sonatype

5 Key Open Source Security Risks and How to Prevent Them

December 01, 2022 By **Luke Mcbride**

5 minute read time

SHARE:

Whether your company or organization is managing it, open source software (OSS) is firmly entrenched in software development strategies.

- 96% of codebases contain at least some open source (Gartner, 2019)
- 90% of enterprise IT leaders are using open source tools and 79% expect adoption of OSS to increase in the coming years. (Red Hat, 2021)
- Over 3 trillion downloads are projected this year alone across the four major ecosystems: Java, JavaScript, Python, and .NET. (Sonatype, 2022)

One reason companies are embracing open source solutions is that they believe it can help protect against cyber threats. The thinking is that open source projects typically address vulnerabilities quicker than their proprietary counterparts do.

While open source certainly has its advantages, using open source solutions doesn't automatically lead to stronger security—just like migrating to the cloud won't automatically make your business more resilient.

In both cases, you must do your due diligence and examine your environment to ensure you're protected.

Open Source Security Risks: The Challenges

What follows are some security issues to keep in mind when leveraging someone else's code to build digital services.

1. Publicity of Exploits

Because OSS code is freely accessible to the public, the issues are also public and visible. As component vulnerabilities are found and shared, it's possible they could be used against other projects that use those same components. This can make it easy for bad actors to discover

vulnerabilities and execute targeted attacks against enterprises—particularly when they're using poorly maintained code.

2. Licensing Management

While OSS code is free to use, there are specific licensing terms to follow. Violating OSS licensing terms can potentially lead to legal action and losing exclusive rights to intellectual property.

Too often, companies develop digital services without knowing what OSS licenses they're using, but companies must both track and fulfill license terms to avoid legal risk.

3. Acquisition Complications

Sooner or later, OSS vulnerabilities will rear their ugly heads, which is just the nature of the beast. Unfortunately, this lack of awareness of vulnerable components can complicate companies wishing to merge or be acquired. When investors and acquiring companies ask to review your underlying code for legal compliance, security, and performance reviews, they are looking for quality source code.

Companies that can't produce documentation or demonstrate they're using reliable code can delay or disrupt potential business deals. This can lead to major consequences for developers, technology leaders, and entire companies.

This is especially problematic for fully digital companies who are entirely dependent on connected applications and services.

4. Managing Code

Developers often select OSS libraries without consideration around track record, health, or viability.

On one hand, this saves a considerable amount of time and effort. But developers don't always take the time to check for security and licensing requirements, which can cause serious problems. They also don't consider

the transitive dependencies that come with these libraries, which might also have security concerns.

Over time, as these components grow your software becomes more complex, you can have an incredibly difficult time monitoring it for security and stability.

5. Lack of Security Expertise

Few engineers are security experts, and some developers may be unaware they're introducing serious security issues in their code. Others have no patience for security concerns and instead bank on security teams catching their mistakes.

Unfortunately, attacks are on the rise and it's possible security professionals will miss vulnerabilities during testing—or even avoid addressing them altogether. This can result in releasing faulty and unsecured applications and services—much to the chagrin of your users.

How to Prevent OSS Risks: an OSS Health Framework

As you can see, using OSS software can lead to a range of security and legal risks. But that doesn't mean your organization should fear OSS or diminish its use – quite the opposite.

Instead, you should respect OSS security vulnerabilities and act to strengthen your security posture and eliminate gaps wherever you can. One way to achieve this is with a framework that helps avoid mishaps and keeps your company safe from external threats. Here's what such a framework could look like:

1. Shift Left and Adopt DevSecOps

In a traditional software development workflow, software testing doesn't occur until the very end of the process. Smart teams are rethinking this

approach, and many of them are integrating security with DevOps.

This is commonly referred to as "shifting left," meaning earlier in the process, which lets you catch security vulnerabilities earlier, saving effort and money. This can also improve collaboration, while removing some of the burden from your security team.

Additionally, building in evaluation steps for project quality and safe open source licenses early in the process avoids legal risk and rework. You can vet potential OSS using project quality metrics to choose optimum open source projects.

Shifting left is also a great way to ease developers into a security mindset and have them take more responsibility for both vetting software and closing gaps proactively.

2. Continuously Monitor Your Environment

OSS code isn't something you can set and forget. If your team is using OSS code, you also have to constantly monitor it and apply security patches as soon as they're available. By doing so, you can identify small issues before they lead to major catastrophes.

3. Create a Software Bill of Materials

When it comes to working with OSS software, documentation is critical for success. Teams should always keep a running software bill of materials (SBOM), which is a complete record of the dependencies and other items that exist in a piece of software.

By maintaining an SBOM, you can track code back to its source location. Smart companies typically use SBOMs to quickly identify where vulnerabilities exist and promptly remediate them upon discovery.

4. Maintain an OSS Security Library Inventory

In addition to tracking software components in an SBOM, it's also a good idea to have a master list of the various libraries you're using across all of

your applications.

One reason it makes sense to track different libraries is to ensure they remain active and in use. Libraries can lose popularity over time. When this happens, supporters can stop updating them, which means that security updates and patches are nonexistent. This puts your products at risk (unless, of course, you want to build them yourself, which is always an option).

In addition to tracking the OSS libraries you're using, you should also include information about licensing agreements, as well as versions and updates.

Improve your OSS Development Strategy With Sonatype

Software development teams are focused on speed and efficiency, which can make it harder to stay on top of evolving OSS security needs. Even more crucial is how the development lifecycle moves too quickly to manually track all security components across the software supply chain.

For this reason, a growing number of companies are automating OSS security analysis and using machine learning to track and analyze open source components. Sonatype Nexus Lifecycle works to improve OSS visibility and governance at scale with integrated supply chain management software.

With help from Sonatype's developer-friendly supply chain management software, your team can accelerate innovation while tightening its overall security posture.

To get an SBOM and look for vulnerabilities in your software, visit the Sonatype Vulnerability Scanner or, to see us in action, request a demo today.

Tags: Nexus Lifecycle, shift left, OSS security, open source security risks

Written by Luke Mcbride

Luke is a writer at Sonatype covering everything from open source licenses and liability to DevSecOps trends and container security.

Learn more on:

AUTHOR POSTS TOPIC POSTS

Another SolarWinds? The Latest Software Supply Chain Attack on 3CX

Luke Mcbride

Visualize Your Open Source Governance With BOM Doctor

Luke Mcbride

What is Hashing? A Look at Unique Identifiers in Software

Luke Mcbride

First Name* Last Name*

Email*

Comment*

protected by reCAPTCHA
Privacy - Terms

SUBMIT COMMENT

Get Started

Videos
Customer Stories

PARTNERS

Find a Partner
Become a Partner
Log in

CUSTOMER PORTAL

Training & Workshops
Documentation
My Sonatype
Customer Support

COMPANY

About
Careers
Newsroom
Investors
Contact
Press Kit
Trust Center

Subscribe for all the latest software security news and events

Subscribe

Terms of Service

Privacy Policy

Modern Slavery Statement

Event Terms and Conditions

Do Not Sell My Personal Information