BLACK DUCK®

GUIDE

FOUR STEPS TO EVOLVING DEVSECOPS AT THE SPEED OF AI

OVERVIEW

There is a critical need for DevSecOps practices to keep pace with the rapid adoption of AI in software development. But many organizations face a dilemma when it comes to AI—***security initiatives are not keeping pace with development practices***.

According to Black Duck's 2024 "Global State of DevSecOps" report, over 90% of the 1,000 organizations surveyed are already using AI to develop software. Over 20% reported that their developers are using AI to code even when official policies forbid it. That reality produces another daunting figure—26% of organizations have little or no confidence that they are adequately prepared to secure the AI-generated code they produce. The report highlights that many organizations are struggling to keep pace with AI adoption and are still in the process of putting policies and tools into place to manage the challenges posed by AI-generated code.

To harness the full power of AI while safeguarding against potential risks, organizations need an approach that enables development teams to advance their AI transformation securely through every phase of the development life cycle. This guide outlines four essential steps organizations can take to evolve their DevSecOps practices in alignment with the rapid adoption of AI in software development.

STEP 1: EMPOWER SECURE CODING IN THE IDE

A fundamental aspect of building secure software, especially when incorporating new technologies like AI, is empowering developers to address security vulnerabilities early in the development process—or to avoid introducing them entirely. A "shift left" approach, where security considerations are moved closer to the beginning of the software development life cycle, is significantly more cost-effective and less disruptive than addressing issues after deployment.

Given how quickly AI can generate code, real-time security scanning within the IDE has become even more critical to identify and address vulnerabilities that might be introduced by AI tools. By providing developers with the right tools and information within their familiar IDE, organizations can foster a culture of secure coding without hindering productivity. IDE-based security scanning is an essential first line of defense against vulnerabilities, and helps minimize a potential flood of insecure artifacts passed to security teams as AI coding assistants churn out code.

IDE security plug-ins play a crucial role by offering real-time feedback as developers—or their AI assistants—write code, allowing them to identify and fix vulnerabilities immediately. These plug-ins often feature inline vulnerability scanning that highlights potential security flaws directly within the code editor. Furthermore, they can provide fix suggestions, guiding developers on how to remediate identified issues, and even deliver detailed explanations of the vulnerabilities' potential impact. IDE security plug-ins geared toward enterprise DevSecOps also give developers clear insight into issues that can appear later in the pipeline, such as security policy violations and other issues that enter the project after leaving the developer's terminal.

STEP 2: INTEGRATE AND AUTOMATE SECURITY TESTING INTO CI/CD PIPELINES

Seamlessly integrating and automating security testing into the CI/CD pipeline is crucial to addressing the two most likely outcomes of AI-enabled development—larger volumes of code to analyze and secure, and faster code delivery cycles with abbreviated testing windows. Integrated testing ensures that security vulnerabilities are identified as quickly as they enter the project and prioritized for remediation before they can be deployed to production. Automation plays a vital role in this process by reducing manual effort, ensuring consistency in testing procedures and risk tolerance, and accelerating the feedback loop for developers.

A comprehensive security testing strategy within the CI/CD pipeline typically involves multiple types of application security testing (AST) tools, including static application security testing (SAST), dynamic application security testing (DAST), and software composition analysis (SCA) as the primary mechanisms to detect issues in proprietary code, third-party and open source components, and at production runtime.

There are several best practices to help ensure that the integration of AST tools into the CI/CD pipeline is effective and does not disrupt developer workflows. Security feedback should be integrated directly into the tools that developers use, such as their IDE, git repositories, issue management systems, and pull requests, and the feedback should provide immediate and actionable insights. It is also crucial to choose tools that return minimal false positives and that align to policies defined and managed by AppSec teams. And the tools should prioritize critical vulnerabilities or those affecting applications that handle sensitive information, to avoid overwhelming developers with irrelevant alerts.

Security analysis should be implemented at multiple stages of the software development life cycle. This should include

- **In source code management systems:** Scanning code as it is committed enables early detection of vulnerabilities and analysis of code that might enter projects through unofficial and unsecured workflows. Once detected, security teams can choose to automatically enforce security policies on code repositories, establishing a resilient safety net for any project.
- **During the build process:** SAST and SCA integrated with build workflows ensures detection of artifacts that may have eluded prior security testing. This is the most common entry point for transitive open source dependencies that might contain high-severity and critical vulnerabilities.
- **In quality assurance (QA) and production environments:** This testing should closely mirror production runtime to identify vulnerabilities in realistic conditions. To optimize efforts, this should include interactive application security testing (IAST), which can run alongside manual or automated functional testing to detect security and compliance issues that manifest at runtime. Integrating a lightweight DAST solution, which does not adversely impact performance of an application in production runtime, provides a valuable mechanism for simulating potential attack vectors and validating security standards implemented through prior architecture, assessment, and remediation.
- **In binary repositories and container registries:** Vulnerability scanning and scanning for known malware in compiled binaries can help identify risks even when source code is not available, which is often the case in commercially acquired software. Testing container registries ensures visibility into new risks from recently modified containers, even when the associated project passes muster in aggregate. Additionally, automation mechanisms allow for proactive risk mitigation, such as preventing any container that violates security policies from being promoted into production.

Although integrating security testing into the CI/CD pipeline is essential, it can introduce delays if not optimized effectively. Ideally, integrated security testing and issue management should fit naturally into existing development and DevOps workflows, without requiring contributors to interact with extraneous tools or modify their current preferred workflows. This helps minimize any negative impact on development speed and ensures that developers do not feel compelled to bypass security checks to stay on schedule.

Techniques like incremental scanning, which analyzes only the changes in code, and parallel testing, which runs multiple tests simultaneously, can help improve the performance of AST tools within the CI/CD pipeline. AST configurations and rules can be tailored to the specific needs and risk tolerance of the organization or project to ensure that the testing is relevant and focused.

STEP 3: FOSTER A "SECURITY-FIRST" MINDSET

Cultivating a strong sense of security responsibility among developers is essential for building applications securely in an AI-enabled development culture. This requires first eliminating implicit trust in AI, which has emerged as a new breed of business risk and is characterized by the notion that AI-generated code is inherently better than code composed by developers. This implicit trust that coding assistants' output will become better over time without human intervention can reduce developers' due diligence efforts during code review and increase the burden on AppSec teams during late-stage testing and triage.

Next, organizations must provide effective security training tailored to the organization's business objectives and aligned to the specific coding best practices relevant to their technology stack. This training should address two key needs to cultivate security-capable developers.

- Provide proactive, modular security risk awareness and avoidance training pertinent to each development team's projects, languages, and frameworks
- Provide clear, responsive remediation guidance associated with risks detected by AST scans and make it available within developer tools like the IDE and issue management tools (e.g., Jira)

AI-powered security resources are emerging as valuable methods for enhancing developer security capabilities. For example, AI tools can help developers focus on the most critical issues and reduce the noise from less-severe findings. And they can provide developers with context-aware guidance and suggest fixes for security vulnerabilities, essentially functioning as a reliable security counterpart to developers' AI coding assistants.

Additionally, investing in developer security training can lead to a significant reduction in the number of vulnerabilities introduced into the code, both limiting overall risk exposure and reducing the backlog of issues that security teams must review and triage. Developers who possess a strong understanding of security principles and common vulnerabilities are better equipped to write secure code from the outset as well as identify issues introduced by AI coding assistants. This helps make the development process both more efficient and more secure.

STEP 4: USE FLEXIBLE AND POLICY-DRIVEN SECURITY TESTING

To effectively manage security across multiple development pipelines, organizations need to implement flexible and policy-driven security testing practices. This involves establishing centralized security policies that can be consistently applied across different teams and projects, ensuring a baseline level of security for all applications and enabling the automated mechanisms outlined in Step 2.

Defining clear security goals and priorities based on the organization's overall business risk is also essential for focusing security efforts on the areas that matter most. Leveraging policy-as-code approaches can automate the enforcement of these security policies within the CI/CD pipeline, reducing the reliance on manual checks and ensuring consistent application.

Solutions that offer centralized visibility and control over security testing results across various projects and pipelines are critical to effectively manage and prioritize issues at the scale required for AI-assisted development. A unified platform that aggregates security findings from all security tests (e.g., SAST, SCA, IAST, DAST) provides a comprehensive view of the organization's security posture while requiring minimal security tooling or workflow variation for the AppSec teams that oversee DevSecOps initiatives. Such platforms often include features like centralized dashboards, detailed reporting capabilities, diverse scan engines, customizable and conditional automation triggers, and seamless integration with developer tools and issue-tracking systems to allow for efficient remediation.

Flexible security policies enable organizations to tailor their security testing practices to the specific risks and requirements of each application. Recognizing that not all applications carry the same level of sensitivity or risk, flexible policies allow stricter security controls for high-risk applications while maintaining agility for those with lower risk profiles. The ability to prioritize issues based on factors such as severity, exploitability, and potential business impact ensures that development teams focus their efforts on the most critical vulnerabilities first, minimizing the risk of missing software shipping deadlines due to wasted effort or misaligned priorities.

Finally, adopting a policy-driven automation approach ensures the consistent application of security standards across the entire organization. This reduces the risk of human error and supports broader compliance requirements, whether they are set by the organization to achieve internal standards for security and performance, or by external third parties like regulatory bodies or governments.

HOW CAN BLACK DUCK HELP?

The four steps outlined in this guide provide a comprehensive framework to help development teams integrate AI into their workflows securely and productively, and help security teams elevate DevSecOps initiatives to accommodate the speed and scale required by AI-enabled development. By providing secure coding resources and fix guidance in the IDE, code repositories, and build pipelines, organizations can put clear and actionable insight directly into the hands of developers. By integrating and automating security testing across CI/CD pipelines and using flexible, policy-driven security testing practices, organizations can confidently and consistently scale their AppSec efforts amid a rapidly evolving AI transformation.

Black Duck's comprehensive suite of tools and capabilities are uniquely positioned to help you with these four steps.

- **Empower secure coding in the IDE.** Code Sight™ IDE Plug-in acts as a "security spellchecker" for developers, highlighting issues as they code and providing clear fix guidance without the need to search for more information. Code Sight ensures clear visibility into the risks detected during pipeline scans, including policy violations, priority issues, and assigned remediation tasks. It also offers risk-relevant remediation guidance from Black Duck cybersecurity research teams, so developers can focus on innovation without being security experts. Additionally, Black Duck offers API-based snippet analysis for AI-generated code, detecting the licenses associated with third-party code snippets that AI coding assistants often reference or copy. This aids in compliance and prevents potential risks to your valuable intellectual property.
- **Integrate and automate security testing into your CI/CD pipelines.** Black Duck integrates seamlessly with CI/CD pipelines to automate SCA and SAST, providing comprehensive visibility into open source risks and insecure proprietary code. It also ensures proper analysis with each edit, commit, and build. Developers can use Black Duck's out-of-the-box plug-ins and automation templates for GitHub, GitLab, Azure DevOps, Jenkins, and Bitbucket to tailor testing activities to their projects and workflows while adhering to AppSec teams' risk tolerance policies.
- **Enhance developer security capabilities.** Black Duck offers developer security training, powered by Secure Code Warrior, to enhance developers' secure coding skills and provide actionable remediation guidance for detected risks. This combination of proactive and responsive education helps reduce the number of issues introduced over time and accelerate the remediation of those that do enter the project. Black Duck's AI-powered security assistant provides summaries of security risks to enhance developers' understanding of the issues, recommends patched versions of vulnerable open source components, and generates recommended code fixes for flawed proprietary code.
- **Implement flexible and policy-driven security testing.** Black Duck offers detailed policy configurations, flexible deployment options, seamless integration into developer workflows, customizable testing configurations, and scalable architecture. Black Duck Polaris™ Platform offers flexible, centralized policies for a suite of AST engines, with both on-premises and as-a-service deployment models. This makes the Polaris platform suitable for complex and global enterprise environments, even as AI tools encourage rapid scale and pipeline evolution.

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.