VERACODE EBOOK

Understanding Your Open Source Risk

VERACODE

The demand on software development teams is greater than ever.

With the cultural move towards DevOps, the implementation of CI/CD systems, and the desire to operate in an agile manner, developers are being asked to push out more software — and in shorter periods of time — than ever before.

In turn, developers are increasingly relying on open source libraries, or pre-built pieces of code available online, which allow them to add functionality to their code without having to build it from scratch. As a result, software today is rarely completely made of first-party code, and is more often "assembled" from other sources.

BACKGROUND KNOWLEDGE

Direct vs Indirect Libraries

When leveraging open source libraries, there are two categories of libraries pertaining to your application.

Direct dependency libraries are all of the pieces of open source code that your developers are directly using and adding to the application they are developing. However, in the open source world, many times these libraries are also relying on other open source libraries for some of their functionality. And it's very common for those libraries to use even more open source libraries, and continue the chain — we've seen anywhere from two to over 10 levels of libraries being called, one after the other. The functionality, and potential issues, are then proliferated down the open source library chain to your application: These second layer removed libraries are called **indirect dependencies.**

Risk of open source libraries

While the use of open source libraries is proliferating, so are the risks.

With reusable code and functionality also comes reusable vulnerabilities, and with very popular libraries used in thousands of applications, a single vulnerability, in a single piece of code, can suddenly make thousands of applications vulnerable to the same exploit. If we look at Java applications alone, we're seeing that 88 percent of Java applications use one or more libraries containing vulnerabilities.

There's also a pervasive false sense of security around open source libraries, and a general lack of understanding about the attack vectors that open source libraries can open up. While it's true that there are more "eyes on the code" when it comes to open source libraries, there's nobody to ensure that these security fixes are being disclosed, and nobody to inform you that there was a vulnerability patched. Ultimately, contributing developers are interested in the success of an open source library, not in the success of your business. It's up to the company implementing the open source library to ensure that it's safe to use and that it's staying up to date on any known vulnerabilities in the library and their fixes.

Simply using open source libraries isn't a security threat to the business. The real problem is not knowing that what you're using contains vulnerabilities and that they're exploitable in your application.

Challenges In Securing Open Source

When assessing the posture of your open source risk, you have to ask three key questions:

QUESTION ONE

Which libraries are we using, and do they contain any vulnerabilities?

QUESTION TWO

Does the vulnerable library actually do anything bad?

QUESTION THREE

Can I react fast enough to new vulnerabilities?

The unknown unknowns

Does my code contain any vulnerabilities?

With 70 percent of our customers leveraging open source libraries, and software being comprised of up to 90 percent open source code, the amount of open source code in use is growing rapidly.

Even though a developer could be pulling in only five open source libraries directly, those libraries could easily pull in hundreds of other open source libraries with them — including all of their vulnerabilities.

There's another fundamental issue that can be broken into two parts:

1. Today the most used source of vulnerability data is the National Vulnerability Database (NVD), which is completely overrun with vulnerability submissions.
2. Everything submitted to the NVD has to be known and disclosed, but often, when a vulnerability is disclosed and assigned in the NVD, it's been patched for several months.

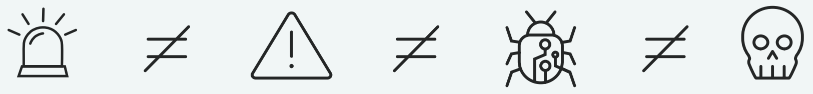
Not all vulnerabilities are created equal

Is the vulnerability in the execution path?

While there are different severity levels of vulnerabilities, there's also a second dimension that people don't, or can't, take into account — and that's whether the vulnerability is actually impacting your code.

When leveraging an open source library, it's very common for a developer to only use a small subset of that library, for a very particular function or capability. Thus, it's very likely that a vulnerability in the library is never actually being called by your application and, in essence, isn't exploitable.

Most companies prioritize high-severity and critical vulnerabilities, but ignore lower-severity vulnerabilities. However, there could be a noncritical vulnerability that's actually impacting your code and could be used to exploit the application. We argue that rather than fixing a high-severity vulnerability in a function that is not called by your application, you should prioritize a medium-severity vulnerability that lies in the execution path and puts your customers at risk.



Exploit attack window is shortening

Can I respond fast enough?

`Time between vulnerability disclosure and exploitation is counted in days, sometimes hours. As in many areas of IT security, the attacker is at an advantage.`

Attackers are known to look through the commit history of open source libraries to find code changes that remediate a vulnerability. Often, developers make these changes without publicly disclosing the vulnerability to the NVD. This means open source users aren't alerted to security issues or urged to update their versions. Meanwhile, attackers can develop exploits based on the vulnerability and often start attacking applications with a "spray and pray" approach across the entire internet. Once they successfully exploit a machine, they can access data or get a session pivot to other parts of the organization's network.

On the flip side, defenders need to be aware of not just this one vulnerability, but of all the vulnerabilities in their open source code, with the added complexity that they are not all listed in the NVD. They also need to work with development teams to fix all vulnerabilities that put the organization at risk. Without the right solutions, this is a near impossible task.

Nobody does it quite like Veracode

While we help our customers protect themselves from open source risks, there are a number of vendors that do the same basic discovery and reporting. However, Veracode's focus is on coverage and actionable results that help development teams maintain velocity while ensuring more secure applications are delivered to production.

DOWNLOAD OUR WHITEPAPER

Learn more about the technology aspects that set Veracode apart from the competition and how we can help solve your open source risk.