Building Secure Software at Enterprise Scale

EXECUTIVE SUMMARY

There are innovative methods for performing static analysis of application code that results in secure, higher-quality software at a significantly lower cost and level of effort. These methods move the testing as close to the coding process as possible, providing real-time analysis in the development environment. These methods drive efficiencies by seamlessly complementing the cadence of software development. In their fully evolved state, these innovative tools provide actionable and contextual help to fix and prevent security vulnerabilities.

When compared to the traditional find-and-fix cycles that occur much later in the development lifecycle, these new methods represent a significant reduction in costs and developer resources. Equally important, use of these evolved tools greatly reduces the need for critical security resources to remediate identified defects. This frees up security personnel to address more strategic security challenges in the organization and removes them from the critical path in completing releases on time.

The benefits to the organization are abundant. Moving testing to the Integrated Development Environment (IDE) results in code that is more secure, which, in turn, reduces security risks. Development timelines are not threatened by security defects detected late in the lifecycle, decreasing the risks of missing deadlines and saving money by more effectively using scarce resources such as security analysts. These new approaches operate at enterprise scale and readily support the growing adoption of agile methodologies.

WHY DOES THIS MATTER?

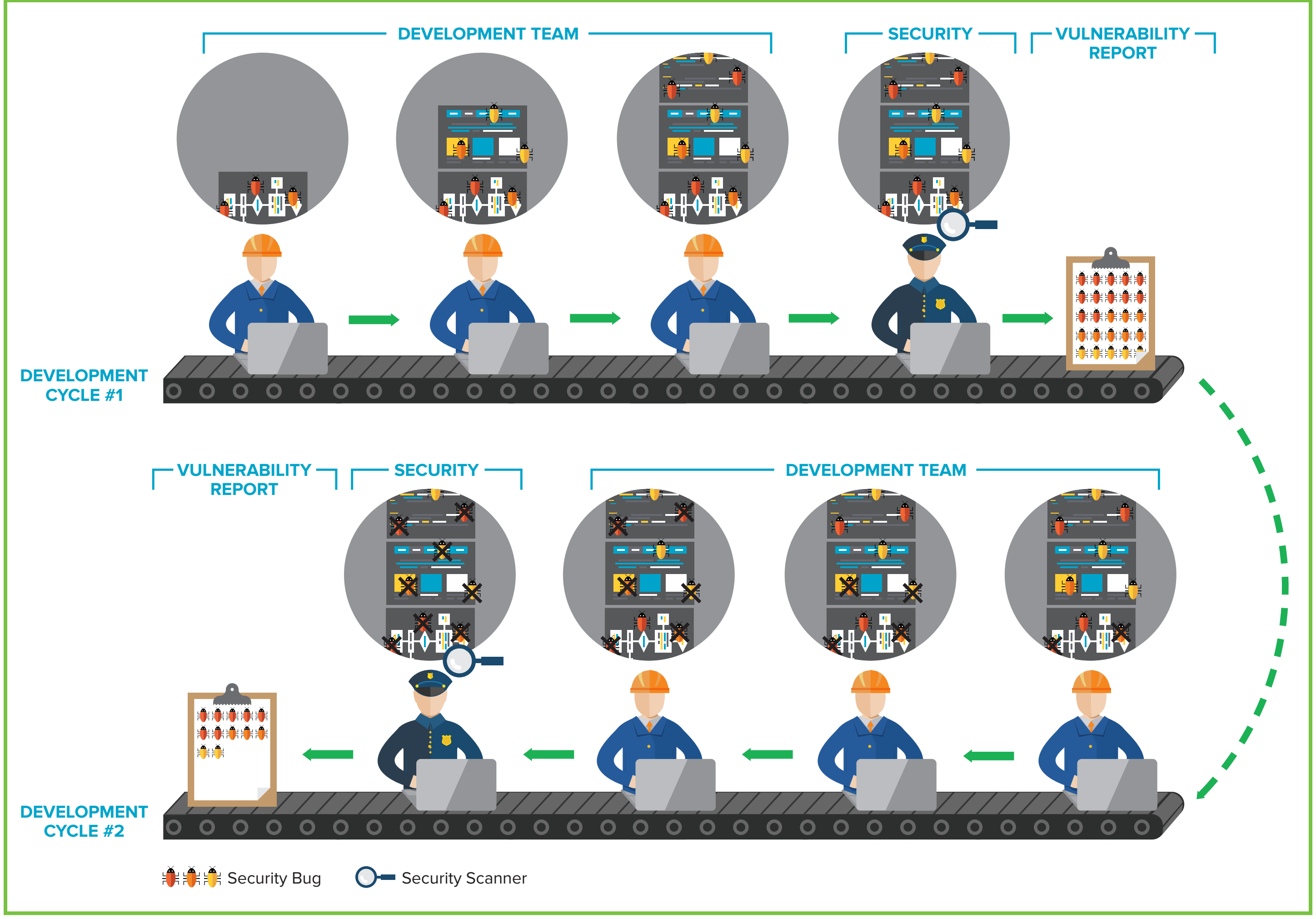
It is an immutable law of application development that the cost of finding and remediating any defect is dramatically reduced when the defect is found early in the development lifecycle – the earlier the better. Finding and fixing security vulnerabilities is a vital and necessary part of the software development process. However, the trial-and-error approach to scanning and remediating code adds a significant amount of time to the development process, which risks delaying the delivery of product upgrades and enhancements—eroding the organization's competitiveness and threatening revenue.

> "It is an immutable law of application development that the cost of finding and remediating any defect is dramatically reduced when the defect is found early in the development lifecycle – the earlier the better."

Application scanning remains a critical tool in ensuring the security of those applications. Unfortunately, because application scanning occurs so late in the development lifecycle, security defects aren't discovered until they're very expensive and time-consuming to fix. Even the costs of hardware, software, and personnel required can be staggering: organizations with comparatively modest application portfolios easily spend millions of dollars per year.

Today, application security professionals sink hours upon hours into scanning applications and working with developers to fix simple bugs rather than applying their security expertise to higher-impact, business-critical issues. The output of these scans—a lengthy report outlining thousands of issues across an entire application—requires hours of prep work by a small team of security personnel before it is ready to hand off to the development team for remediation. With little to no guidance outlining how to fix these vulnerabilities, developers are pushed into another development loop focused on remediating and retesting the application.

DISJOINTED, INEFFICIENT APPROACH TO APPLICATION DEVELOPMENT AND SECURITY



Often, a single bug can take hours to fix, since the developer is forced to go back and work on code that he or she checked in days—or weeks—ago. Occasionally, the original developers are no longer available to work on the fixes; they have been assigned to other projects or have other priorities. This results in new developers repairing someone else's code—and often unwittingly introducing new vulnerabilities.

THE SEARCH FOR ANSWERS

There are many different ways to improve application security (see **Different Approaches to Building Secure Code** side bar starting on page six), but no single approach can guarantee secure, vulnerability-free software. Human-dependent code review—like peer review—is prone to error; conversely, highly automated review is fast and efficient, but fails to detect numerous issues without clearly defined guidelines. The result is that each organization's team of security experts is overly taxed and become a barrier to shipping products, which could impact revenue.

Automated application security has fairly primitive origins. Early code review tools lived on individual developer workstations, which let security teams gather vulnerability information earlier in the development process. By all accounts, this approach was great; it effectively reduced the number of bugs and developers actually used it—but it couldn't catch everything. The tools were extremely limited in the diversity of vulnerabilities they could detect.

Application security assessment vendors pushed to expand the capabilities of their tools to identify a broader spectrum of vulnerabilities. This caused the size of security testing tools to balloon, making them impractical to run on a developer workstation. These new, industrial-strength static security assessment tools needed hours to scan a single application on a centralized server and required someone knowledgeable to manually vet, prioritize, and assign vulnerabilities to developers for repair. As a result, these tools became highly dependent on skilled security personnel to scale.

The deficiencies with these tools became increasingly problematic as adoption of agile development methods expanded. The short, continuous development sprints used by agile developers are incompatible with the labor-intensive process of finding and remediating security defects. For example, agile developers would be tasked with addressing vulnerability scan results from a sprint they completed months earlier. In some cases, the developers are given scan results from several sprints at once, leaving them to map thousands of vulnerabilities back to the exact sprint, developer, and line of code. As a result, many agile shops don't bother scanning their code after every sprint, leaving security vulnerabilities to linger in the code base for months.

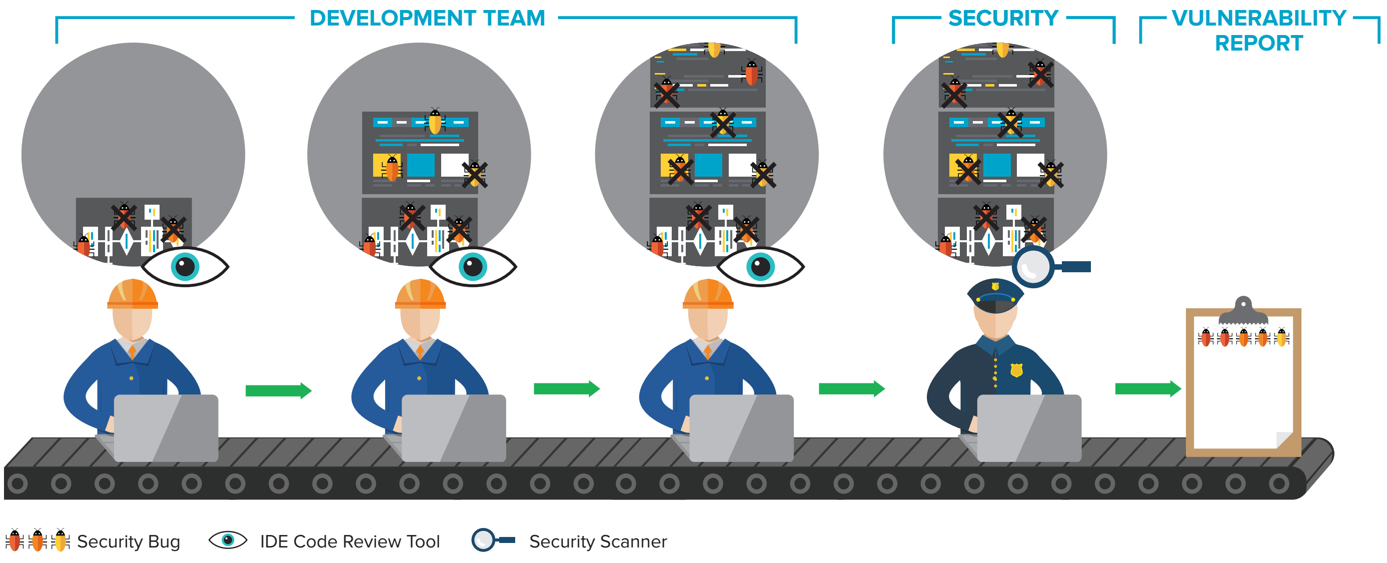
“In their fully evolved state, these innovative tools provide actionable and contextual help to fix and prevent security vulnerabilities.”

THE ANSWER EMERGES IN THE NEXT GENERATION OF IDE-BASED CODE REVIEW TOOLS

Clearly a better approach was needed that enables developers to identify vulnerabilities earlier in the lifecycle and provides actionable guidance to help developers fix the problem. By extension, the process becomes a learning tool that instructs the developer how to avoid the same problem in subsequent code, moving the process from finding and fixing to *preventing* problems. To more effectively target organizational-specific security concerns, the tool would provide the flexibility to incorporate organizational policies and guidance into the process. These possibilities are being realized in a new generation of IDE-based code review tools.

Far more advanced and lightweight than their early predecessors, IDE-based code review tools are fast enough to run behind the scenes while a developer writes code—eliminating the need for a separate, cumbersome scan. Even better, modern tools include security remediation advice *directly within the developer's coding environment*, obviating the need for oversight from the security team. The result is a cleaner, more streamlined process where developers fix a large number of frequently-occurring vulnerabilities before they ever check their code into the codebase, when it is easier and cheaper to do so.

SCALABLE, STREAMLINED APPROACH TO APPLICATION DEVELOPMENT AND SECURITY



Equipping developers with security awareness and responsibility also serves to reduce the workload of application security teams. Rather than spending hours in the trenches vetting thousands of common defects and taking an artisan approach to remediating detected defects, security personnel can spend their time working on high-value security issues. The more advanced tools enable security teams to capture organizational policies and embed them as rules in the analysis tools, enabling institutional knowledge to be integrated into the process.

SUMMARIZING THE BENEFITS

Lightweight IDE-Based Code Review tools offer many advantages and benefits to organizations of all kinds:

- **Reduce Risk.** Help developers fix security vulnerabilities before they are committed to the code base, and prevent common issues from ever being introduced.
- **Decrease Costs.** Enabling developers to fix defects in real-time is orders of magnitude less expensive than scanning completed applications for defects and entering a fix-and-retest cycle. Early detection and remediation may remove potential delays in releases and prevent patch cycles, eliminating the significant costs associated with each.
- **Increase efficiency.** These tools provide a tight feedback loop by alerting developers to each issue and immediately showing them how to fix it.
- **Expand knowledge.** Teach developers the right way to code in their development language, eliminating recurring defects ad further enhancing efficiency.

EVALUATING LIGHTWEIGHT CODE SCANNING TOOLS

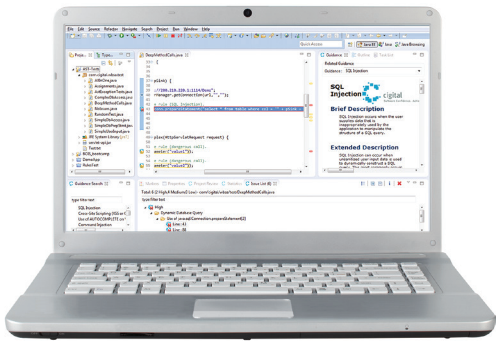
There are already several IDE-based tools on the market, though the feature sets vary widely between each. Many lack critical functionality that enables developers to work smarter and more efficiently in their native workspace. When evaluating IDE-based tools, look for the following key components:

FEATURE/FUNCTIONALITY	WHY IT IS IMPORTANT
Just-in-time scanning	Flags and explains issues as the developer codes *in the IDE*, so it's fixed before the code is checked in
Language-specific guidance	Teaches developers how to fix their code independently with pragmatic, actionable advice
Customization options	Helps organizations find defects more effectively with rules customized to their specific policies (like cryptography)
Reporting capabilities	Delivers insight into wide-spread development issues and highlights improvements over time

IDE-BASED TOOL SPOTLIGHT: CIGITAL SECUREASSIST

Cigital SecureAssist is a lightweight static analysis tool that enables enterprises to find, fix, and prevent vulnerabilities before code leaves developers' desktops. The tool flags vulnerable lines of code, explains why they are a problem, and shows developers the secure way to write the code in their own development language.

As an IDE-based tool, SecureAssist fits perfectly within an agile development workflow. By delivering remediation guidance (written by trusted security experts) while code is being created, the tool allows developers to fix vulnerabilities without slowing them down—and without the need for a cumbersome, time-consuming security scan.



SecureAssist greatly reduces the strain on security teams by eliminating the most common application security problems during the development process. Since fewer problems make it out of development, application security professionals have significantly more time to concentrate on higher-value issues.

The tool is also robust enough to satisfy the governance needs of larger organizations. SecureAssist can be modified to address organization-specific policies and procedures with custom rules and guidance. Enterprise customers can use an online reporting portal to access trend data and get insight into the most pervasive issues—and developers' improvements over time—at their organization. As patterns surface, administrators can identify gaps in security knowledge that they can incorporate into future training.

DIFFERENT APPROACHES TO BUILDING SECURE CODE

PEER REVIEW

Developers review each other's code before it is shipped.

👍 cheap, easy to implement

👎 highly dependent on developers' security knowledge; time-consuming; prone to error

SOFTWARE ASSURANCE POLICY STATEMENT

Formal policy outlines requirements for secure software development, with policy and coding guidance to achieve stated policy.

👍 easy to implement

👎 limited adoption; no enforcement; no verification of results

SECURE CODING EDUCATION

Developers take courses and attend workshops on secure coding practices.

👍 thorough; hands-on

👎 expensive and consequently not provided to all developers; not reinforced

CODE REVIEW "FACTORY," OR CENTRALIZED SCANNING

Code is scanned by a comprehensive tool. The results are triaged by small teams of security experts who then assign each issue to a developer for remediation.

👍 thorough detection of security issues

👎 doesn't work well with agile development processes; tools can be expensive to license and complex to use; fixing vulnerabilities late in the development process is costly; security teams become a bottleneck

CONCLUSION

Development teams are stuck in an onerous position. On one hand, they are under increasing pressure to turn around new code at an ever-increasing pace through approaches such as agile development. On the other, developers face the pressure of defending applications from constantly evolving threats. An effective way to overcome this challenge is to equip developers with a way to detect and remediate defects in real time when the costs and risks are lowest.

The evolution of application security assessments continues as mature software security initiatives strive for greater efficiency. Integrating lightweight secure code development tools early, within the IDE, greatly reduces the time required to find and fix issues from hours to seconds. Developers are provided actionable intelligence to fix defects without the need to involve critical security resources, freeing those resources to address the myriad of other risks facing the organization. This enables organizations to develop their applications more securely and more rapidly, while reducing costs and risks across multiple dimensions.

FOR HELP SCALING SECURE CODE REVIEW AT YOUR ORGANIZATION, CONTACT CIGITAL AT:

North America: +1 (800) 824-0022

EMEA: +44 808-189-0628

DIFFERENT APPROACHES TO BUILDING SECURE CODE CONTINUED

CODE REVIEW AS PART OF BUILD PROCESS

Lightweight static analysis tools review software as code is checked into a repository.

👍 integrates with agile workflow; can create control standards

👎 occurs after code has been submitted (long feedback loop); risk of “breaking the build” during overnight build process

CODE REVIEW IN THE INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

Security reviews conducted within the IDE to find (and fix) vulnerabilities as developers code.

👍 high developer adoption; prevents security defects from entering the code base; lightweight/fast; inexpensive (relative to centralized static analysis tools)

👎 triage and remediation help still required from security team if guidance is weak or false-positives are high; limited breadth of detectable vulnerabilities

cigital

Cigital
21351 Ridgetop Circle
Suite 400
Dulles, VA 20166
www.cigital.com