BLACKDUCK

KNOW YOUR CODE

Don’t Get Blindsided by Open Source Security Risks During Development

EXECUTIVE SUMMARY

Application security is a strategic imperative for organizations developing internal and publicfacing software. Exploits of software security vulnerabilities can result in loss of customer or company information, disruption of business operations, damage to public image, regulatory penalties, and costly litigation.

Adding to the management challenge, the software development lifecycle (SDLC) is increasingly complex. Demands for agility and faster time-to-market, distributed development teams, and rapidly evolving languages and technologies are all contributing factors.

To remain competitive, development teams increasingly rely on open source software – cost effective, reusable software building blocks created and maintained by global communities of developers.

OPEN SOURCE SECURITY RISKS

THE SECURITY RISKS OF OPEN SOURCE

While open source is essential in today's development environment because of economic and time-to-market advantages, its use creates security risks. Unfortunately most organizations lack the visibility and control to address those risks.There are four key areas of open source security risk:

- Hidden Entry – Because developers have easy access to open source, it often enters an organization under the radar. Results from Black Duck On-Demand Audits show that 98 percent of organizations have open source that they are not aware of in their applications. As a result, the apps carry hidden security risks.

- Rich Target – Like businesses, hackers pay attention to ROI. Because use of open source projects is broad, a single vulnerability has a high rate of return, allowing exploits of many websites and applications. At the same time, easy access to the source code as well as the exploits (for examples, just search for "OpenSSL exploit" on YouTube), means uncovering and exploiting vulnerabilities is a low cost venture.
- Varying Quality –Because most projects have no specific team tasked with ensuring quality, it can vary widely from project to project, and even version to version, within a project.
- Moving Target – Even when incorporating a high-quality open source project and version into an application, new known security vulnerabilities often surface after release of the application. Most organizations updated to a "safe" version of OpenSSL after the Heartbleed vulnerability discovery, but an additional 40 vulnerabilities have been found since then. Unless teams are diligent in inventorying and monitoring their open source components, new security vulnerabilities can blindside them.

MANAGING OPEN SOURCE SECURITY RISKS THROUGHOUT THE SDLC

While many organizations use static analysis and dynamic analysis application security testing tools to uncover vulnerabilities in their custom- developed code, they need additional tools to gain visibility into, and control of, their open source to reduce security risk.

Organizations need to know:

- What open source components are in use or planned for use
- What known security vulnerabilities are in their open source
- What is their complete list of open source projects/versions approved for use

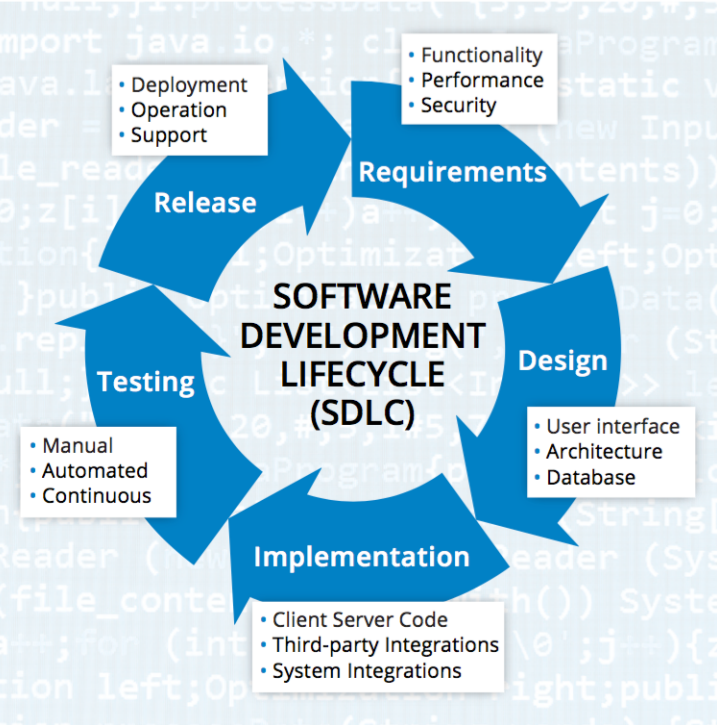
Organizations can be proactive in mitigating risks from open source. This best practices guide details how organizations can identify potential open source vulnerability issues and deal with them at each phase of the SDLC.

OPEN SOURCE AND THE SDLC

While there are numerous development methodologies, virtually all comprise at least five phases.

Organizations often identify and select third- party software and libraries (commercial and open source) during the Design phase and establish a level of visibility and control. However, once implementation starts, developers add open source projects as needed to help them complete the software. Unless the right controls are in place at each phase of the development lifecycle, it's difficult for organizations to identify and track open source components.

The Requirements Phase: Defining the Risks



In the requirements phase, organizations should define the criticality of the application, its risk profile (meaning the level of exposure to, and the ramifications of, any exploit), and the measures required to mitigate risks. Typically, this includes non-functional requirements and risk assessments. One deliverable from this phase should be a policy specifying the characteristics of the open source components acceptable for use in a particular application. The three primary parameters are security, license, and operational risk.

While open source is not less secure than commercial software, what makes open source security management challenging is the availability of older versions of components, many of which may contain previously disclosed vulnerabilities. In some cases, the simple presence of a vulnerability does not preclude the component's use (depending on how that component is used), but an organization's policy should make this clear for each class of applications. More mature organizations may also consider "vulnerability density," or the historical prevalence of security issues in the project.

Older versions also present a higher risk for security vulnerabilities simply because there are fewer developers actively using, testing, and fixing them. Open source use policies should establish rules for use of current vs. older versions. Moreover, if the open source component is critical to an application, and especially if the anticipated life of the product is long, organizations should consider the maturity and vibrancy of the community. Metrics for this can include the number of active community members committing code and the number of commits per year. Organizations may also want to ensure they are using the most current release of a project.

The Design Phase: Vulnerability Prevention

In the design phase, vulnerability prevention is the goal. Identifying preventable risk at this point allows remediation at the lowest possible cost. It is also when application security teams utilize threat modeling to assess how an attacker might compromise the application, and attack-surface analysis to minimize attack vectors.

However, this is also the time to avoid inadvertently building vulnerabilities into the code. While development teams are correctly concerned with design flaws that may result in security issues, from an open source security standpoint, the goal is to ensure that selected components meet the stated open source policies. Companies should include open source selection and

policy enforcement as part of the responsibilities of an architectural review board. They should also develop a proposed open source inventory (or bill of materials) listing all of the components/versions to be included in the application.

The Implementation Phase: Policy Enforcement

Open source enters applications in a variety of ways; from approved libraries, outsourced development, shared in-house developed libraries, and/or commercial software that includes open source components.

During this phase, organizations need processes to ensure they build software in accordance with the parameters, including those for security, defined during requirements and design. In addition to source code reviews and unit testing practices, teams should confirm that only approved open source components – and the specific version of those components – are used.

Software Composition Analysis (SCA) is a process by which a scan of the source code for an application in development identifies and inventories open source components and versions in the application. This is critical as it provides an automated way to validate conformance with the approved bill of materials and detect hidden open source. It is also important, especially for larger applications, to be able to map the list of identified open source components and versions to known security vulnerabilities, including those registered in the NIST National Vulnerabilities Database (NVD). SCA provides the safety net to catch exceptions that make it past upstream processes.

Due to the complexity of code scanning, as well as the large number of open source projects and vulnerabilities to consider, SCA requires use of an automated solution such as the Black Duck Hub.

The Test Phase: Security Verification

In many organizations, security testing using static and dynamic tools occurs at this point. While these are important tools for the discovery of unknown security vulnerabilities in custom application code, they often miss vulnerabilities that enter the application through open source.

For open source security, the goal is to enforce the open source component policies for the application. As in the implementation phase, development and security teams will want to use SCA to itemize the open source components used, correlate this with the approved bill of materials, identify any discrepancies, and either approve or remediate the discrepancies.

Additionally, it's wise to check the components in the bill of materials again for any vulnerabilities disclosed since the last time the analysis was performed.

The Release Phase: Threat Monitoring

Security testing typically stops after deployment or shipment as security teams shift focus to the next release. The thinking here is that because the code is not changing, security testing will not reveal any new information.

While this may be true for proprietary code, open source components are different. While the code may not change, the threat environment changes constantly with the disclosure of new vulnerabilities. Each year, more than 4,000 new vulnerabilities are disclosed in open source software. These vulnerabilities are known to attackers, and often include proof of the vulnerability in the form of a test exploit.

Organizations require formal processes to ensure continuous monitoring for newly emerging vulnerabilities among the open source projects they use. When discovered, they can map vulnerabilities to each application's bill of materials for remediation (typically accomplished by updating to a new version of the component).

KNOW YOUR CODE

Open source is an essential component in the development of today's software applications and services, and there is no sign of this changing in the future. Threats to applications built on open source will also continue to increase. To realize the benefits of open source without being blindsided by security vulnerabilities, organizations need to know what is going into their applications. The best way to do this is to use processes and tools that provide the visibility into application composition that enables control over the use of open source throughout the SDLC.

LEARN MORE: TRY THE BLACK DUB HUB FREE FOR 14 DAYS

Do you know what's in your code? You might be surprised.

With the Black Duck Hub you can scan your applications and containers to identify the open source projects and versions they are using, even if your team has modified them. Leveraging the Black Duck® KnowledgeBase™, the industry's most comprehensive registry of open source projects, the Hub gives you deep insights into these open source projects including known vulnerabilities, license requirements, and project/community activity. Plus, it alerts you when any new vulnerabilities are identified for those projects and gives you tools to track and manage remediation activities.

Find out what's in your code. Try the Black Duck Hub free for 14 days.

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, open source license compliance and operational risk. Black Duck is headquartered in Burlington, MA, and has offices in San Jose, CA, London, 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