GUIDE

The DIY Guide to Open Source Vulnerability Management

Table of contents

According to SAP, more than 80% of all cyberattacks are happening on the application layer,[1] specifically targeting software applications rather than the network.

Hackers take the easiest path when determining exploits and choose applications that offer the best attack surface opportunities. Those opportunities are generally created by unpatched or outdated software.

For example, Heartbleed, a dangerous security flaw, critically exposes OpenSSL, an open source project used in hundreds of thousands of applications that need to secure communications over computer networks against eavesdropping. Yet 56% of all OpenSSL versions that Cisco Security Research examined in its 2015 security report[2] were still vulnerable to Heartbleed, more than two years after the Heartbleed vulnerability was first disclosed and a patched version issued.

This illustrates the difficulty organizations have in inventorying and managing open source components rather than a lack of security diligence. Without a comprehensive list of open source components in use, it is nearly impossible for any organization to identify specific applications that use vulnerable components.

Open source is in nearly every application

Open source use has increased dramatically in recent years, with open source components composing as much as 50% of any given application.

Yet many companies remain ill-informed about the amount of open source they use, and blind to the vulnerabilities that may be in that open source. And open source vulnerabilities abound: Reports show that, on average, 67% of commercial applications contain open source security vulnerabilities.[3] The number of open source vulnerabilities the National Vulnerability Database (NVD) has reported since 2014 now exceeds 6,000.

Because of their ubiquity, attackers see popular open source components as a target-rich environment. To return to OpenSSL as an example, more than 66% of active sites on the web use OpenSSL, as well as email servers (SMTP, POP, and IMAP protocols), chat servers (XMPP protocol), virtual private networks (SSL VPNs), network appliances, and a wide variety of client-side software.

To win the race against hackers, you need to manage open source risk

The fact that open source is an essential element in most applications is not lost upon hackers, who also know that they can often access public information on known open source vulnerabilities along with detailed information on how to exploit those vulnerabilities. When a new open source vulnerability is reported, the race is on between you and hackers.

- Will you know if you're using an open source component with a known vulnerability?
- Will you know if that vulnerability exposes your organization to hackers?
- Will you know how prevalent that open source component is in your organization's internal and public-facing applications?
- Will you know how to effectively manage and mitigate any risk exposed by that vulnerability?

This guide looks at the processes you should consider putting in place to manage your open source use and strategies to manage open source risks. It will walk you through the specific goals and tasks you'll need to implement as part of your strategy and processes and highlight some of the challenges you may face along the way.

Open source components compose as much as 50% of any given application

Step 1: Inventory your open source

You can't secure what you're not tracking, so your first step needs to be the creation of an inventory of all open source components your teams use to develop software.

A complete—and useful—open source inventory must include all open source components, the version(s) in use, and download locations for each project in use or in development. You'll also need to include all dependencies—the libraries your code is calling to and/or the libraries that your dependencies are linked to—in your inventory.

What to take into account

The size of your development team(s)

You have a better chance to accurately track open source use if your development team is small and all in one location. If you're using third-party developers, you'll need to be confident that they will be as diligent about code inventory as an internal team. Having a larger team or more teams can quickly make the inventory process unwieldy and more prone to errors and omissions.

Whether you can assure developer compliance

While your development team may agree—in theory—to document their open source use as it happens, the reality is likely to be after-the-fact recording with inaccuracies, partial listings of components, missing information on versions or download locations, unlisted components that entered via dependencies, and so on.

Mapping your open source against known vulnerabilities must be a continuous process

Step 2: Map open source to known vulnerabilities

Your next step is to compile a list of known vulnerabilities that have been reported against the open source components you've inventoried.

Most DIYers will use the U.S. government vulnerability disclosure database, the National Vulnerability Database (NVD, https://nvd.nist.gov), as their primary source. But be aware that not all vulnerabilities are reported to the NVD, and that the format of NVD records often makes it difficult to determine which versions of a given open source component are affected by a vulnerability.

Other useful sources of information include project distribution sites such as those maintained by the Debian (https://www.debian.org/security) and Python (http://bugs.python.org) projects. Security blogs and message boards such as the US-CERT alerts page (https://www.us-cert.gov/ncas/alerts) and Google's security blog (https://security.googleblog.com) should also be part of your daily vulnerability research.

"Daily" is the key word here. New vulnerabilities are uncovered literally every day. Mapping your open source against known vulnerabilities must be a continuous process in order to be effective. Since 2014 the National Vulnerability Database has reported more than 6,000 new open source vulnerabilities alone.

Outside of the sheer volume of data that you'll need to sift through, other challenges you'll face include prioritizing which vulnerabilities should be addressed immediately and which can be safely ignored, and then mapping vulnerabilities back to their specific locations in your code.

Step 3: Identify other open source risks

Licensing compliance

If you build packaged, embedded, or commercial SaaS software, open source license compliance is a key concern. You'll need to determine the license types and terms for each open source component you use and ensure they are compatible with the packaging and distribution of your software.

Using your list of open source components, you'll want to compile detailed license texts associated with those components so that you can flag any components not compatible with your software's distribution and license requirements and generate a license notices report to include with your shipped software. Unfortunately, there is no uniform approach to component license documentation. You'll need to research the license(s) for each component you're using.

At this point you may also want to involve your organization's general counsel—or seek outside legal advice—as understanding licensing terms and conditions and identifying conflicts among various licenses can be challenging for those not familiar with legal terminology.

Component quality

Security and licensing concerns aside, how do you know you are using high-quality open source components? Are you using a current version of the software? Is it the most stable? Is the component actively maintained by a robust community?

If you are taking a DIY approach, you'll need to go to GitHub or the distribution source for each component to research project activity as well as potential alternatives. Since there is no universal scoring mechanism to aid in comparing different components or versions, this can be both time consuming and subjective, a reason why many organizations struggle with effective open source governance.

Step 4: Manage risk and monitor for new threats

Once you've identified vulnerability, licensing, or component quality risks in your open source, it's time to prioritize those risks and address them. You'll need to determine what remediation—if any—needs to be done, assign the remediation work to the appropriate people, and track the remediation process: what's being reviewed, what's been reviewed, what's been fixed, what fixes have been deferred, and what has been patched.

Challenges you can expect to face include time and cost issues. Manual review tends to result in remediation late in the development cycle, when the cost to fix is high and release deadlines need to be met. Manual review processes are also incompatible with the rapid pace and automation at the core of modern agile build and continuous integration environments.

It requires that all the discovery processes described above be done continuously, and review and approval mechanisms be integrated into the overall SDLC processes. Without the assistance of tools to automate these processes, many organizations struggle to implement mechanisms that are both effective and efficient.

More importantly, the job of open source vulnerability management doesn't stop when the application ships. You'll need to continue to monitor for vulnerabilities as long as the application is in use. An average of 10 new open source vulnerabilities are discovered daily,[4] and most vulnerabilities aren't reported for months or even years after they are introduced into the component.

While it’s possible to wear a toilet bowl as a hat, that doesn’t make it desirable

A better way of open source vulnerability management

Can you create an effective manual process to manage open source use? It’s possible, but while it’s possible to wear a toilet bowl as a hat, that doesn’t make it a desirable fashion statement. The odds are that a manual process will be neither complete nor completely accurate. Black Duck software audits of customer codebases consistently show that most organizations are using twice as much open source as they are able to track manually.[5]

A manual approach to open source vulnerability management has many drawbacks, including poor accuracy, reduced productivity, and high costs. And while it may seem easier to just keep doing what you’re doing and hope for the best, there is an alternative. Many organizations turn to an automated solution such as Black Duck software composition analysis to simplify and automate open source risk management, enabling them to effectively inventory open source in their code, protect against security and other open source risks, and enforce open source use policies.

An automated solution such as Black Duck can provide the tools your organization needs to manage open source risks reliably and cost-effectively. Black Duck helps organizations:

- Inventory and track the use of open source software within their applications and containers.
- Map open source components to known open source vulnerabilities.
- Identify open source license and component quality risks.
- Automate open source policy and risk management.
- Monitor for and alert them about new open source vulnerabilities affecting them.

References

1. Tim Clark, Most Cyber Attacks Occur From This Common Vulnerability, Forbes, May 2015.
2. Cisco, 2015 Annual Security Report, 2015.
3. Mike Pittenger, The State of Open Source Security in Commercial Applications, Black Duck Software, 2016.
4. Tracy Brockman, Managing Accepted Vulnerabilities, SANS Institute, May 2016.
5. Ibid.

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com