

# Puesta en Producción Segura

## Unidad 2.

Determinación de los niveles de seguridad requeridos por las aplicaciones.



*"Los atacantes sólo necesitan encontrar una vulnerabilidad; los defensores tenemos que protegerlas todas."*

# Objetivos

- Conocer las principales listas en fuentes abiertas sobre vulnerabilidades, debilidades, etc., y saber relacionarlas y extraer información de ellas.
- Conocer las vulnerabilidades más críticas.
- Conocer los diferentes niveles de seguridad en las aplicaciones.
- Entender las listas de comprobaciones de ASVS.
- Conocer la existencia de aplicaciones web inseguras y la importancia que tienen en el aprendizaje de ciberseguridad.



# Contenidos

**01** Introducción. Sistemas informáticos seguros

**02** Fuentes abiertas para el desarrollo seguro

**03** OWASP Top 10 y OWASP MOBILE TOP 10

**04** Niveles de seguridad en las aplicaciones

**05** Comprobaciones de seguridad a nivel de aplicación (ASVS y MASVS)

**06** Aplicaciones Web inseguras



01

# Introducción. Sistemas informáticos seguros



# Introducción

- La seguridad es un proceso continuo y heterogéneo.
- Actualizaciones, pentesting, sistemas operativos.
- Es necesaria una metodología o estándar para establecer los requisitos en cuanto a seguridad de los sistemas.
- Los requisitos de seguridad dependen de los activos a proteger y de la infraestructura desplegada.
- Tipos de datos de carácter personal.
- Infraestructuras críticas.
- Debemos ser capaces de evaluar los riesgos de una determinada aplicación.



# ¿Qué es la seguridad de las aplicaciones?

- Se refiere a las medidas de seguridad para impedir el robo o el secuestro de datos o códigos dentro de la aplicación.
  - Abarca las consideraciones al desarrollar y diseñar aplicaciones y los enfoques para protegerlas tras su distribución.
- Un principio básico de la ingeniería de software se resume en una cita Tom DeMarco: "No puedes controlar lo que no puedes medir". Las pruebas de seguridad no son diferentes, aunque medir la seguridad es un proceso verdaderamente difícil.
- La seguridad de las aplicaciones puede incluir hardware, software y procedimientos que identifican o minimizan las vulnerabilidades de seguridad.
  - Un enrutador que impide que se vea la dirección IP de un ordenador desde Internet se puede considerar una forma de seguridad de las aplicaciones mediante hardware.
  - Un cortafuegos de aplicación que defina estrictamente qué actividades están permitidas y cuáles prohibidas sería seguridad mediante software.

# ¿Por qué es tan importante la seguridad de las aplicaciones?

- La seguridad de las aplicaciones es importante porque las aplicaciones actuales suelen estar disponibles a través de varias redes y conectadas al cloud, lo que aumenta las vulnerabilidades, los peligros y las amenazas a la seguridad.
  - Cada vez hay más presión y más alicientes para garantizar la seguridad no solo a nivel de la red, sino también dentro de las propias aplicaciones.
- Uno de los motivos es que los hackers están más interesados que antes en atacar aplicaciones.
- Al realizar pruebas de la seguridad de las aplicaciones, se pueden desvelar puntos débiles de las aplicaciones y ayudar a evitar este tipo de ataques.



# Sistemas informáticos seguros

Los sistemas informáticos seguros cumplen estas características:

- Integridad
- Confidencialidad
- Disponibilidad
- Autenticación
- Irrefutabilidad (No-Rechazo o No Repudio)





02

# Fuentes abiertas para el desarrollo seguro

# Inteligencia de Fuentes Abiertas OSINT

- Uno de los mayores problemas que enfrentan los profesionales de la seguridad es la regularidad con la que los usuarios normales y bien intencionados dejan accidentalmente activos e información sensibles expuestos a Internet.
- La proliferación del uso de Internet y la facilidad de publicación de contenidos a través de diferentes medios como redes sociales o blogs ha favorecido que se almacene una desorbitada cantidad de información online.
- Lo anterior supone una enorme cantidad de datos disponibles en la red de manera pública, y a partir de los cuales se puede obtener información de gran valor y utilidad mediante técnicas como OSINT.
- **Inteligencia de fuentes abiertas u «Open Source Intelligence» (OSINT)** hace referencia al conocimiento recopilado a partir de fuentes de acceso público.
- No se limita a lo que se puede encontrar usando Google, aunque la llamada «web de superficie» es un componente importante. La llamada «web profunda» es una masa de sitios web, bases de datos, archivos y más que no pueden ser indexados por Google pero está disponible usando las herramientas adecuadas.
- El proceso incluye la búsqueda, selección y adquisición de la información, así como un posterior procesado y análisis de la misma con el fin de obtener conocimiento útil y aplicable en distintos ámbitos.

# OSINT en el desarrollo seguro

- Uno de los factores críticos sobre el que gira el mundo de la seguridad es el estudio y control de debilidades, vulnerabilidades vectores de ataque y cómo prevenirlos o mitigarlos.
- Para ello existen organizaciones y proyectos encargadas de tratar estos temas, como es el caso de **NIST**, **MITRE**, **OWASP**, **FIRST** e **ICASI**.
- Toda la información, proyectos y utilidades que elaboran y desarrollan es de carácter público y puede ser consultada libremente.
- Además tiene como objetivo la estandarización para poder identificar unívocamente debilidades, vulnerabilidades, patrones de ataque, etc.



# OSINT en el desarrollo seguro

El uso de fuentes de conocimiento abiertas en el desarrollo seguro es muy recomendable para no repetir errores.

Podemos distinguir dos vertientes de fuentes abiertas en el contexto del desarrollo seguro:

- Fuentes abiertas con amenazas conocidas:
  - Organizaciones, instituciones, proyectos: **MITRE**, **FIRST**, **NIST**, etc.
  - Proyecto **OWASP**
- Aplicaciones completas de código libre o fragmentos funcionales de código de proyecto **OWASP**, **MITRE**, etc.



# Corporación MITRE

**MITRE** es una organización sin ánimo de lucro que provee información de ciberseguridad.

- Es un centro de investigación y desarrollo no comercial, que tiene sus orígenes en el **Instituto Tecnológico de Massachusetts (MIT)**, por sus siglas en inglés), cuando un grupo de científicos desarrolló sistemas de defensa aérea y computadoras durante la Segunda Guerra Mundial. En julio de 1958, por razones administrativas, este equipo del MIT fue transferido a Mitre, que oficialmente abrió sus puertas como un laboratorio de investigación.
- En la práctica, se encarga de registrar y oficializar todos los datos relativos a vulnerabilidades, debilidades y ataques conocidos en el mundo de la seguridad y provee ingeniería de sistemas, investigación y desarrollo y da soporte de tecnologías de la información al gobierno de Estados Unidos de América.



# Corporación MITRE

Entre otras, MITRE mantiene las siguientes listas, estándares o proyectos:

- **CVE**: lista de vulnerabilidades comunes.
- **CWE**: lista de debilidades comunes.
- **CAPEC**: Lista de patrones de ataque.
- **ATT&CK**: Matriz de ataques.





# CVE lista de vulnerabilidades

**CVE (Common Vulnerabilities and Exposures)** <https://cve.org/> es una lista de vulnerabilidades de seguridad de la información públicamente conocidas.

Es quizás el estándar más usado. Permite identificar cada vulnerabilidad, asignando a cada una un código de identificación único:

- Se conoce como identificador CVE (**CVE-ID**) y está formado por las siglas de este diccionario seguidas por el año en que es registrada la vulnerabilidad o exposición y un número arbitrario de al menos cuatro dígitos según el siguiente formato:



Financiado por  
la Unión Europea  
NextGenerationEU



# Ventajas de CVE

La utilidad de este catálogo es múltiple:

- Permite tener una base para la evaluación de las vulnerabilidades.
- Es un estándar muy adoptado para referirse a ellas. En la mayoría de las ocasiones, la asignación de un CVE permite diferenciar vulnerabilidades que, de otra forma, resultarían muy complejas de describir y diferenciar desde un punto vista técnico.
- Realiza un proceso de actualización continua de las vulnerabilidades registradas en la lista.
- La posibilidad de monitorizar cambios o actualizaciones sobre la lista y los contenidos de las vulnerabilidades.
- Supone una revisión exhaustiva de las nuevas vulnerabilidades que podrán ser registradas en el diccionario.
- Podemos descargar el registro en formato JSON para utilizarlo con diferentes utilidades de ciberseguridad.

# ¿Qué información ofrece la web oficial de CVE?

- De cada vulnerabilidad suele recolectarse por lo general sólo una escueta descripción y las referencias que comprueben y apoyen la existencia de la vulnerabilidad.
- Las referencias pueden ser publicaciones o entradas de foros o blogs donde se han hecho públicas las vulnerabilidades, así como demostraciones de su explotación.
- Además, suele también mostrarse un enlace directo a la información de la base de datos de vulnerabilidades del NIST (NVD), en la que pueden conseguirse más detalles de la vulnerabilidad y su valoración.
- Podemos descargarnos el registro de la vulnerabilidad en formato JSON de CVE.

Ejemplo: <https://www.cve.org/CVERecord?id=CVE-2022-25869>





# CWE lista de debilidades

**CWE (Common Weakness Enumeration)** es una lista de tipos de debilidades de software dirigida a desarrolladores y profesionales de la seguridad. Fue creada al igual que CVE para unificar la descripción de las debilidades de seguridad de software en cuanto a arquitectura, diseño y código se refiere.

Esta lista nace como iniciativa del **MITRE** para hacer frente a la diversidad de pensamiento sobre este asunto por parte de tres sectores: el académico, el comercial y el gubernamental.

Se puede ver como un **catálogo de debilidades** documentadas que se suelen cometer programando, y que podría derivar en vulnerabilidades.

Es muy utilizada por distintas herramientas de seguridad encargadas de identificar estas debilidades y para promover la identificación de las vulnerabilidades, mitigación y su prevención.

**CWE** satisface la necesidad que tienen las grandes organizaciones de conocer y tener catalogadas las distintas debilidades existentes, permitiendo asegurar sus productos frente a fallos de seguridad ya conocidos.

En comparación con el catálogo de CVE, lógicamente esta lista contiene un número considerablemente menor de registros.

# ¿Qué información ofrece la web oficial de CWE?

En la lista podemos buscar debilidades por software, hardware y otros criterios.

En la información de un elemento de CWE se ofrece en tres vistas:

- Una de tipo diccionario o conceptual, donde muestra información de la debilidad.
- Una vista Operacional con información detallada y que permite a un usuario ampliar el conocimiento sobre la debilidad: por qué se produce, plataformas, posibles mitigaciones, etc...
- Una nueva vista de árbol o Mapping-friendly con las relaciones con otras debilidades individuales y de las que deriva o derivadas de esta.

Ejemplos: <https://cwe.mitre.org/data/definitions/79.html>

<https://cwe.mitre.org/index.html>

<https://cwe.mitre.org/data/index.html>

[https://cwe.mitre.org/top25/archive/2020/2020\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html)



# CAPEC lista de patrones de ataque.



**CAPEC (Common Attack Pattern Enumeration and Classification)**

<https://capec.mitre.org/> es un catálogo de patrones de ataque perteneciente a **MITRE** que se encarga de recolectar información sobre ellos, junto a un esquema de clasificación exhaustiva.

- Estos patrones de ataques no son más que las descripciones de los métodos comunes utilizados para la explotación de vulnerabilidades.
- Las entradas de esta conocida lista intentan dar a conocer la perspectiva del atacante y los métodos utilizados para explotar los sistemas.
- Al igual que los demás catálogos, CAPEC intenta ofrecer la información necesaria para ayudar a mejorar la seguridad en todo el ciclo de vida de desarrollo de software y también apoyar las necesidades de los desarrolladores.
- Muchas de las vulnerabilidades que se registran comparten patrones de ataques, por tanto, no suelen generarse entradas en esta lista tan comúnmente como en la de vulnerabilidades.

# ¿Qué información ofrece la web oficial de CAPEC?

- Por cada patrón se pueden estudiar sus soluciones o mitigaciones comunes, los recursos necesarios o las habilidades que son necesarias en el atacante para llevarlo a cabo.



- Listados:

<https://capec.mitre.org/data/index.html>

- Ejemplo de ataque:

<https://capec.mitre.org/data/definitions/591.html>



# CPE lista de Plataformas comunes

**CPE (Common Platform Enumeration)** <https://nvd.nist.gov/products/cpe>

permite identificar con un nombre único y estándar los sistemas y software de una manera muy detallada. Proporciona:

- Un formato estandarizado para referirse a productos y plataformas que pueda ser utilizado por otras herramientas.
- Un conjunto de procedimientos para la comparación de nombres.
- Un lenguaje para la construcción de "declaraciones de aplicabilidad", que combinan un CPE con operadores lógicos simples; Un diccionario CPE y una noción general para la interpretación de los nombres CPE contenidos en él.

CPE puede integrarse dentro del **Common Vulnerability Reporting Framework (CVRF)**, utilizada para reportar toda aquella información asociada a una vulnerabilidad.

Dentro de este framework, se utiliza el **CPE** para hacer referencia a tecnologías, herramientas, sistemas o plataformas específicas con la mayor exactitud posible.



Con la versión 2.3 CPE creó un modelo más rico y complejo, además de mantener compatibilidad hacia atrás. Se introdujo el concepto de Well Formed CPE Name o WFN. El WFN viene a ser algo así:

- wfn:[part="a",vendor="microsoft",product="internet\_explorer",version="8.\*",update="sp?",edition=NA,language=ANY]
- Para referirse a Microsoft Internet Explorer 8.Cualquier actualización ServiPack(sin edición y en cualquier lenguaje).
- **cpe:2.3:a:microsoft:internet\_explorer:8:sp?::\*:\*:\*:\***

Ejemplos:

- <https://nvd.nist.gov/products/cpe/detail/EE51662B-B4FF-4E6E-94D2-75DD2F8AC381?nameFormat=2.3&orderBy=CPEURI&keyword=moodle&status=FINAL>
- Búsqueda: <https://nvd.nist.gov/products/cpe/search>



# ATT&AC: Marco de conocimiento de técnicas y tácticas

El marco MITRE **ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge)** <https://attack.mitre.org/> es una herramienta dinámica que utilizan las organizaciones para comprender y mitigar las amenazas de ciberseguridad.

El marco **Mitre ATT&CK** se basa en tres componentes principales: tácticas, técnicas, defensas y conocimiento común:

- **Tácticas (Tactics)**: Por qué de un ataque, objetivos generales, p.e. escalada.
- **Técnicas (Techniques)**: cómo se logra esos objetivos, p.e. Email.
- **Defensas (Defenses)**: Métodos de defensa o mitigación de esos ataques.
- **Conocimiento común (CTI)**: son casos documentados en los que los adversarios utilizan estas tácticas y técnicas de forma espontánea.

El marco también incluye una matriz (**Matriz ATT&CK**) que clasifica las técnicas según sus tácticas respectivas, lo que proporciona una distinción clara entre los objetivos y los métodos utilizados para lograrlos.

# Cómo aprovechar la página ATT&CK

- Es muy útil para los equipos de seguridad que buscan planificar, defender y detectar posibles ataques.
- Tanto en matrices, tácticas, técnicas y defensas nos las podemos encontrar categorizadas, según sean específicas de tecnología móvil (mobile), Sistemas de control industrial (ICS) o generales (Enterprise).
- Ejemplo de troyano bluelight:  
<https://attack.mitre.org/software/S0657>



ATT&CK®

# Corporación NIST



El NIST (National Institute of Standards and Technology) <https://www.nist.gov/> o Oficina Nacional de Normas (NBS) es una agencia de la Administración de Tecnología del Departamento de Comercio de los Estados Unidos. La misión de este instituto es promover la innovación y la competencia industrial en Estados Unidos mediante avances en metrología, normas y tecnología de forma que mejoren la estabilidad económica.

Relacionada con la ciberseguridad provee:

- **NVD:** National Vulnerability Database: Base de datos oficial de EEUU sobre vulnerabilidades y fallos de seguridad.
- Provee también informaciones sobre vulnerabilidades (**CVE**), Plataformas software, sistemas y paquetes(**CPE**), Etiquetas de identificación de software (**SWID**) y utilidades como calculadoras de riesgo de aplicaciones (**CVSS**), Enumeración de Configuraciones (**CCE**), Repositorio Checklist (**NCP**), etc.

# NVD Lista de vulnerabilidades

La **NVD (National vulnerability Database)** <https://nvd.nist.gov/>, está mantenida por el **NIST (National Institute of Standards and Technology)**, y al igual que **CVE** mantiene una base de datos de vulnerabilidades.

Para cada entrada provee de información detallada de la vulnerabilidad, así como de plataformas afectadas **CPE**, debilidad a la que se asocia **CWE**, valoración de la severidad de dicha vulnerabilidad **CVSS**, así como enlaces o referencias a ella por parte de otros proveedores, fabricantes, etc.

- Ejemplo: <https://nvd.nist.gov/vuln/detail/CVE-2016-7108>

## FIRST: Foro global de respuesta a incidentes y seguridad

- El FIRST (Forum of Incident Response and Security) es un foro global de seguridad y respuesta a incidentes <https://www.first.org/>
- Pretende formar y juntar a equipos de las áreas de gobierno, empresa y organización de enseñanza en lo que tiene que ver a respuesta a incidentes.

Entre otros estándares y servicios provee:

- **CVSS-SIG**
- **TRAFFIC LIGHT PROTOCOL (TLP)** que tiene relación con la comunicación de datos sensibles.
- Frameworks **CSIRT** y **PSIRT**.
- **EPSS** The Exploit Prediction Scoring System.



# CVSS Sistema de valoración de vulnerabilidades

## CVSS (Common Vulnerability Scoring System)

<https://www.first.org/cvss/> es un sistema de valoración de vulnerabilidades creado como un método estándar para determinar y clasificar la **criticidad de las vulnerabilidades**.

### !!!IMPORTANCIA DE LA ESTANDARIZACIÓN!!!

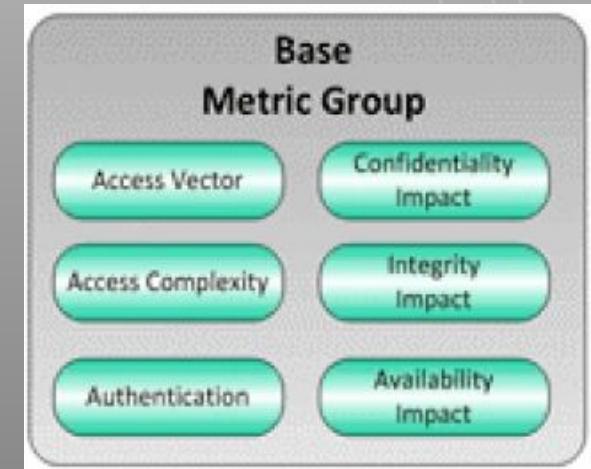
Este estándar pertenece es creado y mantenido por **FIRST**. Para conseguir ser lo más objetivo posible, CVSS se encarga de estudiar las diversas propiedades de una vulnerabilidad y las agrupa en tres métricas diferentes: **métrica base, métrica temporal y métrica del entorno**.

[Documento de especificaciones de CVSS](#)



# CVSS: Cálculo de riesgos (Base metrics)

- La única que es común a cualquier vulnerabilidad.
- El primer factor necesario para calcular esta métrica es el que viene asociado al **impacto**, definido por cómo afecta la vulnerabilidad al activo en cuanto a la **confidencialidad, integridad y disponibilidad** se refiere.
- Los valores que pueden tomar son "**bajo**", "**medio**" y "**alto**" para cada uno de ellos
- El segundo de los factores es el de la **explotabilidad**, que viene definido por el **vector de acceso (remoto/local)**, la **complejidad** del proceso (más subjetivo) y la **autenticación necesaria (sí/no y nº)** para llevar a cabo un ataque que aproveche la vulnerabilidad.



Fuente de la imagen:  
<https://www.first.org/cvss>

# CVSS: Cálculo de riesgos (Temporal metrics)

¿Una vulnerabilidad puede considerarse igual de grave si existe parche o si no, si está siendo explotada por atacantes o si se ha revelado de forma responsable... ? incluso, durante las primeras horas desde que se conoce la vulnerabilidad, el nivel de confirmación "oficial" es relevante para valorarla. Los factores que varían con el tiempo y usamos para calcular en CVSS están:

- **Exploitability:** Este factor mide el nivel de información accesible acerca de la vulnerabilidad. Pretende valorar qué técnicas, informes, exploits funcionales están disponibles, etc.
- **Remediation Level:** Este factor está relacionado con el tipo de solución disponible para la vulnerabilidad que se está valorando. Puede que exista parche oficial (bajaría su gravedad), que sólo se conozca alguna contramedida o no haya más remedio que no usar el producto o servicio.
- **Report Confidence:** establece el grado de confianza en la existencia de la vulnerabilidad y la credibilidad de los detalles técnicos conocidos y reportados sobre ella. Especialmente útil cuando se revelan vulnerabilidades en foros, sin ningún tipo de comprobación oficial sobre su veracidad o alcance.

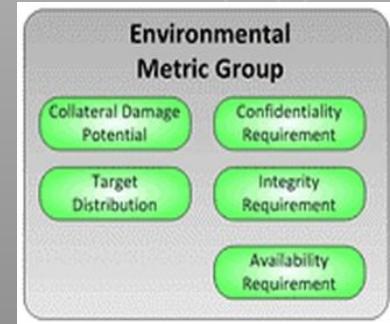


Fuente de la imagen:  
<https://www.first.org/cvss>

# CVSS: Cálculo de riesgos (Environment metrics)

Se encarga de estudiar las características que afectan al entorno donde se encuentra la vulnerabilidad. En realidad, esta métrica resulta en un “ejercicio personal” de los afectados y sus circunstancias únicas. No existe un valor “universal” para esta métrica, como es el de la base, sino que cada administrador podrá calcular el suyo en base a:

- **Collateral Damage Potential:** Mide el potencial de **pérdidas de vidas o bienes materiales** a través del daño o robo de los activos, además de la pérdida económica de la productividad o de los ingresos. ¿Cuánto daño me haría la explotación de esta vulnerabilidad?
- **Target Distribution:** Mide la **proporción de los sistemas vulnerables**. Se encarga de expresar una aproximación del porcentaje de sistemas que podrían verse afectados por la vulnerabilidad.
- **Security Requirements:** Permite personalizar el valor de la métrica de entorno dependiendo de la importancia del activo, medido en términos de confidencialidad, integridad y disponibilidad.

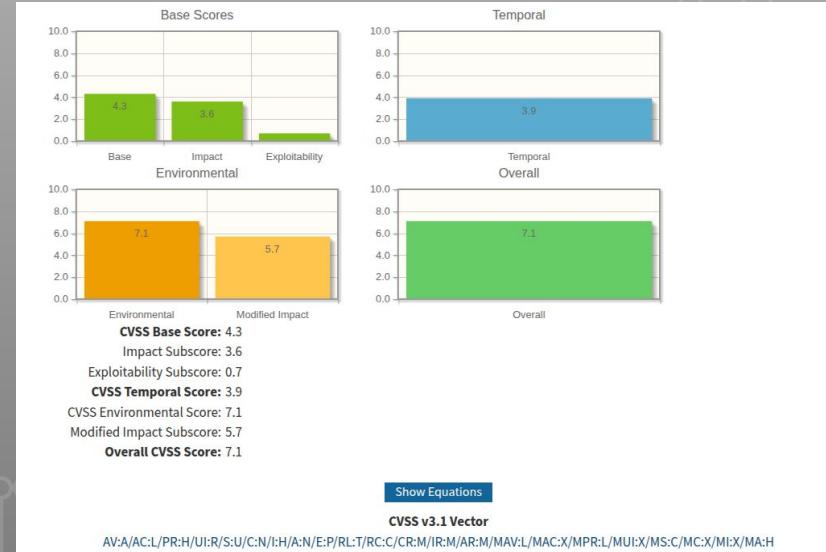


Fuente de la imagen:  
<https://www.first.org/cvss>

# ¿Cómo calcular el riesgo CVSS?

Sobre los valores obtenidos en las métricas **Base**, **Temporal** y de **entorno** se calcula el riesgo de la vulnerabilidad.

Los **valores CVSS** son puntuaciones numéricas que clasifican la gravedad de una vulnerabilidad de seguridad, oscilando entre **0 (muy baja)** y **10 (crítica)**.



Fuente de la imagen:  
<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>



Financiado por  
la Unión Europea  
NextGenerationEU



# ¿Por qué calcular el riesgo CVSS?

Entre las webs más conocidas que almacenan la información de la valoración de las vulnerabilidades se encuentran National Vulnerability Database (NVD), Open Source Vulnerability Database (OSBDV), CVE Details y Qualys Security Alerts.

Además, los programas de análisis y catalogación de vulnerabilidades más reputados deben usar estos estándares para ofrecer la información más completa y objetiva a sus clientes.

## Utilidades de cálculo CVSS:

- **NVD Cvss calculator:**  
<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>
- **FIRST CVSS Calculator:** <https://www.first.org/cvss/calculator/3.1>

## Métodos de reporte e intercambio de información de vulnerabilidades

A la hora de intentar afrontar el problema que surge a la hora de **comunicar vulnerabilidades** tanto entre profesionales de diferentes compañías, agencias estatales y Organizaciones educativas y también en la comunicación entre sistemas automáticos, nos surge la **necesidad de estandarizar los datos** y la **forma de comunicarlos**.

- ¿Qué información debe contener la descripción de una vulnerabilidad?  
¿Qué campos se deben comunicar? ¿En qué formato?

En este sentido tenemos varias alternativas de diferentes organizaciones:

- **CVRF (Common Vulnerability Reporting Framework)** se trata de un lenguaje basado en XML que permite compartir información crítica relacionada con la seguridad en un formato único, permitiendo así un rápido intercambio y gestión de la información.
- **CVE JSON** Nos proporciona la información de CVE en un registro del tipo JSON, ampliamente aceptado.



## Cómo aprovechar la información de las diferentes listas

La información de las listas y estándares que hemos visto, están estrechamente vinculadas, es decir, una **vulnerabilidad** es explotada gracias a una **debilidad** conseguida en una plataforma (**software o hardware**), que a su vez ha sido aprovechada a través de un **patrón de ataque**. Además podemos dar un valor al riesgo de dicha vulnerabilidad.

De esta forma, cada vez que cualquiera de las tres listas recibe una nueva entrada, ésta es considerada, comprobada y estudiada, por lo que muy probablemente se puedan generar o complementar los registros en cualquiera de las otras.

# Cómo aprovechar la información de las diferentes listas

Sabiendo manejar bien las listas podemos:

- Encontrar las vulnerabilidades que afectan a un recurso o plataforma(a través de las búsquedas por CPE)
- Dada una vulnerabilidad podemos conocer su riesgo y además mucha información triangulando los datos CVE, CWE, NVD, CAPEC y otras fuentes acerca de los vectores de ataque y debilidades que aprovecha y las posibles soluciones o mitigaciones.



03

# OWASP Top 10 y OWASP MOBILE TOP 10



# Fundación OWASP

El **Proyecto abierto de seguridad de aplicaciones web**, o OWASP (<https://owasp.org/>) es una organización internacional sin ánimo de lucro dedicada a la **seguridad de las aplicaciones web**. Uno de los principios fundamentales del OWASP es que todos sus materiales están disponibles de forma gratuita y son fácilmente accesibles en su sitio web, lo cual posibilita que cualquiera pueda **mejorar la seguridad** de su propia aplicación web.

- El objetivo de este proyecto es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones.

# Fundación OWASP

Entre otros **proyectos** podemos encontrar en su web:

**OWASP top 10:** <https://owasp.org/www-project-top-ten/> listado de los problemas de seguridad más comunes en aplicaciones web y móviles(OWASP top 10 Mobile).

Listas de comprobación de seguridad de aplicaciones ( **Web Security Testing guide**)

**OWASP ModSecurity Core Rule Set:** Sistema de detección de ataques.

**Software Assurance Maturity Model:** Modelo de madurez de seguridad en software.

**OWASP Dependency Check:** para comprobar las librerías que utilizamos y si son seguras.

Diferentes Aplicaciones (**Juice Shop**, **Zed attack Proxy**, **OWASP Insecure Web App**) y Herramientas de pruebas de seguridad (**OWASP Mobile Application Security**), etc.

# Owasp Top 10

**OWASP Top 10** (<https://owasp.org/Top10/es/>) es un documento de los diez **riesgos de seguridad más importantes en aplicaciones web** según la organización OWASP (Open Web Application Security Project), en español Proyecto Abierto de Seguridad de Aplicaciones Web.

- Los proyectos de desarrollo deben considerar estas vulnerabilidades en sus documentos de requerimientos y diseño, en las construcción y en las pruebas de sus aplicaciones para asegurarse de que no han sido introducidas o que han sido eliminados de forma correcta.
- Los supervisores deben incluir presupuesto y tiempo para formación, una política de desarrollo seguro, pruebas de intrusión y revisión de seguridad en el código fuente.

# Metodología de evaluación de Riesgos OWASP

## ¿Dónde se encuentra el riesgo?

### Riesgos en la Empresa



Fuente de la imagen:  
Análisis de riesgos  
aplicando la metodología  
owasp. Alvaro Machaca  
Tola

# Estimar la probabilidad

Una vez identificados los riesgos, debe estimarse:

- La probabilidad de que una vulnerabilidad en particular sea **descubierta y explotada**.
- Inicialmente es recomendable definir parámetros de calificación **cualitativos** para estimar la probabilidad. Para un calculo con mayor certeza s recomendable el **calculo cuantitativo**.



Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

# Metodología de evaluación de Riesgos OWASP

## Riesgo Tecnológico

Es la probabilidad de sufrir pérdidas por caídas o fallos en los sistemas informáticos o transmisión de datos, errores de programación u otros, siendo un componente del riesgo operativo.

Fuente: ASFI 207/13 – Dic/2013 Directrices Básicas para la Gestión del Riesgo Operativo



JUNTA DE EXTREMADURA



Plan de  
Recuperación,  
Transformación  
y Resiliencia



Financiado por  
la Unión Europea  
NextGenerationEU



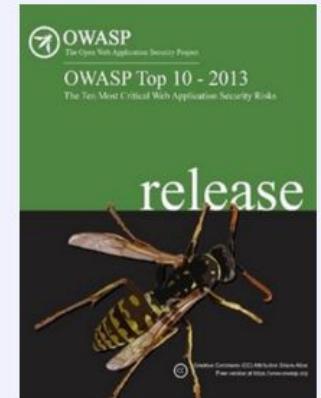
MINISTERIO  
DE EDUCACIÓN,  
FORMACIÓN PROFESIONAL  
Y DEPORTES



## Riesgos en Aplicaciones

Los atacantes pueden usar potencialmente rutas diferentes a través de la aplicación para hacer daño al negocio u organización, estas rutas representan un riesgo que puede, o no, ser lo suficientemente grave como para justificar la atención.

Fuente: OWASP Top 10



# ¿Cómo calcula OWASP el riesgo?

## OWASP Risk Rating Methodology

Para valorar el riesgo, se deben tomar en cuenta los siguientes aspectos:

- Una vulnerabilidad crítica para un tipo de negocio no lo es necesariamente para otro negocio.
- Existen metodologías y estándares internacionales para la gestión de riesgos las cuales deben adaptarse al negocio.

$$\text{Risk} = \text{Likelihood} * \text{Impact}$$

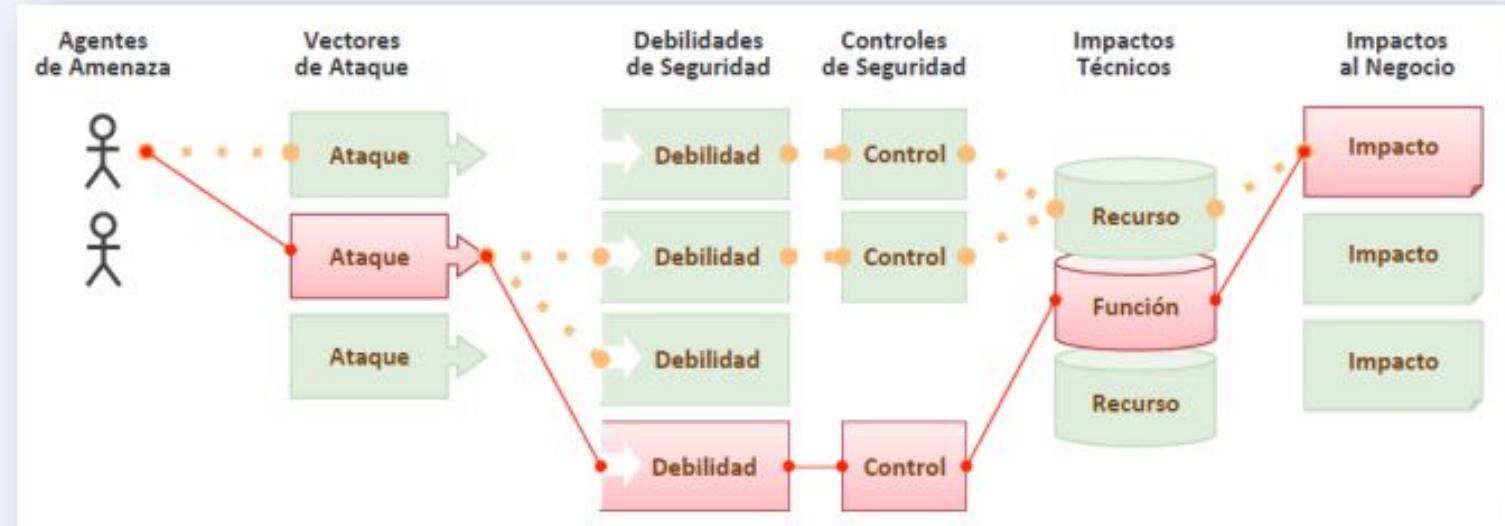
Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

RIESGO OWASP=((explotabilidad + prevalencia+ detectabilidad) /3) \* impacto técnico) + impacto de negocio) /6

Calculadora de riesgos de OWASP: <https://owasp-risk-rating.com/>

# Metodología de evaluación de Riesgos OWASP

## Representación del Riesgo



Fuente: OWASP Top 10 - 2013

JUNTA DE EXTREMADURA

R Plan de  
Recuperación,  
Transformación  
y Resiliencia

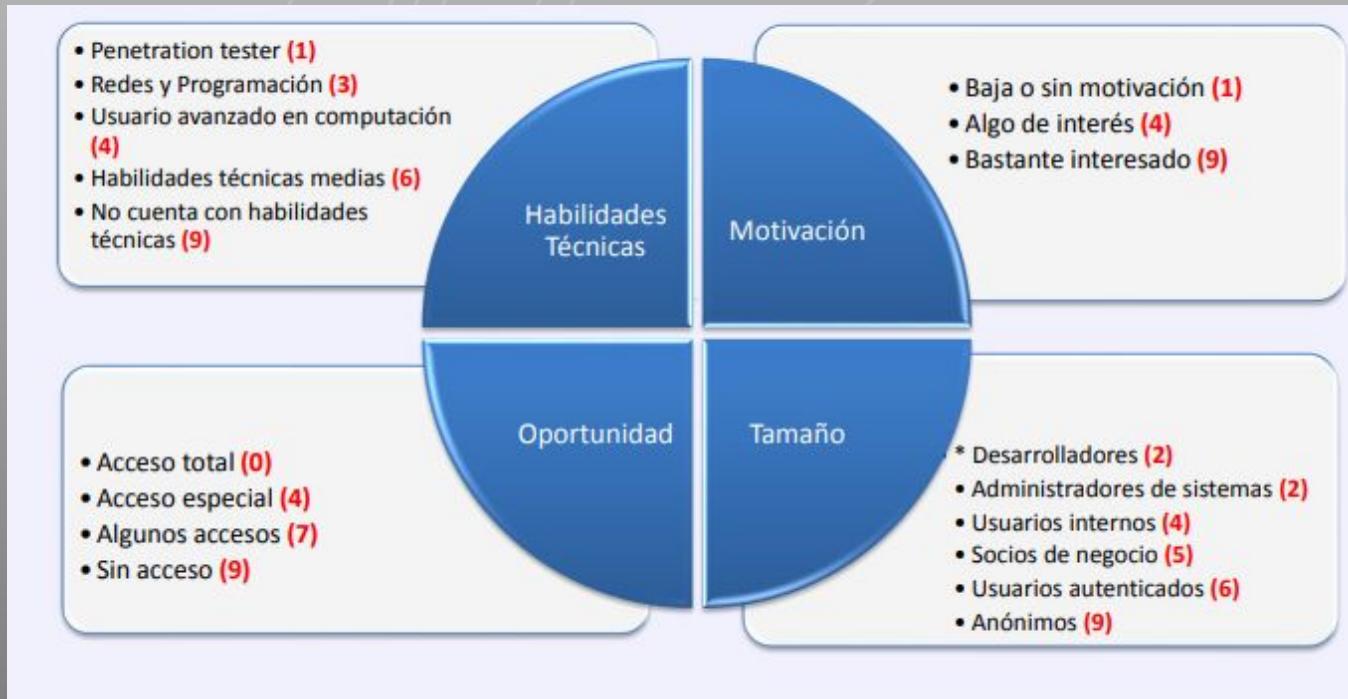


Financiado por  
la Unión Europea  
NextGenerationEU

MINISTERIO  
DE EDUCACIÓN,  
FORMACIÓN PROFESIONAL  
Y DEPORTES

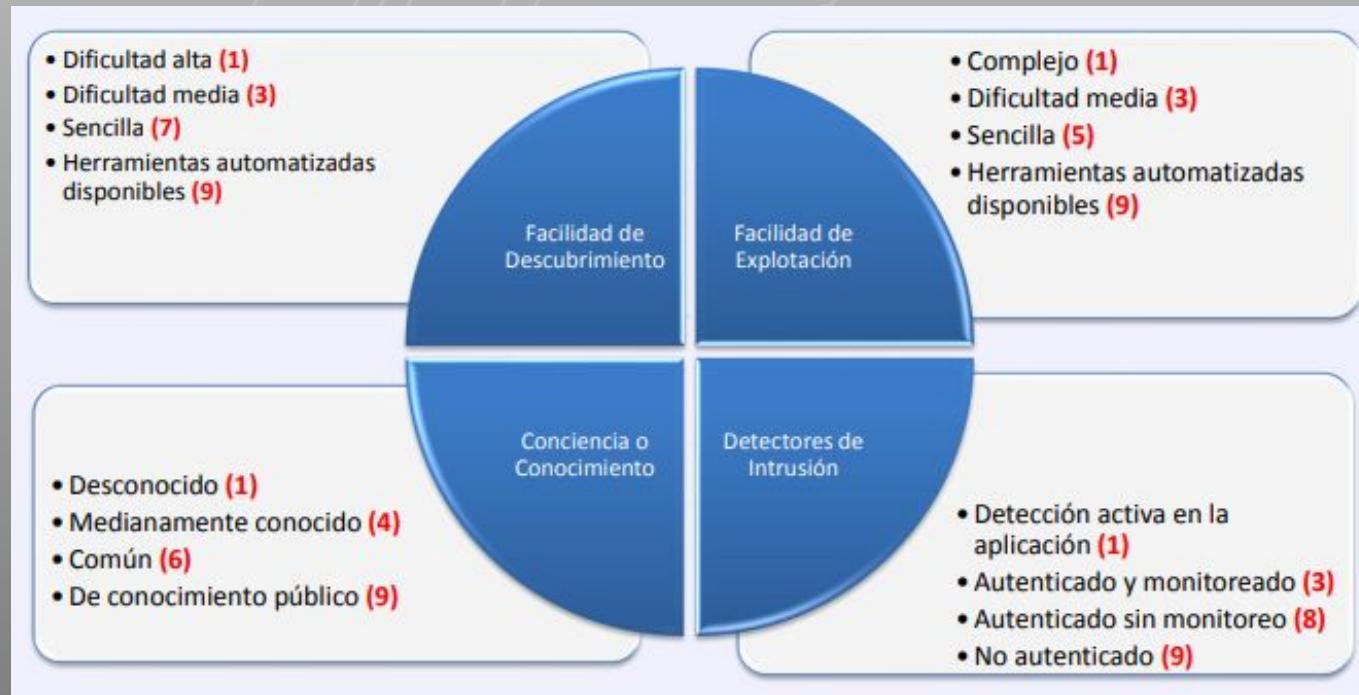


# Probabilidad: Agentes de amenaza



Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

# Probabilidad: Vulnerabilidades



Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

# Estimar el Impacto

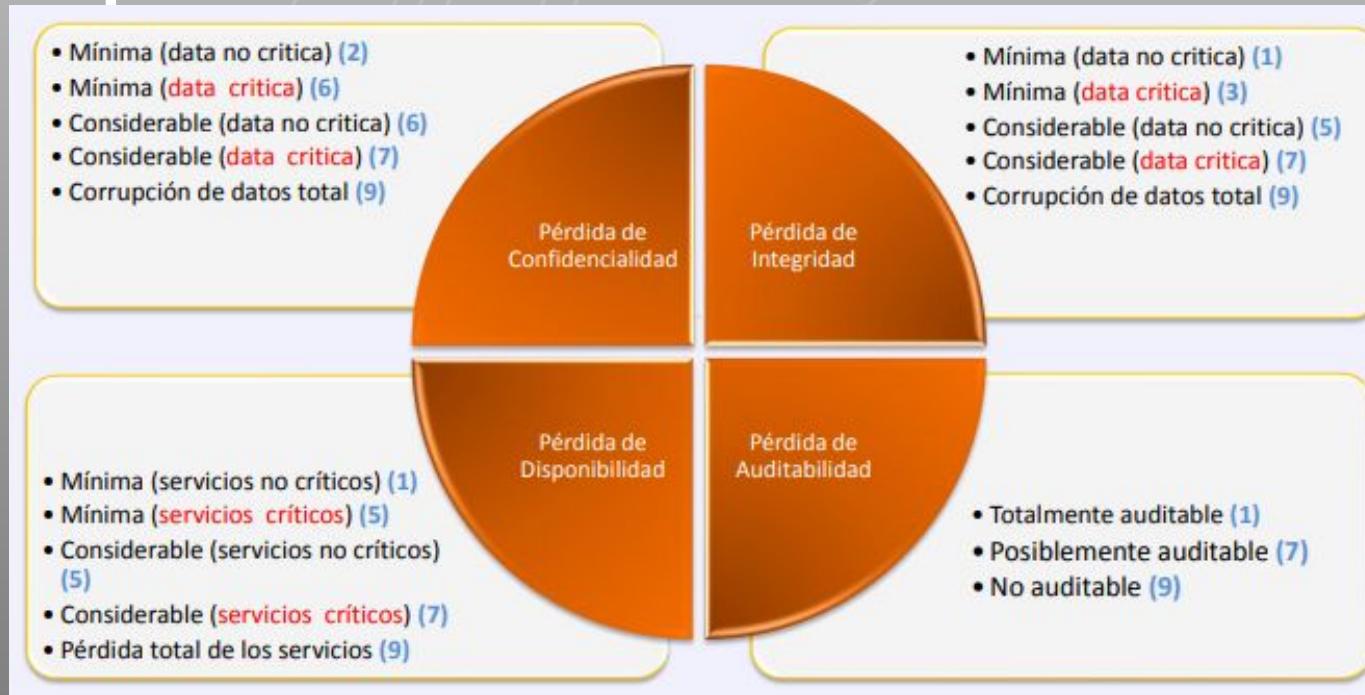
Cuando una amenaza es materializada, deben considerarse dos tipos de impacto:

- Impacto Técnico.
- Impacto sobre el Negocio.



Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

# Impacto técnico



Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

# Impacto en el Negocio



Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

## Determinar la severidad del riesgo

Para determinar la severidad del riesgo, se debe trabajar con los siguientes valores:

- **Probabilidad** de la ocurrencia de la amenaza.
- **Impacto** generado sobre el negocio.

| Likelihood and Impact Levels |        |
|------------------------------|--------|
| 0 to <3                      | LOW    |
| 3 to <6                      | MEDIUM |
| 6 to 9                       | HIGH   |

Fuente de la imagen: Análisis de riesgos aplicando la metodología owasp. Alvaro Machaca tola

| Threat agent factors |        |             |      |
|----------------------|--------|-------------|------|
| Skill level          | Motive | Opportunity | Size |
| 5                    | 2      | 7           | 1    |

| Vulnerability factors |                 |           |                     |
|-----------------------|-----------------|-----------|---------------------|
| Ease of discovery     | Ease of exploit | Awareness | Intrusion detection |
| 3                     | 6               | 9         | 2                   |

Overall likelihood=4.375 (MEDIUM)

| Technical Impact        |                   |                      |                        |
|-------------------------|-------------------|----------------------|------------------------|
| Loss of confidentiality | Loss of integrity | Loss of availability | Loss of accountability |
| 9                       | 7                 | 5                    | 8                      |

Overall technical impact=7.25 (HIGH)

$$\sum \text{ Variables de agentes de amenaza y Variable de factores de vulnerabilidad}$$

8

$$\sum \text{ Variables de Impacto técnico}$$

4

$$\sum \text{ Variables de Impacto sobre el negocio}$$

4

| Business Impact  |                   |                |                   |
|------------------|-------------------|----------------|-------------------|
| Financial damage | Reputation damage | Non-compliance | Privacy violation |
| 1                | 2                 | 1              | 5                 |

Overall business impact=2.25 (LOW)

| Overall Risk Severity |        |        |        |          |
|-----------------------|--------|--------|--------|----------|
| Impact                | HIGH   | Medium | High   | Critical |
|                       | MEDIUM | Low    | Medium | High     |
|                       | LOW    | Note   | Low    | Medium   |
|                       |        | LOW    | MEDIUM | HIGH     |

**Likelihood**

# Lista OWASP Top 10

## M01. Uso inapropiado de la aplicación

- Esta categoría cubre el uso indebido de características de la plataforma o el no uso de los controles de seguridad, permisos de plataforma, uso indebido de Touch ID, servicios de contraseñas (KeyChain IOS) o algún otro control de seguridad que sea parte del sistema operativo móvil. Lo que se valora en el ataque es la seguridad de cualquier API expuesta.

## M02. Almacenamiento inseguro de datos.

- La valoración y por ende el vector de ataque varía mucho. Desde aplicaciones de terceros que utilizan caché, cookies y otra información para recopilar datos protegidos, hasta que un adversario pueda obtener físicamente el dispositivo y ver información, debe manejar el almacenamiento de datos correctamente de varias maneras. Esto incluye la autenticación, el cifrado y el manejo adecuado de todas las funciones de almacenamiento en caché.

## M03 Comunicación insegura.

- Esta categoría cubre los protocolos inseguros de enlace, versiones SSL incorrectas, negociación débil, la comunicación sin cifrar de datos sensibles, etc. Por lo que hay que comprobar que los desarrolladores (a menudo muy preocupados en cuanto a la protección del procedimiento de autenticación y los datos en reposo) hayan implementado un cifrado de los datos que se transmiten correctamente.

# Lista OWASP Top 10

## M04 Autenticación insegura.

- Esta categoría captura las nociones de autenticación del usuario final o gestión de sesión incorrecta. Esto puede incluir: no identificar al usuario en absoluto cuando sea necesario
- No mantener la identidad del usuario cuando se requiere
- Debilidades en el manejo de sesiones.

## M05 Criptografía insuficiente.

- Esta categoría sistematiza los fallos vinculados con la criptografía. Y que pueden llegar a exponer datos sensibles y comprometer a los sistemas en su totalidad, es decir, se tratan, por tanto, de vulnerabilidades vinculadas a la ausencia de protocolos de comunicaciones seguros.

## M06 Autorización insegura.

- Esta es una categoría para capturar cualquier fallo en la autorización (por ejemplo, decisiones de autorización en el lado del cliente, navegación forzada, etc.). Es distinto de los problemas de autenticación (por ejemplo, inscripción de dispositivos, identificación de usuarios, etc.).

Fuente: Fernando Saavedra:  
<https://www.audea.com/owasp-top-ten-mobile-risks/>

## Lista OWASP Top 10

### M07 Calidad del código del cliente.

- Esta categoría se refiere a la captura de todos los problemas de implementación a nivel de código en el cliente móvil. Esto es distinto de los errores de codificación del servidor. Entrarían dentro de ella, cosas como desbordamientos de búfer, vulnerabilidades de cadena de formato y varios otros errores de nivel de código donde la solución es reescribir algún código que se esté ejecutando en el dispositivo móvil. Es decir, se centra en las vulnerabilidades creadas debido a errores de codificación.

### M08 Alteración de código.

- Esta categoría cubre parches binarios, modificación de recursos locales, incorrecta utilización de métodos y modificación de memoria dinámica. Por ende, esta categoría cubre cualquier modificación que el adversario pueda realizar en el código de la aplicación.

Fuente: Fernando Saavedra:  
<https://www.audea.com/owasp-top-ten-mobile-risks/>

# Lista OWASP Top 10

## M09 Ingeniería inversa.

- Esta categoría incluye el análisis del núcleo binario final para determinar su código fuente, bibliotecas, algoritmos y otros activos. Esto puede utilizarse para explotar otras vulnerabilidades nacientes en la aplicación, así como para revelar información sobre los servidores backend, las claves criptográficas y la propiedad intelectual. Por lo que hay que valorar que el propio código se encuentre correctamente ofuscado ya que es la mejor manera de paliar con la ingeniería inversa.

## M10 Funcionalidad externa.

- Esta vulnerabilidad surge cuando los desarrolladores no eliminan funciones adicionales, creadas durante el proceso de desarrollo para facilitar la prueba de la aplicación y que pueden ser explotadas para realizar ataques contra la aplicación.
- Como se puede observar existen diferentes riesgos en las propias aplicaciones móviles (tan de moda en los últimos tiempos en entornos personales y corporativos), por lo que es siempre aconsejable analizar las mismas para que no sean el foco de algún ataque. Fuente: Fernando Saavedra:  
<https://www.audea.com/owasp-top-ten-mobile-risk/>

# Vulnerabilidades OWASP MOBILE TOP 10

## M01. Uso inapropiado de la aplicación

- Esta categoría cubre el uso indebido de características de la plataforma o el no uso de los controles de seguridad, permisos de plataforma, uso indebido de TouchID, servicios de contraseñas (KeyChain IOS) o algún otro control de seguridad que sea parte del sistema operativo móvil. Lo que se valora en el ataque es la seguridad de cualquier API expuesta.

## M02. Almacenamiento inseguro de datos.

- La valoración y por ende el vector de ataque varía mucho. Desde aplicaciones de terceros que utilizan caché, cookies y otra información para recopilar datos protegidos, hasta que un adversario pueda obtener físicamente el dispositivo y ver información, debe manejar el almacenamiento de datos correctamente de varias maneras. Esto incluye la autenticación, el cifrado y el manejo adecuado de todas las funciones de almacenamiento en caché.

## M03 Comunicación insegura.

- Esta categoría cubre los protocolos inseguros de enlace, versiones SSL incorrectas, negociación débil, la comunicación sin cifrar de datos sensibles, etc. Por lo que hay que comprobar que los desarrolladores (a menudo muy preocupados en cuanto a la protección del procedimiento de autenticación y los datos en reposo) hayan implementado un cifrado de los datos que se transmiten correctamente.

# Vulnerabilidades OWASP MOBILE TOP 10

## M04 Autenticación insegura.

- Esta categoría captura las nociones de autenticación del usuario final o gestión de sesión incorrecta. Esto puede incluir:
  - no identificar al usuario en absoluto cuando sea necesario.
  - No mantener la identidad del usuario cuando se requiere.
  - Debilidades en el manejo de sesiones.

## M05 Criptografía insuficiente.

- Esta categoría sistematiza los fallos vinculados con la criptografía. Y que pueden llegar a exponer datos sensibles y comprometer a los sistemas en su totalidad, es decir, se tratan, por tanto, de vulnerabilidades vinculadas a la ausencia de protocolos de comunicaciones seguros.

## M06 Autorización insegura.

- Esta es una categoría para capturar cualquier fallo en la autorización (por ejemplo, decisiones de autorización en el lado del cliente, navegación forzada, etc.). Es distinto de los problemas de autenticación (por ejemplo, inscripción de dispositivos, identificación de usuarios, etc.).

Fuente: Fernando Saavedra:

<https://www.audea.com/owasp-top-ten-mobile-risks/>

# Vulnerabilidades OWASP MOBILE TOP 10

## M07 Calidad del código del cliente.

- Esta categoría se refiere a la captura de todos los problemas de implementación a nivel de código en el cliente móvil. Esto es distinto de los errores de codificación del servidor. Entrarían dentro de ella, cosas como desbordamientos de búfer, vulnerabilidades de cadena de formato y varios otros errores de nivel de código donde la solución es reescribir algún código que se esté ejecutando en el dispositivo móvil. Es decir, se centra en las vulnerabilidades creadas debido a errores de codificación.

## M08 Alteración de código.

- Esta categoría cubre parches binarios, modificación de recursos locales, incorrecta utilización de métodos y modificación de memoria dinámica. Por ende, esta categoría cubre cualquier modificación que el adversario pueda realizar en el código de la aplicación.

Fuente: Fernando Saavedra:  
<https://www.audea.com/owasp-top-ten-mobile-risks>

# Vulnerabilidades OWASP MOBILE TOP 10

## M09 Ingeniería inversa.

- Esta categoría incluye el análisis del núcleo binario final para determinar su código fuente, bibliotecas, algoritmos y otros activos. Esto puede utilizarse para explotar otras vulnerabilidades nacientes en la aplicación, así como para revelar información sobre los servidores backend, las claves criptográficas y la propiedad intelectual. Por lo que hay que valorar que el propio código se encuentre correctamente ofuscado ya que es la mejor manera de paliar con la ingeniería inversa.

## M10 Funcionalidad externa.

- Esta vulnerabilidad surge cuando los desarrolladores no eliminan funciones adicionales, creadas durante el proceso de desarrollo para facilitar la prueba de la aplicación y que pueden ser explotadas para realizar ataques contra la aplicación.
- Como se puede observar existen diferentes riesgos en las propias aplicaciones móviles (tan de moda en los últimos tiempos en entornos personales y corporativos), por lo que es siempre aconsejable analizar las mismas para que no sean el foco de algún ataque.

Fuente: Fernando Saavedra:  
<https://www.audea.com/owasp-top-ten-mobile-risks>



04

# Niveles de seguridad en las aplicaciones



## Estándar de verificación de seguridad en las aplicaciones (ASVS)

**El Estándar de Verificación de Seguridad en Aplicaciones (ASVS de sus siglas en inglés,**

<https://owasp.org/www-project-application-security-verification-standard/>) reúne unos controles funcionales necesarios para diseñar, desarrollar y testear aplicaciones web.

- Fue creado por el proyecto de código abierto **OWASP (Open Web Application Security Project)** dedicado a combatir las causas que hacen que el software sea inseguro.
- ASVS tiene como objetivo ayudar a los desarrolladores a crear y mantener aplicaciones más seguras y establecer un consenso entre proveedores de productos de seguridad y consumidores.

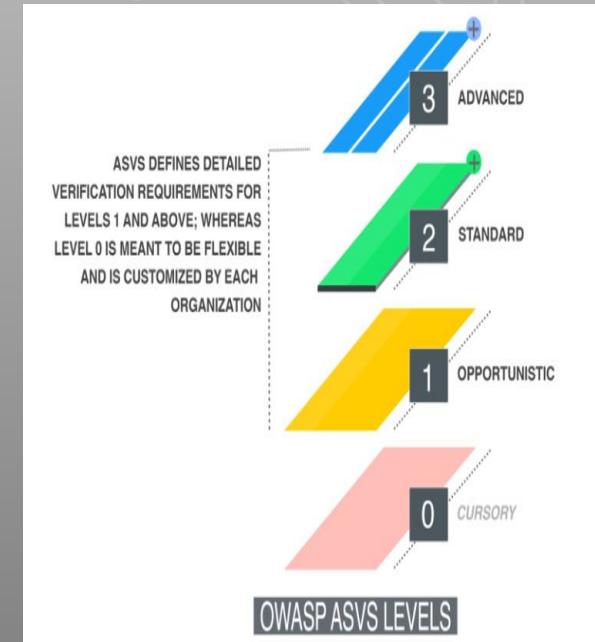
# Niveles de verificación de seguridad en las aplicaciones

**ASVS** define tres niveles de verificación de seguridad, incrementando la profundidad con cada nivel.

Cada nivel ASVS contiene una lista de requerimientos de seguridad que pueden ser funcionalidades específicas de seguridad o capacidades que deben construirse por los desarrolladores de software.

Los **niveles de verificación de seguridad** establecidos son:

- **Nivel 1:** dirigido a todo tipo de software.
- **Nivel 2:** para aplicaciones que manejan datos sensibles.
- **Nivel 3:** que engloba aplicaciones críticas como las que manejan datos médicos o transacciones bancarias.



Fuente de la imagen:  
<https://owasp.org/www-project-application-security-verification-standard/>

VT

# OWASP – Application Security Verification Standard v4.0

The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.

286 Controls and 14 Verification topics



[www.VirtuallyTesting.com](http://www.VirtuallyTesting.com)

Financiado por la Unión Europea

3

Fuente de la imagen: <https://owasp.org/www-project-application-security-verification-standard/>

# Nivel 1: Oportunista

Una aplicación alcanza ASVS **nivel 1** (u **oportunista**) si se defiende adecuadamente contra vulnerabilidades de seguridad de aplicaciones que son **fáciles de descubrir** y se incluyen en el **OWASP Top 10** u otras listas similares.

- Este nivel es apropiado típicamente para **aplicaciones donde se requiere escasa confianza** en el uso correcto de los controles de seguridad, o para proporcionar un análisis rápido a un conjunto de aplicaciones de una organización, o asistir en la elaboración de una lista de requerimientos de seguridad con prioridades como parte de un esfuerzo de múltiples fases.

Los controles de nivel 1 pueden ser asegurados por **herramientas automáticamente** o manualmente sin acceso al código fuente. Consideramos a nivel 1 como el mínimo requerido para todas las aplicaciones.

- Las amenazas a la aplicación probablemente provendrán de atacantes que utilizan técnicas simples y de bajo esfuerzo para identificar vulnerabilidades fáciles de encontrar y de explotar. Esto es en contraste con un ataque dirigido el cual se enfocará específicamente a la aplicación. Si los datos procesados por la aplicación tienen un alto valor, difícilmente sea suficiente con una revisión de nivel 1.

## Nivel 2: Estándar

Una aplicación alcanza ASVS **nivel 2 (o estándar)**, si se defiende adecuadamente contra **la mayoría de los riesgos asociados con el software de hoy** en día.

El Nivel 2 asegura que controles de seguridad se encuentran en el lugar adecuado, son efectivos y son utilizados dentro de la aplicación. Este nivel es generalmente apropiado para **aplicaciones que manejan transacciones business-to-business, información de datos de usuarios**, implementan **funciones sensibles** o críticas para el negocio o incluyen el proceso de otros activos sensibles.

Las **amenazas** para aplicaciones de nivel 2 por lo general están **motivados por atacantes** los cuales **se centran en objetivos concretos**, utilizando herramientas y técnicas efectivas en el descubrimiento y explotación de vulnerabilidades dentro de las aplicaciones.

# Nivel 3: Avanzado

El **nivel 3** en ASVS es el más alto nivel de verificación dentro de ASVS. Este nivel está reservado normalmente para aplicaciones que requieren niveles significativos de verificación de seguridad, como las que se encuentran dentro de **áreas de militares, salud, seguridad, infraestructuras críticas**, etc.

Las organizaciones pueden requerir del nivel 3 para aplicaciones que realizan funciones críticas, donde una falla de seguridad podría afectar significativamente sus operaciones y hasta su supervivencia.

Un ejemplo en la aplicación del nivel 3 de ASVS se proporciona a continuación. Una aplicación alcanza el nivel 3 (o avanzado) si se defiende adecuadamente contra vulnerabilidades de seguridad avanzadas y también demuestra los principios de un **buen diseño de seguridad**.

Una aplicación en el nivel 3 de ASVS, requiere un análisis de mayor profundidad, arquitectura, codificación y Testing en todo nivel. Una aplicación segura es modularizada de forma significativa (para facilitar por ejemplo su resiliencia, escalabilidad, y sobre todo, capas de seguridad), y cada módulo (separados por conexiones de la red o instancias físicas) se encarga de sus responsabilidades de seguridad (defensa en profundidad) la que debe ser debidamente documentada. Las responsabilidades incluyen controles para asegurar la confidencialidad (cifrado, por ejemplo), integridad (transacciones, validación de la entrada), disponibilidad (manejo de carga), autenticación ...



# 05

## Comprobaciones de seguridad a nivel de aplicación (ASVS y MASVS)



## Estándar de verificación de seguridad en las aplicaciones (ASVS)

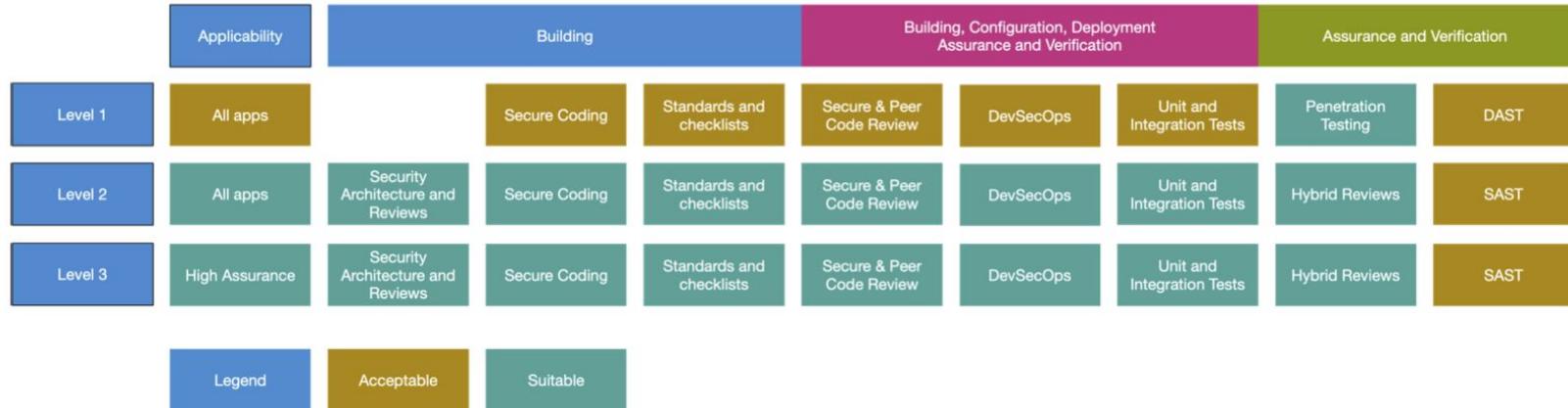
Como hemos comentado, el ASVS, contiene una lista de requerimientos de seguridad que pueden ser funcionalidades específicas de seguridad o capacidades que deben construirse por los desarrolladores de software.

- Estos requerimientos pueden ser diferentes para cada nivel.
- Los requerimientos se agrupan en diferentes capítulos que veremos a continuación.

# OWASP Application Security Verification Standard 4.0.2



The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.



Fuente de la imagen: <https://owasp.org/www-project-application-security-verification-standard/>

# Comprobaciones ASVS

## 1. Arquitectura, diseño y modelado de amenazas:

- Comprobar que todos los componentes de la aplicación se definen en función de las funciones de negocio.
- Comprobar que todos los controles de seguridad tienen una implementación centralizada.
- Comprobar que los componentes están separados unos de otros mediante controles de seguridad.
- Comprobar que todos los componentes de la aplicación están libres de vulnerabilidades conocidas.

# Comprobaciones ASVS

## 2. Verificación de autenticación:

- Comprobar que todas las páginas y recursos requieren autenticación menos las que deban de estar públicas.
- Comprobar que los campos de credenciales no reflejen las contraseñas de usuario.
- Comprobar que los campos de contraseña fomentan utilizar frases como autenticación.
- Comprobar que la funcionalidad de cambio de contraseña requiere de la contraseña anterior y contiene un campo de confirmación.
- Comprobar que las contraseñas almacenadas se utilizan un mecanismo de hashing esto es que no están almacenadas en texto plano.
- Comprobar que las credenciales viajan cifradas.
- Comprobar que no se utilizan contraseñas por defecto en la aplicación.
- Comprobar que existen medidas para bloquear el uso de contraseñas débiles.
- Comprobar que se utiliza doble factor de autenticación.
- Comprobar que las interfaces de administración no son accesibles a cualquier usuario.

# Comprobaciones ASVS

**3. Verificación de gestión de sesiones.** Una aplicación verificada debe cumplir:

- Las sesiones son únicas para cada individuo y no se pueden adivinar ni compartir.
- Las sesiones se invalidan cuando ya no son necesarias y se agota el tiempo de espera durante los períodos de inactividad.

**4. Verificación del control de acceso.** Una aplicación verificada debe cumplir:

- Las personas que acceden a los recursos tienen credenciales válidas para hacerlo.
- Los usuarios están asociados a un conjunto bien definido de roles y privilegios.
- Los metadatos de roles y permisos están protegidos contra la reproducción o la manipulación.

# Comprobaciones ASVS

**5. Verificación de Validación, Desinfección y Codificación.** Una aplicación verificada debe cumplir:

- La validación de entrada y la arquitectura de codificación de salida tienen un canal acordado para evitar ataques de inyección.
- Los datos de entrada están fuertemente tipados, validados, de rango o longitud comprobados, o en el peor de los casos, desinfectados o filtrados.
- Los datos de salida se codifican o escapan según el contexto de los datos lo más cerca posible del intérprete.

**6. Verificación para la criptografía en el almacenamiento.** Una aplicación verificada debe cumplir:

- Todos los módulos criptográficos fallan de forma segura y que los errores se gestionan correctamente.
- Se utiliza un generador de números aleatorios adecuado.
- El acceso a las claves se administra de forma segura.

# Comprobaciones ASVS

## 7. Verificación del manejo del registro y gestión de errores.

- No recopilar o registrar información confidencial a menos que sea específicamente necesario.
- Garantizar que toda la información registrada se maneje de forma segura y protegida según su clasificación de datos.
- Asegurarse de que los registros no se almacenan para siempre, pero tienen una duración absoluta que es lo más corta posible

## 8. Verificación de protección de datos.

Una aplicación verificada debe cumplir:

- Confidencialidad: Los datos deben protegerse de la observación o divulgación no autorizada tanto en tránsito como cuando se almacenan.
- Integridad: Los datos deben protegerse de ser creados, alterados o eliminados maliciosamente por atacantes no autorizados.
- Disponibilidad: los datos deben estar disponibles para los usuarios autorizados según sea necesario.

# Comprobaciones ASVS

**9. Verificación de seguridad en las comunicaciones.** Una aplicación verificada debe cumplir:

- Comprobar que puede construirse la cadena de confianza desde una Autoridad de Certificación para un certificado TLS del servidor.
- Comprobar que se utiliza TLS para todas las comunicaciones.
- Comprobar que existe una única implementación estándar de TLS.
- Comprobar que se utilizan únicamente algoritmos y protocolos de seguridad en comunicaciones fuertes.
- Comprobar que la configuración de TLS está en línea con las mejores prácticas actuales.

**10. Verificación ante el Código Malicioso.** El código debe de cumplir:

- La actividad maliciosa se controla de forma segura y adecuada para no afectar al resto de la aplicación.
- No tiene bombas de tiempo u otros ataques basados en el tiempo.
- No permite "llamar a casa" a destinos maliciosos o no autorizados.
- No tiene puertas traseras, huevos de pascua, salami attacks, rootkits o código no autorizado que pueda ser controlado por un atacante.

**11. Verificación para la lógica de negocio.** Una aplicación verificada debe cumplir:

- Comprobar que el flujo de la lógica de negocio es secuencial y en orden.
- Comprobar que la lógica de negocio incluye límites para detectar y evitar ataques automatizados.

# Comprobaciones ASVS

**12. Verificación de Archivos y Recursos.** Una aplicación verificada debe cumplir:

- Los datos de archivo que no son de confianza deben manejarse en consecuencia de una manera segura.
- Los datos de archivos que no son de confianza obtenidos de fuentes no confiables se almacenan fuera de la raíz web y con permisos limitados.

**13 Verificación de API y Servicios Web.** Una aplicación verificada que utiliza APIs de capa de servicio de confianza (normalmente mediante JSON o XML o GraphQL) debe cumplir:

- Autenticación adecuada, gestión de sesiones y autorización de todos los servicios web.
- Validación de entrada de todos los parámetros que transitan de un nivel de confianza inferior a superior.
- Controles de seguridad eficaces para todos los tipos de API, incluida la nube y los Serverless API

**14 Verificación de Configuración.** Una aplicación verificada debe cumplir:

- Un entorno de compilación seguro, repetible y automatizable.
- La aplicación no incluye la biblioteca de terceros reforzada, la dependencia y la administración de la configuración, de modo que la aplicación no incluya componentes obsoletos o no seguros.

# OWASP Web Security Testing Guide

OWASP, a partir del Estándar de Verificación de Seguridad en las aplicaciones (ASVS, <https://owasp.org/www-project-web-security-testing-guide/>), establece una guía específica de pruebas para aplicaciones y servicios web, **OWASP Security Testing Guide o Guía de pruebas de seguridad OWASP (WSTG)**.

- El objetivo del proyecto es ayudar a las personas a comprender qué, por qué, cuándo, dónde y cómo probar aplicaciones web.
- El proyecto proporciona un marco de prueba completo, no simplemente una lista de verificación o prescripción de problemas que deben abordarse.
- Este marco incluye las técnicas necesarias para ponerlo en práctica puede usarse como plantilla para crear programas de prueba
- Además OWASP pone a nuestra disposición **OWASP CHECK LIST** (<https://mas.owasp.org/checklists/>), un libro Excel que permite realizar la verificación del resultado de cada escenario, dandonos un resultado de la evaluación del nivel de riesgo.

# OWASP Web Security Testing Guide

La guía se estructura en escenarios agrupados en 12 categorías:

- Recopilación de información
- Pruebas de configuración y gestión de implementación
- Prueba de gestión de identidad
- Pruebas de autenticación
- Prueba de autorización
- Pruebas de gestión de sesión
- Prueba de validación de entrada
- Pruebas de manejo de errores
- Pruebas para criptografía débil
- Pruebas lógicas comerciales
- Pruebas del lado del cliente
- Pruebas de API

## Estándares de verificación de seguridad de dispositivos móviles (MASVS)

El Proyecto OWASP también nos proporciona el **Estándar de Verificación de Seguridad en Aplicaciones de dispositivos móviles (MASVS)** de sus siglas en inglés, <https://github.com/OWASP/masvs>) que, al igual que lo hacía el ASVS para aplicaciones web, reúne unos controles funcionales necesarios para diseñar, desarrollar y testear aplicaciones, pero en este caso en dispositivos móviles.

# Estándares de verificación de seguridad de dispositivos móviles (MASVS)

En este caso los controles vienen agrupados en 8 categorías:

- Requisitos de Arquitectura, Diseño y Modelado de Amenazas.
- Requisitos de Almacenamiento de datos y Privacidad.
- Requisitos de Criptografía.
- Requisitos de Autenticación y Manejo de Sesiones.
- Requisitos de Comunicación a través de la red.
- Requisitos de Interacción con la Plataforma.
- Requisitos de Calidad del Código y de la Configuración del Compilador .
- Requisitos de Resistencia ante la Ingeniería Inversa.

# OWASP Mobile Application Security Testing Guide

Al igual que ocurría con las aplicaciones web, OWASP, a partir del Estándar de Verificación de Seguridad en dispositivos móviles (MASVS), establece una guía específica de pruebas para aplicaciones y servicios en aplicaciones móviles, la **OWASP Mobile Application Security Testing Guide (MASTG)**.

- En este caso define un conjunto de casos de pruebas junto con los procesos, técnicas y herramientas a utilizar en dichas pruebas.
- En esta ocasión también disponemos de una hoja de cálculo que permite realizar el proceso de evaluación de los requerimientos, el **MAS Checklist**:  
<https://mas.owasp.org/checklists/>



# 05

## Aplicaciones Web inseguras



# Aplicaciones web inseguras

Como fuentes abiertas también podemos considerar máquinas o fragmentos de código que contienen vulnerabilidades y máquinas que se han creado para ser inseguras en las que podemos realizar pruebas.

## ¿Por qué practicar con aplicaciones web vulnerables?

- Permite conocer nuestras habilidades de piratería y saber cuál es su posición en el mundo de la seguridad.
- Antes de ingresar al mundo profesional de la seguridad de la información es necesario tener experiencia práctica con aplicaciones vulnerables.
- Ayuda en la capacitación en seguridad.
- También ayuda a comprender, identificar y reforzar las debilidades.
- Es bueno difundir conocimientos. Se pueden utilizar estas aplicaciones web para demostrar cómo se explotan las vulnerabilidades comunes a otros.

# Aplicaciones web inseguras

Las distintas herramientas de prueba de penetración o de análisis de seguridad (que se verán en la siguiente unidad) van de la mano con una serie de aplicaciones web deliberadamente vulnerables que sirven como **plataformas de entrenamiento** a la hora de buscar errores y vulnerabilidades.

- Permiten practicar habilidades antes de enfrentarse a escenarios de seguridad del mundo real.
- Van a permitir comprender los procedimientos y métodos desde el punto de vista del atacante, para posteriormente pasar "al otro lado", protegiendo las aplicaciones web y realizando desarrollos seguros.
- De entre la multitud de aplicaciones web rotas o vulnerables que se pueden usar y que pueden alojarse fácilmente en localhost, destacan las descritas a continuación.
  - (Se facilita un conjunto de aplicaciones a modo de catálogo para poder practicar).

# Damn Vulnerable Web App: DVWA

**DVWA - Significa Damn Vulnerable Web App** (<https://github.com/digininja/DVWA>).

Está basado en PHP y se ejecuta en el servidor de base de datos MySQL, que de hecho es muy vulnerable. Tiene tres niveles de seguridad: Bajo, Medio y Alto, que exigen diferentes habilidades. Presenta vulnerabilidades como XSS, CSRF, inyección SQL, inyección de archivos, fallas de carga y más. Está disponible su código fuente.



# Badstore

**Badstore:** Tiene vulnerabilidades como cross-site scripting (XSS), inyección SQL, clickjacking, hash de contraseña (decodificación MD5) y contiene un archivo robot.txt para más exploits. Se incluye en una .iso que puede utilizar desde Virtualbox para emular un servidor.



The screenshot shows a web browser displaying the Badstore.NET homepage. The title bar reads "BADSTORE.NET". The main content area has a green header with the text "Welcome {Unregistered User} - Cart contains 0 items at \$0.00" and a "View Cart" button. Below this, a search bar contains the text "book" and displays the message "No items matched your search criteria:". Underneath, a database query is shown: "SELECT itemnum, sdesc, ldesc, price FROM itemdb WHERE 'book' IN (itemnum,sdesc,ldesc)". At the bottom of the page, a copyright notice reads "BadStore v2.1.2 • Copyright © 2003-2006".

# Metasploitable3

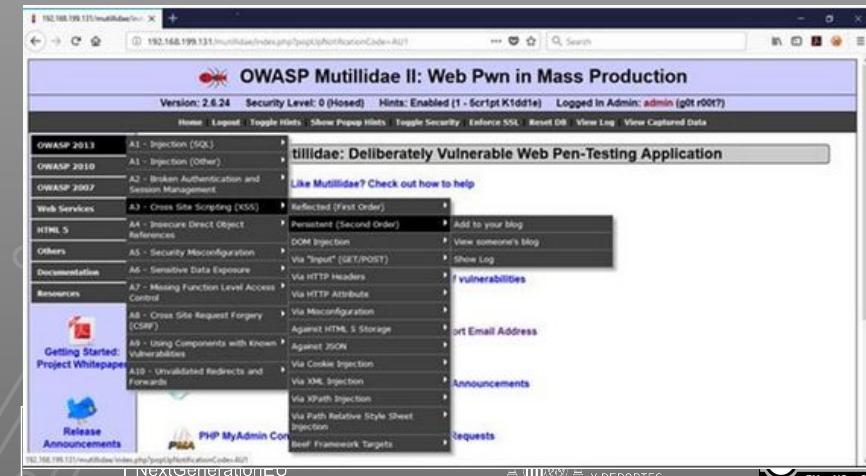
## Metasploitable 3

(<https://sourceforge.net/projects/metasploitable3-ub1404upgraded/>) es la aplicación web vulnerable más común, y se suele combinar con herramientas como Metasploit y Nmap para probarla. Se utiliza principalmente para pruebas de red. Tiene TWiki, phpMyAdmin, WebDAV y DVWA integrados. Se incluye en una .iso o OVA.



# OWASP Mutillidae II

**Mutillidae II** (<https://owasp.org/www-project-mutillidae-ii/>): desarrollada por el propio OWASP, incluye un entorno de laboratorio completo. Tiene vulnerabilidades para probar como XSS, inyección SQL, inyección HTML, clickjacking, bypass de autenticación y muchas otras vulnerabilidades. También tiene subcategorías en su sección de vulnerabilidades que proporciona más opciones, así como un modo seguro/inseguro. Se necesita instalar XAMPP. Incluye un botón de "setup" para restaurar a los valores predeterminados y una página de captura de datos.



# Web Security Dojo - WSD

Web Security Dojo (WSD,

<https://sourceforge.net/projects/websecuritydojo/>): es una máquina virtual (.ova) que contiene muchas herramientas (como Burp Suite, w3af, Ratproxy y SQLmap.) Y máquinas de destino (WebGoat y Hacme Casino, entre otras). Es un entorno de formación de código abierto basado en Xubuntu 12.04. También contiene materiales de capacitación y guías de usuario para algunos objetivos.



# OWASP JUICE SHOP

OWASP Juice Shop: es lo opuesto a una aplicación de mejores prácticas o plantilla para desarrolladores web.

- Es una herramienta de concienciación, capacitación, demostración y ejercicio para los riesgos de seguridad aplicaciones web.
- Enlaces importantes para trabajar con Juice Shop:
  - Repaso:  
<https://hackpuntos.com/owasp-juice-shop-project-laboratorio-para-practicar-las-10-vulnerabilidades-mas-comunes-en-aplicaciones-web/>
  - Libro oficial:  
<https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/>
  - Web oficial: <https://owasp-juice.shop/>

Fuente de la imagen:  
<https://github.com/juice-shop/juice-shop>



## Otras aplicaciones web vulnerables realmente útiles:

- Zero Bank: <http://zero.webappsecurity.com/>
- BWAPP: <http://www.itsecgames.com/>
- WebGoat: <https://owasp.org/www-project-webgoat/>
- Hackxor: <https://hackxor.net/>
- WebMaven:  
<https://www.mavensecurity.com/about/webmaven>
- Vicnum: <https://sourceforge.net/projects/vicnum/>
- XXE: <https://sourceforge.net/projects/xxe/>
- Web Application Exploits and Defenses:  
<https://google-gruyere.appspot.com/>
- Spiracle: <https://github.com/waratek/spiracle>

# Bibliografía y Webgrafía

- Fernando Saavedra:  
<https://www.audea.com/owasp-top-ten-mobile-risks/>
- OWASP TOP 10  
<https://owasp.org/www-project-top-ten/>
- OWASP MOBILE TOP 10 -  
<https://owasp.org/www-project-mobile-top-10/>
- OWASP ASVS  
<https://owasp.org/www-project-application-security-verification-standard/>
- OWASP MASVS - <https://mas.owasp.org/MASVS>
- OWASP Security Testing Guide -  
<https://owasp.org/www-project-web-security-testing-guide/>

# Gracias!

¿Alguna pregunta?



[informatica.iesvalledeljerteplasencia.es](http://informatica.iesvalledeljerteplasencia.es)



[coordinacion.cenfp@iesvp.es](mailto:coordinacion.cenfp@iesvp.es)



C/ Pedro y Francisco González, s/n  
10600, Plasencia (Cáceres)



927 01 77 74

