Agile software development models have grown in popularity in recent years. It’s easy to see why. Agile development practices help organizations prototype, build, test, and deploy code faster and more flexibly than legacy waterfall methods. In particular, agile differs from traditional models by integrating multiple stakeholders into the development process.

Unlike waterfall models, in which requirements are passed to architects and designs are passed to developers, agile practices unite product management, architects, developers, and testers into a single team with responsibility for quick iterations, or “sprints.” This practice enables agile development teams to react quickly to changing requirements and deliver products that address customer needs almost as they emerge.

Using open source software as building blocks is popular with agile teams for good reason. Open source components supply needed functionality without requiring developers to write new code, and thousands of open source components have proven effective for delivering performance and functionality. This mix of availability and functionality forms a solid foundation for agile projects, and it supports the incremental practices that distinguish agile development.

UNDERSTANDING OPEN SOURCE RISKS

Using open source is not without risk, however. Even projects that are well-curated and vetted by project community members can exhibit security vulnerabilities. Components from well-maintained projects are unlikely to include common vulnerabilities such as those found by automated tools. However, they may be subject to the often-subtle, sometimes serious vulnerabilities that are discovered and disclosed almost daily by security researchers.
A simple strategy for agile development teams to mitigate security risks from open source software entails selecting open source components without known vulnerabilities, and continuing to monitor those components throughout the application lifecycle.

OPEN SOURCE DISCOVERY AND SELECTION
Open source components provide critical functionality to all types of software development, and especially to agile projects. In particular, agile efforts benefit from the use of open source application frameworks, utility libraries, and functions for authentication and encryption. When gathering project requirements, teams should assess the criticality of the application and whether and how it is exposed to end users inside and outside the corporate firewall. This assessment lets agile teams determine the level of security required. Internal applications that do not process sensitive information will likely have lower security thresholds, while Internet-facing applications that manage sensitive customer information (such as financial or health information) warrant greater scrutiny.

THE AGILE SOFTWARE DEVELOPMENT ENVIRONMENT: WHY IT PRESENTS SECURITY TESTING CHALLENGES
While developing and adopting an open source usage policy constitutes a good and necessary first step, actual policy enforcement is required to benefit from these efforts. For example, a sensible and easily-implemented policy is to use the most recent version of an open source project (e.g., libraries or middleware) whenever possible. However, even with such a policy in place, developers may inadvertently use older, deprecated versions, often exhibiting a range of vulnerabilities.

The agile environment can present particular challenges to traditional security testing. Static analysis, for example, can often take many hours to process a complex application, making evaluation of analyses difficult prior to subsequent builds. Agile teams need tools that integrate with their continuous integration environment, without hampering quick “edit/build/test” cycles.
DEALING WITH A CHANGING THREAT SPACE
Security testing typically ends when a product is released. However, security professionals understand that the threat environment is constantly evolving. According to the Open Source Vulnerability Database, more than 4,000 new vulnerabilities are disclosed in open source projects each year. These represent new threats to deployed applications, and public exploits are often available.

As open source code proliferates throughout multiple applications, organizations struggle with monitoring the ongoing security of each individual component. And while it’s theoretically possible to manage open source usage and monitor vulnerabilities via spreadsheets and other manual techniques, it’s next to impossible to practice comprehensive and timely open source vulnerability analysis without the use of automation.

To help companies address the ever-changing threat landscape, Black Duck’s solutions constantly monitor multiple public and private sources for reports of new security vulnerabilities in open source software. Through Black Duck tools and customer subscriptions, Black Duck supplies agile teams with timely alerts of new vulnerabilities, descriptions of the scope and severity of those vulnerabilities, and guidance on prioritizing and implementing remediation.

PROVIDING AGILE TEAMS WITH RESPONSIBILITY FOR SECURITY, AND THE RIGHT TOOLS
Both traditional waterfall and agile-based continuous development and integration disciplines look to Black Duck to support security and open source management policies by integrating both threat detection and policy enforcement into the build server architecture. Black Duck’s rapid scanning tools provide reports in minutes, detailing risks associated with all open source code, and deviations from open source usage policies in a given build. By receiving reports in-cycle, agile teams can react quickly to remediate issues, often simply by using updated libraries, or building compensating controls.

THE BLACK DUCK SOLUTION: MEETING THE REQUIREMENTS OF AGILE TEAMS
Open source software and agile development practices bring many benefits to organizations. Avoiding open source components with known vulnerabilities is a simple strategy to improve security. To be effective, however, testing tools must meet the requirements of agile teams; delivering quick, accurate, and actionable results. By integrating Black Duck into the build process, organizations can establish, and enforce, reasonable open source policies without affecting the productivity of agile teams.
LEARN MORE: TRY THE BLACK DUCK HUB FOR FREE

Black Duck is known for its high-performance scanning tools, such as the Black Duck Hub. The Black Duck Hub scans code bases to identify all the open source code in use, including exact versions of open source project code being deployed. The Black Duck® KnowledgeBaseTM (KB), the industry’s most comprehensive compendium of open source project profiles (delivered through the Black Duck Hub) provides up-to-date information on the licenses, security status, and operational risks associated with more than one million open source projects.

The Black Duck Hub allows agile teams, including product management, software architects, and developers, to make informed decisions about open source projects under consideration, including functionality, ease of use, and security, while designing solutions. Visit the Black Duck website today to start identifying and remediating open source vulnerabilities with a free, 14-day trial of the Black Duck Hub.

ABOUT BLACK DUCK SOFTWARE
Organizations worldwide use Black Duck Software’s industry-leading products to automate the processes of securing and managing open source software, eliminating the pain related to security vulnerabilities, license compliance and operational risk. Black Duck is headquartered in Burlington, MA, and has offices in San Jose, CA, Vancouver, London, Belfast, Frankfurt, Hong Kong, Tokyo, Seoul and Beijing. For more information, visit www.blackducksoftware.com.

CONTACT
To learn more, please contact: sales@blackducksoftware.com or +1 781.891.5100
Additional information is available at: www.blackducksoftware.com