Static Analysis, Security Failure


Wayne Ariola

Static analysis for security has a hot topic lately, and I fear we're starting to think of it as a silver bullet. The quest for application security has breathed new life into static analysis technologies, which until recently were primarily perceived as either frivolous beautification tools or burdensome big brother monitoring systems. Surprisingly, the underlying technology was not substantially modified to accommodate the issue of security. Rather, the changes were more like a face lift. As a result, organizations using static analysis still encounter the same challenges in making it sustainable over time.

The secret to making static analysis tools productive is to use them in the proper context. The adoption of this technology should be driven by a policy-based approach. This means establishing a policy that defines requirements, then enforcing that policy consistently. Automation helps ensure that the required practices are sustained, and workflow, task management, and metrics enable you to measure how well the policy is being implemented. In the context of policy, static analysis is elevated from a "nice-to-have" checker to a critical tool for ensuring that code meets the organization's expectations.

How Policy Provides Context

The most likely culprit for the reputation static analysis has as a nonessential technology is its lack of context. Products provide "out-of-the box" support for hundreds of rules which could be important in many different contexts. However, most organizations don't take the time to determine which rules are most important in the context of their own organization, team, and project. Then they require compliance to those carefully-selected rules. As a result, rule violations are perceived as suggestions for general code improvements—not critical coding issues that need to be addressed immediately.

The key to providing the necessary context to static analysis is to take a policy-based approach: use static analysis to monitor a non-negotiable set of expectations around code security, reliability, performance, and maintainability. With this approach, a violation of a particular guideline is not just another suggestion for people building software in an ivory tower—it's notification that the code failed to meet the organization's expectations.

Effective policy management allows an organization to bridge the gap between management expectations and developer performance. Essentially, if a static analysis rule enforces something that is part of the policy, fixing a violation of that rule is non-negotiable. If a developer fails to satisfy the defined policy, he is not executing his job as expected by management.

What's Needed to Make it Work

The rising risk and impact of application-level security attacks has brought static analysis and its challenges into new light. Static analysis has great potential for ensuring that code is written in ways that prevent security vulnerabilities. However, to ensure that static analysis delivers as promised here, it's essential to address the challenges that have traditionally stymied its success. This is where considerations such as policy management, workflow management, and workflow optimization come into play.

For example, using static analysis as an audit that occurs at later stages of the SDLC only exacerbates its tendency to drain development resources. Having an inline process makes the analysis more valuable and more effective. Since the code is still fresh in developers' minds when violations are reported, developers are more likely to learn from their mistakes and remediate problems faster and more easily.

Policy management lies at the core of such an inline process. You should be able to easily configure policies for specific projects without compromising the integrity of the corporate objectives, easily deploy and update both project-specific and organization-wide policies, and automate their application for rapid scanning and reporting. A carefully defined and implemented set of policies establishes a knowledge base that allows developers to increase their relative security IQs.

Putting the policy into practice involves workflow management—defining, automating, and monitoring security verification and remediation tasks, which are ingrained into the team's workflow. These tasks must be optimized to ensure that the static analysis process is both sustainable and scalable. The lack of automation, repeatability, or consistency will degrade any quality initiative that the organization intends to deploy.

Second Time's a Charm?

Static analysis has a history of impacting productivity to the point where developers start ignoring it and achieve little or no code improvement. Now that having secure code is non-negotiable, more people than ever are taking advantage of the many benefits that static analysis can deliver. This is a great opportunity for the industry to reacquaint itself with the technique. With a concerted effort to focus on policy and workflow management and workflow optimization, we can start off on the right foot with static analysis for security—and then continue to build on this new stable foundation to improve quality as well.

Wayne Ariola is vice president of strategy at Parasoft, which recently extended dataflow capabilities in its flagship code analysis tools.