IAST vs DAST and SAST

MAKING SENSE OF IT ALL

To keep up with the fast pace of releases and the speed of DevOps, organizations need accurate and automated security testing tools that can easily scale and produce actionable results.

Historically, AppSec programs were characterized by the use of Static Application Security Testing (SAST) tools which analyze the code or binary itself, and Dynamic Application Security Testing (DAST) tools that simulate attacks to see how an application reacts. Fast forward to 2019 - While SAST is able to fit fast and iterative development processes, point-in-time DAST is slow and manual, rendering it as unfit for DevOps-like processes. This is where the next-generation Interactive Application Security Testing (IAST) comes in.

IAST is a dynamic and continuous security testing solution that detects vulnerabilities on a running application by leveraging existing functional testing activities. IAST is designed to fit agile, DevOps and CI/CD processes. Unlike legacy DAST solutions, IAST does not introduce any delays to the software development lifecycle.

Here we take a look at the core differences between these three testing solutions to help to you decide which tools you need in your application security toolkit.

DevOps-fit

SAST	IAST	DAST
White Box Security Testing • Application is tested from the inside • Developer approach testing	**Grey Box Security Testing** • Application is tested from the inside out and outside in • QA approach testing	**Black Security Testing** • Application is tested from the outside in • Hacker approach testing
Early and Rapid • Vulnerabilities found early in the SDLC, making remediation faster and easier • Code-level guidance is provided on where to fix vulnerabilities in source code	**Fast and Immediate** • Quickly identify a broader range of runtime vulnerabilities providing insight down to the line of code that should be fixed • Provides real-time results, supporting DevSecOps and CI/CD processes	**Slow and Late** • Can't effectively achieve the fast turnaround times required for integration into CI/CD workflows • Offers no code guidance as to where to fix the vulnerability
Used Incrementally During the Development Stage • Runs incrementally only on new or modified code • Integrates with IDEs, build management servers, bug tracking tools and source repositories	**Used Continuously During the Testing Stage** • Runs continuously in parallel with functional testing • Integrates with any existing functional testing processes, whether manual or automated	**Used as a Security Gatekeeper** • Requires dedicated security testing and environment • Heavy reliance on experts to write tests, making it difficult to scale
Scans Code • Scans code or binary without executing the application • Doesn't require a deployed application	**Analyzes Running Application** • Integrates into the existing development and testing cycle, • Doesn't require code or binaries	**Attacks Running Application** • Injects input into external interfaces and observers external output
Covers all Code • Covers all in-house written code • Does not cover 3rd party modules	**Covers all Functional Testing** • Covers runtime vulnerabilities • Covers 3rd party modules	**Covers Only Reflective Vulnerabilities** • Blind as to what is happening inside an application

CHECKMARX

About Checkmarx

Checkmarx is the Software Exposure Platform for the enterprise. Over 1,400 organizations around the globe rely on Checkmarx to measure and manage software risk at the speed of DevOps. Checkmarx serves five of the world's top 10 software vendors, four of the top American banks, and many government organizations and Fortune 500 enterprises, including SAP, Samsung, and Salesforce.com. Learn more at **Checkmarx.com**.