Enterprise Application Security

The 5 Key Benefits of Source Code Analysis

Static Code Analysis: **Binary vs. Source**

Static Code Analysis is the technique of automatically analyzing the application's source and binary code to find security vulnerabilities.
According to Gartner's 2011 Magic Quadrant for Static Application Security Testing (SAST):

> SAST SHOULD BE CONSIDERED A MANDATORY REQUIREMENT FOR ALL IT ORGANIZATIONS THAT DEVELOP OR PROCURE APPLICATION.

In fact, in recent years we have seen a shift in application security, whereas code analysis has become a standard method of introducing secure software development and gauging inherent software risk.
Two categories exist in this realm:

1. **Binary – or byte- code analysis (BCA).** Analyzes the binary/ byte code that is created by the compiler.
2. **Source code analysis (SCA).** Analyzes the actual source code of the program without the requirement of retrieving all code for a compilation.

Both offerings promise to deliver security and the requirement of incorporating security into the software development lifecycle (SDLC). Faced with the BCA vs SCA dilemma, which should you choose?

THE INHERENT FLAWS OF BINARY CODE ANALYSIS (BCA)

On the one hand, BCA saves some of the code analysis efforts since the compiler automates parts of the work such as resolving code symbols. Ironically, however, it is precisely this compiler off-loading which presents the fundamental flaw with BCA. In order to use BCA, all code must be compiled before it is scanned. This raises a plethora of problems that push back the SDLC process and gives security a bad, nagging name.
Issues include:

- **Vulnerabilities exposed too late in the game.** Since all the code must be compiled prior to the scan, security gets pushed to a relatively late stage in the SDLC. At this point, the scan usually finds too many vulnerabilities to handle, no time to fix, and pressure from sales and marketing teams to release the product. As a result, these vulnerabilities – albeit being uncovered – are pushed to release. In fact, actual vulnerabilities have already slipped through the scanning process in real-world projects, such as occurred in a Linux OS distribution release.
- **Compiler optimization hurts the accuracy of the results.** One of the many roles compilers fulfill is to optimize code in terms of efficiency and size. However, this optimization may come at the expense of the accuracy of results. For example, compilers might remove so-called "irrelevant" lines, aka dead code. These are lines of code that developers insert as part of their debugging process. While the compiler removes these code snippets, they can contain code that breaches corporate standards.

- **PaaS-providers incapable of retrieving the byte-code.** In a Cloud Computing scenario, the PaaS-provider is responsible for validation, proprietary compilation and execution of the programs. However, the PaaS provider cannot retrieve the byte-code, or has no manifestation as byte-code or binary.

BENEFITS OF SOURCE CODE ANALYSIS (SCA)

By scanning the source code itself, SCA can be integrated smoothly within the SDLC and provide near real-time feedback on the code and its security. Source code analysis comes to compensate for BCA's shortcomings and provide an efficient, workable alternative. How?

1

Scans Code Fragments and Non-Compiling Code

An SCA tool is capable of scanning code fragments, regardless of compilation errors arising from syntactic or other errors. Developers can scan incomplete code in the midst of the development process, ultimately allowing the discovery of vulnerabilities much earlier during the Software development Life Cycle (SDLC).

2

Supports Cloud Compiled Language

New coding language breeds have developed under cloud computing scenarios. In these cases, the developer codes in the PaaS-provider's language, while the PaaS-provider is responsible for the validation, proprietary compilation and execution of the programs. In these cases, the code has no manifestation as byte-code nor as binary, and the SCA must be done on the source code itself. The most known example is the Force.com platform supplied by Salesforce.com. This platform is based on the server-based language called Apex, and client-based language called Visualforce. Only an SCA product can support this new paradigm.

3

Assesses Security of Non Linking Code

In the case where the code references infrastructure libraries for which their source is missing, the BCA tools immediately fails on the unfortunate "Missing Library" message. Days may be spent building stubs for these missing parts, just to make the code compile – a lot of hard work without any added value.
An SCA product easily identifies vulnerabilities, such as SQL Injection - even when the actual library code of the executing SQL function call is missing.

4

Compiler Agnostic

In a multi-compiler environment- typically found at code auditors and large corporations- the SCA standard provides a one solution fits all. This is starkly opposed to the BCA which must support an endless number of compilers and versions. The reason? Each compiler transforms source code into its own version of binary/ byte code forcing the BCA tool to read, understand and analyze the different outputs of different compilers. However, since an SCA tool runs on the code itself – and not post-compilation, the SCA provides a single standard irrelevant to the compiler version or compiler upgrades.

5

Platform Agnostic

Similarly, when integrating SCA into the SDLC, the exact same tool can be used to scan the code anywhere – regardless of the operating system or development environment. This eliminates the inherent redundancy of BCA which must deliver separate scanning tools for each platform.