Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de la Matanza
ISSN: 2525-1333
Vol.: 7 - Nro. 2 (DICIEMBRE 2022)

Artículo original

Características de las herramientas de pruebas estáticas de seguridad de las aplicaciones

Characteristics of static application security testing tools

Jorge Eterovic[(1) (2)], Valeria Silvestari[(1)], Martin Zeballos[(1)], Andrea Vera[(1)]

[(1)] Universidad Nacional de La Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
San Justo, Pcia. de Buenos Aires, Argentina
[(2)] eterovic@unlam.edu.ar

Resumen:

Las herramientas de Pruebas Estáticas de Seguridad de las Aplicaciones (SAST: Static Application Security Testing), permiten automatizar la detección de vulnerabilidades y se pueden integrar al sistema de distribución de las aplicaciones (CI/CD: Integración Continua/Distribución Continua) para que detecten las vulnerabilidades en etapas tempranas del ciclo de vida, lo que llevaría a implementar un ciclo de vida seguro de desarrollo de software.

Integrar una herramienta SAST al proceso de CI/CD permite detectar vulnerabilidades en la etapa de desarrollo, en vez de esperar a la etapa de prueba o que se detecten directamente en producción. En este trabajo se analizan las fortalezas, las debilidades, las características a considerar para la selección de una herramienta y se muestra un listado de las más usadas, ya sean open-source o pagas.

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de la Matanza
ISSN: 2525-1333
Vol.: 7 - Nro. 2 (DICIEMBRE 2022)

Abstract:

Static security test tools for applications (SAST: Static Application Security Testing), allow automating vulnerabilities detection and can be integrated into the applications distribution system (CI/CD: Continuous integration/continuous distribution) to detect Vulnerabilities in early stages of the life cycle, which would lead to implementing a safe software development life cycle.

Integrating a SAST tool into the CI/CD process allows you to detect vulnerabilities in the development stage, instead of waiting for the test stage or to be detected directly in production. In this work the strengths, weaknesses, characteristics to consider for the selection of a tool are analyzed and a list of the most used ones is shown, whether they are open-source or paid.

Palabras Clave: Herramienta SAST; Detección de Vulnerabilidades; Análisis Estático; Seguridad de las Aplicaciones.

Key Words: SAST tool; Vulnerability Detection; Static Analysis; Application Security.

Colaborador: Alesio Sinopoli

ISSN: 2525-1333. Vol.: 7 - Nro. 2 (DICIEMBRE-2022)

I. INTRODUCCIÓN

Existen muchas definiciones diferentes de DevOps disponibles en libros, en artículos de revistas o en Internet. A raíz de esta disparidad en las definiciones, surgen diversos estudios que tratan de darle una descripción académica [1] [2]. Según estos estudios, podemos definir DevOps como una cultura que trata de aunar a los equipos de desarrollo y operaciones, basándose en una serie de principios y prácticas [2] que pretenden acelerar las entregas del producto mejorando el feedback de los clientes y la capacidad de reacción ante los cambios [3].

El término DevOps surge a finales de los 2000, en un contexto en el que las metodologías de desarrollo ágiles cada vez tomaban mayor relevancia en la industria del desarrollo de software. La velocidad a la que se desarrollaban nuevas características o se corregían bugs distaba mucho de la velocidad a la que se realizaban los despliegues de estos cambios, con lo que el ciclo de desarrollo del software se veía ralentizado. Esta ralentización era resultado de la falta de comunicación existente entre el equipo de desarrollo y el de operaciones.

Fue Patrick Debois quien, en 2007, tras una experiencia frustrante trabajando en la migración de un gran centro de datos, se percató de cómo esta falta de comunicación entre desarrolladores y administradores de sistemas afectaba al flujo de trabajo [4]. En 2009, John Allspaw y Paul Hammond, ingenieros de Flickr, presentaron su charla "10 Deploys a Day: Dev and Ops Cooperation at Flickr" [5] donde propusieron integrar desarrollo y operaciones en un flujo automatizado. Patrick Debois, tras esta conferencia, decidió organizar una similar en Bélgica, a la que llamó DevOpsDays, de donde surge el término DevOps [6].

Un aspecto importante para seguir exitosamente la cultura DevOps es la automatización de procesos, al ser lo que permite mantener la agilidad durante el desarrollo del software. La importancia de la automatización de procesos ha hecho surgir una gran cantidad de herramientas que contemplan la construcción del software, la integración y el despliegue continuos, la gestión de logs y la monitorización [7].

Esto tiene grandes ventajas, pues permite tener feedback temprano del cliente para mejorar el producto desde las primeras fases de desarrollo, mejorando la adaptabilidad del producto con el entorno y permitiendo marcar la diferencia con la competencia gracias a la implementación de nuevas funcionalidades y la mejora del funcionamiento de las ya existentes [8].

DevSecOps surge para incluir la seguridad en DevOps, alineando los equipos de desarrollo, de operaciones y de seguridad durante todo el ciclo de vida de desarrollo. Esto se consigue haciendo participar al equipo de seguridad desde las primeras etapas del desarrollo [9]. De esta manera, al tenerla en cuenta desde el diseño de la aplicación, es posible realizar los controles de seguridad necesarios a lo largo del ciclo de desarrollo del software y automatizarlos para que sean rápidos, escalables y efectivos [10].

Al igual que en DevOps, las herramientas juegan un papel muy importante. Existen una gran cantidad de herramientas para llevar a cabo DevSecOps. Una referencia relevante es la guía OWASP DevSecOps

Guideline [11] para establecer los aspectos más importantes relativos a la seguridad: la detección de vulnerabilidades, las Pruebas Estáticas de la Seguridad de la Aplicación (SAST), las Pruebas Dinámicas de la Seguridad de la Aplicación (DAST), el escaneo de la infraestructura y la comprobación del cumplimiento normativo.

II. PRUEBAS ESTÁTICAS DE LA SEGURIDAD DE LA APLICACIÓN (SAST)

Las pruebas estáticas de seguridad analizan el código fuente de la aplicación sin ejecutarlo, tratando así de encontrar vulnerabilidades o bugs [12]. Los análisis estáticos se pueden realizar de diferentes formas, desde las más sencillas y rápidas que contemplan sólo un análisis del código fuente en base al árbol, hasta las más complejas que combinan diversas representaciones del código como grafos de control y flujo de datos para realizar un análisis semántico en busca de patrones vulnerables [12].

Los factores a tener en cuenta para elegir una herramienta son la velocidad a la que se quiere realizar el análisis y la profundidad del mismo, ya que a más profundidad, más se tardará en realizar y viceversa. Además, se ha de considerar el porcentaje de falsos positivos que puede dar la herramienta y si contempla el lenguaje de programación de la aplicación.

III. CARACTERÍTICS DE LAS HERRAMIENTAS SAST

Se estima que el 90 por ciento de los incidentes de seguridad son el resultado de ataques que explotan errores de software conocidos, por lo que, eliminar esos errores en la fase de desarrollo podría reducir los riesgos de seguridad.

Las herramientas de Pruebas Estáticas de Seguridad de Aplicaciones analizan el código fuente o las versiones compiladas del código tratando de encontrar vulnerabilidades o fallas de seguridad antes de que se conviertan en una versión final de software.

Las herramientas SAST se pueden agregar al Entorno de Desarrollo Integrado (IDE: Integrated Development Environment). La retroalimentación de la herramienta SAST puede ahorrar tiempo y esfuerzo, especialmente en comparación con la búsqueda de vulnerabilidades en fases avanzadas del ciclo de vida del desarrollo de software.

Las fortalezas de las herramientas SAST son:

- Escalabilidad: se puede ejecutar en una gran cantidad de lenguajes de software y se puede ejecutar repetidamente (por ejemplo: compilaciones nocturnas o integración continua).
- Identifica ciertas vulnerabilidades bien conocidas, tales como:
  - Desbordamientos de búfer
  - Defectos de inyección de SQL
- La salida ayuda a los desarrolladores, ya que las herramientas SAST resaltan el código con problemas, por nombre de archivo, ubicación, número de línea e incluso el fragmento de código comprometido.

También presentan debilidades, tales como:

- Búsquedas difíciles de automatizar para muchos tipos de vulnerabilidades de seguridad, que incluyen:
  - Problemas de autenticación

- Problemas de control de acceso
- Uso inseguro de la criptografía

- Las herramientas SAST actuales son limitadas. Pueden identificar automáticamente solo un porcentaje relativamente pequeño de las fallas de seguridad de las aplicaciones.
- Existe un alto número de falsos positivos.
- Con frecuencia no se pueden encontrar problemas de configuración, ya que no están representados en el código.
- Es difícil 'probar' que un problema de seguridad identificado es una vulnerabilidad real.
- Muchas herramientas SAST tienen dificultades para analizar código que no se puede compilar.
- Con frecuencia, los analistas no pueden compilar código a menos que tengan:
  - Bibliotecas correctas
  - Instrucciones de compilación
  - Todo el código requerido

Algunas de las herramientas SAST más usadas son las siguientes:

Nombre	***Tipo de Licencia***
Bandit	open-source
Brakeman	open-source
Checkmarx	comercial
CodeQL	Gratis para repositorios Open Source. Paga para repositorios privados.
Contrast Scan	comercial (con Edición Comunitaria Gratuita)
Coverity Scan	comercial
Fortify Static Code Analyzer	comercial
HCL AppScan	comercial (AppScan CodeSweep gratis)
Kiuwan Code Security	comercial
Klocwork	comercial (con un Free Trial)
LGTM.COM	comercial (libre para proyectos open-source)
Reshift	comercial (Gratis para usuario individual)
Semgrep	comercial (con Edición Comunitaria Gratuita)
Snyk	comercial (con edición de prueba limitada gratuita)
SonarQube	comercial (con edición Free Community)
Veracode Static Analysis	comercial

Fuente: recopilación propia

IV. CONCLUSIONES

Para seleccionar una herramienta SAST, se pueden seguir los siguientes criterios:

- Que contemple el lenguaje de programación utilizado.
- Que tenga capacidad para detectar vulnerabilidades, en base a:

- El Top Ten de OWASP (Open Worldwide Application Security Project). Representa un amplio consenso sobre los riesgos de seguridad más críticos para las aplicaciones Web.
- Otros criterios como:
  - OSSTMM (Open-Source Security Testing Methodology Manual). Proporciona una metodología para una exhaustiva prueba de seguridad a nivel operacional.
  - Prueba de penetración, también llamada pen test o hacking ético, es una técnica de seguridad cibernética que las organizaciones utilizan para identificar, probar y resaltar vulnerabilidades a su seguridad.

- Precisión:
  - Tasas de falso positivo/falso negativo.
  - Puntaje de referencia del Benchmark OWASP. OWASP Benchmark Project es un conjunto de pruebas en Java diseñado para evaluar la precisión, la cobertura y la velocidad de las herramientas SAST. Sin la capacidad de medir estas herramientas, es difícil comprender sus fortalezas y debilidades y compararlas entre sí.
- Capacidad para comprender las bibliotecas y los marcos que necesita.
- Requisito para el código fuente compilable.
- Capacidad para ejecutarse contra binarios (en lugar de fuente).
- Disponibilidad como complemento en los IDE usado por los desarrolladores.
- Facilidad de configuración y uso.
- Capacidad para ser incluidos en herramientas de integración/implementación continua (CI/CD).
- Costo de la licencia (puede variar según el usuario, la organización, la aplicación o las líneas de código).
- Interoperabilidad de salida:
  - SARIF (Static Analysis Results Interchange Format: Formato de Intercambio de Resultados de Análisis Estático). Es un estándar que define un formato de archivo de salida. El estándar SARIF se utiliza para optimizar la manera en el que las herramientas SAST comparten sus resultados.

V. REFERENCIAS

[1] de Franca, B. B. N., Jerónimo, H., & Travassos, G. H. (2016). Characterizing DevOps by Hearing Multiple Voices. Proceedings of the 30th Brazilian Symposium on Software Engineering - SBES '16. Published. https://doi.org/10.1145/2973839.2973845

[2] Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016). What is DevOps? Proceedings of the Scientific Workshop Proceedings of XP2016. Published. https://doi.org/10.1145/2962695.2962707

[3] Virani, M. (2015). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. Fifth International Conference on the Innovative Computing Technology (INTECH 2015). Published. https://doi.org/10.1109/intech.2015.7173368

[4] Watts, S. (2019, 29 March). A Brief History of DevOps. BMC Blogs. https://www.bmc.com/blogs/devops-history/

[5] Allspaw, J., & Hammond, P. (2009, Jun 22). 10+ Deploys per Day: Dev and Ops Cooperation at Flickr [Talk]. O'Reilly Velocity Conference, San Jose, California. https://www.youtube.com/watch?v=LdOe18KhtT4

[6] Debois, P. (s. f.). About devopsdays. DevOpsDays. Recuperado 18 de enero de 2022, de https://devopsdays.org/about/

[7] Akshaya, H. L., Nisarga Jagadish, S., Bidya, J., & Veena, K. (2015). A Basic Introduction to DevOps Tools. International Journal of Computer Science and Information Technologies, 6(3). http://ijcsit.com/docs/Volume%206/vol6issue03/ijcsit2015060382.pdf

[8] Beck, K., Fowler, M., Martin, R. C., Beedle, M., Cockburn, A., Cunningham, W., Thomas, D., Mellor, S., Schwaber, K., Sutherland, J., Bennekum, A. V., Grenning, J., Highsmith, J., Hunt, A., Je_ries, R., Kern, J., & Marick, B. (2001, 13 February). Principios del Manifiesto Ágil. Agile Manifesto. http://agilemanifesto.org/iso/es/principles.html

[9] Shackleford, D. (2016, 8 March). A DevSecOps Playbook. SANS. https://www.sans.org/webcasts/devsecops-playbook-101472

[10] Amazon Web Services. (2016, 9 November). Introduction to DevSecOps on AWS. https://www.slideshare.net/AmazonWebServices/introduction-todevsecops-on-aws-68522874

[11] The OWASP Foundation. (s. f.). OWASP DevSecOps Guideline. OWASP. Recuperado 24 de enero de 2022, de https://owasp.org/www-projectdevsecops-guideline/

[12] Adkins, H., Beyer, B., Blankinship, P., Lewandowski, P., Oprea, A., & Stubble _eld, A. (2020). Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems (Illustrated ed.). O'Reilly Media. https://sre.google/books/building-secure-reliable-systems/

Recibido: 2022-12-01
Aprobado: 2022-12-16
Hipervínculo Permanente: https://doi.org/10.54789/reddi.7.2.3
Datos de edición: Vol. 7 - Nro. 2 - Art. 3
Fecha de edición: 2022-12-29