BLACK DUCK®

WHITE PAPER

FIRST LINE OF DEFENSE

DEVELOPER SECURITY TOOLS IN THE IDE

One of the challenges of implementing resilient software security is that, historically, security has been owned and managed by security teams while its implementation has been owned and managed by development teams. Security teams detect, identify, and prioritize risks for remediation, a process they undertake late in the software development life cycle (SDLC), after developers have completed the build work. But with this approach, security issues found late in the SDLC pose a problem: either the code is sent back to developers to be fixed, which could mean pushing the release date back, or software is pushed out, despite known issues, to a repo or production, with the hope that the potential risk doesn't incite a security incident.

As software development and deployment methodologies evolved and got faster, security responsibilities began to "shift left"—be performed earlier in the SDLC—and involve security, operations, and infrastructure teams. At the same time, the tools each team used to detect and mitigate risks diverged, with only tangential connections via APIs and reports. This often complicated communication and collaboration across teams and introduced noise into DevSecOps initiatives.

We all want to produce better and more secure software, and we want to do that faster than we ever have before. For developers, this means taking on more responsibility for security without sacrificing velocity, as well as learning new tools and processes that may have been prescribed by teams that are disconnected from the development process. By bringing security detection and remediation into the IDE and delivering that information to developers as they work, security-focused IDE plugins such as Black Duck Code Sight™ IDE Plug-in help build security into the code without sacrificing velocity.

There are three essential factors that must be implemented into the SDLC and DevOps workflows to accomplish this.

Risk awareness **Risk prioritization** **Risk remediation**

RISK AWARENESS

Implementing an effective risk awareness program is the first step to shifting security left and enabling developers to begin securing the software they create. There are several reasons why developers could be unaware of application security issues. These include the inconsistent nature of security education, disparity among developers' experience, the proliferation of open source and third-party components, and the late stage at which traditional security teams discover vulnerabilities. Let's break these down a little more.

Inconsistent security education

Most university computer science programs offer few, if any, security courses, leaving developers to learn secure coding practices on the job or on their own. Security education in general entails gaining an awareness of risks, and gaining an awareness of more-secure alternatives to use instead of less-secure methods. Developers can only address code quality issues in code they are writing if they are aware that the code they wrote is insecure. And once they become aware of it, they need to know of more-secure methods to accomplish the same task or function within the parameters of the application, framework, language, and other technical restrictions. Likewise, in third-party software, developers need to be able to both recognize a security issue and either fix it or replace it with a more secure component.

Disparity among developers' experience

As every business continues to become a software business, the need for developers grows. This has brought new developers into security roles despite scant security experience, and has increased the demand for developers who know a range of tools and methodologies. This wide difference in skill levels can pose a challenge for organizations that centralize security responsibilities within one team.

Proliferation of open source and third-party components

Increasingly, developers are leveraging third-party and open source components to accelerate development and benefit from the work of other developer communities. But these assets can pose a number of security issues, as they are created by developers who are subject to the same potential limitations as in-house development teams. In truth, using open source and third-party components means that developers are outsourcing aspects of application security and relegating their risk profile to the standards of another organization or developer. This obfuscates security risk awareness at the source code level, and that can often delay issue resolution or require a patchwork of code to be layered atop vulnerable components. These components also often entail adding dependencies—both known and transitive—into the built project. These dependencies are subject to the same security nuances. This is why there's been such focus on software Bills of Materials (SBOMs) in recent years. A good SBOM is a critical risk awareness tool because it catalogs software assets and can help developers find affected code and artifacts if they are alerted to a newly discovered risk.

Pro tip:

IDE plugins can detect code quality issues and open source vulnerabilities in projects as code is being written. Many even deliver remediation advice upon detection.

Late-stage security risk detection

As development and release methodologies evolve from waterfall, to agile, to DevOps and continuous integration / continuous delivery (CI/CD), security standards are undergoing similar changes. Postponing the discovery of security vulnerabilities until after assets have left the developer's hands is a relic of earlier methodologies. Modern, more rapid software development workflows require faster and earlier security review to avoid missing shipping deadlines. Late-stage discoveries force developers to return to work they may have already moved on from. This means they have to interrupt their current workflow to fix code that may no longer be the focus of their task at hand.

RISK PRIORITIZATION

When security practitioners discuss prioritization, the fundamental elements include risk severity, exploitability, exposure to the public (e.g., commercial software, external server deployment), and patch availability. There are two primary complicating factors in risk prioritization: the diverse range of application security testing (AST) tools, and the complex and often subjective task of identifying the greatest return on investment (ROI) for remediation or mitigation efforts. These factors are also influenced by the inconsistent toolsets, risk tolerances, and workload availability of individuals across teams spanning development, testing, and production operations. Let's examine these challenges more closely.

Contributor perspectives

Deciding which risks are most important to resolve can be difficult to determine because each stakeholder has their own perspectives on the matter. Assessing risk and prioritizing remediation can be subjective, and can put team members at odds with one another. Security teams often manage testing across hundreds or thousands of applications in their organizations, with each application containing potentially vulnerable components. Developers often focus on a portion of the software load that passes through security and are intimately familiar with the code, components, and structure that compose that set of projects. Operations teams take ownership of software assets in production environments, so they require clear communication with security and development teams to patch and replace vulnerable software affected by emerging threats.

Diverse AST toolsets

According to the Global State of DevSecOps 2023 report, the majority of organizations are dissatisfied with their AST tools. These tools are commonly focused on satisfying the requirements of security teams, with a secondary focus on development or DevOps teams—often via integrations and API support. Organizations experience challenges when teams implement disparate tools, configured for their specific risk tolerances and project requirements. Each tool's scanners, analysis engines, reporting, and remediation capabilities vary, exacerbating the siloed contributor perspective.

Ultimately, fast-paced DevOps workflows cannot support compliance requirements and customer demands for consistent, resilient application security when teams and tools do not function in unison. Developers need the tools and insight to detect and prioritize risks as they write and build software, so they can prevent as many potential security risks as possible downstream. Risk prioritization may be inadvertently inconsistent, but if issues are remediated earlier in the SDLC, they won't have the chance to be overlooked or undetected by downstream tools and contributors.

Pro tip:

IDE-based security plugins provide the most direct and frictionless way to catch issues early. They can highlight known vulnerabilities in open source components and their dependencies, and reveal code quality risks that create potentially exploitable weaknesses.

RISK REMEDIATION

After detecting code quality and security risks as early as possible in the SDLC, and prioritizing them based on relevant criteria (e.g., risk severity, exploit availability, application accessibility), developers bear the responsibility for remediation.

Developers must navigate complex file structures and a huge amount of code to remediate an issue. An IDE-based security tool can simplify this step, as the tool highlights the at-risk file or links to the location of the issue. For code written in-house, effective remediation is predicated on knowledge of secure coding practices. If, for example, a developer writes code that leaves software open to SQL injection, that developer must learn the secure way to write the same function in order to fix the issue. This is true for any insecure coding practice, spanning languages, frameworks, and IDEs.

Open source components and other third-party assets add a layer of complexity to remediation, effectively limiting an organization's risk posture to the security preparedness of software vendors and community contributors. The owners and maintainers of vulnerable third-party assets should incorporate a fix into their deliverables, but they first need to be aware of the security risk. When this doesn't happen, developers using these assets must either remediate the issue themselves or remove the vulnerable asset entirely.

Sometimes a fix is available in the form of a newer, more-secure software version or an analogous component with stronger security. This greatly helps developers act on the risk insight they receive from security tools. Implementing a DevSecOps program with automated and integrated systems that are easy to use and that deliver diagnostic and remediation advice right to developers is the best way to secure code without slowing development and DevOps workflows.

Pro tip:

To get a sense of how many code quality risks and security vulnerabilities developers can be responsible to fix, check out CWE.Mitre.org and CVE.org.

THE DEVSECOPS APPROACH

DevSecOps expands the collaboration between development and operations teams and integrates security teams. DevSecOps requires a change in culture, process, and tools across these core functional teams and makes security a shared responsibility. Integrating automated systems into DevOps workflows and CI/CD pipelines enables developers to perform quick security tests as they code and get remediation information without leaving the IDE. This type of security-first approach to development is key to implementing a DevSecOps program in any organization.

Automating risk detection through IDE-based security plugins or AST integrations makes it easier for development teams to code securely without impeding velocity. Black Duck Code Sight, for example, is a developer-centric security plugin that performs code analysis and open source risk analysis, known as static application security testing and software composition analysis, right from the IDE where developers work. Using IDE-based security tools like Code Sight helps developers find and fix code quality issues and security risks quickly. This helps developers ship fewer security risks and improve the security risk posture of the organization as a whole.

The resulting DevSecOps program is perhaps best evaluated by consistent software pipeline velocities and reduced time to remediation. Organizations at the leading edge often consider the reduced number of new security risks shipped to production as a metric for success, as it indicates an evolution of the security capabilities of development teams.

DEVSECOPS REQUIRES A CONCERTED PLAN

The first step to implementing a DevSecOps program is a conversation between development, security, and operations teams. Everyone should understand the role of each team in the process, and how each team measures or qualifies success at their corresponding touchpoints. They should also identify the tools and techniques used to assess and prioritize security risks, and define a path to a mutually beneficial improved state. Although certain teams may realize greater benefits, that should not interfere with the path forward that benefits the organization as a whole.

A robust DevSecOps program does not limit itself to “shift left” security practices; rather, it strives for a “shift everywhere” approach to security for sustained security postures and compliance with regulatory standards.

Companies should consider this a starting point for planning systems and controls, with the goal to exceed them as teams develop their security skills. Many organizations focused on elevating their standard for security undergo detailed assessments, such the Black Duck Maturity Action Plan (MAP). A MAP provides an actionable roadmap for security and development teams. Whether you want to move applications to the cloud, manage open source risks, improve developer security education, or build security into your SDLC or DevOps initiatives, a MAP outlines the steps to get you there.

ABOUT BLACK DUCK

Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.