BLACK DUCK®

GUIDE

STRATEGIES FOR AI-POWERED SOFTWARE DEVELOPMENT

Whether embedded in a handheld device or a commercial jet, software is everywhere. But in some machines (that jet, for example), a software failure could result in injury or death. So it is crucial that developers of safety-critical software prioritize reliability and security.

Many developers have adopted AI coding assistants for the productivity benefits they deliver. However, research has shown that AI-generated code frequently includes defects and security vulnerabilities. To use these tools safely, organizations must take measures to mitigate these risks.

In this guide, we'll examine the emerging risks that can be introduced by AI coding assistants, proven mitigation strategies to address those risks, and what to look for when selecting tools to improve the reliability and security of your software.

AI-GENERATED CODE RISKS

A recent study by Princeton University, MIT, Microsoft Corp., and the University of Pennsylvania revealed that developers using GitHub Copilot already see a 26.08% increase in completed tasks. Which means, as data scientist Sahin Ahmed pointed out, using the AI coding assistant effectively turns an 8-hour workday into 10 hours of output.

However, research has also found that AI coding assistants generate a lot of incorrect code. That's because these tools are trained on large datasets that include open source and proprietary code, which contains flaws and vulnerabilities. As a result, they produce code that contains similar flaws and vulnerabilities.

A study conducted by the Center for Security and Emerging Technology found that approximately 48% of the code snippets produced by AI coding assistants contained memory-related bugs that could lead to malicious exploitation. And Bilkent University research revealed that ChatGPT, GitHub Copilot, and Amazon CodeWhisperer generated correct code only 65.2%, 46.3%, and 31.1% of the time, respectively.

There is also concern that over-reliance on AI coding assistants could erode developers' problem-solving and foundational coding skills over time. Recent Stanford University research found that developers who used AI code assistants "wrote significantly less secure code" but were "more likely to believe they wrote secure code."

Still, for many organizations, the productivity benefits of AI coding assistants will outweigh the potential risks—especially as these tools evolve and become more effective. So, it is critical to put measures in place to minimize or eliminate the risks that AI coding assistants can introduce.

MITIGATING RISKS TO SOFTWARE SAFETY AND RELIABILITY

In 2024, a misconfigured update to a CrowdStrike Falcon sensor caused 8.5 million Windows devices to crash. This outage was not caused by AI-generated code, but it demonstrates how a code flaw can cause widespread damage. The incident knocked out safety-critical applications in industries including transportation, healthcare, financial services, and more. Losses from the incident are estimated to be at least $10 billion worldwide.

Safety-critical software—the software used in safety-related systems—has the potential to cause harm or damage if it fails or malfunctions. Code flaws in safety-critical software can have severe consequences, including loss of life, financial damage, and environmental disasters. Robust testing and validation procedures are vital to ensuring the reliability and safety of these applications. This includes evaluating code and addressing potential vulnerabilities before they are deployed in critical systems.

REGULATORY COMPLIANCE LEGISLATION AND STANDARDS

Regulatory guidelines and standards provide a framework for organizations to manage risk, protect sensitive data, and maintain the trust of their stakeholders. They often cover areas such as functional safety, data protection, cybersecurity, and financial reporting. Guidelines and standards are typically enforced by government agencies or industry regulatory bodies. Noncompliance can result in significant penalties, reputational damage, and financial losses.

Automated testing tools can help organizations address code defects and security vulnerabilities early in the software development life cycle. They allow organizations to track progress toward achieving compliance, prioritize issues, and report progress to leadership.

WHAT TO LOOK FOR WHEN EVALUATING TOOLS

The right testing tools can help identify vulnerabilities, ensure compliance with regulatory requirements, and improve the overall quality of software, whether written by human developers or AI coding assistants. When selecting a tool, organizations should look for several key features that can significantly impact the effectiveness and efficiency of their development process.

Integration into Developer Workflows

The best security tools seamlessly integrate into existing developer workflows. Tight integration ensures that testing and security practices are adopted without causing significant disruptions or slowdowns in the development process.

Tools like Coverity® Static Analysis can be integrated into popular development environments, version control systems, and continuous integration/continuous deployment pipelines, making it easier to adopt security practices without disrupting the development workflow.

SAST for Multiple Languages

When choosing a static application security testing (SAST) solution, organizations should ensure that it supports the programming languages and frameworks used in their software, as well as the coding standards that their software must adhere to. The ideal solution will perform comprehensive static analysis across a variety of programming languages and leverage insights from common frameworks to improve the accuracy of results.

Coverity supports more than 20 programming languages and 200 application frameworks. It scans source code for security flaws, coding errors, and compliance issues, ensuring that all parts of the codebase are thoroughly vetted.

SBOM Creation and Software License Tracking

A Software Bills of Materials (SBOM)—a complete inventory of everything in a codebase—is vital for gaining visibility into the components and dependencies within your software, including those introduced by AI. Creating an SBOM is the best way to be certain that third-party and open source software used in your projects complies with legal, security, and consumer requirements.

The SBOM creation tools in Black Duck® SCA can automatically generate and maintain SBOMs, providing detailed information about each component, including version numbers, licenses, and known vulnerabilities.

Unknown Vulnerability Identification Using Fuzzing

Fuzz testing is a technique that simulates hacker behavior to identify unknown defects and vulnerabilities in software. While SAST tools rely on rules-based analysis to identify the types of defects and vulnerabilities that are well-known to developers and security professionals, fuzz testing tools can identify unknown issues that stem from vulnerable code inadvertently generated or inserted by an AI coding assistant. It does this by generating and sending unexpected or random data to the software, revealing weaknesses that could cause the software to crash or be exploited by malicious actors.

By leveraging Defensics® Fuzzing fuzz testing capabilities, organizations can complement their static analysis and uncover issues that might otherwise remain undetected.

Reducing Noise and Prioritizing Issues

Many static analysis tools require users to sort through endless false positives and low-priority issues. Coverity can help reduce this noise by filtering out irrelevant findings and prioritizing issues based on industry standards and best practices. This ensures that developers can focus on the most critical defects and vulnerabilities, improving the overall quality of the software.

By carefully evaluating these features, organizations can select the right tools to enhance the reliability and security of their software, ultimately leading to more robust and trustworthy products.

CONCLUSION

Organizations must implement effective strategies and comprehensive testing to address quality and security issues in their software, including those introduced by AI coding assistants. Adopting a multifaceted approach to risk management, which includes automated testing, compliance and governance expertise, and secure development practices, is essential.

By integrating with existing development workflows, automating testing to find defects as early as possible, and prioritizing issues based on coding standards, organizations can produce high-quality, compliant software at the pace that their business demands.

Ultimately, prioritizing software reliability and security is essential for long-term success. By adopting the strategies outlined in this guide, organizations can minimize risks and ensure the quality and reliability of their products.

BLACK DUCK®

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.