

# Puesta en Producción Segura

Unidad 3.

Herramientas de detección de vulnerabilidades.



*"Probar la seguridad no es un obstáculo: es la mejor manera de aprender a construir software fuerte y confiable."*

# Objetivos

- Definir y diferenciar los distintos enfoques de Application Security Testing (SAST, DAST, IAST, MAST, RASP, etc.).
- Reconocer las ventajas, limitaciones y casos de uso de cada tipo de prueba de seguridad en el ciclo de vida de desarrollo de software.
- Identificar herramientas representativas (tanto de código abierto como comerciales) para cada categoría de AST.
- Relacionar las pruebas de seguridad con el modelo DevSecOps, comprendiendo cómo se integran en pipelines de CI/CD.
- Desarrollar un criterio crítico para seleccionar la herramienta o enfoque más adecuado según el contexto del proyecto (web, móvil, cloud, etc.).
- Promover la cultura de seguridad desde el diseño, entendiendo la importancia de aplicar pruebas continuas en todas las fases del SDLC.



# Contenidos

**01** Herramientas SAST

**02** Herramientas DAST

**03** Herramientas IAST

**04** Herramientas MAST

**05** Otras herramientas AST

# Herramientas de detección de vulnerabilidades

Existen diferentes tipos de herramientas de detección de vulnerabilidades que permiten mejorar la seguridad y la calidad del código:

- Herramientas de prueba de seguridad de aplicaciones **estáticas (SAST)**.
- Herramientas de prueba de seguridad de aplicaciones **dinámicas (DAST)**.  
(Principalmente para aplicaciones web)
- Herramientas de prueba de seguridad de aplicaciones **interactivas (IAST)**.  
(Principalmente para aplicaciones web y API web)
- Herramientas de prueba de seguridad de **aplicaciones móviles (MAST)**.
- Herramientas de **calidad de código** estático.
- Herramientas de análisis de dependencias para mantener bibliotecas de código abierto actualizadas (para evitar el uso de componentes con vulnerabilidades conocidas).



01

# Herramientas de pruebas de seguridad estáticas (SAST)



# Herramientas de análisis estático de código (SAST)

- Entre los tipos de pruebas que pueden utilizarse para comprobar que el código software de una aplicación funciona correctamente, tenemos el **análisis estático de código**, que ayuda a poner de manifiesto problemas no funcionales de la aplicación y a prevenirlos.
- Las herramientas SAST hace referencia a aquellas técnicas de revisión de seguridad de tipo **caja blanca** aplicadas sobre el código fuente del software: su objetivo es la búsqueda de defectos de seguridad en el código fuente de las aplicaciones.
- Una ventaja importante del análisis estático de código es que proporciona importantes ahorros de costes. Al poder anticipar posibles problemas antes de que se hagan realidad: no olvidemos que los gastos aumentan exponencialmente a medida que va avanzando el ciclo de vida.

# Herramientas de análisis estático de código (SAST)

- En el momento en que el análisis estático de código se introduce en el ciclo de vida se produce un cambio en los equipos de desarrollo: se dan ocasiones en las que se adoptan malas prácticas simplemente por el hecho de que “siempre se ha hecho así”. En estos casos, el efecto de evidenciar el problema, explicar y justificar la forma correcta de hacerlo, suele ser inmediato.
- Por otra parte, habría que decidir la conveniencia de acudir al análisis manual o al automatizado.
- Los criterios para elegir uno u otro vienen condicionados por aspectos como:
  - – La tecnología que se pretende analizar.
  - – Las licencias de las herramientas.
  - – El establecimiento de prioridades, es decir, hasta qué punto profundizar en el análisis.



# Herramientas de análisis estático de código (SAST)

- Como todo software, las herramientas SAST también se pueden dividir en **libres y privativas**.
  - – **Libres**: FingBugs, Kiuwan, SonarQube, CodeWarrior, JSHint, NodeJsScan...
  - – Dentro de las **privativas** destaca la solución de **Synopsys**, la de **Veracode**, **Fortify** (solución de Micro Focus) y **Checkmarx**.
- Para poderlas probar, el proyecto NIST Software Assurance Metrics And Tool Evaluation (**SAMATE**) que está dedicado a mejorar la seguridad del software mediante el desarrollo de métodos que permitan evaluar las herramientas de análisis de software y medir su eficacia.
- Para ampliar:  
[https://owasp.org/www-community/Free for Open Source Application Security Tools](https://owasp.org/www-community/Free_for_Open_Source_Application_Security_Tools) y [https://owasp.org/www-community/Source Code Analysis Tools](https://owasp.org/www-community/Source_Code_Analysis_Tools)



# ¿Qué es la deuda técnica?

- El término deuda técnica no solo es aplicable al análisis estático de código. Se trata de un concepto que pretende poner en términos financieros cualquier carencia técnica de un producto.
- En el caso que nos ocupa, la deuda técnica se define al coste de desarrollo para eliminar los riesgos debidos a la calidad de código en un entorno productivo. Adicionalmente, podemos incluir:
- - El coste asociado a la complead,
  - - La falta de documentación,
  - - Difícil testeo, etc.,
- Según la herramienta que se utilice, la deuda técnica puede venir dada en jornadas o en un valor monetario.



# ¿Qué es la deuda técnica?

Independientemente de cómo se exprese, el cálculo se realiza en base a los siguientes factores:

- Asignación de un tiempo de corrección a cada incumplimiento de cada regla.

- Ponderación de dicho tiempo con un factor de complejidad asociado al elemento en el que se produce el incumplimiento.

- De esta forma, los incumplimientos de la misma regla, resultan más costosos de corregir según la complejidad del objeto en el que se producen.





02

# Herramientas de Pruebas dinámicas de seguridad (DAST)

# Herramientas de prueba de seguridad dinámicas (DAST)

- Las herramientas **DAST (Dynamic Application Security Testing)** analizan las aplicaciones en su **estado dinámico de ejecución** durante las fases de prueba o de operación.
- Se ejecuta en el código operativo para detectar problemas con interfaces, solicitudes, respuestas, secuencias de comandos, inyección de datos, sesiones, autenticación y más.
- Simulan ataques contra una aplicación (generalmente aplicaciones y servicios habilitados para la web) y analiza las reacciones de la aplicación para determinar si es vulnerable.
- El probador no tiene conocimiento previo del sistema (en las pruebas SAST, sí):
- Detectan condiciones que indican una vulnerabilidad de seguridad en una aplicación en su estado de ejecución.
- Hacen uso de fuzzing: lanzar casos de prueba conocidos, no válidos e inesperados y, a menudo, en gran volumen.

# Escáneres de vulnerabilidades web

- Un escáner de seguridad de aplicaciones web es un programa de software que realiza pruebas automáticas de caja negra en una aplicación web e identifica vulnerabilidades de seguridad.
- Los escáneres no acceden al código fuente; solo realizan pruebas funcionales e intentan encontrar vulnerabilidades de seguridad.
  - – Tienen una base de datos de vulnerabilidades en función de la cual realizan la verificación en el host remoto. Esta base de datos contiene toda la información requerida (servicio, puerto, tipo de paquete, una ruta potencial para explotar, etc.) para verificar el problema de seguridad.
  - – Pueden escanear la red y los sitios web en busca de miles de vulnerabilidades, proporcionar la lista de problemas según el riesgo y sugerir también la solución.

# Escáneres de vulnerabilidades web

- Un escáner de seguridad de aplicaciones web es un programa de software que realiza pruebas automáticas de caja negra en una aplicación web e identifica vulnerabilidades de seguridad.
- Los escáneres no acceden al código fuente; solo realizan pruebas funcionales e intentan encontrar vulnerabilidades de seguridad.
  - – Tienen una base de datos de vulnerabilidades en función de la cual realizan la verificación en el host remoto. Esta base de datos contiene toda la información requerida (servicio, puerto, tipo de paquete, una ruta potencial para explotar, etc.) para verificar el problema de seguridad.
  - – Pueden escanear la red y los sitios web en busca de miles de vulnerabilidades, proporcionar la lista de problemas según el riesgo y sugerir también la solución.

# Algunas herramientas DAST de código abierto

- OWASP ZAP (proxy de ataque Zed): Es simple y fácil de usar. Se puede utilizar para encontrar una amplia gama de vulnerabilidades, como interceptor proxy, escáner automático, arañas tradicionales (pero poderosas), fuzzer, soporte web socket, soporte plug-n-hack, soporte de autenticación, API basada en REST, certificados SSL dinámicos, compatibilidad con tarjetas inteligentes y certificados digitales de clientes.
  - Se puede usar como un escáner ingresando la URL para realizar el escaneo o como un proxy de interceptación para realizar pruebas manualmente en páginas específicas.
- W3af: es un marco de auditoría y ataque de aplicaciones web popular. Fue desarrollado usando Python y es capaz de identificar más de 200 tipos de vulnerabilidades. Viene con una interfaz gráfica y de consola. Su uso es sencillo, gracias a su intuitiva interfaz.
  - – Si un sitio web necesita autenticación, también puede utilizar módulos de autenticación para escanear las páginas protegidas por sesión.



OWASP  
Zed Attack Proxy

JUNTA DE EXTREMADURA



Plan de  
Recuperación,  
Transformación  
y Resiliencia



Financiado por  
la Unión Europea  
NextGenerationEU



w3af

Web Application Attack and Audit Framework



# Algunas herramientas DAST de código abierto

- **Wapiti:** Realiza pruebas de caja negra escaneando páginas web e inyectando datos a través de línea de comandos. Más compleja. Intenta inyectar cargas útiles y ver si un script es vulnerable. Admite ataques GET y POSTHTTP.
  - Detecta múltiples vulnerabilidades, entre las que destacan inyección CRLF, inyección SEL, inyección Xpath y configuración de .htaccess débil
- **Nikto:** Simple de usar para hallar vulnerabilidades conocidas / problemas de configuración de servidores/servicios web. Buen complemento.
  - No va a cubrir toda la profundidad de bugs complejos, poco adaptado para APIs modernas o aplicaciones JS pesadas.



# Catálogo comercial DAST

- **Invicti:** Escaneo “proof-based” (pruebas que confirman la vulnerabilidad), buen soporte para APIs, integración con CI/CD, buen reporting, reducción de falsos positivos en muchos casos.
- **Acunetix:** Muy usada para aplicaciones web, APIs, buena interfaz, bastante estable, detección de muchas vulnerabilidades comunes, integración con DevSecOps.
- **Rapid7 InsightAppSec:** Enfoque cloud, bastante automatización, integración con otras herramientas de seguridad y monitoreo, suitable para escenarios de producción y APIs modernas.



# Catálogo comercial DAST

- **Burp Suite:** es una herramienta basada en Java.
- La versión gratuita de Burp Suite tiene las siguientes características:
  - Un proxy de interceptación para analizar y manipular la solicitud y la respuesta del backend.
  - Burp Spider para rastrear las páginas y el enlace de la aplicación de destino.
  - Repetidor para manipular y reenviar la solicitud varias veces.
  - Secuenciador para analizar la aleatoriedad y la fuerza del token de sesión.
  - Burp Intruder para realizar un ataque automatizado personalizado para encontrar y explotar vulnerabilidades.
    - Algunas características que solo están presentes en la **versión profesional** incluyen:.
  - Un escáner de aplicaciones web avanzado para detectar automáticamente las vulnerabilidades.
  - Burp Extension que permite escribir complementos propios para realizar la tarea de manera personalizada.
  - La opción de guardar el estado actual y usarlo más tarde.
  - La capacidad de generar un informe de escaneo.





03

# Herramientas de prueba de seguridad interactivas (IAST)



# Herramientas de pruebas de seguridad interactivas (IAST)

- **IAST (Interactive Application Security Testing)** es un tipo de prueba de seguridad de aplicaciones que combina lo mejor de SAST (análisis estático de código) y DAST (análisis dinámico).
- Se ejecuta **desde dentro de la aplicación** mientras esta corre (normalmente en un entorno de pruebas o staging).
- Utiliza instrumentación en tiempo de ejecución (agentes dentro de la JVM, .NET CLR, etc.) para observar:
  - El flujo real de datos.
  - La interacción con la memoria, librerías y frameworks.
  - Cómo se ejecutan realmente las peticiones y consultas (p. ej. SQL, LDAP, XML, etc.).
- Permite detectar vulnerabilidades en tiempo real, con menos falsos positivos que SAST y más precisión que DAST.
- Aporta contexto: sabe en qué línea de código ocurre la vulnerabilidad y qué llamada concreta la provoca.
- Es especialmente útil en DevSecOps, ya que se integra en CI/CD y permite feedback rápido y contextual.

# Principales herramientas IAST de código abierto

Las **herramientas IAST puras open source** son muy escasas (la mayoría de las opciones IAST potentes son comerciales), pero existen proyectos e iniciativas relacionados:

- **Contrast Community Edition**: Tiene versión gratuita limitada (IAST para Java y .NET).
- **AppScan CodeSweep Community (IBM)**: más un SAST, pero con algunos elementos IAST en entornos de prueba.
- Algunos investigadores han creado instrumentación DIY con agentes Java (**Byteman**, **ASM**, **AspectJ**) para **simular IAST**, pero no son productos listos para producción.

# Principales herramientas IAST privativas

Como decíamos, la mayor parte de herramientas IAST potentes, son privativas:

- **Contrast Security (Contrast Assess)**: es uno de los referentes en IAST.
- **Synopsys Seeker**: es una solución IAST muy implantada en entornos enterprise.
- **Micro Focus Fortify IAS**: parte del ecosistema Fortify.
- **HCL AppScan IAST**: es la evolución de AppScan con motor de instrumentación.
- **Veracode IAST**: está integrado con la plataforma de análisis de Veracode.



04

# Herramientas de prueba de seguridad de aplicaciones móviles (IAST)



# Pruebas de seguridad de aplicaciones móviles (MAST)

- MAST (Mobile Application Security Testing) es el proceso de evaluar la seguridad de aplicaciones móviles. Incluye pruebas tanto automáticas como manuales para identificar vulnerabilidades y puntos débiles en aplicaciones de iOS, Android y otras plataformas móviles. Muchas empresas de tecnología también usan el término mast security testing o security mast para referirse a estas pruebas especializadas.
- Las herramientas MAST tienen características especializadas que se centran en problemas específicos de las aplicaciones móviles, como el desbloqueo o el enraizamiento del dispositivo, las conexiones WI-FI falsificadas, el manejo y la validación de certificados, la prevención de fugas de datos y más.



# Herramientas MAST Open Source

## MobSF (Mobile Security Framework)

- Una de las más populares para análisis estático y dinámico de apps Android/iOS.
- Permite análisis de binarios (APK/IPA), auditorías de código fuente y hooking dinámico.
- Soporta integración en pipelines CI/CD.

## QARK (Quick Android Review Kit)

- Creado por LinkedIn. Se centra en Android.
- Analiza código fuente y APKs buscando malas prácticas de seguridad.
- Genera exploits PoC automáticos.



# Herramientas MAST Open Source

## Drozer

- Framework para pruebas de seguridad en Android, especialmente para componentes (intents, actividades, servicios).
- Muy útil en análisis dinámico.

## Frida / Objection

- Frida: framework de instrumentación en tiempo real, muy usado para reverse engineering y bypass de protecciones.
- Objection: interfaz más sencilla sobre Frida, orientada a pentesters móviles.

## Androguard

- Framework en Python para análisis estático de APKs.
- Muy usado en ingeniería inversa y automatización.



# Herramientas MAST Privativas

## NowSecure Platform

- Plataforma SaaS de análisis automatizado para Android/iOS.
- Genera reportes con referencia a OWASP MASVS.

## Pradeo Security

- Orientada a la seguridad corporativa (MDM/MAM + análisis de apps).
- Detecta comportamiento malicioso y riesgos de privacidad.

## Zimperium (zScan / zIAP)

- Enfocada en MAST y RASP (protección en tiempo de ejecución).
- Muy usada en entornos empresariales.



# Herramientas MAST Privativas

## **AppScan (HCL AppScan, antes IBM AppScan)**

- Solución más general de AST, con soporte para móviles.
- Integración en entornos DevSecOps.

## **Veracode Mobile Security Testing**

- Servicio de análisis estático y dinámico, parte de la suite de Veracode.

## **Checkmarx MAST**

- Basado en su motor de SAST, pero con análisis extendido para Android/iOS.





05

# Otras herramientas AST



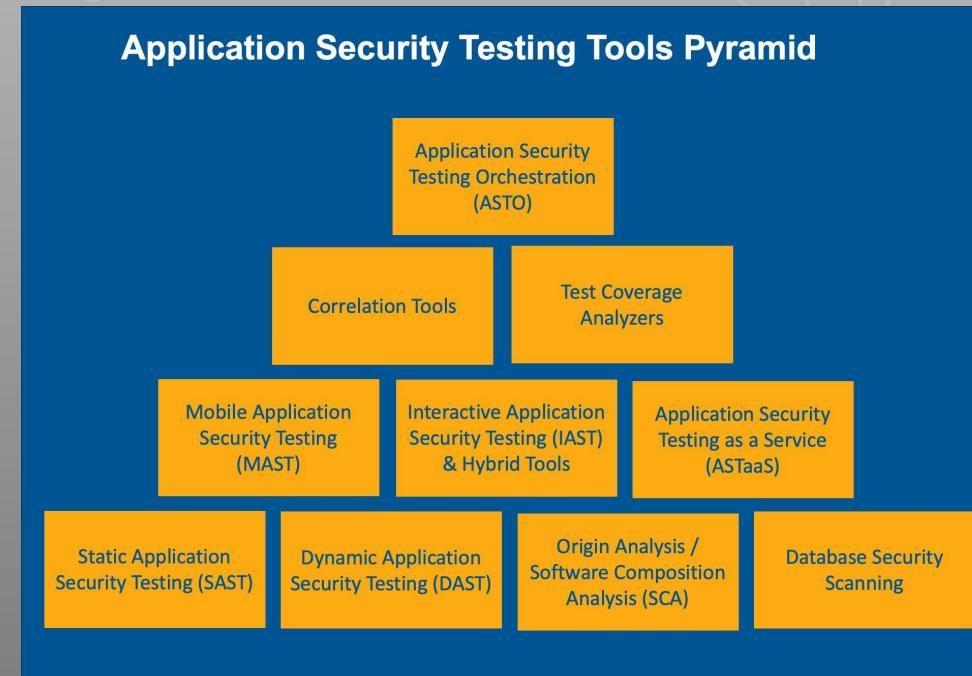
# Otras tecnologías AST

Algunos proveedores de AST también han comenzado a ofrecer:

- **RASP (Autoprotección de aplicaciones en tiempo de ejecución o Run-time Application Security Protection)**: Esta tecnología aprovecha los principios de instrumentación IAST para ofrecer una solución de protección que compita con los Web Application Firewall (WAF). La mayoría de los equipos comienzan probando la instrumentación en modo IAST (sin bloqueo) y, una vez que se sienten cómodos, pasan a RASP (modo de bloqueo completo).
- **SCA (Análisis de composición de software)**: es una tecnología utilizada para identificar componentes de fuente abierta y de terceros en uso en una aplicación y sus vulnerabilidades de seguridad conocidas. Las herramientas de SCA examinan el software para determinar los orígenes de todos los componentes y bibliotecas dentro del software. Estas herramientas son altamente efectivas para identificar y encontrar vulnerabilidades en componentes comunes y populares, particularmente en componentes de código abierto. Sin embargo, no detectan vulnerabilidades para los componentes desarrollados por la propia empresa. Ya hemos hablado de ellos cuando hemos visto vulnerabilidades de librerías de terceros.

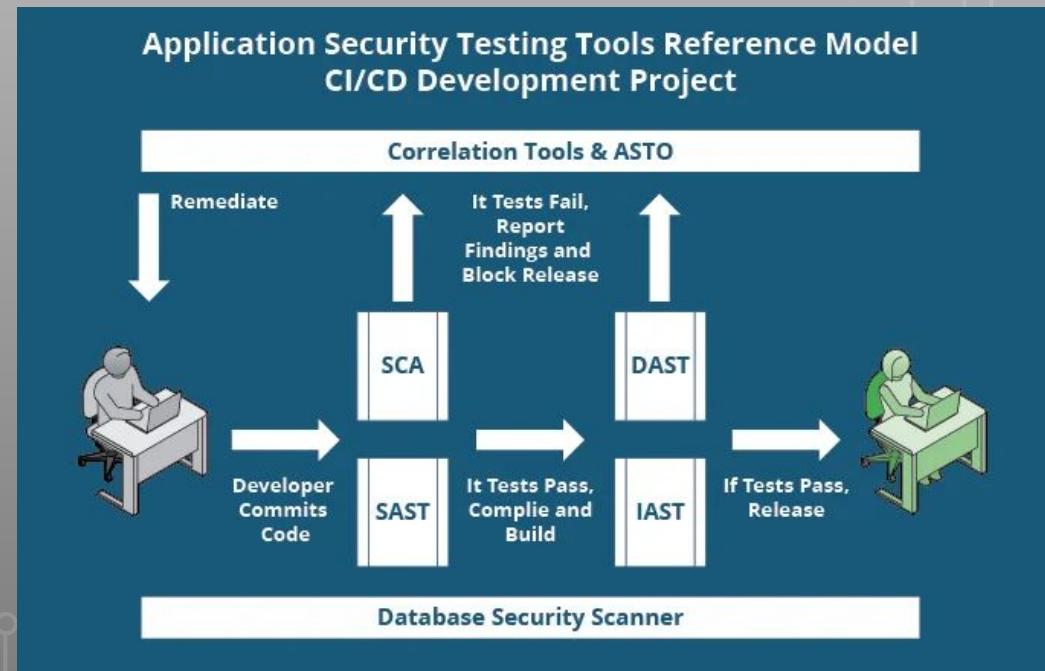
# Otras tecnologías AST

Este gráfico muestra un resumen de las diferentes clases o categorías de herramientas de prueba de seguridad de aplicaciones:



# Otras tecnologías AST

A medida que se analiza los resultados con una herramienta, puede ser conveniente introducir herramientas adicionales en su entorno. Como ejemplo de referencia, el gráfico a continuación describe cuántas clases de herramientas podrían implementarse de manera efectiva en un proceso de desarrollo de integración continua y entrega continua (CI/CD).



Fuente de la imagen:

<https://www.sei.cmu.edu/blog/10-types-of-application-security-testing-tools-when-and-how-to-use-them/>

# Bibliografía y Webgrafía

- **Presentaciones de Puesta en Producción Segura.** *Rafael López García.*
- **Seguridad de aplicaciones.** José Losada Pérez.
- **Presentaciones de Puesta en Producción Segura.** *Rafael Fuentes Ferrer.*
- [\*\*https://www.sei.cmu.edu/blog/10-types-of-application-security-testing-tools-when-and-how-to-use-them/\*\*](https://www.sei.cmu.edu/blog/10-types-of-application-security-testing-tools-when-and-how-to-use-them/)



# 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

