerable si:

- No conoce las versiones de todos los componentes que utiliza (tanto del lado del cliente como del servidor). Esto incluye componentes utilizados directamente como sus dependencias anidadas.
- El software es vulnerable, no posee soporte o se encuentra desactualizado. Esto incluye el sistema operativo, servidor web o de aplicaciones, DBMS, APIs y todos los componentes, ambientes de ejecución y bibliotecas.
- No se analizan los componentes periódicamente ni se realiza seguimiento de los boletines de seguridad de los componentes utilizados.
- No se parchea o actualiza la plataforma subyacente, *frameworks* y dependencias, con un enfoque basado en riesgos. Esto sucede comúnmente en ambientes en los cuales la aplicación de parches se realiza de forma mensual o trimestral bajo control de cambios, lo que deja a la organización abierta innecesariamente a varios días o meses de exposición a vulnerabilidades ya solucionadas.
- No asegura la configuración de los componentes correctamente (vea A6:2017-Configuración de Seguridad Incorrecta).

Cómo se previene

- Remover dependencias, funcionalidades, componentes, archivos y documentación innecesaria y no utilizada.
- Utilizar una herramienta para mantener un inventario de versiones de componentes (por ej. *frameworks* o bibliotecas) tanto del cliente como del servidor. Por ejemplo, *Dependency Check* y *retire.js*.
- Monitorizar continuamente fuentes como CVE y NVD en búsqueda de vulnerabilidades en los componentes utilizados. Utilizar herramientas de análisis automatizados. Suscribirse a alertas de seguridad de los componentes utilizados.
- Obtener componentes únicamente de orígenes oficiales utilizando canales seguros. Utilizar preferentemente paquetes firmados con el fin de reducir las probabilidades de uso de versiones manipuladas maliciosamente.
- Supervisar bibliotecas y componentes que no poseen mantenimiento o no liberan parches de seguridad para sus versiones obsoletas o sin soporte. Si el parcheo no es posible, considere desplegar un parche virtual para monitorizar, detectar o protegerse contra la debilidad detectada.

Cada organización debe asegurar la existencia de un plan para monitorizar, evaluar y aplicar actualizaciones o cambios de configuraciones durante el ciclo de vida de las aplicaciones.

Ejemplos de escenarios de ataque

Escenario #1: típicamente, los componentes se ejecutan con los mismos privilegios de la aplicación que los contienen y, como consecuencia, fallas en éstos pueden resultar en impactos serios. Estas fallas pueden ser accidentales (por ejemplo, errores de codificación) o intencionales (una puerta trasera en un componente). Algunos ejemplos de vulnerabilidades en componentes explotables son:

- CVE-2017-5638, una ejecución remota de código en *Struts 2* que ha sido culpada de grandes brechas de datos.
- Aunque frecuentemente los dispositivos de Internet de las Cosas (IoT) son imposibles o muy dificultosos de actualizar, la importancia de éstas actualizaciones puede ser enorme (por ejemplo en dispositivos biomédicos).

Existen herramientas automáticas que ayudan a los atacantes a descubrir sistemas mal configurados o desactualizados. A modo de ejemplo, el motor de búsqueda Shodan ayuda a descubrir dispositivos que aún son vulnerables a Heartbleed, la cual fue parcheada en abril del 2014.

Referencias

OWASP

- OWASP Application Security Verification Standard: V1 Architecture, design and threat modelling
- OWASP Dependency Check (for Java and .NET libraries)
- OWASP Testing Guide: Map Application Architecture (OTG-INFO-010)
- OWASP Virtual Patching Best Practices

Externos

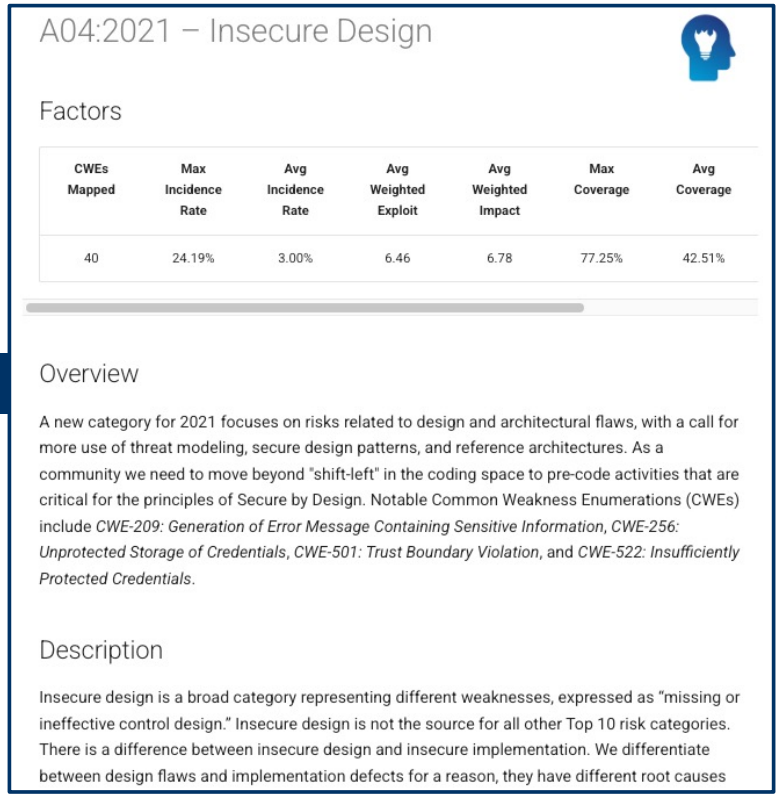
- The Unfortunate Reality of Insecure Libraries
- MITRE Common Vulnerabilities and Exposures (CVE) search
- National Vulnerability Database (NVD)
- Retire.js for detecting known vulnerable JavaScript libraries
- Node Libraries Security Advisories
- Ruby Libraries Security Advisory Database and Tools

OWASP

Top Ten

Ejemplo 2021

- Solo HTML.
- https://owasp.org/Top10/A04_2021-Insecure_Design/



A04:2021 – Insecure Design

Factors

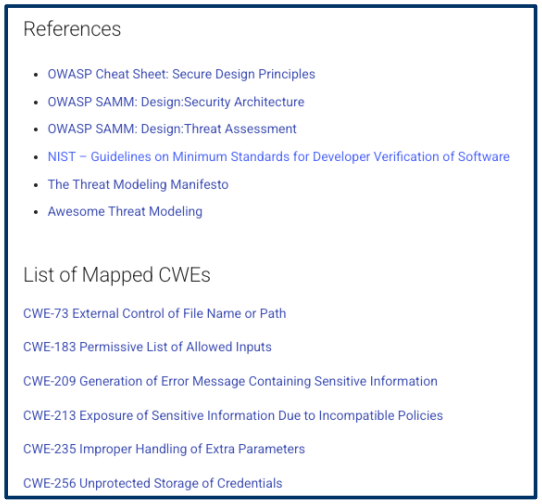
CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage
40	24.19%	3.00%	6.46	6.78	77.25%	42.51%

Overview

A new category for 2021 focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include *CWE-209: Generation of Error Message Containing Sensitive Information*, *CWE-256: Unprotected Storage of Credentials*, *CWE-501: Trust Boundary Violation*, and *CWE-522: Insufficiently Protected Credentials*.

Description

Insecure design is a broad category representing different weaknesses, expressed as "missing or ineffective control design." Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes



References

- OWASP Cheat Sheet: Secure Design Principles
- OWASP SAMM: Design:Security Architecture
- OWASP SAMM: Design:Threat Assessment
- NIST – Guidelines on Minimum Standards for Developer Verification of Software
- The Threat Modeling Manifesto
- Awesome Threat Modeling

List of Mapped CWEs

CWE-73 External Control of File Name or Path

CWE-183 Permissive List of Allowed Inputs

CWE-209 Generation of Error Message Containing Sensitive Information

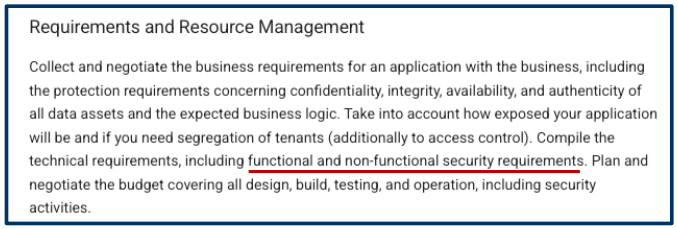
CWE-213 Exposure of Sensitive Information Due to Incompatible Policies

CWE-235 Improper Handling of Extra Parameters

CWE-256 Unprotected Storage of Credentials

Secure Development Lifecycle

Secure software requires a secure development lifecycle, some form of secure design pattern, paved road methodology, secured component library, tooling, and threat modeling. Reach out for your security specialists at the beginning of a software project throughout the whole project and maintenance of your software. Consider leveraging the OWASP Software Assurance Maturity Model (SAMM) to help structure your secure software development efforts.



Requirements and Resource Management

Collect and negotiate the business requirements for an application with the business, including the protection requirements concerning confidentiality, integrity, availability, and authenticity of all data assets and the expected business logic. Take into account how exposed your application will be and if you need segregation of tenants (additionally to access control). Compile the technical requirements, including functional and non-functional security requirements. Plan and negotiate the budget covering all design, build, testing, and operation, including security activities.

OWASP

Top Ten

LLM (*Large Language Model*)

- Son modelos de Redes Neuronales Profundas que han sido entrenados con muchos datos (archivos en internet) para que sepan procesar lenguajes (varios idiomas) y ”razonen” con ellos.
- Existen modelos especializados en varios temas: en Código, en problemas de matemáticas, etc.
- Estos LLMs pueden ser atacados para alterar su comportamiento
- URL
  - https://owasp.org/www-project-top-10-for-large-language-model-applications/

OWASP

Top Ten

T10 FOR GEN AI ▪ RESOURCES

WHITEPAPERS/GUIDES

OWASP Top 10 for LLM Applications 2025

November 17, 2024

About

The OWASP Top 10 for Large Language Model Applications started in 2023 as a community-driven effort to highlight and address security issues specific to AI applications. Since then, the technology has continued to spread across industries and applications, and so have the associated risks. As LLMs are embedded more deeply in everything from customer interactions to internal operations, developers and security professionals are discovering new vulnerabilities—and ways to counter them.

Download

OWASP Mobile Application Security

MAS

resumen

OWAS

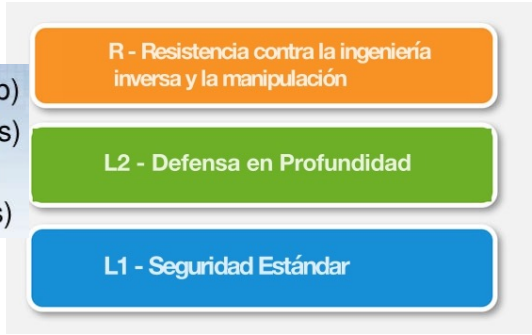
MAS

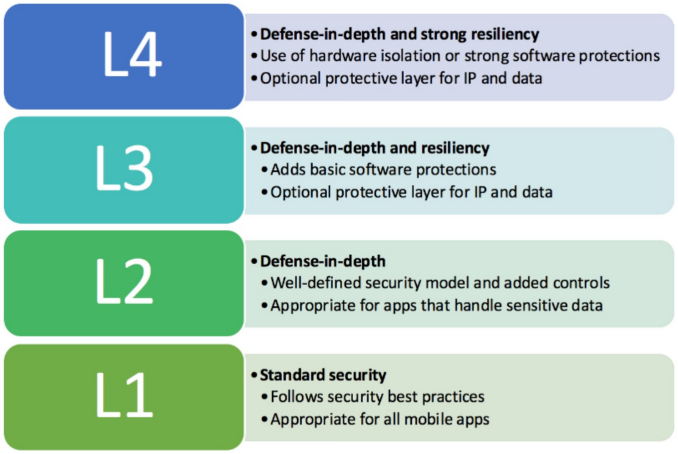
Antes

- *Open Web Application Security Project*

Ver 1.0

- MASVS-L1 (e.g. social media app)
- MASVS-L1+R (e.g. mobile games)
- MASVS-L2 (e.g. healthcare app)
- MASVS-L2+R (e.g. banking apps)



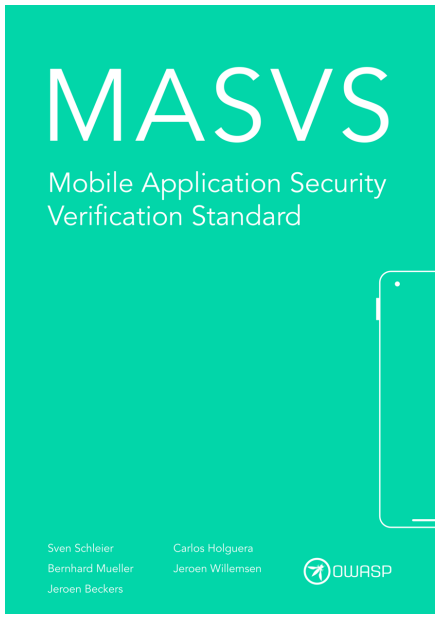


OWASP MASVS LEVELS

OWAS

MAS

Ahora

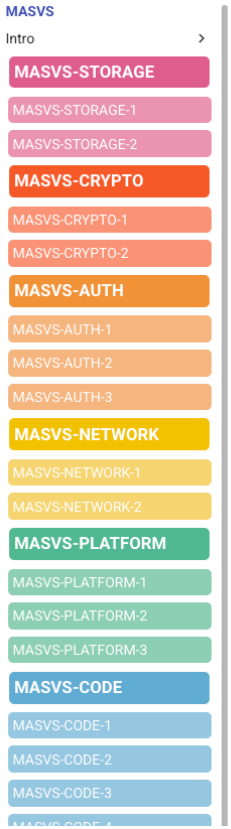


https://mas.owasp.org/

OWAS

MAS

https://github.com/OWASP/owasp-masvs/releases



The **OWASP MASVS (Mobile Application Security Verification Standard)** is the industry standard for mobile app security. It can be used by mobile software architects and developers seeking to develop secure mobile applications, as well as security testers to ensure completeness and consistency of test results.

To complement the MASVS, the OWASP MAS project also provides the OWASP Mobile Application Security Testing Guide (MASTG), the OWASP Mobile Application Security Weakness Enumeration (MASWE) and the OWASP MAS Checklist which together are the perfect companion for verifying the controls listed in the OWASP MASVS and demonstrate compliance.

Download the MASVS

The MASVS Control Groups

The standard is divided into various groups of controls, labeled **MASVS-XXXXX**, that represent the most critical areas of the mobile attack surface:

- **MASVS-STORAGE**: Secure storage of sensitive data on a device (data-at-rest).
- **MASVS-CRYPTO**: Cryptographic functionality used to protect sensitive data.
- **MASVS-AUTH**: Authentication and authorization mechanisms used by the mobile app.
- **MASVS-NETWORK**: Secure network communication between the mobile app and remote endpoints (data-in-transit).
- **MASVS-PLATFORM**: Secure interaction with the underlying mobile platform and other installed apps.
- **MASVS-CODE**: Security best practices for data processing and keeping the app up-to-date.
- **MASVS-RESILIENCE**: Resilience to reverse engineering and tampering attempts.
- **MASVS-PRIVACY**: Privacy controls to protect user privacy.

OWAS

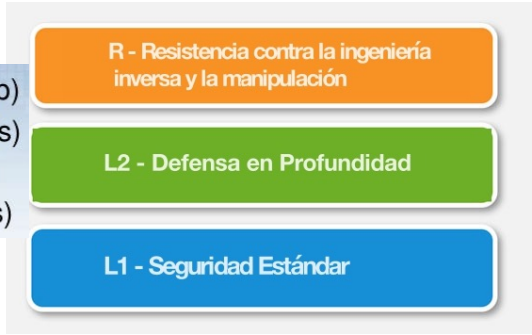
MAS

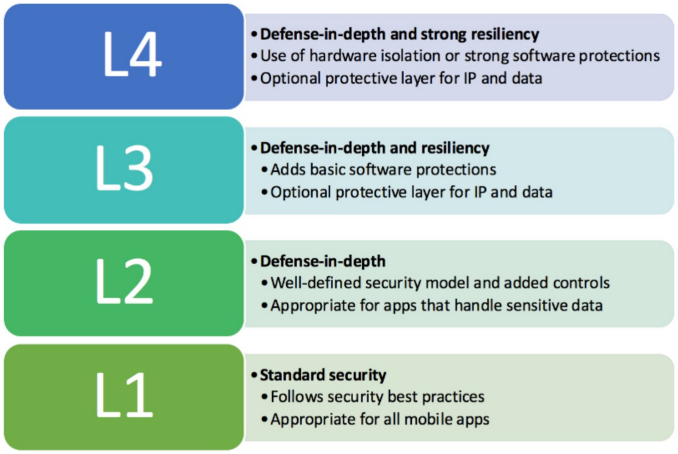
A

- *Open Web Application Security Project*

Ver 1.0

- MASVS-L1 (e.g. social media app)
- MASVS-L1+R (e.g. mobile games)
- MASVS-L2 (e.g. healthcare app)
- MASVS-L2+R (e.g. banking apps)





OWASP MASVS LEVELS

Varios temas

Consultar

Resumen

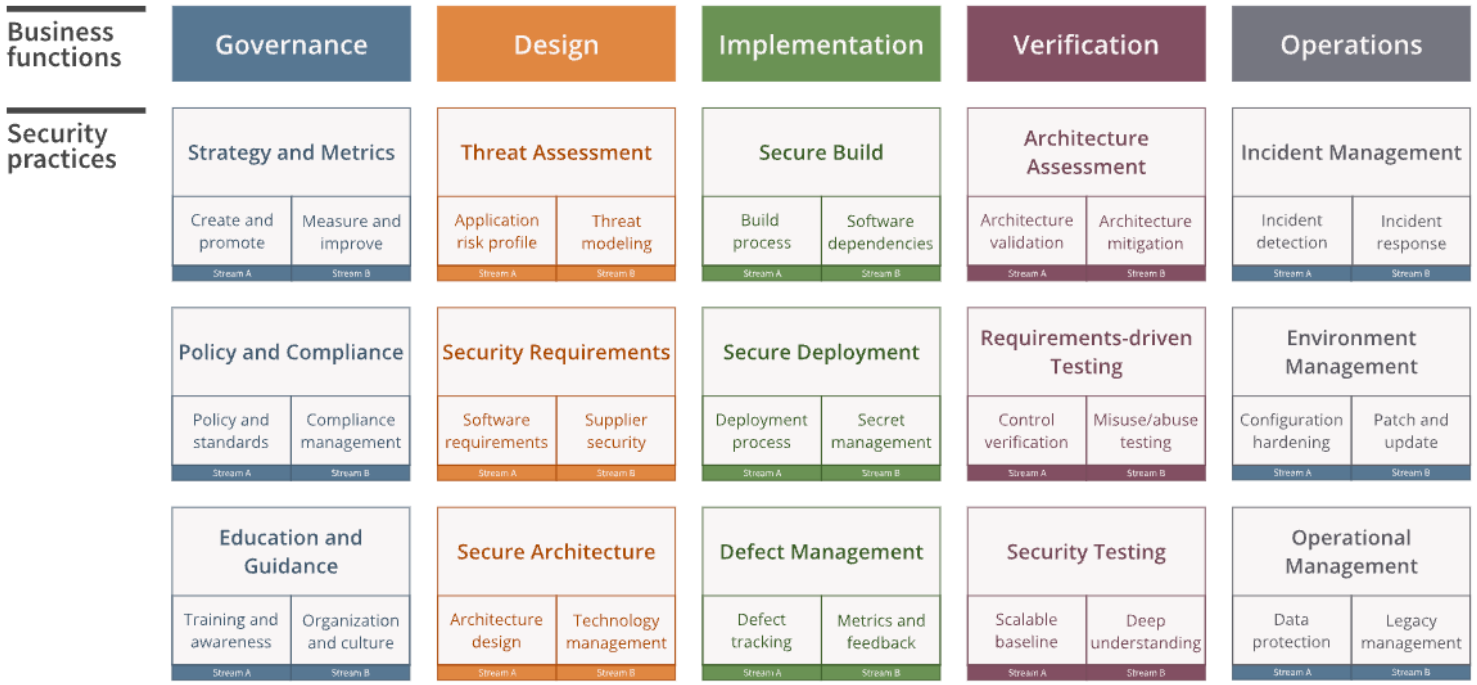
URLs
- https://owaspsamm.org/
- PDF
  - https://drive.google.com/file/d/1cI3Qzfrly_X89z7StLWI5p_Jfqs0-OZv/view?usp=sharing

Varios

Temas

Sencillo, pero por lo mismo puede ser un buen inicio.

SAMM (Software Assurance Maturity Model)

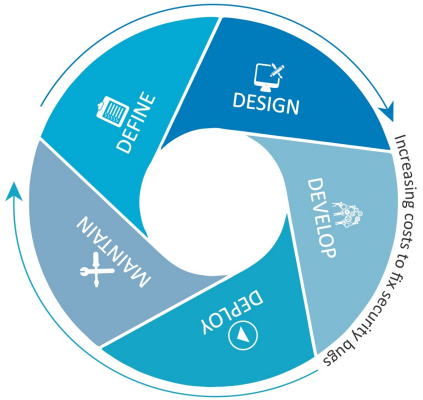


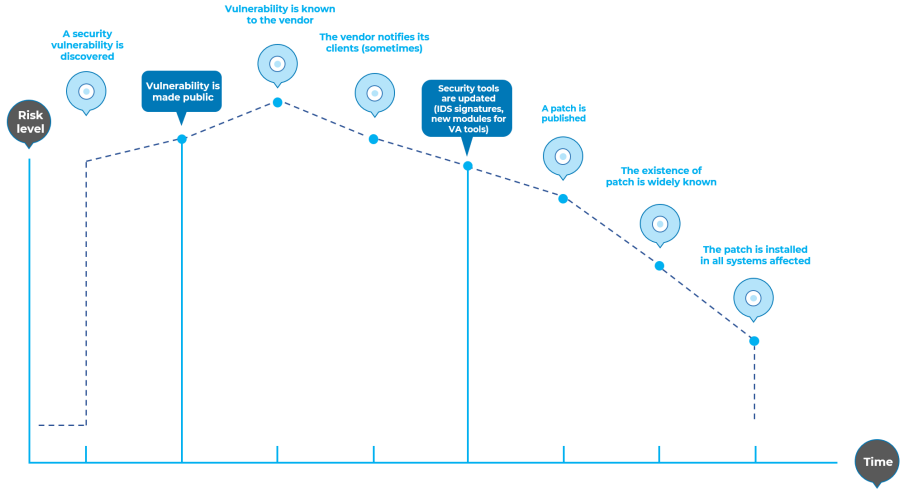
URLs
- https://owasp.org/www-project-web-security-testing-guide/stable/

Varios

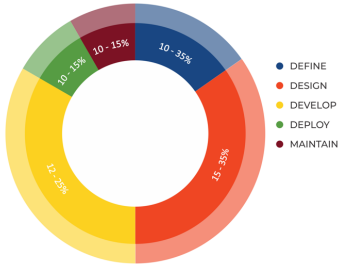
Temas

WSTG (*Web Security Testing Guide*)

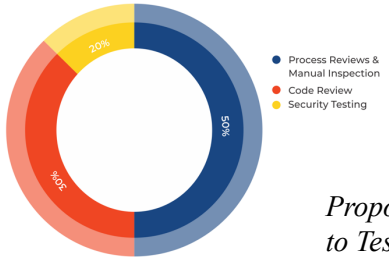




Window of Vulnerability



Proportion of Test Effort in SDLC



Proportion of Test Effort According to Test Technique

Varios

Temas

Herramientas

- En varios documentos cita el uso de herramientas que se pueden utilizar y todas son open source.
- Algunos ejemplos:
  - Zed Attack Proxy (ZAP), https://www.zaproxy.org/.
  - SqlMap, https://sqlmap.org/.
  - Web Fuzz, https://github.com/xmendez/wfuzz/tree/master.
  - Fingerprint Web Application, https://www.wappalyzer.com/

The end

Contacto

Raúl Acosta Bermejo

http:www.cic.ipn.mx
http://www.ciseg.cic.ipn.mx/

racostab@ipn.mx
racosta@cic.ipn.mx

57-29-60-00
Ext. 56652