CHECKMARX
Source Code Analysis Made Easy

The AppSec How-To: Choosing a SAST Tool

GIVEN THE WIDE RANGE OF SOURCE CODE ANALYSIS TOOLS, SECURITY PROFESSIONALS, AUDITORS AND DEVELOPERS ALIKE ARE FACED WITH THE QUESTION:

How to assess a Static Application Security Testing (SAST) tool for deployment?

Choosing the right tool requires different considerations during each stage of the SAST tool evaluation process.

Evaluation Preparation

The following qualifiers are required prior to testing the SAST tool in order to set initial expectations:

1. **List of languages.** Ensure that the SAST tool supports the languages in the development environment.

2. **Access to source and binary files.** Some SAST tools run only on the source code files (pre-compilation scanning), while others run on the binaries (post-compilation scanning). As opposed to scanning on the source code, post-compilation scanning requires all project dependences in order to run the scan.

3. **Deployment.** Confirm the SAST tool supports the preferred mode of operation - on premise or on-demand.

4. **Parties within the organization responsible for code security.** Define how code security is managed within the organization. For example, one organization might prefer having a dedicated team – such as code auditors or an application security team – which provides the security services to the organization.
   While another organization might decide that each development team has an individual responsible for the security. Each of these management models influences the SAST tool architectural setup- including licenses, deployment and tool's usage.

STAGE 1 Installation

Ease of installation during this step includes:

1. **Resources.** Evaluate whether installation of the SAST tool is manual or automated. If manual, consider whether installation requires specialized knowledge as well as the number of installation man-hours.

2. **Scalability.** Client software installation requires developer down-time during installation and additional installation time per endpoint. A centrally-managed installation is a one-time only process where additional servers can be added without the need for system duplication.

3. **Licenses.** Some licensing schemes are distributed where each endpoint requires its own license. In other cases, the license is centrally-managed and is on a per-user basis, eliminating the need for multiple licenses.

STAGE 2 Set-up

Two measuring factors need to be considered:

1. **Effort and complexity**
   - o **Simplicity.** Scanning overhead should be kept to a minimum. Scanning source code should not require the user to perform excessive operations to start running the tool.
   - o **Scaling to other languages.** Adding a new language should be seamless to the environment and should not entail a new scanning setup to support the language.

2. **Time Scanning** – regardless of the SAST tool- takes time.
   The point here is to consider the SAST features, or the different scanning methods, that the SAST tool provides to speed up the scanning process. For example, being able to scan portions of the code is particularly helpful when there are lots of developers and code to scan.

STAGE 3 Scan capabilities

Scanning capabilities include:

1. **Range of supported languages.** The SAST tool should not merely support the current development languages (as specified when qualifying the tool). It should also support emerging technologies as these may prove to be significant in the long run. For example, mobile or updated development languages (e.g Android, Objective C, Ruby on Rails).

2. **Range of supported frameworks.** Supporting the development's framework allows the SAST tool to identify coding vulnerabilities, as well as to eliminate any false reporting that results from not recognizing the framework.

3. **Multiple scans.** The ability to run simultaneous scans or support multi-chaining, multi-threading or multi-core processing environments.

4. **Vulnerability coverage.** There are different classes of vulnerabilities that the SAST tool should address:
   - ***Technical security vulnerabilities.*** Detection of common vulnerabilities as identified by different industry standards such as OWASP Top 10, SANS and CWE. Since the vulnerability taxonomy and ratings differ by each SAST vendor, it is necessary to receive from each SAST vendor their list and normalize them one against the other for a true vulnerability coverage comparison.
   - ***Business logic flaws.*** These include authentication by-passing mechanisms, as well as backdoors in the application.
   - ***Best coding practices.*** For example, error handling, elements usage and race conditions.

5. **Result accuracy.** To ensure the accuracy of the results, the tool should scan and its output compared against a test application for which the results are known a-priori. One such common test bed is OWASP's WebGoat project. However, the real test should be against an in-house application- unknown to the tool – to prevent the tool from being tuned in advanced to the testing environment.

   Result accuracy is measured by:
   - ***Amount of True Positives (TPs).*** The percentage of results that have been correctly identified as actual vulnerabilities.
   - ***Amount of False Positives (FPs).*** Although there is no such SAST tool today that will output a totally FP-free scan, the ideal is to achieve a minimal amount – up until a handful of these.

6. **Customizability.** The ability to adapt the scan results to the specific software frameworks and business logic of the organization. Each organization uses its' own framework for accessing databases and sanitizing input data and so the SAST tool must be customizable to the proprietary code. This capability also eliminates false positives that occur due to the custom code and the organization's business logic.

7. **Ability to aggregate scans.** Aggregation allows all the scans of the project to be displayed as a whole.

STAGE 4 Results Management

Scan results need to be presented in a clear manner to enable convenient and quick fixing.

1. Results analysis and management tools. Results analysis should provide the user with the relevant security intelligence and tools to remediate flaws in virtually zero-time.
   - o ***Vulnerability flow.*** Visibility into the code flow down to the exact line of the vulnerable code helps developers to understand the vulnerability flow and its meaning.
   - o ***Best fix locations.*** Optimal vulnerability remediation can be presented in textual or visual formats. For example, the ability to pinpoint the precise vulnerability which- if fixed-eliminates all vulnerabilities that depend on that particular code flaw.
   - o ***Tagging and filtering capabilities.*** Users should be able to group results according to policies, and prioritize results from highly important to un-exploitable. Further, the tools should provide the ability to filter out results as in the case of a test directory.
   - o ***Ability to track projects.*** The scan tool should be able to keep the status of vulnerabilities between scans for tracking purposes.
   - o ***Scan comparison.*** The SAST tool should enable the comparison of results from one scan to another to monitor the state of vulnerabilities.

2. Reports. The tool should provide multiple layers of reporting.
   - o ***Dashboard.*** Provides a typical executive summary section with a high-level overview of the state of the application's code.
   - o ***Reports per policy.*** The ability to configure a report to present only relevant information. For example, PCI.

STAGE 5 Integration into the SDLC

There are both logical and technological aspects when integrating source code analysis within the Software Development Life Cycle (SDLC):

1. **SDLC model.** Measurements include:
   - o ***Early-stage scanning.*** Scanning early supports SDLC's fundamental concept of fixing code flaws – including security vulnerabilities- as early as possible within the development process. Various SAST tools provide the ability to scan code prior to code compilation, or before the code's check-in.

- o ***Support for secure Agile development and Continuous Deployment environments.*** Agile and Continuous Deployment (aka DevOps) mandate that scanning must be done within minutes, and cannot tolerate any latency due to excessive processing, scanning overhead and fixing. Accordingly, the SAST tool should enable the developers to perform ad-hoc scanning from within their development environments.
- o ***Rescanning.*** Rescanning a project should not require the redundant scanning of files previously analyzed. For example, SAST tools with incremental scanning features scan only the code – and its dependencies – that were modified from the previous scan.

2. **SDLC tools.** The SAST tool should be able to incorporate, as-if naturally, within the enterprise systems without requiring extra tuning or configuration. The point here is not only to save developer time but also making security part of the development process.
   Suggested integration points include:
   - o ***Development environment.*** The SAST tool needs to seamlessly fit into the development environment – regardless of language and compiler versions. This also includes integration within the IDE-development tool (e.g. Visual Studio, Eclipse, IntelliJ).
   - o ***Build management tools.*** e.g. TeamCity, Bamboo, Jenkins, Maven and Ant.
   - o ***Source-code repositories.*** e.g. GIT, SVN, TFS, Mercurial, ClearCase. Several SAST tools can run within the source code repository, without even requiring a build management system.
   - o ***Bug-tracking system.*** The SAST tool should be able to inject results of the scan into bug tracking systems to prioritize vulnerability fixing according to release schedule, time to fix, vulnerability impact, and how it fits with other tasks.

STAGE 6 Responsiveness and Support of Vendor

Last but not least, a SAST purchase is an ongoing process. Just like any tool, there may be questions regarding its usage, best practices and of course, customizability aspects. Consider the following services from the vendor:

- Implementation of customized SAST queries (aka rules) and policies for your proprietary code
- Engineer support and training for the SAST tool users
- Account manager to accompany your organization throughout the lifetime of the SAST tool
- Availability and responsiveness to inquiries throughout the lifetime of the SAST tool