RogueWave SOFTWARE

Open Source Support

Reduce your open source security risk: strategies, tactics, and tools

OPEN SOURCE SECURITY MUST BE A PRIORITY

While there's no doubt that open source software (OSS) is here to stay, that doesn't mean that developers can feel free to use all and any open source software components with no thought to the vulnerabilities and security issues they may introduce into their development projects. The fact is, there's no such thing as bulletproof, bug-free, and automatically license compliant software. Not in the open source world and not in the commercial off the shelf (COTS) world.

While IT systems have become ubiquitous in virtually every aspect of business operations and consumer behavior and lifestyles, the productivity, innovation, economic impact, and convenience imparted by these systems don't come without risks. We see these risks everyday in news reports of data breaches, first-hand experience with systems that falter or fail, leading to frustration, reduced productivity, customer defections, and more.

So, it's incumbent on developers, project team leaders, IT managers, CIOs, line of business owners, all the way up to C-level executives to ensure that there are sound strategies and tactics in place to make it easy to acquire, distribute, use, monitor, analyze, and keep track of open source software to reduce the risk of vulnerable and buggy software and applications to an absolute minimum.

Challenges abound

Consider these statistics that illustrate the sources and breadth of the challenges facing open source developers:

- Applications are the number one attack vector leading to breaches.
- 90 percent of typical applications comprise open source components.
- Some 11 million developers worldwide make 13 billion open source requests each year.
- 46 million vulnerable open source components are downloaded each year.
- 76 percent of organizations using open source don't have meaningful controls over what components go into their applications.
- Only 20 percent of developers say they have to prove they're using secure open source components.
- While organizations may screen for common vulnerabilities and exposures (CVEs) as open source code comes in the door, only about 20 percent track vulnerabilities over time.
- Only 56 percent of organizations say they have open source policies in place. But less than 70 percent say they follow their own policies.
- 77 percent of organizations surveyed say they've never banned a component.
- 31 percent of organizations suspect (not know, but suspect) they've experienced a breach attributable to an OSS component.
- 51 percent of organizations aren't concerned about license risk.

Keep in mind that most open source software is created to fill a technical gap and is delivered "as is." Security is rarely top of mind in the development process.

There are many more hair-raising statistics that could be listed, but these should make the point that most organizations are sorely in need of well-conceived and defined strategies and tactics to enable them to manage the security risks inherent in using OSS.

Still, in spite of these and other risks, OSS adoption by companies and developers continues to grow year on year, owing to the competitive features and technical capabilities it provides, its ability to accelerate development and time to market, and of course, its ability to reduce costs. There's often also a general assumption that with so many individuals involved in the development of OSS components, it must necessarily be very well vetted and quite secure. In a way, OSS has over time engendered a level of brand trust, similar to or greater than the trust accorded to COTS brands.

Ask questions

You may already be concerned about the security of OSS, perhaps prompted by a first-hand experience with flaws and vulnerabilities that have created migraine-inducing headaches for you and your business.

If not, consider yourself lucky, but not off the hook. Ask yourself a few questions to assess your potential risk and liabilities. For instance:

- Have you selected, architected, and implemented OSS with security in mind?
- Do you know what OSS is present and in use in your organization?
- How do you stay on top of new vulnerabilities as they are discovered and reported?
- Do you know if you are in compliance with OSS license terms and conditions?
- Does your organization have a system in place to communicate information about CVEs to the pertinent stakeholders?
- Do you have the expertise in place to plug security holes in production systems?

If you've answered no to any of these questions, the only comfort you can take is that you're not alone. Of course, that doesn't solve the problem. So, what to do?

Quality and peace of mind lie in sound strategy and tactics

Every organization employing OSS must have an open source management strategy in place that articulates the challenges presented by OSS and the organization's objectives in terms of usage, support, and security. That strategy must be executed through the development and diligent implementation of tactics that enable and enforce OSS acquisition, tracking, monitoring, support, and analysis best practices. Wondering where and how to begin?

Following are five straightforward tactics that significantly reduce your OSS risks.

Create OSS policies

If your organization is committed to reducing the risk associated with using OSS, you must create policies that define and communicate usage rules and guidelines.

> Only 56 percent of organizations say they have open source policies in place. But less than 70 percent say they follow their own policies.

Based on the combined knowledge of the stakeholders in this process regarding the various OSS components in use across your organization, you can begin with a simple process of categorizing those components into black, white, and grey lists. According to this simple scheme, components on the black list would be banned. Those on the white list would be pre-approved. And those on the grey list would require review in order to determine approval.

These initial lists will get you started, but, this is not a one-time process. As additional OSS components are identified in your enterprise, and as new components become available, they'll have to be categorized.

Once this process is established, you may want to add another category: conditional usage approval. For instance, you might stipulate that Bouncy Castle Cryptos is approved for use only if you use version X.5 and above. Defining conditional usage can grow complex over time, so it's very important that you be crisp and clear in the language you employ to communicate such usage.

While providing clear guidelines for the use of all aspects of OSS, your open source management policies should also contain provisions that make it the responsibility of each developer to comply with the policies by tracking, recording, monitoring, and updating OSS usage.

For policies to be effective, they must be living, evolving things.

Create an OSS acquisition process

The next tactic to help you reduce your OSS risk profile is to create a process that defines how developers may acquire OSS. This tactic addresses the difference between can and may. Anyone can easily download or otherwise import OSS components from a plethora of sources. The point here is to manage how developers may acquire OSS and incorporate it in their work.

> 76 percent of organizations using open source don't have meaningful controls over what components go into their applications.[1]

While there are several ways to approach this, the best approach is to create a formal approval process that requires developers to formally request the use of a particular component that is not already white listed per your OSS policy. At a minimum, an effective approval process should require the developer to identify him/herself, describe the component being requested, list any reported security vulnerabilities, and detail how, when, and where the component will be used.

[1] Survey: Control and security of corporate open source projects proves difficult, JavaWorld, April 30, 2013

If for some reason, such a request for approval process seems too onerous for your organization, you might employ a declaration process in which developers, as they download the OSS in question, tag it with the same basic who, what, where, when, and how information described above. Those declarations could then go into a tracking database (to be discussed shortly) so that there is at least a clear record of the acquisition and intended use. This process could be further minimized by simply requiring developers to email the same information to their management teams.

While the first method seems clearly the best, the most important thing is to agree on a method that will be used. Thus, the inclusion of the less formal alternatives.

In determining what approach to take to managing the acquisition of OSS, you also need to create a list of key stakeholders to whom the acquisition request and associated information is communicated. In considering who should be on this list, include developers, project team managers, line of business managers, legal departments, and others who might be affected in the event of a subsequent problem.

Managing provisioning

The acquisition approval process naturally leads to the issues of where the OSS will be acquired and how it will be distributed across your enterprise.

To address this issue, it's recommended that your organization build a centralized, internal repository for approved components. Many organizations have already done this and it works. Once an organization creates a library of open source components it wants its developers to use and reuse, it's gone a long way toward enhancing the security and reliability of its code.

> "While organization may screen for CVEs as open source code comes in the door, only about 20 percent track vulnerabilities over time."

In the absence of creating such a centralized repository, some organizations have chosen to rely upon and use approved, external repositories such as those established and maintained by groups including Maven Central, The Apache Software Foundation, and the Eclipse Software Foundation. Using these sources provides a level of confidence that the OSS available through them has been vetted for reliability and security.

Beyond these two approaches, things can get risky. A lot of OSS is available through blogs and forums, which while readily and easily available, may enter your organization with some rude surprises.

Returning to the recommendation that you build your own repository, here are a few more recommendations:

- In addition to populating your repository with approved OSS components, consider including the metadata associated with those components. So, for instance, if a developer wants to use a particular Apache plug-in, he can view the metadata that might say "our security team has reviewed this and you have to use version 2.7 or higher."

- As you build your repository, consider building in functionality that will allow you to reach out and query the National Vulnerability Database (NVD), for example, and let you store the information contained there alongside the metadata you've already stored in your repository.
- Finally, and this refers back to the creation of your OSS policies, you should have a process in place that checks to ensure that all components stored in your repository have met your organization's criteria for security.

Tracking usage

Up to this point, tactics have focused on processes and activities designed to minimize the chance that flawed, buggy, vulnerable OSS components find their way into your organization, projects, and products.

Equally important is the fact that when a security issue arises, you need a way to identify what open source components you have in place and where they are being used in order to rapidly remediate the issue.

> "Only 20 percent of developers say they have to prove they're using secure open source components."

Tracking that kind of information manually would be difficult enough if your organization and its employees were never changing. But the fact is that project teams change, employees come and go, and all of this throws a wrench into orderly knowledge transfer.

When a problem arises — be it a malware issue, a compliance issue, or a patent issue — a tracking database can prove invaluable. You simply cannot rely on individual human recall, particularly if the individual who might have the answer is no longer working at your organization.

The goal in tracking open source usage is to provide a mechanism you can tap in the event of a security issue to find out what OSS you have, where it is, and how it is being used. You want to know how it's been modified over time. You want to know who the project owner is. You may also, for example in a merger or acquisition situation, need proof that your organization has complied with the appropriate licenses.

When a situation arises there are always more questions to be answered than any person or group of people can answer off the top of their heads.

In the case of OSS, this is made even more complex when you consider that a substantial percentage of OSS downloaded today contains additional open source components, each of which may have security issues or be licensed differently.

So you need to create a tracking system that tracks the who, what, where, and how of your OSS usage plus other attributes that may be uniquely important to your organization. This is a daunting task to be sure, but there are tools, such as Rogue Wave Klocwork, that detect security, safety, and reliability issues in real-time by using static code analysis and provide actionable reporting functionality to produce answers quickly.

Ongoing monitoring, analysis, and support

Ensuring the efficacy of your open source management strategy and supporting tactics requires an ongoing commitment on the part of your organization and each individual involved in open source development.

Once you've created policies to govern your organization's use of OSS, and developed processes and mechanisms to manage OSS acquisition, provisioning, and tracking, you must continue to monitor OSS updates for security issues, proactively analyze your OSS for security flaws, and establish mechanisms of support for when a new vulnerability or package update is discovered.

In the case of monitoring, that means:

- Developing and tracking sources for security information, such as the NVD, Open Source Vulnerability Database (OSVDB), and other established, reliable sources.
- Determining how you will monitor updates (i.e. what tools you'll use).
- Deciding how often you'll monitor based on business critical usage factors that you'll determine.
- Tasking this responsibility to an in-house owner or to a trusted outsourced service provider.

In the case of analyzing your OSS code, that means:

- Proactively scanning all code for security flaws (rather than waiting for problems to occur).
- Automating issue discovery through the use of static code analysis tools to ensure process efficiency.
- Distinguishing between real risks as opposed to typing errors — risks such as buffer overflows, memory leaks, uninitialized data, string taint, and process/OS specific flaws.
- Ensuring compliance with security standards (such as CWE, OWASP, CERT, and DISA).

In the case of support, that means:

- Establishing a remediation process to address new vulnerabilities and security updates.
- Acquiring the skills and training necessary to understand code security and fixes as it relates to your systems.
- Ensuring third-party support is available when a potential security issue is beyond existing skills, especially for systems in production.

> "80% of all OSS support ticket issues were either a lack of product knowledge, or something in the environment outside of the package."[2]

This brings the tactical approach described here full circle back to the key objective of your open source management strategy — to reduce the risks associated with using OSS to the greatest extent possible in order to fully and uninterruptedly realize the many benefits of employing OSS.

[2] 2017 Open Source Support Report, Rogue Wave Software, 2017

Don’t panic. Help is available.

Creating an open source management strategy and executing the tactics described in this document require time, effort, and consistent participation of stakeholders. It takes a lot of work.

The good news is that there are many tools and services on the market that can help make the process easier.

For more information on how you can reduce your open source risk through enterprise-grade guidance, security reviews, and production-level support, feel free to visit the Rogue Wave Software website or email us at info@roguewave.com.