DevSecOps

Security testing Automation

Un guía práctica sobre la nueva era del testing de seguridad

DEVSECOPS Argentina

¿Que veremos?

- Glosario
- Introducción a DevOps seguro
- Mejor seguridad con DevOps
- Application Security Testing: estado actual
- Application Security testing tools: cuándo y cómo usarlas
- Application Security testing tools: criterios de selección
- Application Security testing tools: lista por categoría
- Conclusiones
- Autores
- Links y Referencias

Imágenes creadas por dooder

Glosario

Imágenes creadas por dooder

Glosario

DevOps: es una filosofía, una cultura o metodología que revoluciona el modo en que se gestiona el ciclo de desarrollo software

DevSecOps: acrónimo inglés de la unificación de development (desarrollo), Security (Seguridad) y operations (operaciones)

Desarrollo Ágil: El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.

AST: Application Security Testing, Son técnicas de prueba de seguridad que buscan vulnerabilidades o agujeros de seguridad en las aplicaciones.

SDLC: son las siglas de: Systems Development Life Cycle, también conocido como "System Design Life Cycle" (ciclo vital del desarrollo/diseño de sistemas).

Pipeline: es un concepto para evitar el desperdicio en el proceso de desarrollo de software , y se utiliza para proporcionar comentarios rápidos al equipo durante la implementación

IAC: A veces denominada "infraestructura programable", la infraestructura como código (IaC) trata la configuración de la infraestructura exactamente como el software de programación.

CD Despliegue continuo: La entrega continua es en realidad una extensión de la Integración Continua, en la cual el proceso de entrega de software se automatiza para permitir implementaciones fáciles y confiables en la producción, en cualquier momento.

CI Integración continua: consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. la compilación y ejecución de pruebas de todo un proyecto.

Dependencia: es una aplicación o una biblioteca requerida por otro programa para poder funcionar correctamente

Software Security: La seguridad del software es una idea implementada para proteger el software contra ataques maliciosos y otros riesgos de piratas informáticos para que el software continúe funcionando correctamente bajo tales riesgos potenciales.

Glosario

ASTO: Herramienta para la orquestación centralizada y coordinada y la generación de informes de todas las diferentes herramientas de AST

SAST: o prueba de seguridad de aplicación estática, también conocida como "prueba de caja blanca“ Permite a los desarrolladores encontrar vulnerabilidades de seguridad en el código fuente de la aplicación.

DAST: o prueba de seguridad de aplicación dinámica, también conocido como prueba de "caja negra", puede encontrar vulnerabilidades y debilidades en la seguridad de una aplicación en ejecución, normalmente aplicaciones web.

IAST: Pruebas de seguridad de aplicaciones interactivas. Es una combinación entre las técnicas de los análisis estáticos y dinámicos (SAST + DAST) generando un análisis global de todo el sistema.

SCA: Software Composition Analysis se encarga de identicar, monitorear y crear un inventario de todos los componentes de terceros en el software con el fin de responder frente a vulnerabilidades en dichos componentes**.**

RASP: o protección de seguridad de la aplicación en tiempo de ejecución Al igual que con IAST, RASP o Runtime Application Security Protection, funciona dentro de la aplicación, pero es menos una herramienta de prueba y más una herramienta de seguridad.

ASTaaS: Application Security testing as a Service o pruebas de aplicaciones como servicios.

Correlation: Tools Estas herramientas ayudan a reducir parte del ruido al proporcionar un depósito central para los hallazgos de otras herramientas AST

MAST: Mobile Application Security testing permite implementar el código del lado del cliente, el código del lado del servidor y el análisis de bibliotecas de terceros en sus aplicaciones móviles, sin la necesidad de un código fuente

Database Security Scanning: herramienta que ayuda a identificar las áreas donde la configuración, operación o implementación de su base de datos introduce riesgos y recomienda cambios y controles para mitigar esos riesgos.

Test-Coverage Analyzers: Los Test-coverage analyzers miden cuánto del código total del programa ha sido analizado.

Glosario

Vulnerabilidad de día cero: es un ataque contra una aplicación o sistema que tiene como objetivo la ejecución de código malicioso gracias al conocimiento de vulnerabilidades que, por lo general, son desconocidas para la gente y el fabricante del producto.

Fuzztesting: Es una técnica de pruebas de software, a menudo automatizado o semiautomatizado, que implica proporcionar datos inválidos, inesperados o aleatorios a las entradas de un programa de ordenador.

Test de penetración: o "pentest", es un ataque a un sistema informático con la intención de encontrar las debilidades de seguridad y todo lo que podría tener acceso a ella, su funcionalidad y datos.

Framework: o marco de trabajo2 es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Vulnerabilidades: es una debilidad o fallo en un sistema de información que pone en riesgo la seguridad de la información pudiendo permitir que un atacante pueda comprometer el mismo.

Testeo automatizado: Cuando hablamos de pruebas automatizadas nos estamos refiriendo a: un método de ejecutar una prueba sin intervención humana, que de lo contrario lo requeriría.

Backlog: es una lista de tareas asociadas a un proyecto de desarrollo de software

cross-script scripting (XSS): es un tipo de vulnerabilidad informática típica de las aplicaciones Web.

SQL injection: es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.

Server file system traversal: consiste en explotar una vulnerabilidad informática que ocurre cuando no existe suficiente seguridad en cuanto a la validación de un usuario, permitiéndole acceder a cualquier tipo de directorio superior (padre) sin ningún control.

Glosario

Parches: un parche consta de cambios que se aplican a un programa, para corregir errores, agregarle funcionalidad, actualizarlo, etc.

Componentes: es una unidad modular de un programa software con interfaces y dependencias bien definidas que permiten ofertar o solicitar un conjunto de servicios o funcionales.

código abierto: es un modelo de desarrollo de software basado en la colaboración abierta.

Riesgo: Riesgo es una medida de la magnitud de los daños frente a una situación peligrosa.

Black box o Black hat: En una asignación de prueba de caja negra, el probador de penetración se coloca en el rol de pirata informático promedio, sin conocimiento interno del sistema objetivo.

White box o White hat: Es el extremo opuesto del espectro de las pruebas de caja negra y los probadores de penetración tienen acceso completo al código fuente, a la documentación de arquitectura, etc

OWASP: es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.

OWASP Top 10: es un ranking de los diez riesgos de seguridad más importantes en aplicaciones web según la organización OWASP

SANS Top 25: una lista de los errores más comunes y críticos que pueden provocar graves vulnerabilidades en el software

CERT Coding Standards: es un estándar de codificación seguro. Está desarrollado por la división CERT del Instituto de Ingeniería de Software de la Universidad Carnegie Mellon.

ACL: Una ACL especifica a qué usuarios o procesos del sistema se les otorga acceso a los objetos, así como qué operaciones se permiten en los objetos dados.

API: La interfaz de programación de aplicaciones, conocida también por la sigla API, es un conjunto de subrutinas, funciones y procedimientos

Glosario

IDE: es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software

CVE: es una lista de información registrada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación único.

Falso positivo: Se genera un falso positivo cuando un antivirus o sistema de seguridad interpreta que un código, programa, aplicación, dirección web, archivo etc. legítimo está infectado por un malware o tiene un vulnerabilidad, sin que en realidad sea así.

Rama: Una definición muy simplificada sobre lo que es una rama en un software de gestión de código es que es un duplicado del código de la rama principal, en un determinado momento en la línea temporal de desarrollo

Autoria: La autoría en el contexto de las pruebas de seguridad de la aplicación se refiere a quién desarrolla el código fuente bajo evaluación

Stack: Una pila es una lista ordenada o estructura de datos que permite almacenar y recuperar datos, el modo de acceso a sus elementos es de tipo LIFO. Esta estructura se aplica en multitud de supuestos en el área de informática debido a su simplicidad y capacidad de dar respuesta a numerosos procesos

FISMA: Ley Federal de Gestión de Seguridad de la Información

HIPAA: La Ley de Transferencia y Responsabilidad de Seguro Médico (Health Insurance Portability and Accountability Act)

SOX: Ley Sarbanes-Oxley

PCI: El Estándar de Seguridad de Datos para la Industria de Tarjeta de Pago o PCI DSS fue desarrollado por un comité conformado por las compañías de tarjetas más importantes

COBIT: Objetivos de Control para Información y Tecnologías Relacionadas, es una guía de mejores prácticas presentada como framework para TI

Normas ISO: Las normas ISO son documentos que especifican requerimientos que pueden ser empleados en organizaciones para garantizar que los productos y/o servicios ofrecidos por dichas organizaciones cumplen con su objetivo

Introducción a DevOps seguro

Imágenes creadas por dooder

Introducción a DevOps seguro

En base a nuestra experiencia llevando adelantes mas de 100 procesos de implementación de seguridad en el SDLC ciclo de vida del desarrollo de software podemos afirmar que el término **"seguridad de software"** usualmente genera sentimientos negativos entre los equipos de operaciones y desarrolladores de software, ya que está asociado con el esfuerzo de adicional, la incertidumbre y stopers en el desarrollo Ágil y el ciclo de lanzamiento. Para asegurar el software, los desarrolladores deben seguir numerosas pautas que, si bien están destinadas a satisfacer algunas regulaciones u otras demandas, pueden ser muy restrictivas y difíciles de entender. Como resultado, una gran cantidad de temor, incertidumbre y duda pueden rodear la seguridad de software.

Este e-book es la tercer entrega de DevSecOps Argentina con la colaboración de **Cloud Legion** en El, vamos a trata de describir cómo el movimiento **DevOps Seguro** y DevSecOps.org pueden ayudar a transformar el entorno reactivo que rodea la seguridad del software, impulsando un cambio de paradigma de las reglas y directrices en busca de soluciones creativas para problemas difíciles de seguridad.

DEVSECOPS Argentina

CLOUD LEGION

Introducción a DevOps seguro

Un enfoque proactivo en la seguridad del software 1|2

Al enfatizar en principios de seguridad, los desarrolladores que se guían hoy solo por los principios de DevOps pueden aprender más sobre lo que están desarrollando y cómo pueden ser explotados por otros. En lugar de solo seguir ciegamente las prácticas de seguridad requeridas y los controles de seguridad identificados, los desarrolladores pueden aprender cómo pensar en hacer que sus aplicaciones sean seguras y como resultado, pueden derivar sus propias formas creativas de resolver problemas de seguridad como parte de la comprensión de los desafíos asociados con el **desarrollo de software seguro**.

En lugar de reaccionar a cada nuevo ataque recién cuando este aparece, el software seguro debe centrarse proactivamente en sobrevivir. Esto es posible proporcionado software con una superficie de ataque reducida que sea rápida de implementar y restaurar.

“En otras palabras, los desarrolladores deberían preocuparse menos por ser pirateados y más por prevenir ataques predecibles y recuperarse rápidamente de incidentes cibernéticos como parte de sus actividades de desarrollo”

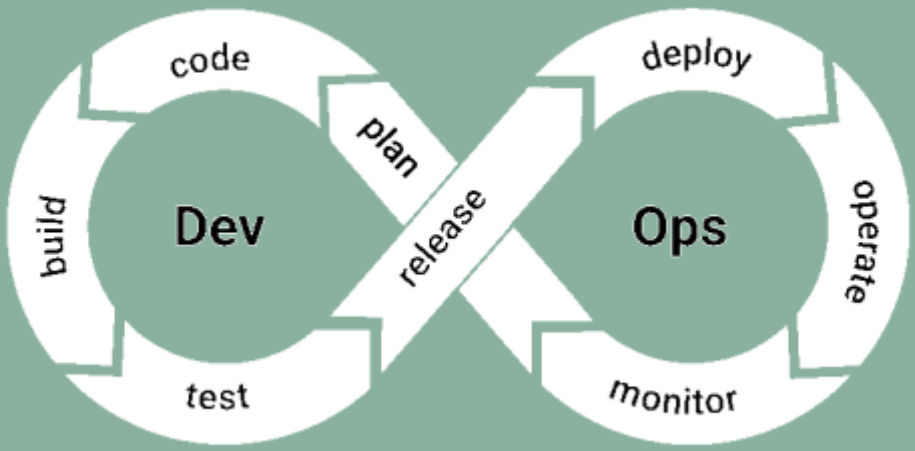
Imágenes creadas por dooder

Introducción a DevOps seguro

Un enfoque proactivo en la seguridad del software 2|2

En el pasado, la **seguridad del software** se centraba en la naturaleza y el origen de los ataques, así como en las medidas para prevenirlos. Sin embargo, dado que la mayoría de los ataques, especialmente los ataques sofisticados, no pueden anticiparse, las correcciones activaban solo a medida que se descubrían nuevos ataques.

La incapacidad de anticipar los ataques es la razón por la que surgen parches en respuesta a las nuevas **vulnerabilidades de día cero**. Los desarrolladores de DevOps seguro prefieren que su software absorba los ataques y continúen funcionando. En otras palabras, debe **doblarse pero no romperse,** y para ello se requiere un cambio de mentalidad en la forma de enfrentar los ataques. Asegurar un ciclo de vida seguro requiere que el equipo de desarrollo se centre en la integración continua, la Infraestructura como código, la implementación continua y la plataforma automatizada de desarrollo integrado.



Introducción a DevOps seguro

Aplicación adecuada de los principios de DevOps al desarrollo

Los conceptos en desarrollo de DevOps incluyen una serie de mejores prácticas que pueden aplicarse para aumentar la seguridad de las aplicaciones desarrolladas. Dentro de estas mejores prácticas se incluyen las siguientes

- **Agrega técnicas de testeos de seguridad automatizadas**, como fuzztest y test de penetración de software, al SDLC ciclo de vida del desarrollo de software o al ciclo de integración del sistema
- **Estandarización del ciclo de integración** para reducir la introducción de fallas.
- **Introducción de problemas y limitaciones de seguridad** a los equipos de desarrollo de software y sistemas al inicio de los proyectos, en lugar de aplicarlos después del hecho

La aplicación de principios de DevOps puede tener un gran impacto en la creación de un entorno que sea resistente y seguro. Profundizaremos mas adelante…

Imágenes creadas por dooder

Mejor seguridad con DevOps

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Cuando se trata de servicios de tecnología de la información orientada a clientes, las organizaciones tradicionales tienden a **favorecer la estabilidad sobre el cambio.** Según una encuesta de Netcraft realizada en marzo del 2017, había 185 millones de sitios web alojados en Windows 2003, un sistema operativo que ha estado fuera de soporte desde julio de 2015.

Muchos de estos servidores todavía se están ejecutando debido al lema **"si no está roto, no lo arregles".** Si bien la reducción del software y la rotación del sistema parece ser la mejor forma de promover la estabilidad, puede dañar la seguridad y la estabilidad de la aplicación.

Cuando el producto principal de una organización no es un software, el tiempo de actividad del sistema es importante para poder vender el producto principal. Tomaremos el ejemplo de **Equifax** y lo desarrollaremos. Asi, si **Equifax** no puede procesar las solicitudes de informes de crédito, perdería una importante fuente de ingresos, que es similar a una línea interminable de clientes detrás de una caja registradora rota.

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Desafortunadamente, en un intento por reducir la deserción de clientes, las organizaciones como **Equifax** continúan ejecutando sistemas operativos de servidor no compatibles, frameworks web obsoletos o servidores de aplicaciones no parcheados, básicamente porque aun parecen funcionar correctamente. Irónicamente, **Equifax** se quemó por un defecto en el framework web de Apache Struts, que podría haberse evitado si la aplicación en cuestión hubiese tenido un pipeline de DevOps automatizado. En muchos casos, estas situaciones se derivan de la falta de recursos que se pueden dedicar a actualizaciones de aplicaciones, migraciones de servidores o testeos manuales.

Aunque la mentalidad de **"si no está roto, no lo arregle**s" no se puede cambiar de la noche a la mañana, los beneficios del **software automatizado y los cambios en el sistema** se pueden evidenciar aunque las prácticas de DevOps se adopten lentamente como soporte de proyectos o aplicaciones.

Dado que la estabilidad y el cambio son métricas que pueden medirse, otros proyectos y equipos de aplicación estarán interesados en reducir el riesgo, lo que ayudará a difundir rápidamente la cultura de DevOps en toda la organización.

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Infraestructura como código 1/4

Aunque la configuración y la implementación varían de una organización a otra, un ingeniero de sistemas generalmente configura el sistema operativo del servidor base, dejando la configuración y la implementación de la aplicación al equipo de la aplicación. Cuando se debe mover una aplicación a un sistema operativo compatible, el equipo del servidor crea un nuevo servidor y lo entrega al equipo de la aplicación. El equipo de la aplicación configurará el servidor utilizando una lista de verificación o por prueba y error. Este proceso suele estar plagado de errores de configuración que se descubren mediante testeos manuales o, peor aún, en producción.

Por ejemplo, los errores pueden ocurrir debido a las diferencias en las imágenes del sistema operativo que son entregadas sin explicar, a los equipos de aplicación, los escenarios de caso extremo, nuevas versiones de software sin documentación adecuada de cambios o pasos que son omitidos de la lista de verificación de configuración.

Una vez finalizadas las pruebas y corregidos los errores, el entorno de testing se clona para el uso de producción o se mueve directamente a la producción. Estas arduas tareas deben completarse para cada migración del servidor.

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Infraestructura como código 2/4

DevOps puede mejorar este proceso al automatizar la configuración del servidor y la aplicación. DevOps también requiere que los sistemas y los equipos de aplicaciones trabajen juntos, ya que la aplicación en ejecución es el producto final, no solo un servidor y por separado la aplicación.

Para mantener entornos consistentes, se puede usar una práctica de DevOps, conocida como Infraestructura como Código (IaC). El concepto de IaC es simple; La configuración de su servidor se almacena en un repositorio de código fuente con la fuente de la aplicación. Las herramientas de automatización, como **Ansible, Chef o Puppet,** tomarán el artefacto de configuración (normalmente escrito en un lenguaje simples o scripting, como YAML o Ruby) y aplicar cada tarea al sistema deseado.

Estas herramientas son fáciles de aprender y pueden automatizar virtualmente cualquier tarea.

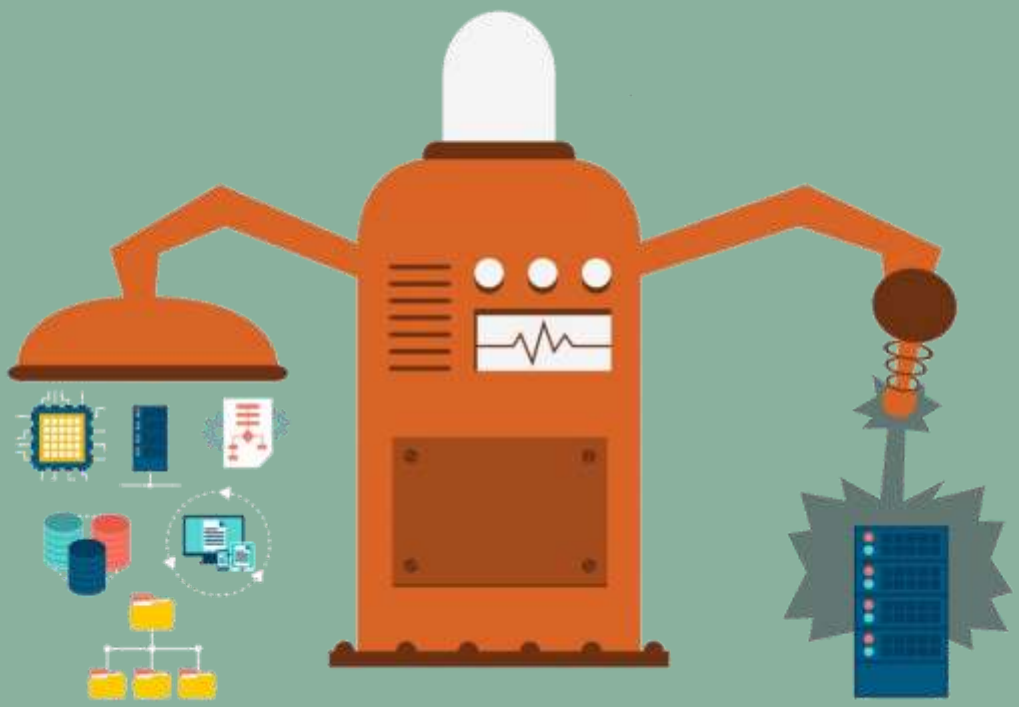
Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Infraestructura como código 3/4

La herramienta de configuración de su elección puede configurar las cuentas de usuario de la aplicación, instalar paquetes para satisfacer las dependencias, ajustar el entorno del sistema, mantener la configuración del servidor web, crear carpetas o configurar permisos de carpetas y archivos.

Además de la configuración, las herramientas de automatización pueden cerrar la brecha entre la implementación manual de la aplicación y un pipeline de entrega continua completo. Estas herramientas también pueden aplicar y revertir automáticamente cualquier cambio manual ejecutando las verificaciones de validación programadas del sistema; sin embargo, este es un proceso más avanzado y requiere alguna configuración adicional.



Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Infraestructura como código 4/4

Al automatizar, aunque solo sea, una pequeña parte de la configuración de su entorno, usted aumenta las posibilidades de una migración exitosa; Además, la misma IaC se puede utilizar también para migrar su aplicación a la nube si se desea en el futuro.

Además de la configuración del entorno repetible, IaC también le permite realizar un seguimiento de los cambios del entorno. Dado que IaC se almacena en un repositorio de código fuente, automáticamente tendrá un historial de todos los cambios de entorno, dependencias y versiones de dependencia.

Por ejemplo, si estuviera usando Apache Struts, podría identificar rápidamente una versión insegura del framework.

También sabría cuándo se introdujo un riesgo de seguridad a través de una dependencia de terceros.

Esta información ayudará a proporcionar una línea de tiempo si se identifican las vulnerabilidades de seguridad de su Aplicación, identificando el que, el cuando y el donde se produjo el cambio

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Testeos automatizados 1/2

Para garantizar la compatibilidad y la estabilidad, todas las migraciones y los procesos de actualización deben tener un plan de regresión.

Desafortunadamente, en base a nuestra experiencia vemos que muchas aplicaciones carecen de un plan de pruebas o de un conocimiento previo de la funcionalidad empresarial.

Estas migraciones o actualizaciones requieren un nuevo plan de pruebas y **numerosas horas de trabajo**, algunos podrían interpretar como tiempo perdido. La mano de obra adicional no siempre está disponible, por lo que cualquier actualización, parche de seguridad o migración se coloca en un Backlog, a veces indefinidamente.

Algunas organizaciones consideran que el desarrollo de pruebas automatizadas es demasiado costoso de crear. Si bien las pruebas automatizadas pueden tomar una cantidad significativa de tiempo y esfuerzo para crear, pero podemos asegurar que el beneficio supera ese costo.

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Testeos automatizados 2/2

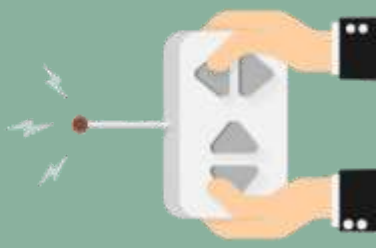
Por ejemplo, inicialmente se puede crear una pequeña serie de testeos de extremo a extremo, y se pueden agregar más testeos a medida que surjan los problemas. Este conjunto de pruebas se puede ejecutar antes y después de cualquier migración o actualización. El conjunto de pruebas identificará cualquier problema conocido, ahorrando tiempo de testeos manuales. Los evaluadores manuales pueden centrarse en probar las funciones más complejas que no pueden automatizarse rápidamente.

Después de que la cobertura de la suite de pruebas se acerca al 100 por ciento de la aplicación, puede migrar, aplicar parches o implementar su aplicación de manera segura. Los testeos automatizados no deben limitarse a Los testeos específicos de la aplicación; Se pueden ejecutar numerosas pruebas de seguridad contra la aplicación para exponer cualquier vulnerabilidad. Las exploraciones podrían intentar **“cross-script scripting (XSS), SQL injection, server file system traversal u otras vulnerabilidades de seguridad”** para exponer las vulnerabilidades de la aplicación.

Además de las exploraciones de vulnerabilidad, el conjunto de pruebas también puede realizar análisis de código estático y fuzzing para encontrar fallas de seguridad más comunes.

Imágenes creadas por dooder

Mejorando la seguridad con DevOps



Automatización de despliegue 1/2

El despliegue de software en muchas organizaciones abarca desde copiar archivos a un recurso compartido hasta un pipeline de despliegue completamente automatizado. En la mayoría de los casos, la implementación es manual. Se realiza una copia de seguridad de la aplicación original, el nuevo código se copia en una ubicación en un servidor y, tal vez, se establecen algunos permisos de archivo o se reinicia un proceso de servidor. Si bien la implementación es relativamente simple para aplicaciones pequeñas, puede ser muy compleja si la aplicación toca muchos servidores. Sin embargo, cuanto más complejo es el flujo de trabajo, más fácil es experimentar una interrupción o error debido a un paso de implementación perdido.

La **Automatización de despliegue** es el primer paso para crear un pipeline de entrega continua (CD), que es otra práctica de DevOps.

Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Automatización de despliegue 2/2

El objetivo de la entrega continua (CD) es la capacidad de hacer un cambio de código, compilar y probar la aplicación, e implementar la nueva compilación en un entorno de producción o testing de aceptación del usuario. Sin embargo, las organizaciones que son nuevas en DevOps primero deben centrarse en crear un proceso de Automatización de despliegue .

La Automatización de despliegue limita la posibilidad de que el usuario cometa un error y evita cualquier duda vaga al implementar una aplicación. Dado que la automatización elimina el error del usuario del proceso de implementación, haciendo confiable la implementación de un parche de seguridad en su aplicación, corrección de errores o para un nuevo entorno. No habrá ninguna preocupación de que se pierda un paso en particular; Cada despliegue seguirá el mismo patrón que el anterior.

Asimismo, se reduce el tiempo de despliegue. En ciertos casos, la implementación puede reducirse de horas a minutos, dependiendo de la complejidad del sistema.

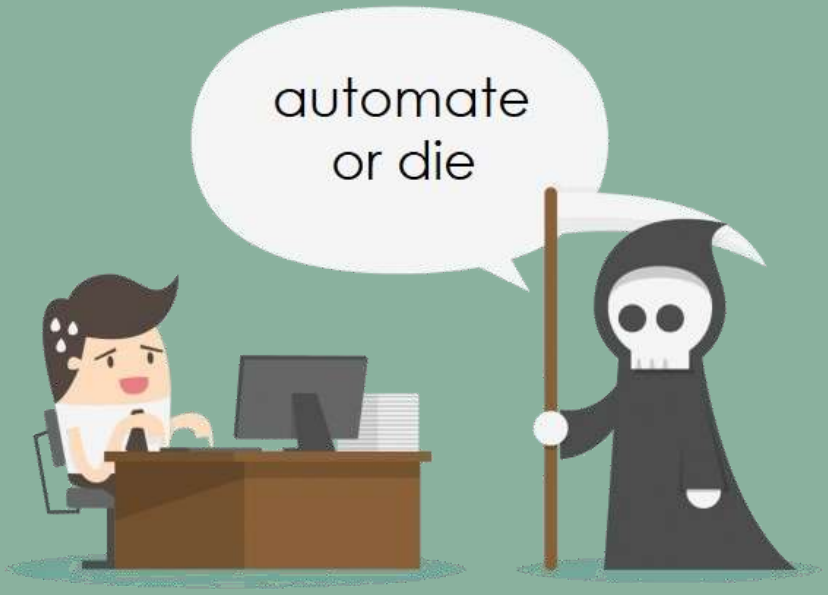
Imágenes creadas por dooder

Mejorando la seguridad con DevOps

Los procesos DevOps pueden reducir la mano de obra manual (no automatizada), reducir los riesgos al eliminar las diferencias de configuración de ambientes, al tiempo que aumenta la confianza de los socios de ingeniería y negocios. Aunque las mentalidades tradicionales son difíciles de cambiar, las prácticas de DevOps se pueden inyectar lentamente en aplicaciones y equipos de soporte.

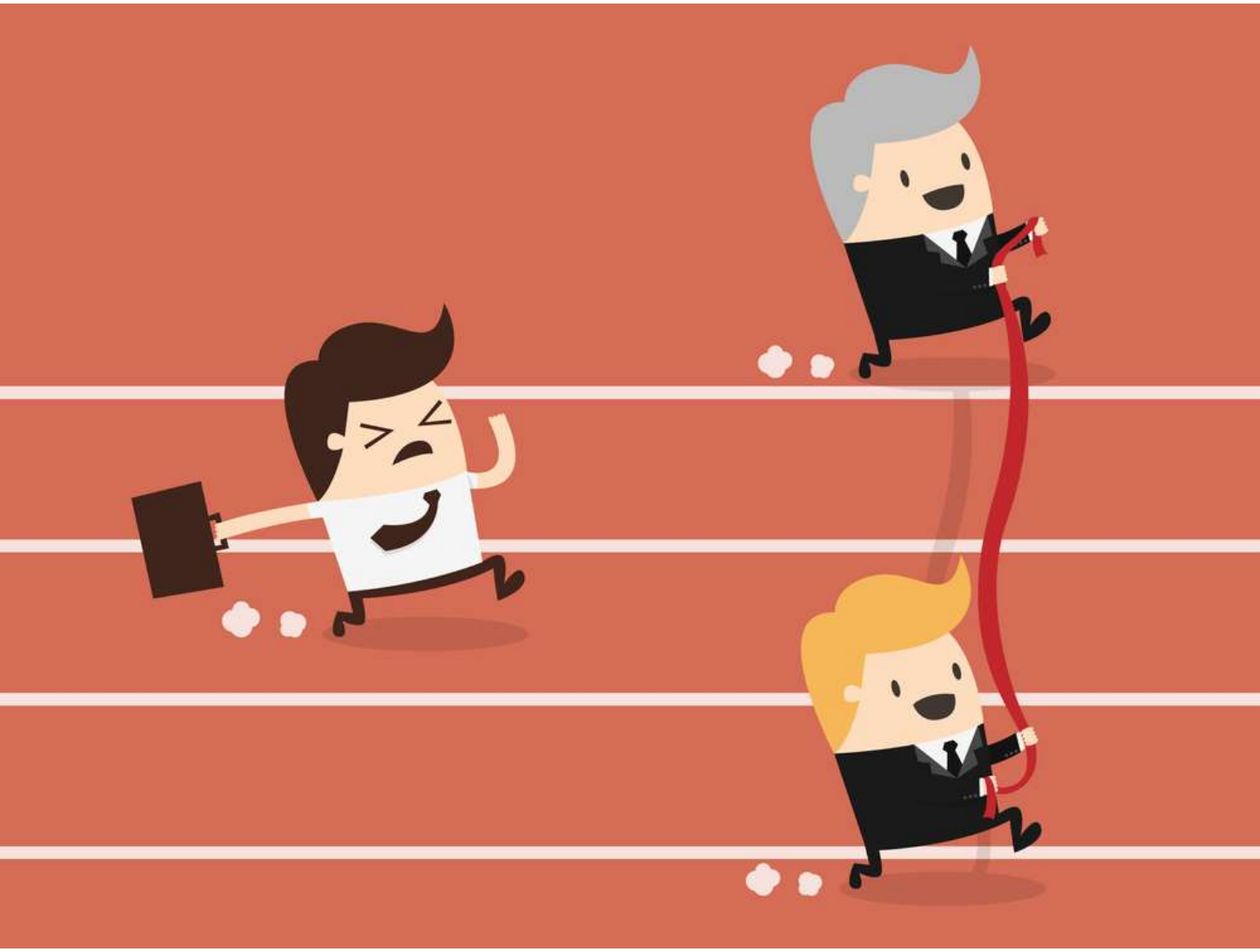
Una vez que se obtienen los beneficios, la integración de DevOps puede **cambiar la cultura** de una organización al reducir los riesgos y aumentar la colaboración entre equipos.

Pudimos ver algunas Practicas e ideas básicas de como DevOps nos proporciona soluciones para problemas comunes que afectan el SDLC ciclo de vida del desarrollo de software y sobre todo como ayuda desde la automatización a minimizar los errores y regresiones por temas de seguridad, sumando un detección temprana.



Imágenes creadas por dooder

Application Security Testing: Estado actual



Imágenes creadas por dooder

Application Security Testing: Estado actual

En los capítulos anteriores vimos una introducción a DevOps y algunos de los beneficios de seguridad relacionados con la adopción de la cultura en el SDLC ciclo de vida del desarrollo de software.

Donde quedo claro la necesidad de integrar la seguridad en los flujos de trabajo de integración continua y entrega continua (CI / CD), Pudimos entender que uno de los limitantes en el estado actual de madurez de muchas empresas con relación a DevSecOps es que los equipos de DevOps y SecOps continúan funcionando en diferentes silos.

Como era de esperar, la seguridad sigue estando rezagada en los entornos DevOps. Una encuesta reciente a 350 responsables de la toma de decisiones de TI realizada por la firma 451 Research mostró que la mitad de todos los equipos de DevOps todavía no han incorporado la seguridad de las aplicaciones en sus flujos de trabajo de CI / CD, a pesar de tener una gran conciencia de la necesidad de hacerlo.

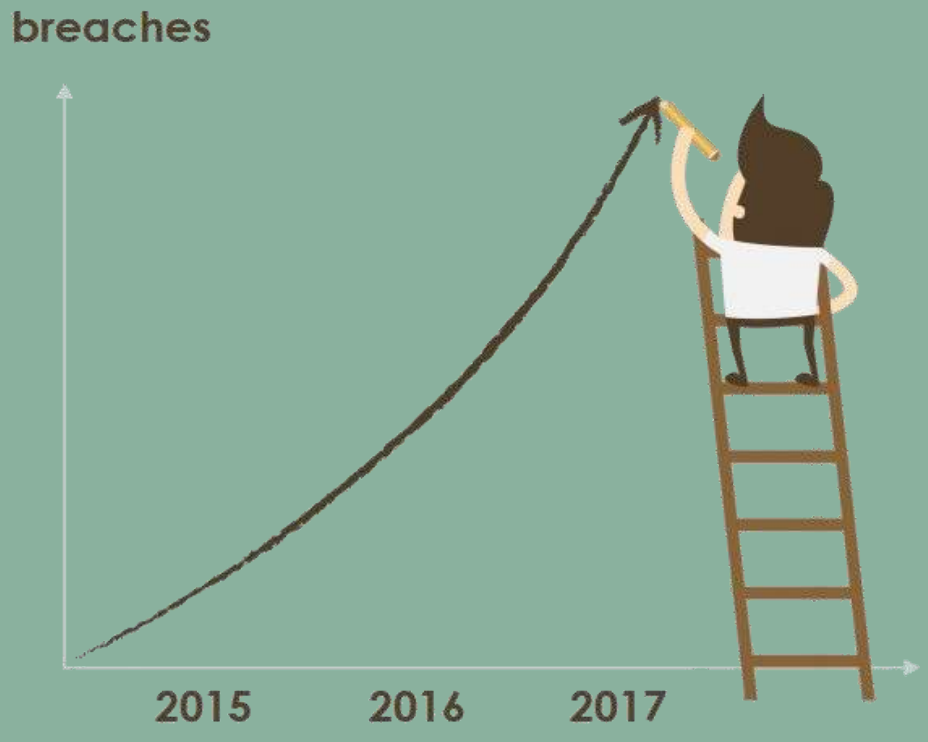
Imágenes creadas por dooder

Application Security Testing: Estado actual

Aunque los equipos de DevOps han comenzado a trabajar en proyectos cada vez más grandes y están lanzando software más rápido que antes, a menudo lo hacen sin una estrategia clara para integrar la seguridad en el proceso.

En otra encuesta a más de 2.050 profesionales de la comunidad de DevSecOps, el **72%** de los encuestados describió la función de seguridad como un "fastidio", y el 48% de los desarrolladores admitió que, aunque la seguridad era importante, no tenían tiempo suficiente para gastar en eso.

Como el mundo fue testigo de **brechas récord en 2017**, los lideres de equipos de TI empezaron a integrar y automatizar más prácticas de seguridad a lo largo del SDLC ciclo de vida del desarrollo de software para fortalecer mejor las aplicaciones y proteger sus datos.

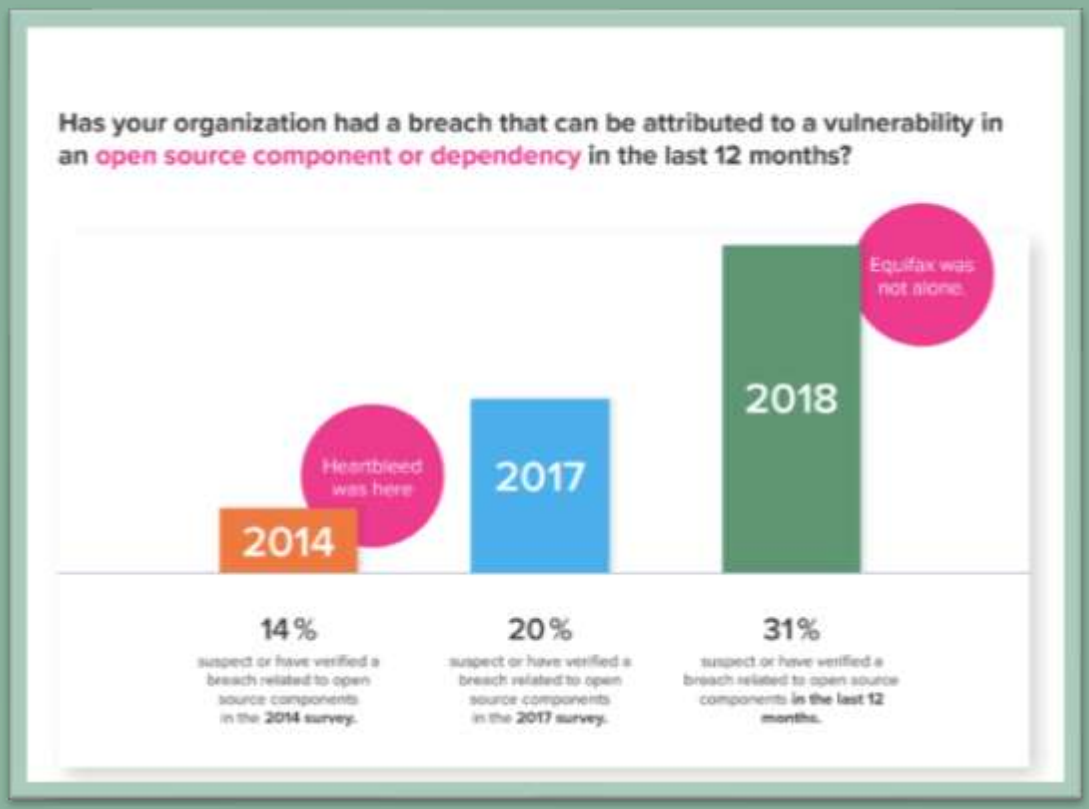


Imágenes creadas por dooder

Application Security Testing: Estado actual

Equifax no estaba sola: Han transcurrido varios meses desde que **Equifax** reveló públicamente una brecha importante en sus sistemas que surgió de un hack que se dirigió a componentes vulnerables de código abierto. Donde **Equifax** se encontró sola en el medio de la tormenta, pero los resultados de las ultimas encuestas cuentan una historia diferente. Equifax no estaba sola.

Algunos meses después del incidente de **Equifax** La investigación de **Sonartype** mostro que **las brechas fijadas a los componentes de software de código abierto aumentaron un 55% año traz año.** La encuesta de la comunidad DevSecOps 2018 de **Sonatype** informó que se registraron brechas en el 31% de las empresas representadas por los 2.076 profesionales de TI que participaron en la encuesta de este año.



Application Security Testing: Estado actual

Uno podría preguntarse si la persistencia de los titulares de las noticias de Equifax influyó en el aumento año tras año. Si bien no tenemos manera de demostrar esta teoría, podemos comparar los resultados de 2018 con otras respuestas de la misma encuesta del 2014 para una mejor perspectiva. Desde la encuesta de 2014 de Sonatype, las **brechas relacionadas con el software de código abierto aumentaron 121%.** Curiosamente, la encuesta de 2014 se llevó a cabo durante el mes de abril, cuando se anunció la notoria vulnerabilidad de Heartbleed y fue lo más importante para muchos de los encuestados.

La Encuesta de la Comunidad DevSecOps 2018 también arrojó noticias positivas. Los resultados año tras año apuntan a un aumento constante del 15% en la seguridad automatizada que se integra a lo largo del SDLC ciclo de vida del desarrollo de software.

Application Security Testing: Estado actual

La carrera por la seguridad automatizada.

En un momento en donde las noticias de violaciones de seguridad son persistentes, es alentador ver que se empezaron a realizan inversiones en la comunidad de DevOps para reducir el riesgo de entrada ilegal y robo de datos por parte de piratas informáticos.

El aumento de las inversiones no fue la única señal alentadora de los datos de la encuesta. Cuando se les pidió que calificaran su preparación para la ciberseguridad, los encuestados de las prácticas de DevOps se calificaron a sí mismos un 85% más altos que aquellos sin prácticas de DevOps.

Con la realización de mayores inversiones en la seguridad de las aplicaciones automatizadas a lo largo del SDLC ciclo de vida del desarrollo de software, las prácticas de "seguro por diseño" aumentaron la confianza de los desarrolladores y los equipos de DevOps.

Imágenes creadas por dooder

Application Security Testing: Estado actual

La automatización de las pruebas de seguridad es difícil de ignorar

En los últimos años, vemos que la preocupación de las empresas por buscar nuevos estándares, métodos, practicas y herramientas esta reflejando en un crecimiento de las iniciativas empresariales de DevSecOps que han incorporado con éxito prácticas de seguridad automatizadas, fortaleciendo su postura de ciberseguridad y preparando para las regulaciones gubernamentales en el horizonte. Estas prácticas de DevSecOps continúan madurando rápidamente y que, una vez automatizada las pruebas, la seguridad es difícil de ignorar.

Si bien algunos números no muestra la realidad de todas las empresas esperamos que les sirva como base y los alienten a comenzar nuevas conversaciones con sus colegas y el resto de la industria.

En el próximo capitulo empezaremos a conocer algunas de las AST (Application Security testing tools) Tipos, como, cuando usarlas y como seleccionar las que mas valor puede surmarle dependiendo el nivel de madurez que se encuentre su pipeline de DevOps.

Application Security testing tools: cuándo y cómo usarlas

Imágenes creadas por dooder

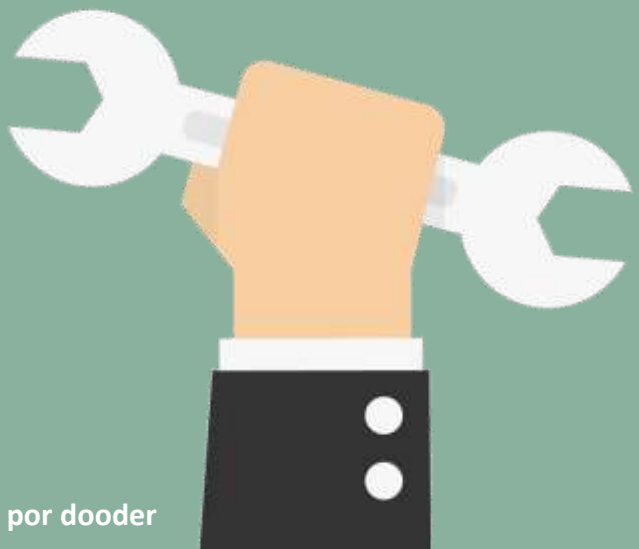
Application Security testing tools

Contexto

¿Por qué surge la necesidad de renovar o sumar nuevas técnicas, practicas y Herramientas de testing de seguridad en nuestro pipeline de desarrollo?

Los errores y las debilidades en el software son comunes: el 84 por ciento de las brechas de software explotan vulnerabilidades en la capa de aplicación donde entre un 70% a 80% de los ataques exitosos suceden por vulnerabilidades “conocidas”. La prevalencia de problemas relacionados con el software es una motivación clave para usar Application Security testing tools (AST). Con un número creciente de Application Security testing tools disponibles, puede ser confuso para los líderes, desarrolladores e ingenieros de tecnología de la información (TI) saber qué herramientas abordan qué problemas.

En este capitulo trataremos de describir y hablar sobre Application Security testing tools , vamos a categorizar los diferentes tipos de herramientas AST disponibles y a proporcionar orientación sobre cómo y cuándo usar cada clase de herramienta.



Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

La seguridad de la aplicación no es una simple opción binaria, ya sea que usted tiene seguridad o no. La seguridad de la aplicación es más bien una escala móvil donde proporcionar capas de seguridad adicionales ayuda a reducir el riesgo de un incidente, con suerte a un nivel aceptable de riesgo para la organización. Por lo tanto, Los testeos de seguridad de la aplicación reducen el riesgo en las aplicaciones, pero no pueden eliminarlo por completo. Sin embargo, se pueden tomar medidas para eliminar los riesgos que son más fáciles de eliminar y para fortalecer el software en uso.

La motivación principal para usar herramientas AST es que las revisiones del código manuales y los proyectos de testeos tradicionales llevan mucho tiempo, y las nuevas vulnerabilidades están siendo continuamente introducidas o descubiertas. En muchos dominios, existen directivas regulatorias y de cumplimiento que exigen el uso de herramientas de AST.

Además, y quizás lo más importante, los individuos y grupos que intentan comprometer los sistemas también usan herramientas, y los encargados de proteger esos sistemas deben seguir el ritmo de sus adversarios.

Imágenes creadas por dooder

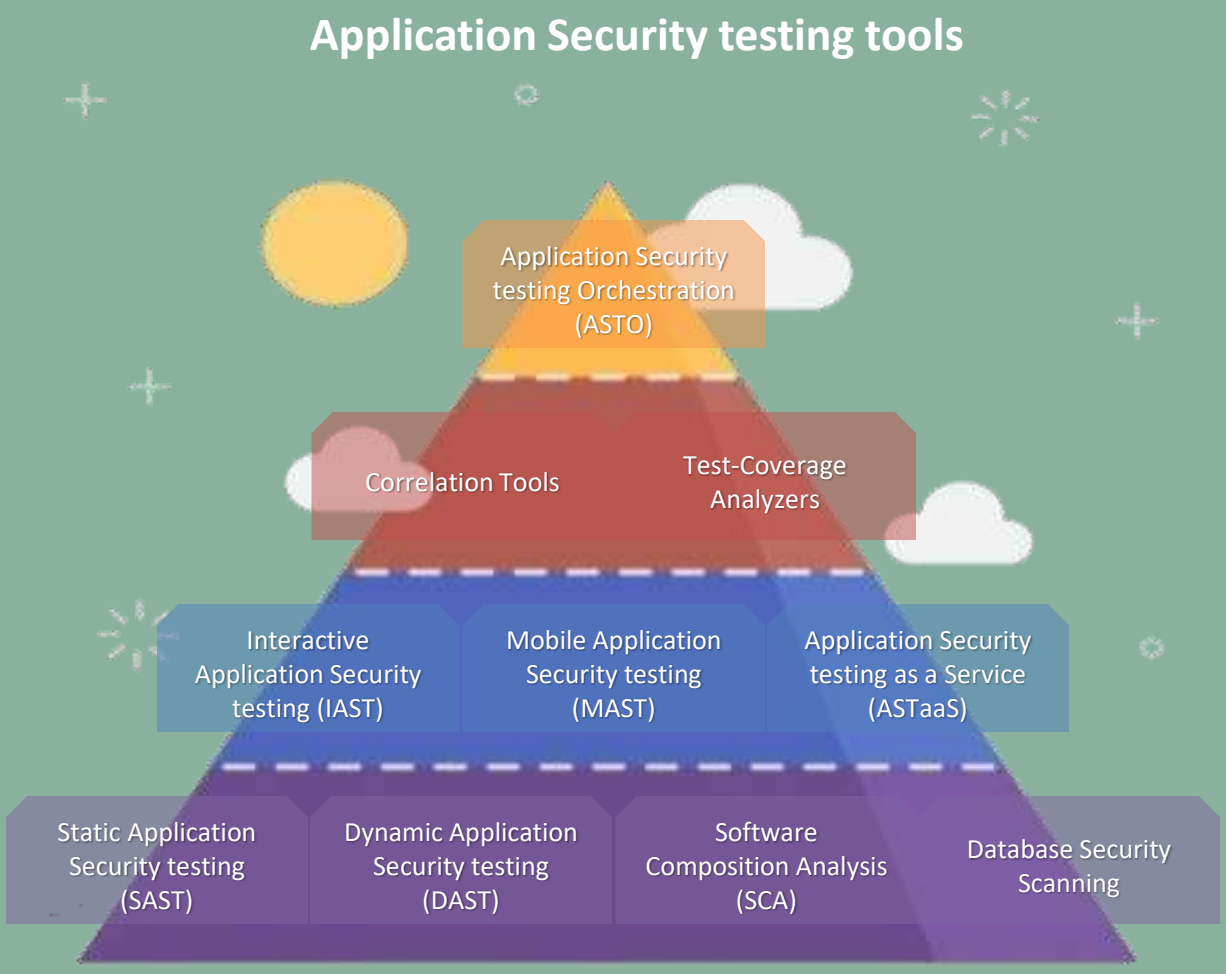
Application Security testing tools : cuándo y cómo usarlas

El uso de herramientas AST tiene muchos beneficios, aumenta la velocidad, la eficiencia y las rutas de cobertura para probar aplicaciones. Los testeos que realizan son repetibles y se escalan bien: una vez que se desarrolla un caso de prueba en una herramienta, se puede ejecutar contra muchas líneas de código con poco costo adicional.

Las herramientas de AST son efectivas para encontrar vulnerabilidades, problemas y debilidades conocidas, y permiten a los usuarios clasificar sus hallazgos. También se pueden usar en el flujo de trabajo de remediación, particularmente en la verificación, y se pueden usar para correlacionar e identificar tendencias y patrones

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas



Este gráfico representa clases o categorías de Application Security testing tools . Los límites se difuminan a veces, ya que determinados productos podrían realizar elementos de varias categorías, pero son aproximadamente las clases de herramientas dentro de este dominio. Hay una jerarquía áspera en que las herramientas en la parte inferior de la pirámide son fundacional y como la competencia se gana con ellos, las organizaciones podrían buscar utilizar algunos de los métodos más progresivos más altos en la pirámide

Imágenes creadas por dooder

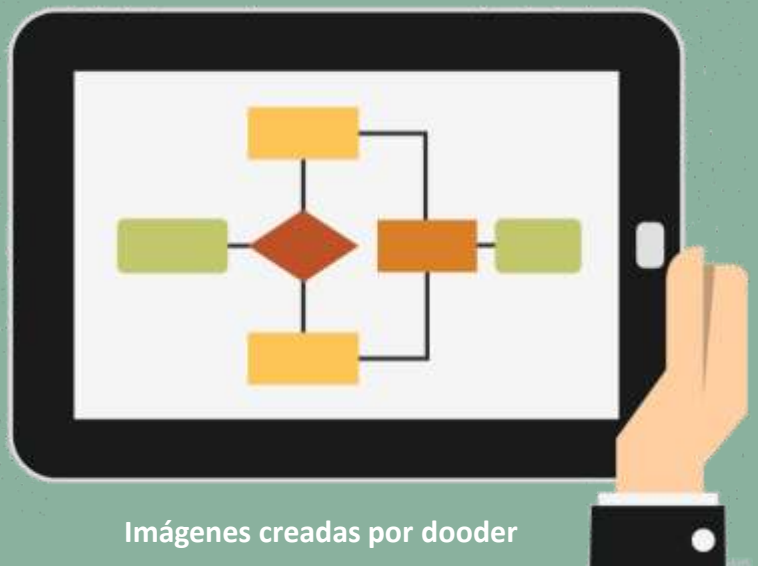
Application Security testing tools : cuándo y cómo usarlas

Static Application Security Testing (SAST)

Las herramientas de la categoría SAST se conocen como herramientas de testeos White hat o White box blanca, donde el evaluador conoce información sobre el sistema o software que se está probando, incluyendo un diagrama de arquitectura, acceso al código fuente, etc.

Las Herramientas SAST examinan el código fuente (en reposo o estático cono se suele nombrar) para detectar y reportar debilidades que podrían convertirse en vulnerabilidades seguridad.

Los analizadores de código fuente podrían ejecutarse en código no compilado para comprobar defectos como errores numéricos, validación de entrada, condiciones, path traversals, punteros y referencias, etc. Los analizadores de código binario y de byte hacen lo mismo en el código compilado. Algunas herramientas se ejecutan únicamente en código fuente, algunas solo en código compilado y otras en ambas.



Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Dynamic Application Security Testing (DAST)

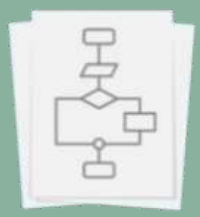
A diferencia de las herramientas SAST, las herramientas DAST se podrían denominar como herramientas de testeos Black-Hat o Black-Box, donde el probador no tiene conocimiento previo del sistema. Detectan condiciones que indiquen una vulnerabilidad de seguridad en una aplicación en su estado de ejecución.

Las herramientas DAST se ejecutan en código operativo para detectar problemas con interfaces, solicitudes, respuestas, scripts (es decir, Javascript), inyección de datos, sesiones, autenticación y más.

Las herramientas DAST emplean técnicas de Fuzzing: lanzar casos de testeos conocidos, no válidos e inesperados en una aplicación, a menudo en gran volumen.

Imágenes creadas por dooder

Application Security testing tools: cuándo y cómo usarlas



Software Composition Analysis (SCA)

Los procesos de gobernanza de software que dependen de la inspección manual son propensos al fracaso. Las herramientas SCA examinan el software para determinar los orígenes de todos los componentes y librerias 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 suelen detectar vulnerabilidades para los componentes desarrollados a medida por la propia empresa. Funcionan comparando los módulos conocidos que se encuentran en el código con una lista de vulnerabilidades conocidas. Las herramientas SCA encuentran componentes que tienen vulnerabilidades conocidas y documentadas y, a menudo, aconsejan si los componentes están desactualizados o tienen parches disponibles. Para hacer esta comparación, casi todas las herramientas SCA utilizan la Base de datos de vulnerabilidad nacional del NIST scannea y exposiciones comunes (CVEs) como fuente de vulnerabilidades conocidas. Muchos productos SCA comerciales también utilizan el VulnDB base de datos de vulnerabilidad comercial como fuente, así como algunas otras fuentes públicas y propietarias.

Las herramientas SCA podrían ejecutarse en código fuente, código de byte, código binario o alguna combinación.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Database Security Scanning

El gusano SQL Slammer del 2003 explotó una vulnerabilidad conocida en un sistema de administración de bases de datos que tuvo un parche lanzado más de un año antes del ataque. Aunque las bases de datos no siempre se consideran parte de una aplicación, los desarrolladores de aplicaciones a menudo dependen en gran medida de la base de datos, y las aplicaciones a menudo pueden afectar mucho a las bases de datos. Las herramientas de escaneo de seguridad de la base de datos verifican parches y versiones actualizadas, contraseñas débiles, errores de configuración, lista de control de acceso (ACL) y más.

Algunas herramientas pueden extraer registros que buscan patrones o acciones irregulares, como acciones administrativas excesivas.

Los escáneres de bases de datos generalmente se ejecutan en los datos estáticos que están en reposo mientras el sistema de administración de bases de datos está en funcionamiento. Algunos escáneres pueden monitorear los datos que están en tránsito

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Interactive Application Security Testing (IAST)

Los enfoques híbridos han estado disponibles durante mucho tiempo, pero más recientemente se han categorizado y discutido usando el término IAST. Las herramientas IAST utilizan una combinación de técnicas de análisis estático y dinámico. Pueden probar si las vulnerabilidades conocidas en el código son realmente explotables en la aplicación en ejecución.

Las herramientas IAST utilizan el conocimiento del flujo de la aplicación y el flujo de datos para crear escenarios avanzados de ataque y utilizan los resultados del análisis dinámico de forma recursiva: a medida que se realiza un análisis dinámico, la herramienta aprenderá cosas sobre la aplicación en función de cómo responde a los casos de prueba. Algunas herramientas utilizarán este conocimiento para crear casos de prueba adicionales, que luego podrían generar más conocimiento para más casos de prueba y así sucesivamente. Las herramientas IAST son expertas en reducir el número de falsos positivos, y funcionan bien en entornos Agile y DevOps, donde las herramientas tradicionales independientes DAST y SAST pueden requerir demasiado tiempo para el SDLC ciclo de vida del desarrollo de software.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Mobile Application Security Testing (MAST)

El Open Web Application Security Project (OWASP) enumeró los 10 principales riesgos móviles en 2016 y 2017 tales como:

- M1: Improper Platform Usage
- M2: Insecure Data Storage
- M3: Insecure Communication
- M4: Insecure Authentication
- M5: Insufficient Cryptography
- M6: Insecure Authorization
- M7: Client Code Quality
- M8: Code Tampering
- M9: Reverse Engineering
- M10: Extraneous Functionality

Las herramientas MAST son una combinación de análisis estático, dinámico y forense. Realizan algunas de las mismas funciones que los analizadores estáticos y dinámicos tradicionales, pero permiten que el código móvil se ejecute también en muchos de esos analizadores. Las herramientas MAST tienen características especializadas que se centran en problemas específicos de las aplicaciones móviles, como el jailbreak o el rooting del dispositivo, las conexiones WI-FI falsas, el manejo y la validación de certificados, la prevención de fugas de datos y más.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Application Security Testing as a Service (ASTaaS)

Como su nombre lo indica, con ASTaaS, le paga a alguien para que realice pruebas de seguridad en su aplicación. El servicio generalmente será una combinación de análisis estático y dinámico, pruebas de penetración, pruebas de interfaces de programación de aplicaciones (API), evaluaciones de riesgo y más. ASTaaS puede usarse en aplicaciones tradicionales, especialmente aplicaciones móviles y web.

El impulso para el uso de ASTaaS proviene del uso de aplicaciones en la nube ,donde los recursos para las pruebas son más fáciles de reunir. Se proyecta que el gasto mundial en computación en nube pública de $67B en 2015 a $162B en 2020.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Correlation Tools

Tratar con falsos positivos es un gran problema en las pruebas de seguridad de aplicaciones. Las herramientas de correlación pueden ayudar a reducir parte del ruido al proporcionar un depósito central para los hallazgos de otras herramientas AST.

Las diferentes herramientas de AST tendrán diferentes hallazgos, por lo que las herramientas de correlación analizan los resultados de diferentes herramientas de AST y ayudan con la validación y la priorización de los hallazgos, incluidos los flujos de trabajo de remediación. Mientras que algunas herramientas de correlación incluyen escáneres de códigos, son útiles principalmente para importar resultados de otras herramientas.

Imágenes creadas por dooder

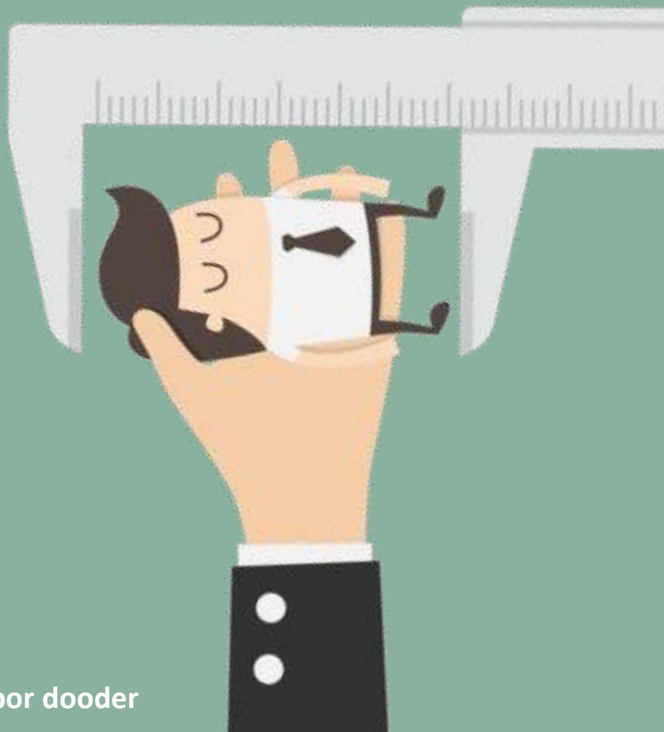
Application Security testing tools : cuándo y cómo usarlas

Test-Coverage Analyzers

Los Test-coverage analyzers miden cuánto del código total del programa ha sido analizado. Los resultados se pueden presentar en términos statement coverage (porcentaje de líneas de código probado) o branch coverage (porcentaje de rutas disponibles probadas).

Para aplicaciones grandes, los niveles de cobertura aceptables pueden determinarse por adelantado y luego compararse con los resultados producidos por los analizadores de cobertura de prueba para acelerar el proceso de prueba y liberación. Estas herramientas también pueden detectar si determinadas líneas de código o ramas de lógica no pueden alcanzarse durante la ejecución del programa, lo que es ineficiente y una posible preocupación de seguridad.

Algunas herramientas SAST incorporan esta funcionalidad en sus productos, pero también existen productos independientes



Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Application Security Testing Orchestration (ASTO)

ASTO integra herramientas de seguridad a lo largo de un ciclo de vida de desarrollo de software (SDLC).

Si bien el término ASTO fue recientemente acuñado por Gartner ya que este es un campo emergente, hay herramientas que ya han estado haciendo ASTO, principalmente aquellas creadas por proveedores de herramientas de correlación. La idea de ASTO es tener una administración centralizada y coordinada y la generación de informes de todas las diferentes herramientas de AST que se ejecutan en un ecosistema.

Todavía es demasiado pronto para saber si el término y las líneas de productos durarán, pero a medida que las pruebas automáticas se vuelven más omnipresentes, ASTO satisface una necesidad.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

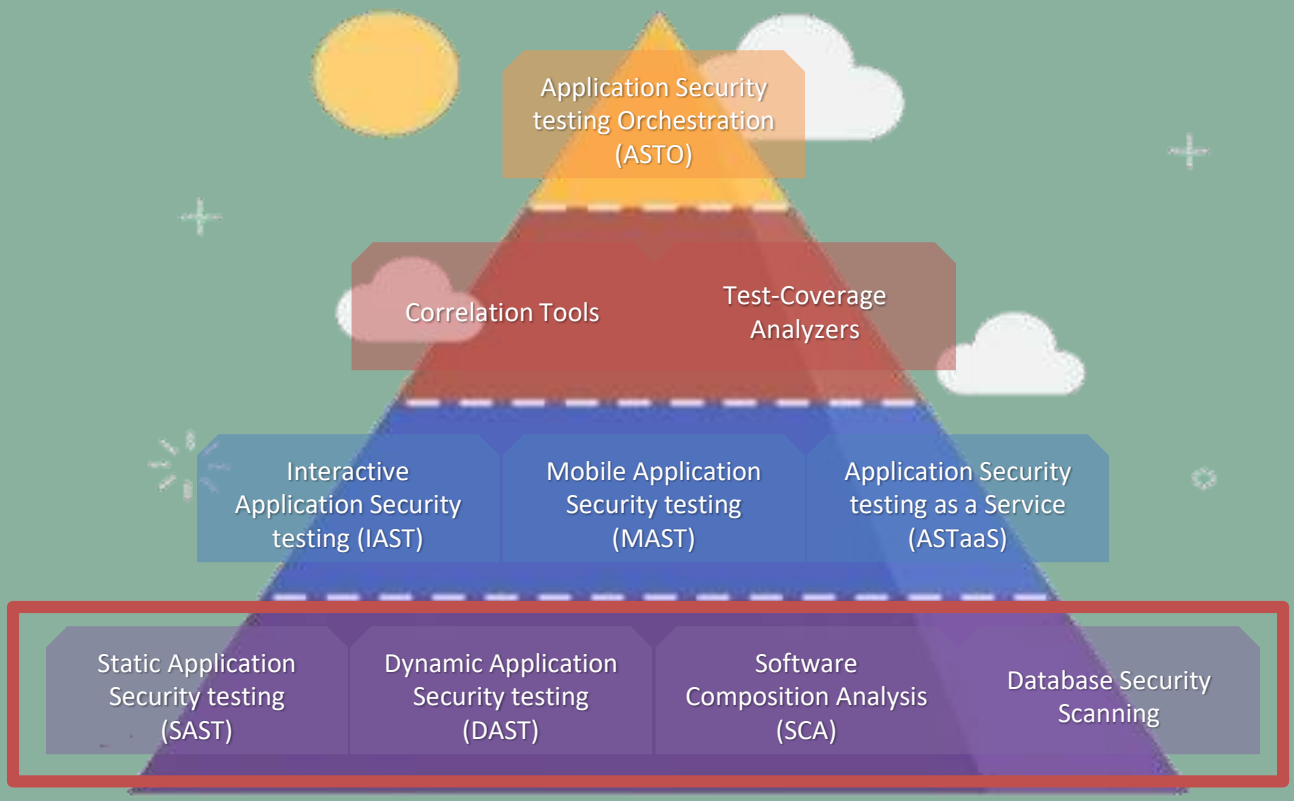
Hay muchos factores a considerar cuando se selecciona entre estos diferentes tipos de herramientas AST. Si se está preguntando cómo comenzar, la decisión más importante que tomará es comenzar a usar las herramientas.

Hay factores que lo ayudarán a decidir qué tipo de herramientas AST usar y determinar qué productos usar en una clase de herramientas AST. Es importante tener en cuenta, sin embargo, que ninguna herramienta única resolverá todos los problemas. Como se dijo anteriormente, la seguridad no es binaria; El objetivo es reducir el riesgo y la exposición. Antes de mirar productos específicos de AST, el primer paso es determinar qué tipo de herramienta AST es adecuada para su aplicación. Hasta que las pruebas de software de su aplicación crezcan en sofisticación, la mayoría de las pruebas se realizarán utilizando herramientas AST desde la base de la pirámide, que se muestra resaltada en la siguiente figura.

Estas son las herramientas AST más maduras que abordan las debilidades más comunes.

Imágenes creadas por dooder

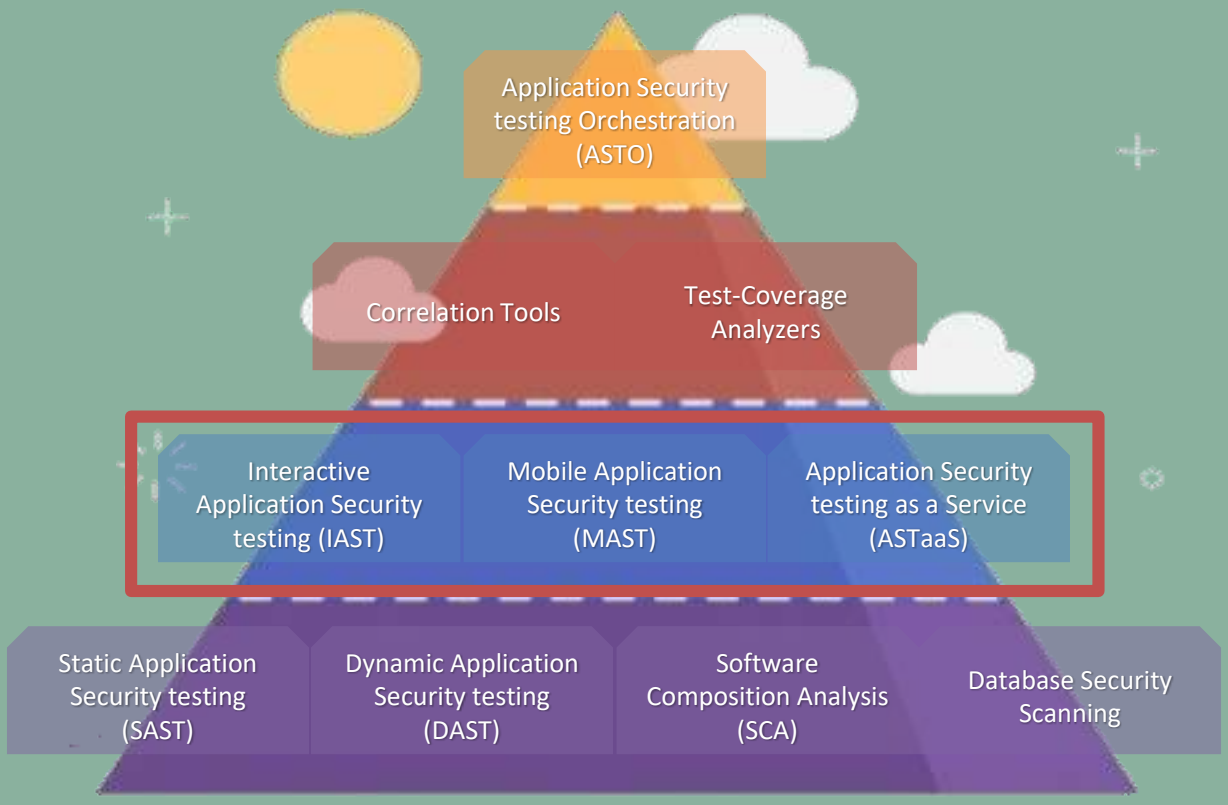
Application Security testing tools : cuándo y cómo usarlas



Una vez que adquiera la habilidad y la experiencia, puede considerar agregar algunos de los enfoques de segundo nivel que se muestran en la pagina a continuación. Por ejemplo, muchas herramientas de prueba para plataformas móviles proporcionan marcos para que usted escriba scripts personalizados para prueba. Tener algo de experiencia con las herramientas DAST tradicionales le permitirá escribir mejores scripts de prueba. Del mismo modo, si tiene experiencia con todas las clases de herramientas en la base de la pirámide, estará en mejor posición para negociar los términos y las características de un contrato ASTaaS.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas



La decisión de emplear herramientas en los tres recuadros superiores de la pirámide está dictada tanto por cuestiones de gestión y recursos como por consideraciones técnicas.
Si solo puede implementar una herramienta AST en las próximas paginas dejaremos algunas pautas para elegir el tipo de herramienta:

Imágenes creadas por dooder

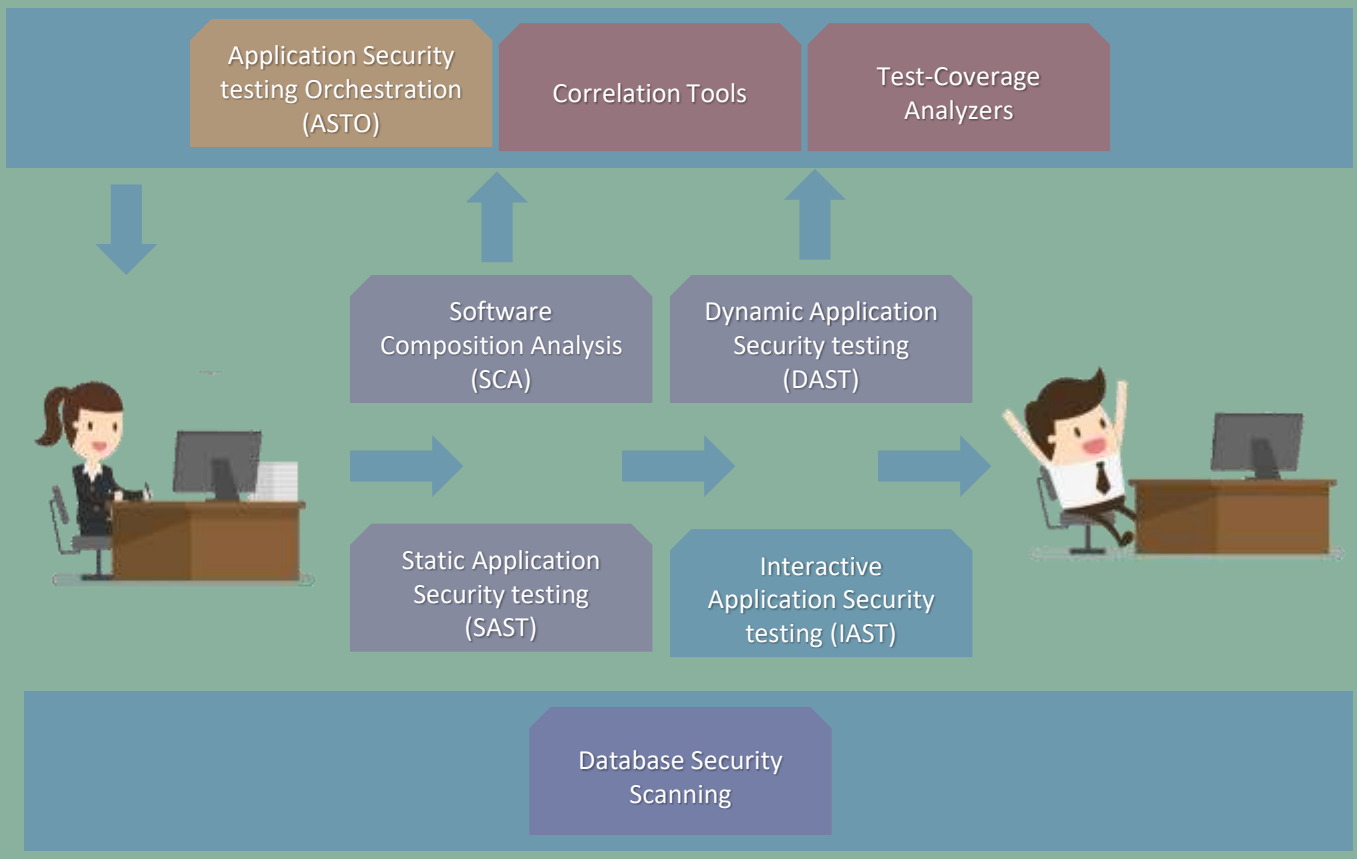
Application Security testing tools : cuándo y cómo usarlas

- Si la aplicación está escrita internamente o si tiene acceso al código fuente, un buen punto de partida es ejecutar una herramienta de seguridad de aplicación estática (SAST) y verificar los problemas de codificación y el cumplimiento de los estándares de codificación. De hecho, SAST es el punto de partida más común para el análisis de código inicial.
- Si la aplicación no está escrita de forma interna o, de lo contrario, no tiene acceso al código fuente, la prueba dinámica de seguridad de la aplicación (DAST) es la mejor opción.
- Ya sea que tenga acceso al código fuente o no, si se sabe que se utilizan muchos componentes de terceros y de fuente abierta en la aplicación, entonces las herramientas de análisis de origen / análisis de composición de software (SCA) son la mejor opción. Idealmente, las herramientas SCA se ejecutan junto con las herramientas SAST y / o DAST, pero si los recursos solo permiten la implementación de una herramienta, las herramientas SCA son imperativas para las aplicaciones con componentes de terceros, ya que comprobarán las vulnerabilidades que ya son ampliamente conocidas.

Imágenes creadas por dooder

Application Security testing tools : cuándo y cómo usarlas

Application Security Testing Tools Reference Model CI/CD Development Project



Estas herramientas también tienen muchos mandos y botones para calibrar la salida, pero lleva tiempo establecerlos en un nivel deseable. Tanto los falsos positivos como los falsos negativos pueden ser molestos si las herramientas no se configuran correctamente.

En el próximo capitulo, abordaremos algunos de estos factores de decisión con mayor detalle y presentaremos una guía en forma de listas que los responsables de las pruebas de seguridad de aplicaciones pueden usar fácilmente.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

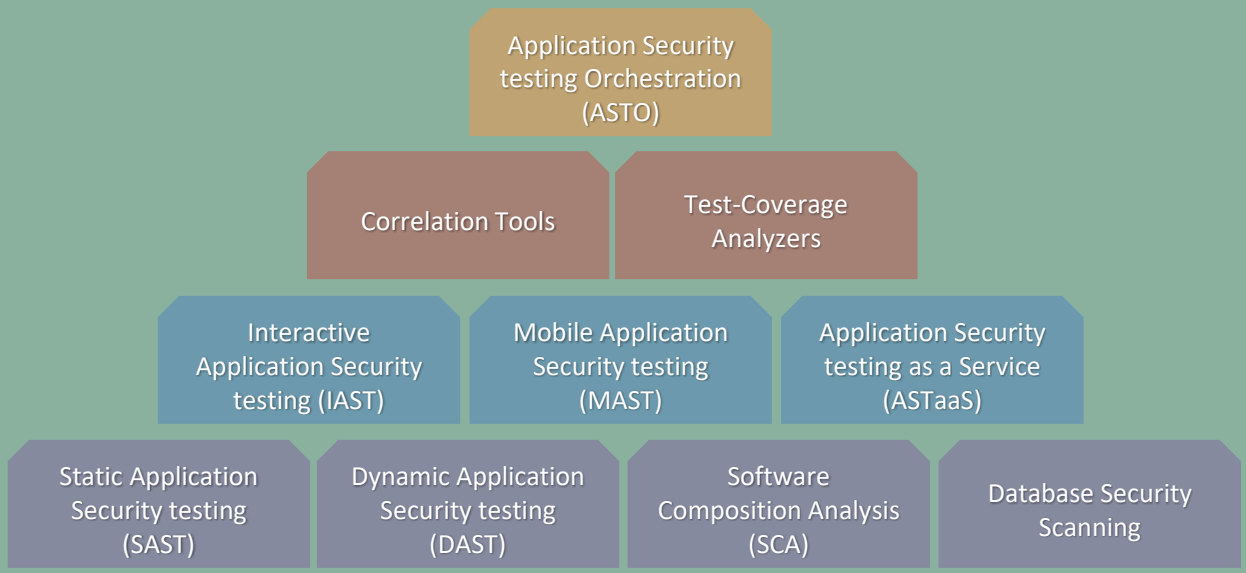
Antes vimos los diferentes tipos de Application Security testing tools (AST) y vimos cuándo y cómo usarlas.
Ahora profundizaremos en los factores de toma de decisiones que se deben tener en cuenta al seleccionar una herramienta AST y presentar una guía en forma de listas a las que los responsables de las pruebas de seguridad de aplicaciones pueden hacer referencia fácilmente como listas de verificación.
Antes de mirar productos específicos de AST, deben determinar qué clase de herramientas de AST son adecuadas para su aplicación. Para lograr esta tarea, hay muchos factores que lo ayudarán a decidir qué clase o tipo de herramientas AST utilizar.
A continuación presentaremos algunos de los factores más destacados junto con una guía de apoyo para tomar una decisión. Si bien cada factor se presenta y discute individualmente, todos los factores deben considerarse colectivamente en relación unos con otros durante el proceso de toma de decisiones.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

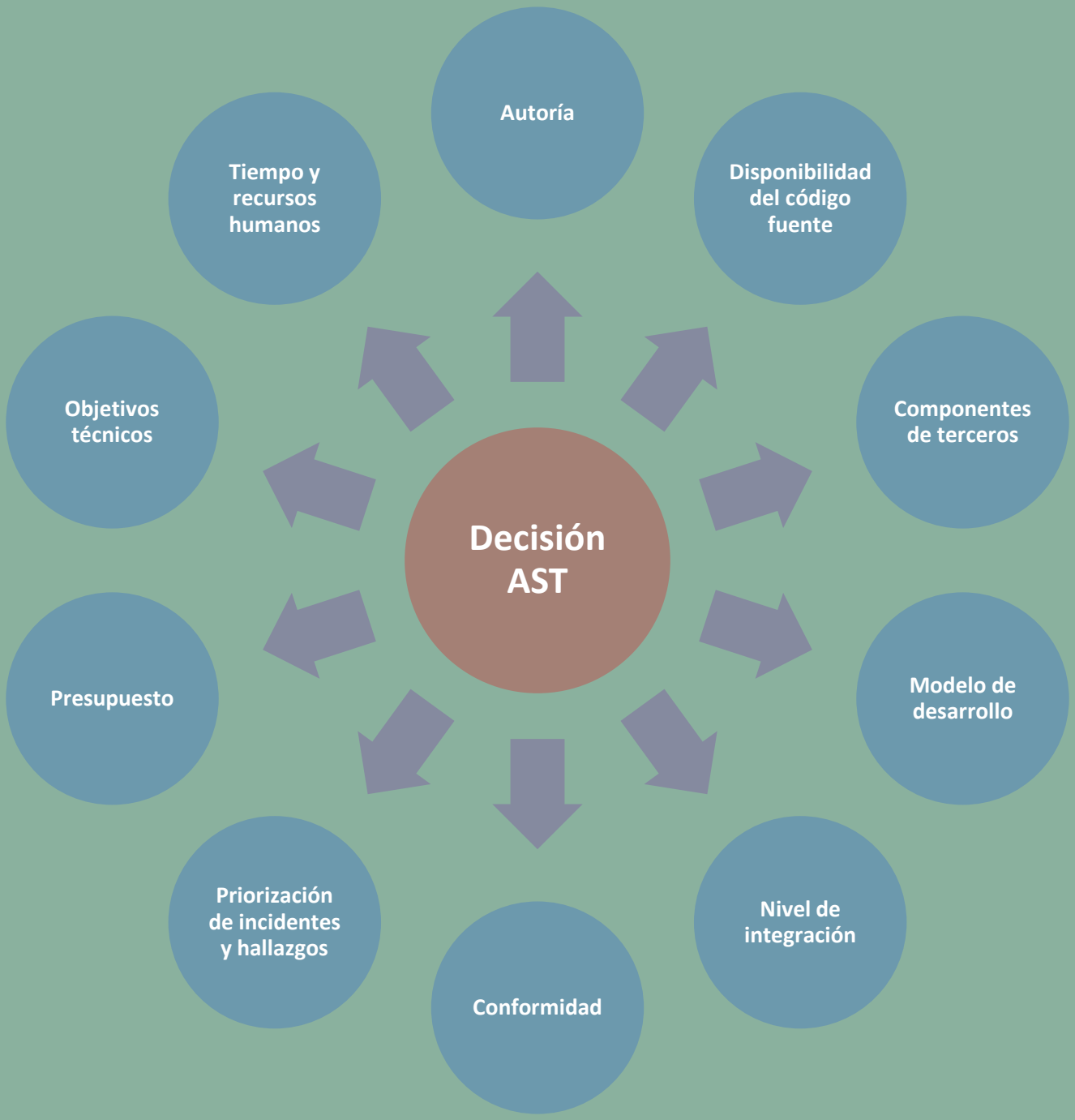
Tenga en cuenta que no existe un único tipo de herramienta que aborde todos los problemas, por lo que a largo plazo se recomienda una combinación de herramientas. Sin embargo, los factores aquí se discuten principalmente desde la perspectiva de seleccionar una única herramienta AST para comenzar con la automatización de la seguridad antes de que se introduzcan herramientas adicionales en el entorno. Más comúnmente, el primer tipo de herramienta utilizado será una herramienta de prueba de seguridad de aplicación estática (SAST), prueba de seguridad de aplicación dinámica (DAST) o una herramienta de análisis de composición de software (SCA) las herramientas en la parte inferior de la pirámide.

En última instancia, la mejor decisión es comenzar la prueba y utilizar cualquier herramienta de AST, pero a continuación se analizan algunos factores para ayudarlo a tomar la mejor decisión posible.



Application Security testing tools: Criterios de selección

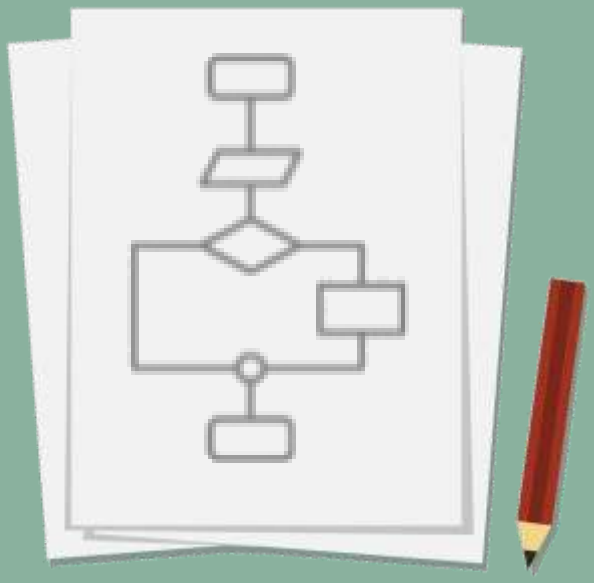
Factores de decisión



Application Security testing tools: Criterios de selección

Autoría

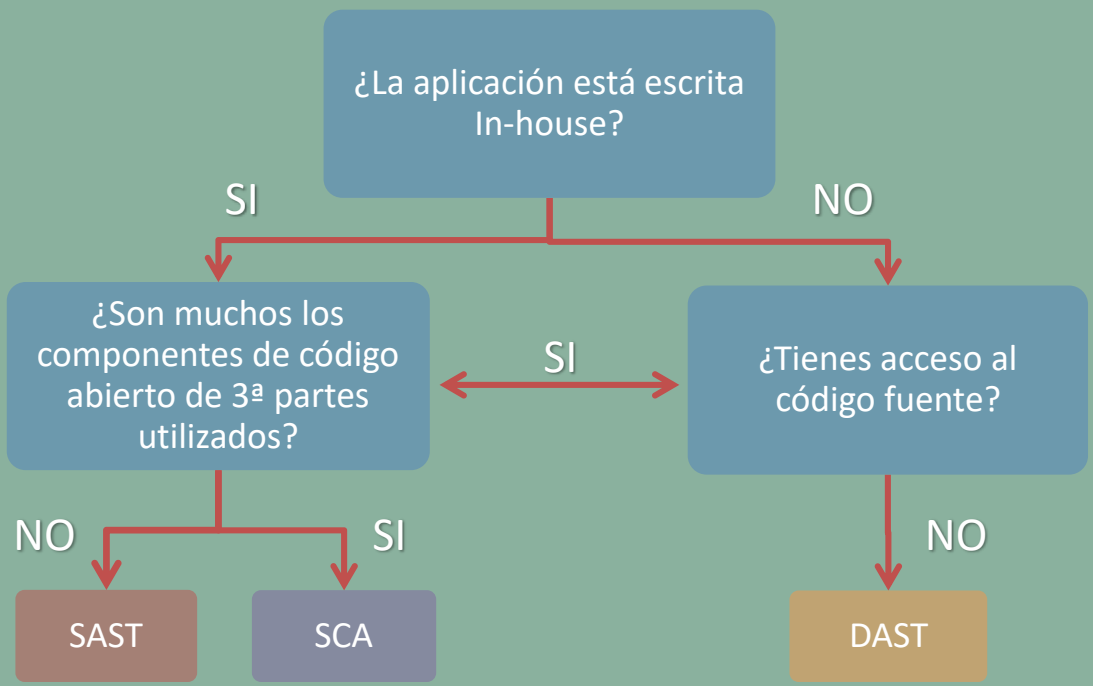
La autoría en el contexto de las pruebas de seguridad de la aplicación se refiere a quién desarrolla el código fuente bajo evaluación. Por lo general, el código fuente es desarrollado internamente por la organización que utilizará el código o el desarrollo está completamente subcontratado a un tercero. Sin embargo, hay muchos otros escenarios de desarrollo, como cuando partes del desarrollo de código se subcontratan a varios terceros diferentes y se integran de manera interna. Conocer el modelo de autoría puede influir en las estrategias de prueba de la aplicación. Cuando sea posible, es una buena idea usar las herramientas SAST y DAST independientemente de la autoría. Sin embargo, al seleccionar un único tipo de herramienta como punto de partida para la prueba, la autoría puede influir en las decisiones.



Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

- Si el código de la aplicación fue escrito solo o en gran parte en casa, las herramientas SAST deberían ser la primera opción. Las herramientas SAST son las más robustas y deben usarse siempre que sea posible.
- Si el código de la aplicación fue escrito por un tercero y entregado como un ejecutable, las herramientas DAST deberían ser la primera opción. Si bien todos los factores de decisión juntos darán forma a la decisión final, estas son las mejores opciones si la autoría es lo único que se sabe sobre una aplicación.
- Las aplicaciones internas también se prestan mejor para la correlación, el análisis de cobertura y las herramientas de Application Security Orchestration (ASTO) que los ejecutables de terceros.



Application Security testing tools: Criterios de selección

...01101
10001110
...01101

Disponibilidad del código fuente

- Si el código fuente está disponible, las herramientas SAST son la mejor opción. Como se mencionó anteriormente, las herramientas SAST son las más robustas y deben usarse siempre que sea posible. Si el código fuente no está disponible, es mejor usar las herramientas DAST.
- Si el código fuente está disponible y otros factores lo permiten, se deben usar las herramientas DAST y SAST. Las pruebas de seguridad de aplicaciones interactivas (IAST) y las herramientas híbridas también se convierten en una opción en este caso.
- Si la aplicación fue escrita por un tercero y el código fuente no está disponible, se deben usar herramientas y técnicas de falsificación y pruebas negativas además de las herramientas tradicionales de DAST.
- Si la aplicación fue escrita por un tercero y el código fuente está disponible, considere la posibilidad de ejecutar las herramientas de compilación y comparación para verificar que los ejecutables entregados pueden generarse exactamente de la misma manera que el código fuente entregado.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Componentes de terceros

- Si la aplicación hace un uso intensivo de librerías y componentes de terceros o los tiene en áreas críticas, las herramientas SCA son la primera opción, incluso antes que las herramientas SAST. Las herramientas de SCA pueden encontrar componentes y librerías que no se pensaron que estaban en la aplicación; por esta razón, las herramientas SCA deben usarse siempre que sea posible.
- Si la aplicación hace poco o ningún uso de librerías y componentes de terceros, use las herramientas SAST como primera opción.
- Si la aplicación se escribió en gran parte de forma interna y hizo un uso menor de las librerías, use SAST y SCA.
- Si la Aplicación fue escrita por un tercero y no está seguro del uso de la librerías, use SCA y DAST

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Modelo de desarrollo y plataforma de destino 1/2

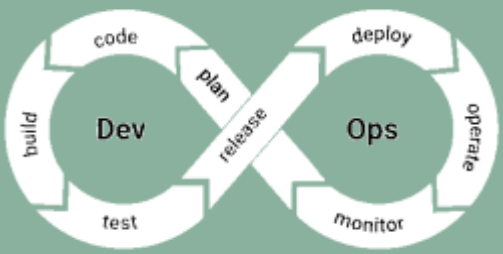
La plataforma de destino se refiere al entorno y el Stack técnico donde se implementará la aplicación. Obviamente, si la plataforma de destino es un dispositivo móvil, las herramientas de prueba diseñadas para aplicaciones móviles serían las más interesantes. Del mismo modo, si la aplicación se dirige a una implementación en la nube, las pruebas de seguridad de la aplicación como servicio (ASTaaS) pueden ser atractivas porque se pueden integrar fácilmente con muchas ofertas en la nube. Si una aplicación se implementará en la Internet pública, se recomiendan las herramientas DAST porque la aplicación estará expuesta a ataques de sombrero negro por parte de cualquier persona en Internet.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Modelo de desarrollo y plataforma de destino 2/2

- Si utiliza un ciclo de vida de desarrollo de software de cascada tradicional (SDLC), las herramientas SAST se adaptan bien a este proceso. Las herramientas DAST también encajan bien.
- Si utiliza un enfoque Agile , posiblemente con DevOps , IAST y las herramientas híbridas generalmente se ajustan mejor porque las herramientas tradicionales DAST y SAST independientes pueden requerir demasiado tiempo para el SDLC ciclo de vida del desarrollo de software.
- En un modelo de integración continua / entrega continua (CI / CD) , IAST y las herramientas híbridas vuelven a encajar mejor, pero generalmente un SCA es más importante en estos modelos de desarrollo si los componentes de terceros se utilizan con frecuencia.
- Si la aplicación está escrita por un tercero, el modelo de desarrollo no es tanto un factor de decisión, pero es posible que desee elegir ASTaaS por este motivo.
- Si se usan modelos de desarrollo Agile y / o CI / CD, y se sabe que se utilizan librerías y componentes de terceros, SCA se convierte en una necesidad, tan pronto como sea posible en el SDLC ciclo de vida del desarrollo de software.



Application Security testing tools: Criterios de selección

Nivel de integración

El factor de nivel de integración se refiere a la capacidad y viabilidad de agregar herramientas AST al SDLC ciclo de vida del desarrollo de software. Un principio general aquí es el cambio a la izquierda: integre las herramientas AST lo antes posible en el proceso.

- Si las herramientas se pueden integrar al principio del proceso, las herramientas SAST y SCA son altamente recomendables.
- Si las herramientas no se pueden integrar al principio del SDLC ciclo de vida del desarrollo de software, use las herramientas DAST junto con las herramientas de prueba de fuzzing y negativo.
- Si las herramientas no se pueden integrar en el SDLC ciclo de vida del desarrollo de software, considere ASTaaS.
- Si la aplicación está escrita por un tercero y dirigida a la nube, y si las herramientas no pueden integrarse en el desarrollo, use ASTaaS centrado en DAST y SCA.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Conformidad

Los procesos o controles de seguridad a menudo están dictados por políticas, contratos y regulaciones. Algunos ejemplos comunes son el Marco de Gestión de Riesgos (RMF) , la Ley Federal de Gestión de Seguridad de la Información (FISMA) , la Ley de Responsabilidad y Portabilidad del Seguro de Salud (HIPAA) , la Ley Sarbanes-Oxley (SOX) , el cumplimiento de la industria de tarjetas de pago (PCI) , los Objetivos de Control para Información y tecnología relacionada (COBIT) , y las Normas ISO. Algunas herramientas incluyen la asignación a estas políticas, contratos y regulaciones.

Si cree que no tiene estándares a seguir, considere listas como OWASP Top 10 , SANS Top 25 y CERT Coding Standards . Debido a que muchas directivas de cumplimiento involucran el almacenamiento y la protección de datos, los escáneres de seguridad de bases de datos son particularmente útiles. Las herramientas de correlación y ASTO también son útiles para los informes.

Si la Aplicación se escribe en forma interna utilizando un SDLC de cascada, y se necesita una estricta adherencia y reporte de la Ley de Portabilidad y Responsabilidad del Seguro de Salud (HIPAA), use SAST con una herramienta de correlación y una herramienta de escaneo de bases de datos.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Priorización de incidentes y hallazgos

- Si tiene un conocimiento histórico de las debilidades de aplicaciones anteriores, puede usarlas para ayudar a informar su estrategia para las pruebas de software de aplicaciones.
- Si su aplicación fue explotada anteriormente utilizando una vulnerabilidad conocida, asegúrese de usar una herramienta SCA fuerte o posiblemente múltiples herramientas.
- Si las revisiones de código manuales muestran prácticas de codificación débiles, implemente las herramientas SAST al principio del proceso de desarrollo.
- Si la aplicación está escrita de forma interna y es principalmente una aplicación web, pero también crea una versión móvil que ha recibido muchas quejas de clientes sobre fallas y errores, use las herramientas SAST, DAST y de prueba de seguridad de aplicaciones móviles (MAST). Recuerde que los bugs son puertas de acceso a una mayor explotación.

Application Security testing tools: Criterios de selección

Resumen de los factores de decisión para los tipos de herramientas AST

Analizar cada factor le permitirá crear una lista de tipos de herramientas AST para considerar. Los factores se presentan individualmente arriba, pero deberán ser considerados en forma conjunta al tomar decisiones. Algunos factores pueden empujarlo a cierto tipo de herramienta y otros factores pueden alejarlo de ese tipo de herramienta. Lo ideal es que implementes una combinación de herramientas. SAST, DAST y SCA deben usarse en combinación siempre que sea posible. Las herramientas IAST e híbridas se pueden usar si es necesario para obtener la mayor cobertura.

En los casos en los que solo se pueden considerar uno o dos tipos de herramientas, los factores de decisión anteriores deberían ayudarlo a priorizar lo que se puede hacer. Sin embargo, tenga en cuenta que una sólida comprensión de SAST tradicional, DAST y SCA es útil para tomar decisiones sobre MAST, IAST y ASTaaS. Las herramientas de correlación, cobertura de prueba y ASTO pueden mejorar el rendimiento y la efectividad de los otros tipos de herramientas AST, pero generalmente no son los primeros tipos de herramientas que se implementan.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Factores de decisión para la selección de herramientas AST

Después de saber qué tipos de herramientas AST se elige para una aplicación, la selección real de herramientas no es muy diferente de cualquier otra evaluación de tecnología.

A continuación se presentan algunos factores a considerar al seleccionar productos AST particulares

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Presupuesto

Afortunadamente, hay muchas herramientas gratuitas de código abierto de AST, especialmente SAST, DAST y SCA. Es una buena idea probar su uso de herramientas AST con productos de código abierto antes de seleccionar un proveedor comercial. Sin embargo, los proveedores comerciales de herramientas AST ofrecen herramientas que normalmente tienen más características y capacidades. Si su presupuesto solo permite una o dos herramientas comerciales de AST, puede complementarlas con herramientas de código abierto. Las herramientas IAST e híbridas pueden ayudarlo a controlar los costos al combinar las características de las herramientas SAST y DAST en un solo producto, en lugar de comprar y operar múltiples productos.

El ASTaaS puede ser costoso, pero puede ser más barato a largo plazo y, a menudo, facilita la realización de cálculos de presupuesto porque, por lo general, estos acuerdos pueden negociarse como servicios de tarifa plana con acuerdos multianuales..

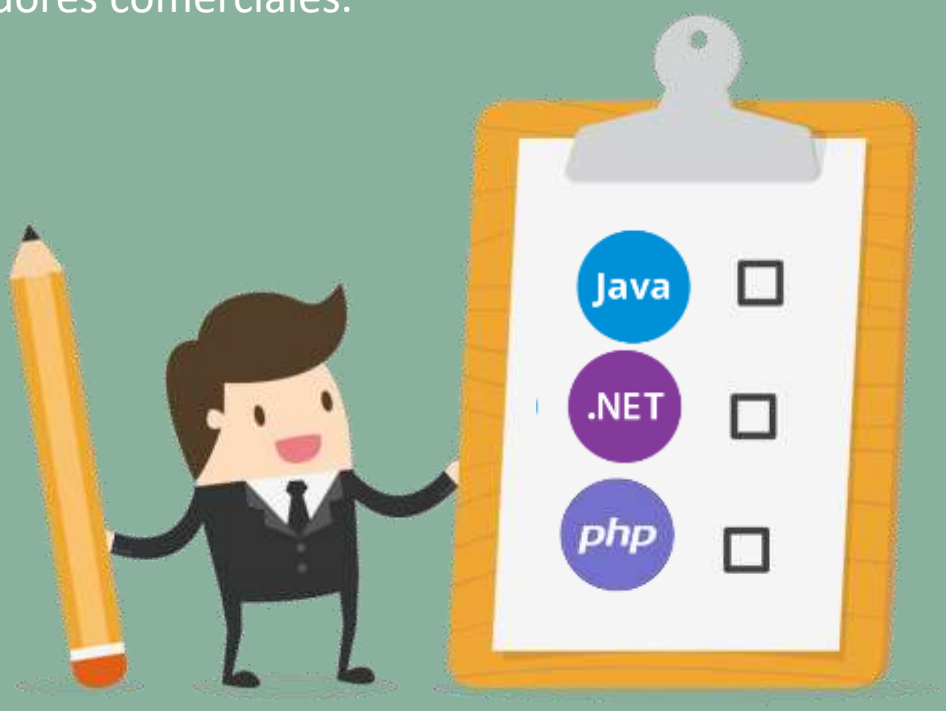
Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Lenguaje de desarrollo y Stack Tecnico

Obviamente, debe seleccionar una herramienta AST que sea compatible con su lenguaje de desarrollo, lo que ayuda a limitar la herramienta a los candidatos. Hay muchas opciones para lenguajes populares como Java, .NET y C / C ++. Busque herramientas que se integren con los entornos de desarrollo integrado (IDE) de sus desarrolladores . Algunos tienen integración nativa o API robustas.

Las grandes herramientas comerciales de AST son compatibles con varios lenguajes, lo que puede generar ahorros de costos en todos los proyectos. Algunas de las herramientas AST más pequeñas admiten solo un lenguaje, pero lo hacen muy bien. Sin embargo, si usa un lenguaje de nicho, sus opciones pueden ser limitadas y es posible que tenga que comprar un complemento opcional de proveedores comerciales.



Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Objetivos técnicos

Puede haber objetivos técnicos específicos descritos en los requisitos de la aplicación, políticas de la organización, directivas de cumplimiento, contratos y orientación similar. Los ejemplos incluyen controles relacionados con vulnerabilidades y exposiciones (CVE) conocidas, inyección de SQL , validación de entradas y desinfección , desbordamientos de búfer , secuencias de comandos entre sitios , gestión de contraseñas , ACL y menos privilegios , etc.

Estas son todas las cosas que las herramientas de AST están diseñadas para combatir. , pero si tiene requisitos específicos para ciertos aspectos, debe asegurarse de que su AST los maneje adecuadamente. El cumplimiento y la administración de licencias son dos ejemplos con los que las herramientas de SCA pueden ayudar.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Tiempo y recursos humanos

A largo plazo, la incorporación de herramientas AST en el proceso de desarrollo debería ahorrar tiempo y esfuerzo en la repetición del trabajo al detectar problemas antes. Sin embargo, la implementación de herramientas AST requiere una inversión inicial de tiempo y recursos.

Las herramientas de AST pueden producir muchos resultados, y alguien tiene que ser capaz de gestionarlos y actuar sobre ellos. Estas herramientas también tienen una gran cantidad de botones y botones para calibrar la salida, pero lleva tiempo establecerlos en un nivel deseable. Tanto los falsos positivos como los falsos negativos pueden ser molestos si las herramientas no se configuran correctamente. Las herramientas de correlación y ASTO pueden ayudar a administrar el flujo de trabajo de remediación y eliminar los problemas según corresponda.

Imágenes creadas por dooder

Application Security testing tools: Criterios de selección

Terminando y mirando hacia adelante

Hay muchos tipos de herramientas disponibles para realizar pruebas automatizadas de seguridad de aplicaciones. De acuerdo con el espíritu de la defensa en un profundo adagio de seguridad, cuantas más herramientas ejecute, más probabilidades tendrá de reducir las debilidades en sus aplicaciones. Sin embargo, es costoso y lento introducir herramientas en un entorno de desarrollo. Por lo tanto, recomendamos comenzar con una herramienta, dominarla y luego agregar herramientas cuando el tiempo y los recursos lo permitan. La guía anterior está diseñada para ayudarlo a seleccionar un punto de inicio adecuado para estas actividades.

Anteriormente hemos descrito una introducción general a las Application Security testing tools.

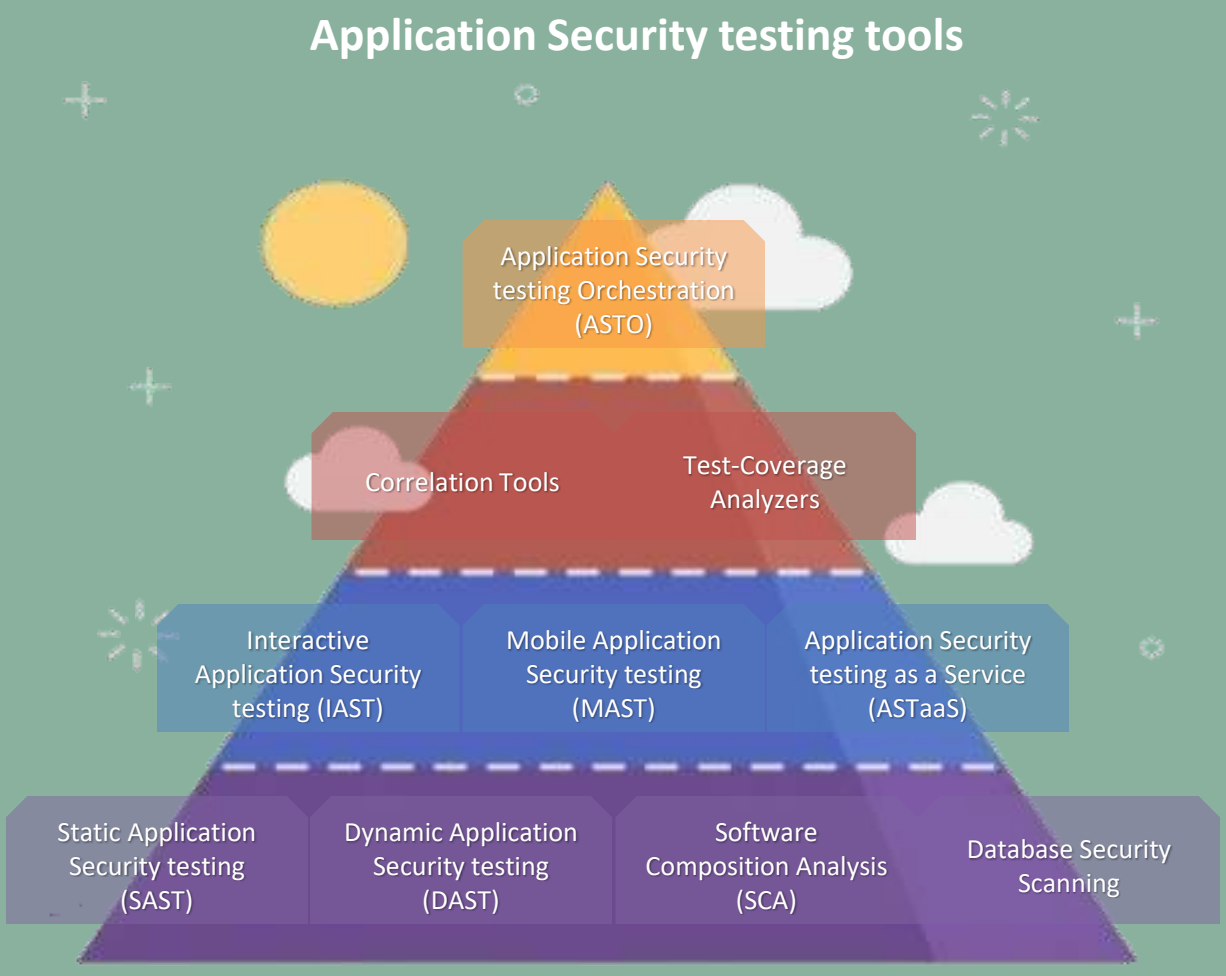
En el próximo E-book conoceremos a fondo el modelo de madurez de DevSecOps

Imágenes creadas por dooder

Application Security testing tools: lista por categoría

Imágenes creadas por dooder

Application Security testing tools: lista por categoría



A continuación veremos una lista con algunas de las herramientas mas conocidas en cada una de las categorías, ya sean las mismas open source o pagas.

Imágenes creadas por dooder

Application Security testing tools: lista por categoría

Application Security testing Orchestration (ASTO)

CODEDX: https://codedx.com/

QUALITEST: https://www.qualitestgroup.com/

CYBRIC'S: https://www.cybric.io/

KONDUKTO: https://kondukto.io/

Correlation Tools

CODEDX: https://codedx.com/

THREADFIX: https://threadfix.it/

ORCHESTRON: https://www.orchestron.io

DEFECTDOJO: https://github.com/DefectDojo

Application Security testing tools: lista por categoría

Test-Coverage Analyzers

CODE PULSE: http://code-pulse.com/

COBERTURA: http://cobertura.github.io/cobertura/

CODECOVER: http://codecover.org/

COVERAGE.PY: https://coverage.readthedocs.io/en/coverage-4.4.1/

EMMA: http://emma.sourceforge.net/index.html

GRETEL: http://www.cs.uoregon.edu/research/

HANSEL: https://sourceforge.net/projects/hansel/files/

JACOCO: http://www.eclemma.org/jacoco/

JCOV: https://wiki.openjdk.java.net/display/CodeTools/jcov

NOUNIT: http://nounit.sourceforge.net/

PITEST: http://pitest.org/

SERENITY BDD: http://www.thucydides.info/

ATLASSIAN CLOVER: https://www.atlassian.com/software/clover

BULLSEYE COVERAGE: http://www.bullseye.com/

FROGLOGIC COCO: https://www.froglogic.com/coco/

TESTWELL CTC++: http://www.testwell.fi/ctcdesc.html

NCOVER: http://www.ncover.com/

PARASOFT JTEST: https://www.parasoft.com/

DOTCOVER: https://www.jetbrains.com/dotcover/features/

OPENCOVER: https://github.com/OpenCover/opencover

VISUAL STUDIO: https://msdn.microsoft.com/en-IN/library/dd537628.aspx

Application Security testing tools: lista por categoría

Mobile Application Security testing (MAST)

ZED ATTACK PROXY: https://www.owasp.org/

MICRO FOCUS: https://software.microfocus.com/

KIUWAN: https://www.kiuwan.com/

QARK: https://github.com/linkedin/qark

ANDROID DEBUG BRIDGE: https://developer.android.com/

CODIFIEDSECURITY: https://codifiedsecurity.com/

DROZER: https://labs.mwrinfosecurity.com/tools/drozer

WHITEHAT SECURITY: https://www.whitehatsec.com/

SYNOPSYS: https://www.synopsys.com/

VERACODE: https://www.veracode.com/

MOBSF: https://github.com/MobSF/Mobile-Security-Framework-MobSF

PRADEO: https://www.pradeo.com/en-US/application-security-testing

APPKNOX: https://appknox.com/know-more/#mast-zone

PROMON: https://promon.co/

APP-RAY: https://app-ray.co/

APPTHORITY: https://www.appthority.com/solution/overview/

CHECKMARX: https://www.checkmarx.com/

DATA THEOREM: https://www.datatheorem.com/

KRYPTOWIRE: https://www.kryptowire.com/

NOWSECURE: https://www.nowsecure.com/

Application Security testing tools: lista por categoría

Application Security testing as a Service (ASTaaS)

WHITEHAT SECURITY: https://www.whitehatsec.com/

CHECKMARX: https://www.checkmarx.com/

MICROFOCUS: https://www.microfocus.com

VERACODE: https://www.veracode.com

SINOPSYS: https://www.synopsys.com

GITLAB: https://about.gitlab.com/solutions/dev-sec-ops/

Interactive Application Security testing (IAST)

ACUNETIX: https://www.acunetix.com/

GLASS BOX: https://www.ibm.com

CONTRAST: https://www.contrastsecurity.com/

SEEKER: https://www.synopsys.com/

BURPSUIT: https://portswigger.net/burp

VERACODE: http://www.veracode.com/

FORTIFY : http://www8.hp.com/

CHECKMARX: https://www.checkmarx.com/

PARASOFT: https://www.parasoft.com/

QUOTIUM: http://www.quotium.com

Imágenes creadas por dooder

Application Security testing tools: lista por categoría

Static Application Security testing (SAST)

BANDIT: https://wiki.openstack.org/wiki/Security/Projects/Bandit

BRAKEMAN: http://brakemanscanner.org/

CODESAKE DAWN: http://rubygems.org/gems/codesake-dawn

FINDSECBUGS: https://find-sec-bugs.github.io/

FLAWFINDER: http://www.dwheeler.com/flawfinder/

GRAUDIT: https://github.com/wireghoul/graudit/

LGTM: https://lgtm.com/help/lgtm/about-lgtm

PMD: http://pmd.sourceforge.net/

PROGPILOT: https://github.com/designsecurity/progpilot

PREFAST: http://msdn.microsoft.com/en-us/library/ms933794.aspx

PUMA SCAN: https://pumascan.com/

.NET SECURITY GUARD: https://dotnet-security-guard.github.io/

RIPS: http://rips-scanner.sourceforge.net/

PHPCS: https://github.com/FloeDesignTechnologies/phpcs-security-audit

SONARQUBE: http://www.sonarqube.org/

APPLICATION INSPECTOR: https://www.ptsecurity.com/ww-en/products/ai/

APPSCAN: http://www-01.ibm.com/

BLUECLOSURE BC DETECT: http://www.blueclosure.com/

BUGSCOUT: https://buguroo.com/

CODACY: https://www.codacy.com/

CAST AIP: http://www.castsoftware.com/

Application Security testing tools: lista por categoría

Static Application Security testing (SAST)

CODESONAR: https://www.grammatech.com/products/codesonar

CONTRASTE: http://www.contrastsecurity.com/

COVERITY: http://www.coverity.com/products/code-advisor/

CXSAST: https://www.checkmarx.com/

FORTIFY : http://www8.hp.com/

JULIA: http://www.juliasoft.com/solutions

KLOCWORK: http://www.klocwork.com/

KIUWAN: https://www.kiuwan.com/code-analysis/

PRUEBA DE PARASOFT: http://www.parasoft.com/

PITSS.CON : https://pitss.com/products/pitss-con/

PT APPLICATION INSPECTOR: https://www.ptsecurity.com/

PUMA SCAN PROFESSIONAL: https://pumascanpro.com/

PVS-STUDIO: http://www.viva64.com/en/

RESHIFT: https://www.softwaresecured.com/reshift

SECUREASSIST: https://www.synopsys.com

SENTINEL SOURCE: https://www.whitehatsec.com/

SEEKER: https://www.synopsys.com/

THUNDERSCAN: https://www.defensecode.com/thunderscan.php

VERACODE STATIC ANALYSIS: http://www.veracode.com/

XANITIZER: http://www.xanitizer.net/

GITLAB: https://about.gitlab.com/solutions/dev-sec-ops/

Application Security testing tools: lista por categoría

Dynamic Application Security testing (DAST)

ACUNETIX WV: http://www.acunetix.com/

APPLICATION SECURITY ON CLOUD: https://www.ibm.com/

APPSCAN: http://www-03.ibm.com/

APP SCANNER TRUSTWAVE: https://www.trustwave.com/

APPSPIDER: http://www.rapid7.com/products/appspider/

ARACHNI: http://www.arachni-scanner.com/

AVDS BEYOND SECURITY: https://www.scanmyserver.com/

BLUECLOSURE BC DETECT: https://www.blueclosure.com/

BURP SUITE: http://www.portswigger.net/

CONTRAST: https://contrastsecurity.com/

DETECTIFY: https://detectify.com/

DIGIFORT- INSPECT: http://www.digifort.se/en/scanner

EDGESCAN: https://www.edgescan.com/

GAMASCAN: http://www.gamasec.com/Gamascan.aspx

GRABBER: http://rgaucher.info/beta/grabber/

GRAVITYSCAN: https://gravityscan.com/

GRENDEL-SCAN: http://sourceforge.net/p/grendel/

GOLISMERO: http://www.golismero.com/

IKARE ITRUST: http://www.ikare-monitoring.com/

IMMUNIWEB: https://www.htbridge.com/immuniweb/

INDUSFACE: https://www.indusface.com/index.php/

N-STEALTH: http://www.nstalker.com/

NESSUS TENABLE: https://www.tenable.com/

Application Security testing tools: lista por categoría

Dynamic Application Security testing (DAST)

NEXPOSE: http://www.rapid7.com/

NIKTO: http://www.cirt.net/nikto2

PAROSPRO: http://www.milescan.com/

PROBE.LY: https://probely.com/

PROXY.APP: http://www.websecurify.com/desktop/proxy.html

QUALYSGUARD: http://www.qualys.com/products/qg_suite/was/

RETINA: http://www.beyondtrust.com/

SECURUS: http://www.orvant.com/

SENTINEL: http://www.whitehatsec.com/

SOATEST: http://www.parasoft.com/

TINFOIL SECURITY: https://www.tinfoilsecurity.com/

TRUSTKEEPER SCANNER: https://www.trustwave.com

VEGA: https://subgraph.com/vega/

WAPITI: http://wapiti.sourceforge.net/

WEB SECURITY SCANNER: https://www.defensecode.com/

WEBAPP360: http://www.tripwire.com/

WEBINSPECT: http://www8.hp.com/

WEBREAVER: http://www.websecurify.com/

WEBSECURIFY: https://suite.websecurify.com/

WIKTO: http://www.sensepost.com/research/wikto/

W3AF: http://www.w3af.org/

ZED ATTACK PROXY: https://www.owasp.org/

NETSPARKER: http://www.mavitunasecurity.com/

GITLAB: https://about.gitlab.com/solutions/dev-sec-ops/

Application Security testing tools: lista por categoría

Software Composition Analysis (SCA)

WHITESOURCE: https://www.whitesourcesoftware.com/

BLACK DUCK HUB: https://www.blackducksoftware.com/

SONATYPE NEXUS LIFECYCLE: https://www.sonatype.com/

VERACODE SCA: https://www.veracode.com/

SYNOPSYS PROTECODE: https://www.synopsys.com

FLEXNET CODE INSIGHT: https://www.flexera.com/

GITLAB: https://about.gitlab.com/solutions/dev-sec-ops/

Database Security Scanning

APPDETECTIVEPRO: https://www.trustwave.com

BSQL HACKER: http://labs.portcullis.co.uk/application/bsql-hacker/

SQLRECON: https://www.specialopssecurity.com/tools/sqlrecon/

SCUBA: https://www.imperva.com/lg/lgw_trial.asp?pid=213

PFCLSCAN: http://www.pfclscan.com/

DBSAT: https://www.oracle.com/

ORASCAN: https://www.nccgroup.trust/uk/

MCAFEE: https://www.mcafee.com/

PRIAMOS: http://www.priamos-project.com/whatis.htm

Conclusiones

Conclusiones

Cuando hablamos de incluir seguridad y sus pruebas en el mundo devops (DevSecOps) lo primero que nos viene a la mente es **“DevSecOps es un esfuerzo de equipo”** y cuando los equipos trabajan juntos, todos ganan.

Desarrollo: al integrar y automatizar los controles y pruebas de seguridad en todo el SDLC, los equipos de desarrollo optimizan el desarrollo y crean un software seguro y de alta calidad.

Seguridad: Cuando los equipos de seguridad trabajan con el equipo de desarrollo para llevar las pruebas de seguridad lo mas temprano en el SDLC, los costos y el esfuerzo disminuyen y las tasas de cumplimiento aumentan.

Operaciones: Los equipos de operaciones pueden evitar vulnerabilidades de las aplicaciones en producción cuando tienen una visión temprana de los componentes utilizados en las aplicaciones y los riesgos asociados a esos, implementado en todo caso controles compensatorios .

AUTORES

Christian Ibiri

www.linkedin.com/in/christian-ibiri

https://twitter.com/Christianibiri

Soy una persona apasionada por la tecnología y sobre todo las disruptivas o las que transforman la forma en que estamos acostumbrados de hacer las cosas, como computación en la nube, metodologías Agile, o Infraestructuras Agiles.

La verdad me encanta la época que estamos viviendo hoy por hoy donde uno puede levantar varias instancias para correr lo que se necesite sin tener que contar con un parque computacional enorme, mucho mas si me acuerdo de mis principios donde teníamos que desplegar maquinas físicas, armar clusters, “virtualización si teníamos suerte”.

Cofundador de DevSecOps Argentina, tiene 10 años de experiencia en IT, de los cuales los últimos 5 años en las áreas de Infraestructuras hibridas, cloud, DevOps y tooling automation.

Participe en varios proyectos de infraestructura, networking, migración y implementación de clouds privadas y publicas, automatización, comunicaciones unificadas y colaboración, acompañando a las demás partes involucradas en los mismos, desde la etapa de requerimientos hasta la implementación y pos-implementación. Creo en la interoperabilidad y que el mundo opensouce puede tranquilamente coexistir con tecnologías propietarias sin ningún tipo de limitaciones.

Entusiasta de DevOps, y infraestructuras agiles o infra como código.

Luciano Moreira

https://www.linkedin.com/in/lucianomoreiradacruz/

https://twitter.com/Luciano_m_cruz

Soy una persona apasionada por la tecnología y sobre todo por la seguridad de la misma en todos sus estados. creo que el conocimiento debe ser libre y que las personas deberían buscarlo todo el tiempo.
Tengo un Master en Ciberseguridad por la Universidad Camilo José Cela. Primer Auditor CSA STAR certificado en la región SOLA, Fui Elegido Cybersecurity Consultant of the Year en los premios Cybersecurity Excellence Awards 2016, 2017 y 2018, MVP - Most Valuable Professional Azure (Security, Infrastructure & Storage), Auditor Líder ISO 27001:2013, 27018, 27017 y 9001:2015.
Cofundador y presidente de DevSecOps Argentina, Presidente del capítulo argentino de la CSA Cloud Security Alliance. Miembro de ISSA, ISACA, OWASP, del comité académico de los eventos E-gisart y Cyber de ISACA y del comité científico en Ciberseguridad del evento IEEE ARGENCON, Orador e Instructor de seguridad en EducacionIT, FSA, I-SEC, entre otros. Cuento con más de 17 años de experiencia en IT, de los cuales los últimos 13 años trabaje en una docena de proyectos en todas las capas de seguridad y infraestructura hibridas. Responsable del diseño, implementación y gestión de proyectos con actividades como el análisis de riesgos, análisis de las deficiencias, cumplimiento, desarrollo de políticas de seguridad, seguridad en desarrollo, recuperación de desastres, plan de continuidad de negocio y seguridad en cloud.
Sólidos conocimientos y experiencia en proyectos de implementación y mejora de sistemas de gestión de la calidad y de seguridad de la información.

Miembro del Board Of Advisory como “DevSecOps strategy” en Cloud Legion

Santiago Cavanna

https://www.linkedin.com/in/santiagocavanna/

https://twitter.com/SCavanna

Soy Especialista en Seguridad de la información y Transformación Digital. Mas de 20 años de especialización en los mercados de Argentina, Chile, Uruguay, Paraguay y Bolivia.
Amplio conocimiento de la evolución de las tecnologías de información dentro de las diferentes industrias. Conocimiento de Estándares, Leyes y Regulaciones y su impacto sobre Nuevas Tecnologías de Información y Procesamiento en el ámbito privado o estatal embarcados en procesos de transformación digital
Desde el año 2003 he participado activamente en asociaciones de usuarios y profesionales de tecnología de la información y seguridad con el objetivo de promover el uso de mejores practicas de seguridad de la información a través de conferencias, trabajos de investigación y otras actividades de carácter publico complementado las actividades de igual índole dentro de mis funciones laborales.
Mi objetivo es poner en lenguaje accesible y comprensible la problemática de transformación digital y su impacto en la Ciberseguridad de las Personas, Organizaciones y Empresas. Guiar la adopción de estándares y buenas prácticas de Ciberseguridad y Asesorar tanto a empresas y organizaciones como a los diferentes cuerpos legislativos del territorio nacional, en su esfuerzo por construir o adecuar las políticas o leyes acordes a los desafíos que los tiempos modernos imponen.
Cofundador y miembro activo de ISSA Capitulo Argentina
Cofundador y miembro activo de ISC2 Capitulo Argentin
Miembro de ADACSI
Miembro del Comité Académico de Segurinfo, (2005-2015)

Miembro del Board Of Advisory como “Security Digital Transformation strategy” en Cloud Legion

Julio Cesar Balderrama

https://www.linkedin.com/in/juliocesarbalderrama/

https://twitter.com/juliobalderrama

Soy un profesional con más de 20 años de experiencia en las áreas de tecnologías de la información y comunicaciones; y más de 10 años de experiencia en seguridad de la información, continuidad del negocio y gestión de riesgos.
Instructor certificado, acreditado y autorizado por organizaciones internacionales como IRCA (The International Register of Certificated Auditors, Inglaterra), y PECB (Professional Evaluation and Certification Board, Canadá) para los entrenamientos profesionales de certificación y acreditación internacional bajo el estándar ISO/IEC 17024.Auditor Líder de sistemas de gestión certificables para las normas ISO/IEC 27001 Seguridad de la Información, ISO 22301 Continuidad del Negocio, ISO 20000 Gestión de Servicios de TI e ISO 9001 Gestión de la Calidad.
Participo activamente en diferentes ONGs y grupos de interés, entre ellos se encuentran ISOC Cybersecurity SIG - secretario del Board, en el capítulo argentino de la Cloud Security Alliance - consejero del directorio, en ISSA Argentina - miembro del Board, IGF Argentina, miembro de ISOC, ISACA entre otros.
Expositor e Instructor Internacional de cursos, seminarios, conferencias y congresos con una fuerte vocación y compromiso en la enseñanza, compartiendo los conocimientos y experiencias en materias relacionadas con la Continuidad de Negocio, Gestión de Riesgos, Seguridad de la Información y auditorías de Sistemas, todas estas actividades son realizadas en Washington DC, Miami (FL), Puerto Rico, México, El Salvador, Guatemala, Costa Rica, Panamá, Colombia, Perú, Chile, Bolivia, Paraguay, Brasil, Uruguay y Argentina.

Miembro del Board Of Advisory como “Compliance strategy” en Cloud Legion

Federico Pacheco

https://www.linkedin.com/in/federicopacheco/

https://twitter.com/FedeQuark

Soy un especialista en seguridad de la información.
A lo largo de mi carrera profesional he prestado servicios tanto a empresas locales y multinacionales, como a entidades de gobierno.
Cuento con más de quince años de experiencia docente en distintos niveles de enseñanza, y formo parte de la cátedra de "Seguridad Informática" en la Universidad Tecnológica Nacional (UTN) y en la Universidad Nacional de Quilmes (UNQ). Además participo periódicamente como expositor en eventos nacionales e internacionales sobre seguridad de la información, tecnología, software libre, y educación.
Actualmente trabajo en el área de Cybersecurity de JP Morgan.
Fui Presidente de la Comisión Directiva 2015-2017 del capitulo de ISSA Argentina (Information Systems Security Association) y colaboro en diferentes organizaciones sin fines de lucro. También soy Ambassador para Argentina de la red profesional de negocios europea XING y soy miembro de la organización internacional IEEE, perteneciendo a los grupos “Education Society”, “Security & Privacy”, “Computer Society” y “Power & Energy Society”.
Llevo publicados 4 libros y diversos trabajos de investigación, y estoy certificado en seguridad informática por la universidad de Stanford y el Instituto Tecnológico de Massachusetts (MIT).

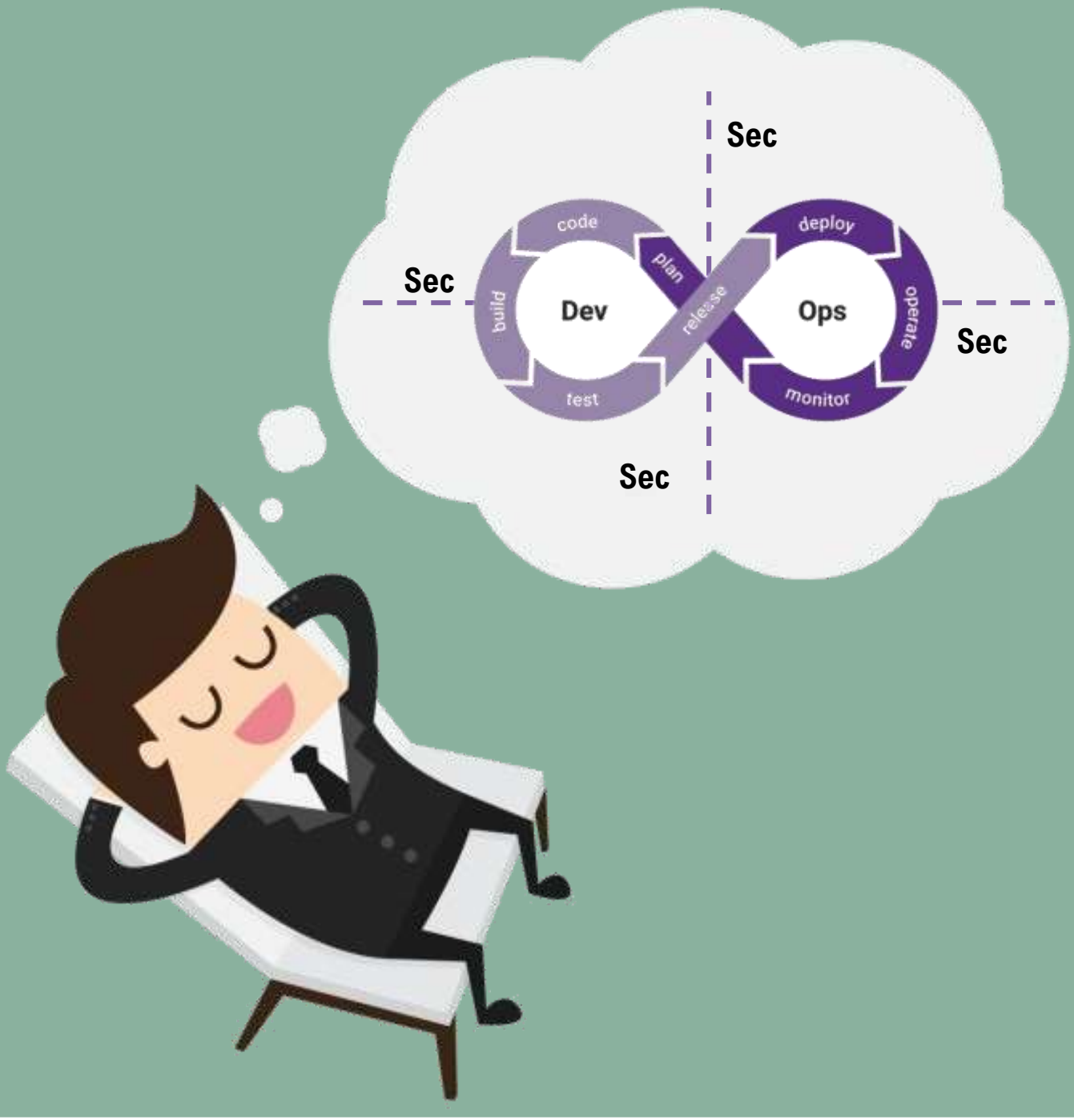
Miembro del Board Of Advisory como “Security Education & research” en Cloud Legion

Links y referencias

- https://cloudlegion.com.ar/
- https://www.ruggedsoftware.org/
- https://devsecops.org/
- https://news.netcraft.com/archives/2017/03/24/march-2017-web-server-survey.html
- https://en.wikipedia.org/wiki/Infrastructure_as_Code
- https://en.wikipedia.org/wiki/Wikipedia:If_it_ain't_broke,_don't_fix_it
- https://theconversation.com/why-dont-big-companies-keep-their-computer-systems-up-to-date-84250
- https://www.sonatype.com/2018survey
- https://en.wikipedia.org/wiki/Application_security#Security_testing_for_applications
- https://en.wikipedia.org/wiki/White-box_testing
- https://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html
- https://en.wikipedia.org/wiki/Black-box_testing
- http://www.gartner.com/it-glossary/dynamic-application-security-testing-dast
- https://en.wikipedia.org/wiki/Code_injection

Links y referencias

- https://www.forrester.com/report/Use+DevOps+And+Supply+Chain+Principles+To+Automate+Application+Delivery+Governance/-/E-RES118681
- https://nvd.nist.gov/
- https://vulndb.cyberriskanalytics.com/
- http://en.wikipedia.org/wiki/SQL_Slammer
- http://en.wikipedia.org/wiki/Access_control_list
- https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks
- https://en.wikipedia.org/wiki/Data_loss_prevention_software
- http://whatis.techtarget.com/definition/jailbreaking
- https://www.forbes.com/sites/louiscolumbus/2017/04/29/roundup-of-cloud-computing-forecasts-2017/#449ad54e31e8
- https://en.wikipedia.org/wiki/False_positives_and_false_negatives
- https://www.gartner.com/doc/3772095/hype-cycle-application-security
- https://en.wikipedia.org/wiki/Static_program_analysis
- https://en.wikipedia.org/wiki/Dynamic_testing
- https://www.forrester.com/report/Vendor+Landscape+Software+Composition+Analysis/-/E-RES122796



DEVSECOPS Argentina

CLOUD LEGION

https://www.linkedin.com/groups/12033278

https://twitter.com/devsecops_ar