Mitigating the Risk of OSS-based Development

If an organization is developing custom software, the odds are extremely high that open-source components are being used as building blocks for the organization’s applications. Open-source components lower costs, improve quality, advance innovation and speed software development processes. Because of these benefits, open-source use and component consumption are booming across multiple industries, including finance and banking, public accounting, government and the public sector, utilities, and manufacturing. In a recent Accenture survey, nearly 80 percent of large enterprises said that they use open-source technology and that their usage would be expanding. In fact, the Gartner Group estimates that, by 2013, open-source technology will be included in 85 percent of all modern commercial software solutions.

However, while open-source technology offers significant benefits, using open-source components carries technical and business risks that cannot be ignored. Factors such as quality, security and licensing can cause significant problems if they are not properly managed. IT professionals and enterprise leaders, including chief information officers (CIOs), risk management officers, and security and compliance executives, need new ways to mitigate the risk of open-source technology across the application development life cycle without disrupting their current development processes.

THE NEED FOR GOVERNANCE

To balance the wide-ranging benefits of open-source technology against its attendant risks, an open-source governance or management program is necessary.

According to the Gartner Group’s Research Vice President Mark Driver:

When unaudited and unmanaged, open-source assets seep into and proliferate in an enterprise’s software portfolio as hidden time bombs that can eventually result in catastrophic technical failures, security failures, audit compliance violations and intellectual property (IP) risks that create a significant loss of IT value—and, subsequently, broader business value.

Many organizations are downloading tens of thousands of open-source components without any reliable way to monitor or control their usage, which leaves the organizations vulnerable to security threats and licensing risks that can derail development processes or even established infrastructure. The recent Sonatype Software Development Infrastructure Survey of 1,600 software developers, team leads and architects provides a glimpse at the unchecked nature of open-source governance: 87 percent of respondents stated that open-source component use is unmanaged within their organization’s development process (figure 1). Even when companies have established open-source artifact rules, it is difficult to know whether developers are actually complying with or sidestepping component usage guidelines. Outsourcing to third-party development agencies or acquiring applications from independent software vendors further limits open-source software (OSS) visibility and control.

Figure 1—Lack of Governance Over Open-source Components

Do you have a standard means of choosing which open-source components are used across your organization?

- 13% We have no standards.
- 45% We have some standards but they are not enforced.
- 42% We must follow corporate standards.
VERSION CONTROL
As applications become more complex, it becomes increasingly difficult to keep up with the number of components and individual component variations that are being used across an enterprise. Open-source artifacts are continually modified, and changes occur quickly. The usage data for the Central Repository (http://search.maven.org/), the leading Internet resource for Java open-source components, shows that active products are updated nearly four times per year, or more often for important enhancements. With nearly 300,000 components in the Central Repository and few established processes for monitoring component changes and updates, it is difficult for most organizations to keep up. In the Sonatype survey, 58 percent of respondents stated that they discover artifact changes by searching the web, and 28 percent said that there is no good way to find out when an artifact changes at all. It is easy to see how multiple versions of open-source components can move into an enterprise.

For example, a well-known global financial services firm was using the Central Repository to download open-source artifacts to build applications. During development, the firm downloaded 184,000 artifacts, including 4,400 unique artifacts per month. When 22 different versions of a single artifact were brought in during a single month, version management was not only chaotic, it put the quality and integrity of the firm's applications at risk.

Because typical applications are comprised of dozens or even hundreds of OSS components and each of these, in turn, depends on additional components, dependency management and version control often become costly, time-consuming manual processes that can stall software development and threaten quality. Figure 2 shows a relatively common example of the complex dependency tree associated with most projects. While repositories enable centralized distribution of approved components, they also serve to enable centralized redistribution of defective components. Worse, without established usage controls or dependency management, the ungoverned use of OSS components can lead to devastating security, quality and licensing issues.

SECURITY VULNERABILITIES
Even when security warnings are posted and easily accessible, they are often overlooked. In March 2009, the US Computer Emergency Readiness Team (US-CERT) and the US National Institute of Standards and Technology (NIST) issued a warning that the Legion of the Bouncy Castle Java Cryptography API artifact was extremely vulnerable to remote attacks. In January 2011, almost two years later, 1,651 different organizations downloaded the vulnerable version of the artifact from the Central Repository within a single month. In January 2010, US-CERT/NIST posted an alert via the National Vulnerability Database that Jetty, an open-source servlet container, had a critical security flaw that could allow remote attackers to modify a window's title, execute arbitrary comments or overwrite files, and allow unauthorized disclosure of information. Despite the warning, in December of 2010, nearly a year later, approximately 11,000 different organizations downloaded the vulnerable version of Jetty from the Central Repository in a single month.

LICENSE COMPLIANCE
When it comes to open-source governance, cutting through the complexity of acquiring and evaluating external components and the associated legal obligations can be difficult and time-consuming. There are multiple types of open-source licenses, each with different terms and conditions that must be met. In fact, several common license types are incompatible with each other and cannot be combined into a new application.
One of the most critical considerations for choosing a license is whether the component licensing imposes restrictions on the resulting application. Copyleft licensing, such as the GNU General Public License (GPL) and its derivatives, is probably the most problematic for commercial entities. This license type allows anyone to use a copyleft-licensed component in projects, but, in turn, the completed application must maintain the same rights, including making the derived application and its source code freely available. In other words, copyleft is a general method for making a program free and requiring all modified and extended versions of the program to be free as well. This type of license is often called “viral.”

Another problem arises as a result of dependencies (figure 2). A viral license may be nested deep within a dependency tree. This license contaminates the entire application, and, therefore, represents a significant IP risk.

As ignoring or forgetting a license obligation can result in legal action, it is critical that a policy be in place to provide developers with guidance and to ensure that they include only components with license obligations that an enterprise is willing to meet.

SOLVING OPEN-SOURCE CHALLENGES

Enterprises are now searching for solutions to help them avoid the technical failures, security risks and legal issues that can result from ungoverned artifact usage and the proliferation of potentially flawed OSS components. Enterprises need solutions that will allow them to capitalize on all of the benefits of open-source technology while avoiding the risk inherent to these critical components—without disrupting development operations.

When evaluating strategies for effectively managing OSS, several factors should be considered. First, an open-source governance program should be established to filter, audit, track and manage open-source assets in the enterprise. Without a governance program, open-source assets that come into or leave the enterprise cannot be managed, audited or tracked. Guidance on quality, security and licensing—the three major risk areas to consider when using OSS components—should be included. Once an open-source governance program is in place, enterprises should:

• Benchmark current usage of open-source components to understand from where the enterprise is starting and to help set realistic goals. Evaluate compliance to existing open-source policies. Identify which groups are using open-source components, including what they are downloading from outside sources. Identify problematic components that are being used in development or in production applications. Classify existing projects based on business importance to identify potential risks and to provide a mechanism to establish the role and value of OSS within the enterprise’s existing software portfolio.

• Develop an enforceable open-source governance program. Determine the rules by which the application development organization can use open-source components. Consider quality, security and licensing factors. Ensure that there are effective enforcement mechanisms throughout the application development life cycle.

• Establish mechanisms to monitor the effectiveness of the open-source governance program. Enterprises will need to monitor open-source consumption from outside sources, such as the Central Repository. Discover potential problems by identifying groups that may not be adhering to the OSS policies. Audit applications to ensure the included components meet enterprise guidelines. Analyze all software delivered by subcontractors or software suppliers to ensure that they are meeting enterprise requirements.

• Build open-source management into the software development process. Avoid disrupting development to minimize project delays or increased costs. Developers must be provided with the tools needed to standardize a set of thoughtfully chosen components that are free of license and security defects. Automated analysis can be used during the

Enjoying this article?

• Read Apache Web Services Server Audit/Assurance Program.
  www.isaca.org/research-deliverables

• Read UNIX/LINUX Operating System Security Audit/Assurance Program.
  www.isaca.org/research-deliverables

• Discuss and collaborate on risk management in the Knowledge Center.
  www.isaca.org/topic-risk-management
build process to ensure that projects do not include flawed components.

- **Evaluate open-source components before using them in development.** Determine whether the project is mature enough for use and whether it meets the enterprise’s requirements for quality, security and licensing. Some of the metrics to consider include the size of the community, the availability of support, the project update frequency, and the reliability of the organization or person running the project.

- **Standardize based on a common set of open-source components.** Lower maintenance costs by reducing the number of components that need to be supported and by limiting the number of components that need to be evaluated.

- **Analyze and continuously monitor all applications for security vulnerabilities and licensing issues.** Examine the complete bill of materials for applications, not just first-level dependencies—flawed components may be hidden deep within an enterprise’s applications. Quickly learn about and repair newly discovered vulnerabilities. Analyze existing applications, too, not just new ones; it is never too late to find and fix issues. For mission-critical applications, consider using heavyweight scanners that examine the complete source code.

- **Establish well-defined channels of acquisition for each open-source component leveraged.** An enterprise must have a trusted source, such as the Central Repository, for each component.

**CONCLUSION**

There are tremendous benefits to using OSS in development, but organizations need to take the proper actions to mitigate the risks. Having an open-source governance program is critical. Only with the right policies, procedures and tools in place can an enterprise be assured that its custom applications meet the necessary license, security and quality standards.

**ENDNOTES**


2. Driver, Mark; “What Every IT Practitioner Needs to Know About OSS,” Gartner RAS Core Research Note G00207329, USA, 5 October 2010

3. Driver, Mark; “A CIO’s Perspective on Open-Source Software,” Gartner RAS Core Research Note G00209216, USA, 31 January 2011


