LEGAL TECHNOLOGY SPECIAL

CIOReview

The Navigator for Enterprise Solutions

JUNE 10, 2016 CIOREVIEW.COM

NightOwl Discovery:
Continuous Innovation
Takes Top Priority

Andrea Wallack,
CEO

$ 15US

CIO REVIEW
44790 S Grimmer Blvd.
#202, Fremont, CA-94538

Staying Secure with Open Source

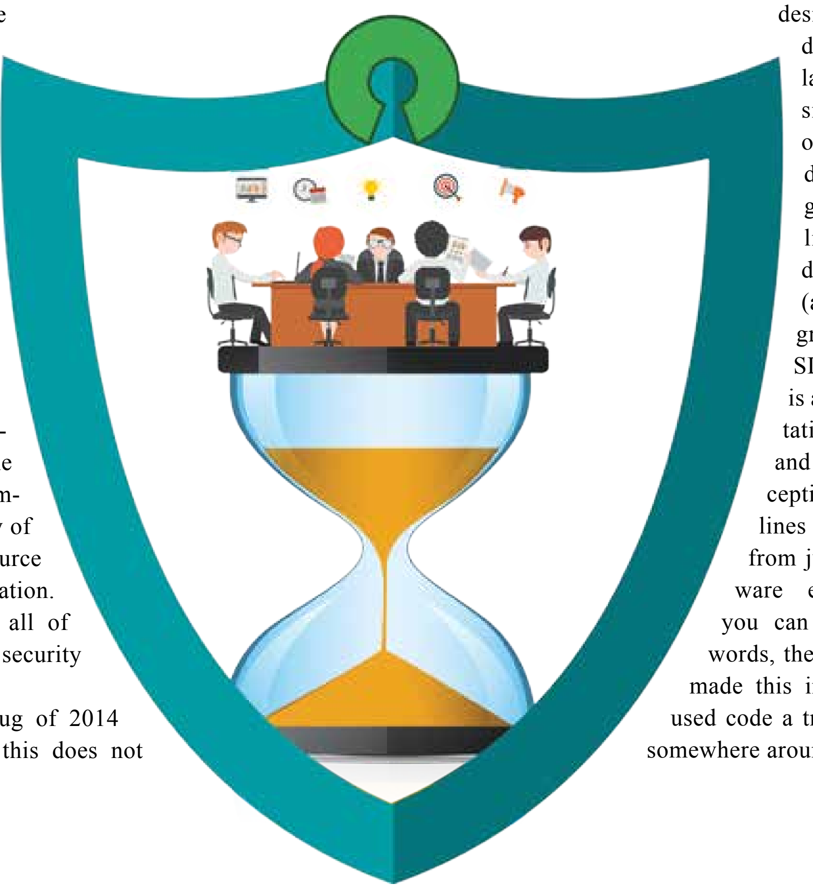
By **Aaron Tantleff**, Partner, Foley & Lardner LLP ,
Jason White, CAO & General Counsel, Toshiba America Business Solutions

Companies today are faced with ever-increasing pressure to innovate; quickly. For most, it is a function of market reality–the company is being asked to do more, with fewer engineers, and in less time. If the software is old, outdated, or it has been more than a year since the last major release with new "killer" features, customers may perceive your company as behind the times. As a result, many companies resort to the use of Open Source Software (OSS), usually under an Apache or GNU General Public License, for parts of their software offering. On the surface, this makes sense. Why should a company expend its valuable resources to reinvent the wheel when it can just leverage what somebody else has done for free?

Relying on OSS saves time and money but can lead to unintended and highly consequential results. For instance, OSS may compromise security. It is widely believed that OSS is more secure than proprietary software. Some believe that many programmers, from a wide variety of disciplines, review the source code and test the application. At least one will catch all of the software bugs and security flaws, right?

As the Heartbleed bug of 2014 should have shown us, this does not really happen in practice. Heartbleed is widely considered to be one of the biggest failures to date. Heartbleed effected an application called OpenSSL, which was used to encrypt the information for hundreds of millions of websites. A bug permitted invalid input (remember, hackers don't think in terms of valid inputs –they think in terms of invalid inputs) to cause a website to dump all of the information left around in its buffer. That information could include anything that was left lying around – account numbers, passwords, social security numbers, etc.

Nobody caught it because software engineers, under the pressure of short schedules and to get the latest and greatest gadget out, were fooled into believing that because OSS can be more secure, it is infused with the magic security pixie dust such that it is more secure.



Why did Heartbleed fail? One reason, while OSS may have more eyeballs on it, it suffers from inconsistent coding methodology.

While a company may (and if not, should) have some standard coding guidelines and conduct design reviews to ensure those guidelines are consistently followed for production code, OSS has none of that. While some of it may be great, particularly the code designed by a group of developers who collaborate closely with similar well thought out and reproducible design methodologies, not all OSS is like that. One security developer at FreeBSD (an Open Source group) noted, "OpenSSL… sucks. The code is a mess, the documentation is misleading, and the defaults are deceptive. Plus, its 300,000 lines of code that suffers from just about every software engineering ailment you can imagine." In other words, the poor design process made this incredibly commonly used code a train wreck that costs somewhere around $500M.

In addition to the lack of structured code in some OSS, there is a hidden danger for companies using OSS in their products. Because OSS is designed, developed, and ultimately maintained on a volunteer basis, it may become orphaned, without any individual or group of individuals to maintain it. Who can blame them?

Eventually even the best of us has to make a living and sometimes life gets in the way and there is no time for such volunteer work. Like the new security vulnerabilities are found all the time in proprietary code (Microsoft is famous for "patch Tuesday"), those vulnerabilities may also be found in OSS code. When it is, and the OSS is orphaned, the vulnerability may be around for years afterwards. Verizon's 2015 Data Breach Report reported that 97 percent of all exploited vulnerabilities were a result of just 10 vulnerabilities published in Mitre's Common Vulnerability Exploits (CVE). Eight of these 10 had been known and publicized for more than five years. In fact, since 2013, the CVE database has experienced a huge spike in known vulnerabilities for common libraries such as Open SSL. The older the code, the more likely it is that there is a CVE that affects it.

Jason White

While not all unfixed vulnerabilities can be attributed to OSS, the chances of a bug not being fixed in orphaned OSS code are significantly higher than it may be for proprietary code where a company's reputation may be on the line.

The U.S. Department of Homeland Security estimates that 90 percent of a typical application is comprised of OSS components, and as much as 71 percent of applications have a critical or severe vulnerability in their OSS components. Take, for example, the IoT market. As new devices come out, old models lose their support as the company moves their engineers over to develop next year's product.

The best solution may lie in the old adage "trust, but verify"

So what is a technology company to do? It is doubtful they will stop using OSS, and it would put them at a competitive disadvantage to do so. The best solution may lie in the old adage "trust, but verify." It's fine to use OSS, but a company shouldn't believe that it is without bugs or security vulnerabilities because it is OSS. Instead, companies should test the package, and their overall product, for security vulnerabilities.

At a minimum, the company's software test engineers should be testing against CVE's with a score of seven or higher, and maybe even 4 or higher.

Aaron Tantleff

(Heartbleed itself was originally scored as a five, but the scoring system has been revised to place it significantly higher.) The company should also be vigilant during code reviews for evidence of issues identified in the CWE/SANS Top 25 Most Dangerous Software Errors database, the sister security issue list to the CVE database and publicized from Mitre. There is a multitude of commercial tools to analyze source code for security vulnerabilities, including the CWE/SANS Top 25 Most Dangerous Software Errors. Which begs the question, which tests the commercial security tools for security issues?

Through such vigilance, and a healthy skepticism about the use of OSS, companies can continue to reap the efficiencies of OSS. Homeland Security estimates that if only 50 percent of software vulnerabilities were addressed prior to production, costs could be reduced by 75 percent. Plus, there is a side benefit to verifying code before production - minimizing the chances of becoming the latest security issue on the front page news. Thanks Steven Millendorf at Foley & Lardner LLP for all his assistance with this article. CR