ENTORNOS DE DESARROLLO

FORO EVALUABLE 2

ALUMNO CESUR

24/25

Alejandro Muñoz de la Sierra

PROFESOR

Diego Tinedo Rodríguez

Pruebas de seguridad en aplicaciones: SAST vs DAST

Introducción

En un primer encuentro se habló de lo vital que es cuidar la seguridad en cada paso del desarrollo; ahora, vamos un poco más allá, echando un vistazo a dos métodos muy utilizados para descubrir vulnerabilidades: uno examina el código sin necesidad de ponerlo en marcha y el otro pone a prueba la aplicación ya en funcionamiento. La idea es sencilla, pero, ya sabes, combinar ambas estrategias realmente potencia la protección del producto final.

¿Qué es SAST (Static Application Security Testing)?

SAST se centra en revisar el código fuente sin ejecutarlo —sí, es posible hacerlo antes incluso de compilar. Esto resulta súper útil en las etapas iniciales del desarrollo, ya que permite encontrar problemas antes de que se conviertan en dolores de cabeza. Por ejemplo, se pueden detectar inyecciones SQL, errores de lógica, usos indebidos de variables y prácticas cuestionables en la programación. Al final, identificar estos fallos desde el principio no solo mejora la calidad del software sino que, en la mayoría de los casos, reduce los costes para arreglarlos más adelante.

¿Qué es DAST (Dynamic Application Security Testing)?

Por el lado del DAST, aquí la prueba ocurre cuando la aplicación ya corre en un entorno real –sea en pruebas o incluso en producción– simulando ataques desde fuera, como si un usuario malintencionado estuviera intentando vulnerarla. Esta técnica ayuda a descubrir, por ejemplo, fallos en la autenticación, errores en la configuración de seguridad o problemas en la comunicación entre cliente y servidor. En definitiva, se trata de ver cómo reacciona el software en situación activa.

¿Cuál conviene más?

La respuesta no es elegir uno y desechar el otro. De entrada, SAST es ideal para sembrar seguridad en la base del código durante el desarrollo. Luego, cuando la aplicación ya está corriendo, DAST se encarga de probarla contra ataques externos, dándonos una imagen más realista de sus debilidades. Es como mirar la misma moneda desde dos ángulos distintos; en efecto, usarlos en conjunto ofrece una protección mucho más completa y robusta.

Herramienta recomendada para SAST

Una opción muy extendida en el análisis estático es SonarQube. Esta plataforma escanea el código para detectar errores, bugs o vulnerabilidades, y es compatible con lenguajes tan variados como Java, Python, C# o JavaScript. Además, se integra fácilmente en entornos de integración continua —Jenkins, GitHub o GitLab— lo que la hace especialmente práctica para equipos de desarrollo que buscan simplificar sus flujos de trabajo, sin demasiadas complicaciones.

Conclusiones

En suma, tanto SAST como DAST son pruebas clave que ayudan a salvaguardar la seguridad de nuestras aplicaciones. SAST, al analizar el código sin ejecutarlo, permite detectarlo todo desde el inicio, mientras que DAST pone a prueba la aplicación en situaciones reales, simulando ataques para descubrir vulnerabilidades. Y, sinceramente, no tienen por qué competir: al contrario, se complementan. Integrar herramientas como SonarQube en procesos de desarrollo modernos es, en definitiva, una estrategia que refuerza notablemente la seguridad y fiabilidad del software.

Referencias

https://www.opentext.com/what-is/sast?

https://www.opentext.com/what-is/dast?

https://www.sonarsource.com/solutions/security/

https://en.wikipedia.org/wiki/Dynamic_application_security_testing

https://www.sonarsource.com/learn/sast/

https://circleci.com/blog/sast-vs-dast-when-to-use-them/