opentext™

SAST vs. DAST: Top 10 differences (and 6 similarities)

Both static application security testing (SAST) and dynamic application security testing (DAST) are necessary for a complete picture of code security. But what are the differences and similarities between them?

SAST

White-box security testing
SAST analyzes an application from the "inside out," searching for vulnerabilities in the source code.

For source code
SAST doesn't require a deployed application. It analyzes the source code without executing it.

Reporting during development
SAST educates developers about security while they work through real-time recommendations, accelerating vulnerability discovery and collaborative auditing.

Covers languages developers use
SAST meets developers where they're at by integrating with their CI/CD build tools.

Scans many types of software
SAST tools can identify issues unique to certain programming environments or frameworks.

Fewer false negatives
But more false positives.

Relatively fast
Delivers results in minutes for normal cases, hours for slow cases.

"Start Left"
SAST tools can be configured to automatically scan codebases within the CI/CD pipeline at the start of the SDLC.

Check against security standards
SAST tools check code against established security standards/guidance (such as OWASP Top 10, CWE/SANS Top 25) and regulatory compliance (PCI, GDPR, HIPAA)..

VS

DAST

Black-box security testing
DAST attacks the application like a malicious user would, from the "outside in."

For a running application
DAST simulates external attacks on a running application, looking for unexpected results and identifying security vulnerabilities.

Reporting in testing or production
DAST tools can function in a dynamic environment, so they can detect run-time flaws that SAST tools can't identify.

Requires some security knowledge
Some security knowledge is needed for developers to interpret DAST reports.

Only scans web apps and web services
DAST is not useful for non-web applications.

Fewer false positives
But more false negatives.

Slower than SAST
Although DAST is seen as suited for later stages of development, it now integrates more seamlessly with CI/CD pipelines.

"Shift Left, to a point"
DAST requires accessible applications. It can be limited by the deployment environment's accessibility or configurations that might block testing efforts.

Understand complex transactions
DAST understands complex transactions that span multiple layers, such as authentication flows, session management vulnerabilities, and business logic errors.

What SAST and DAST can both detect

Injection flaws:
SAST detects potential injection points by analyzing source code for patterns that lead to injection. DAST finds them by attempting injections.

Cross-site scripting (XSS):
SAST identifies these flaws by analyzing code for improper input handling. DAST injects scripts and checks if they are executed.

Insecure deserialization:
SAST can look for code that deserializes objects without adequate checks. DAST can attempt to deserialize data to see if the application behaves unexpectedly.

Directory traversal:
SAST can detect directory traversal flaws by analyzing how applications access files. DAST can attempt to access files beyond the application's scope.

Security misconfiguration:
SAST can spot misconfigurations by analyzing configuration files and code settings. DAST probes applications to discover information leaks or authentication bypasses.

Insecure direct object references (IDOR):
SAST can find instances of IDOR by analyzing access control checks in the code. DAST attempts to access unauthorized objects.

Both SAST and DAST are needed for comprehensive AppSec testing.

Fortify by opentext™

Read our flyer on why SAST + DAST with Fortify makes sense

opentext.com