EBOOK

Contrast SECURITY

The Devsecops Guide To Managing Open-Source Risk

Executive Overview

Open source has become critical to software development by accelerating time to market while reducing operating costs. At the same time, open-source software (OSS) components can introduce security vulnerabilities, licensing issues, and DevOps workflow challenges. Successfully managing OSS increasingly depends on automated application security (AppSec) processes. Automation helps organizations track all the open-source components in use, identify any associated risks, and enable effective mitigation actions so that teams can safely use open source without obstructing development and delivery.

This eBook examines these issues and how defining actionable policies, gaining comprehensive visibility of OSS components, and instituting pipeline controls can improve OSS vulnerability remediation without impacting DevOps objectives.

> “Oss is practically everywhere—embraced by major corporations, including walmart, Jpmorgan chase, and even microsoft.[1]

[1] “The WIRED Guide to Open Source Software,” WIRED, April 24, 2019.

Table of contents

01 Open-Source Components Introduce Rapidly Growing Risks

02 Oss Management Starts with Policies

03 Establishing Continuous Visibility

04 Embedding and Enforcing Controls by Pipeline

05 Ensuring Fast and Effective Remediation

06 Software Security is Ephemeral –Monitor, Monitor, Monitor

01

Open-Source Components Introduce Rapidly Growing Risks

Software today is often built from as much as 90% open-source code—including hundreds of discrete libraries in a single application.[2] While OSS can help DevOps teams save time and money, unmanaged use of open source introduces significant risks—both in terms of licensing complexities and security.

Open-source vulnerabilities typically stem from poorly written code that leave gaps, which attackers can use to carry out malicious activities—such as extracting sensitive data or damaging a system.[3] And with thousands of new vulnerabilities being discovered each year, it becomes critical for organizations to monitor sources continuously for information on new vulnerabilities and then dynamically map those to the components used in the projects and environments where these components are run.

> Use of open-source code by developers grew at 40% from 2018 to 2019 and will Continue (though tapering down to 14% by 2023).[4]

[2] "The Hidden Vulnerabilities of Open Source Software," Harvard Business School, February 24, 2020.
[3] "The Dangers of Open-Source Vulnerabilities, and What You Can Do About It," Security Today, August 19, 2019.
[4] "The Application Security Market Will Exceed $7 Billion By 2023," Forrester, October 4, 2018.

02

Oss Management
Starts with Policies

The first step in managing open source is to define the characteristics of all components allowed in an application. Open-source policies and procedures should provide guidance as to the proper management of OSS components, including which type of OSS licensing is permitted, which type of components should be used, when vulnerabilities should be patched, and how to prioritize vulnerabilities.[5] Organizations that are unsure of their obligations under license can experience problems with intellectual property rights or monetary losses.[6]

To mitigate license risk, organizations must determine which licenses are acceptable by use case and environment. In software that will be distributed externally, statically linked libraries issued under a general public license (GPL) should be prohibited in order to avoid compliance problems. However, that same license for components could be allowable for an internal-only application.

While risk and legal team policies need to be mindful of license usage, AppSec team policies need to be cognizant of vulnerabilities. But a component with even a "high" severity vulnerability may be acceptable in an application that manages data that is neither critical nor sensitive and that has a limited attack surface. But according to policy, the same severe vulnerability would obviously need to be unacceptable in a publicly facing application that manages credit card data. Policies may also dictate the expected remediation time frames that development teams must adhere to when vulnerable components are identified in their projects.

> "While the benefits of oss are clear, it is also clear that oss can pose significant legal risks that must be addressed. The best way to manage these risks is to have a clearly written and enforced oss policy.[7]

[5] "How to Make Your CSO Happy with Your Open Source Components," CPO Magazine, August 28, 2019.
[6] "The Risks and Potential Impacts Associated with Open Source," DevOps.com, January 27, 2020.
[7] "Open Source Software Policies—Why You Need Them And What They Should Include," National Law Review, June 18, 2019.

03

Establishing Continuous Visibility

omprehensive visibility is critical to securing testing, development, and production of applications.[8] A software bill of materials (SBoM) is a definitive list of all the components (including OSS) used in an application. Like a parts list in a manufacturing environment, it provides a simple way to inventory and locate vulnerable components when necessary. Integrating open-source discovery into the development process allows teams to ensure that all direct and transitive dependencies (viz., dependencies-of-dependencies) are accurately identified and added to the inventory. The result is an SBoM against which organizations can track license, quality, and security variables.

SBoMs have the additional benefit of helping to standardize components. A rollup of the various SBoMs provides an overall inventory of all OSS elements used across the organization. This process typically reveals multiple versions of the same component (often in the same application) and multiple libraries with similar functionality. Standardizing on a limited set of components helps the development team simplify software maintenance.

Beyond automating the creation of an SBoM, a critical aspect of maintaining an effective inventory is to ensure that it accurately and dynamically represents the relationships between components, applications, and servers—so that DevOps organizations always know what is deployed, where each component resides, and exactly what needs to be secured.

> “According to gartner, one of the first steps to improving software security is ensuring that an sbom exists for every software application.[9]

[8] “The Six Pillars of DevSecOps,” Cloud Security Alliance, August 7, 2019.

[9] “Gartner: The Crucial Role of Open Source Software License Compliance,” ITAM Channel, December 23, 2019.

Once an automated SBoM is built, it should be mapped to a reliable knowledge base of license, quality, and security data. For example, the National Institute of Standards and Technology (NIST) sponsors the National Vulnerability Database (NVD)—a public repository for information on software vulnerabilities, including those in open-source software.[10] This searchable database also includes descriptions and scoring for Common Vulnerabilities and Exposures (CVE).

While the NVD is a good resource, a robust strategy requires that organizations monitor additional primary sources such as ecosystem security advisories and project repositories for updates addressing security and quality issues. Aggregating disclosures from all these sources provides for a more comprehensive understanding of the risks associated with the use of open-source components and ensures early alerts on a larger set of critical vulnerabilities.

> To maintain a healthy codebase, organizations should keep an eye on the oss components they have introduced into their software. Visibility gives security Experts the opportunity to respond to security events in a timely manner.[11]

[10] "National Vulnerability Database," NIST, accessed April 15, 2020.
[11] "National Vulnerability Database," NIST, accessed April 15, 2020.

04

Embedding and Enforcing Controls by Pipeline

The next step for securing use of OSS in DevOps environments is to embed automated controls in continuous integration/continuous deployment (CI/CD) processes. Organizations, in turn, can leverage this intelligence to alert developers and security staff of any risks detected and then automatically enforce predefined policies by pipeline.

When properly operationalized, an open-source management solution can automatically analyze all dependencies in a project. If vulnerable components are detected in a build, an automated policy check should trigger a post-build action failing or mark the build as unstable based on set parameters.

Simultaneously, a Slack notification would be sent to the appropriate team with contextual information regarding the component in question, the policy violation, mitigation guidance, and so forth. Subsequently, a ticket would automatically be created in the team's issue-tracking system (e.g., Jira) to report on subsequent remediation work.

This is just one basic example to demonstrate how an automated workflow can be integrated into a team's existing process and toolchain. Regardless of the specific process and tooling an organization has in place, the goal should always remain the same: deliver immediate and accurate feedback to developers so that they can take direct action to keep the application secure and functional.

> Oss elements do not pass the same quality and standards checks as proprietary code. Unless each os component is evaluated before implementation, it is easy to incorporate low-quality code containing vulnerabilities, lowering the overall quality of the code.[12]

[12] "5 Best Practices for Managing Open-Source Components," DevOps.com, September 11, 2019.

05

Ensuring Fast and Effective Remediation

An essential part of an effective open-source risk management program is the ability to deliver the quickest possible turnaround for resolving issues once they emerge. This is crucial, since attackers may have free and almost immediate access to exploit kits after a CVE is disclosed. In the case of zero-day attacks against OSS components in production applications, organizations are at immediate risk for the duration of the vulnerability exposure window.

To put this in perspective, it can take teams months to apply a fix and push it to production.[13] With cyber criminals often launching attacks on newly exposed vulnerabilities in hours or days, a different approach to protection is needed. It requires an AppSec solution that can immediately protect against exploitation of open-source vulnerabilities, which provides time for development teams to resolve issues based on prioritized risk. Such a solution is comprised of two components:

1. Runtime protection. The solution must be designed to continuously monitor production applications and automatically block attacks on vulnerable OSS components to prevent exploitation at runtime. This is achievable with runtime application self-protection (RASP), which acts as an immediate compensating control by detecting and blocking attacks against any actual vulnerabilities (including OSS) from within the application itself. RASP runs security checks continuously and responds to live attacks by terminating the bad actor's session and alerting the InfoSec team of the attack.[14]

2. Runtime usage analysis. Given that an application typically includes hundreds of open-source components, it is likely that many issues will be detected. Often the number of potential OSS problems grows so high that developers become overwhelmed.

   As a result, organizations must be able to analyze runtime usage to directly observe and measure the behavior of the running application. This analysis enables effective prioritization of vulnerable components that are actually used by the application, and deprioritization of those that are not. Other security approaches that rely on static code analysis to guess if a component might be used by the application are plagued with false positives and false negatives. These inaccuracies create noise that interferes with effective and automatic prioritization of vulnerabilities.[15]

License compliance risk should be addressed according to organizational policies. This often requires replacing any problematic open-source components with those that offer similar functionality issued under an approved license.

> "As much as 25% of a security analyst's time is spent chasing false positives—sifting through erroneous security alerts or false indicators of confidence—before being able to tackle real findings.[16]

[13] "Why Fixing Security Vulnerabilities Is Not That Simple," Security Intelligence, October 1, 2019.
[14] "Test and Identify Security Vulnerabilities in Applications," Morgan Franklin, March 12, 2020.
[15] "From my Gartner Blog – Considering Remediation Approaches For Vulnerability Prioritization," Security Boulevard, May 2, 2019.
[16] "Every Hour SOCs Run, 15 Minutes Are Wasted on False Positives," Security Boulevard, September 2, 2019.

Contrast SECURITY

06

Software Security is Ephemeral —Monitor, Monitor, Monitor

Security is not a fixed state. Even if an application is released without any known risks (viz., CVEs), new OSS vulnerabilities are disclosed every day. These must also be mapped to existing SBoMs and monitored in production. Hackers are constantly watching for new vulnerabilities; organizations must do the same.

As a result, organizations must automate all activities discussed above. It starts with creating an SBoM and cross-referencing it with a robust knowledge base to detect risk. The AppSec solution must then enforce policies across pipelines to flag violating components while continuously streaming actionable feedback back to security and development teams. That leads to effectively prioritizing findings and automatically protecting production applications where needed.

> Trying to keep track of thousands of vulnerabilities manually has become an impossible task for humans alone. Automation tools can help you create a vulnerability inventory, keep track of the vulnerabilities, and prioritize remediation.[19]

[17] "How to Make Your CSO Happy with Your Open Source Components," CPO Magazine, August 28, 2019.

Contrast Security provides the industry's most modern and comprehensive Application Security Platform, removing security roadblocks inefficiencies and empowering enterprises to write and release secure application code faster. Embedding code analysis and attack prevention directly into software with instrumentation, the Contrast platform automatically detects vulnerabilities while developers write code, eliminates false positives, and provides context-specific how-to-fix guidance for easy and fast vulnerability remediation. Doing so enables application and development teams to collaborate more effectively and to innovate faster while accelerating digital transformation initiatives. This is why a growing number of the world's largest private and public sector organizations rely on Contrast to secure their applications in development and extend protection in production.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com