REPORT REPRINT

Securing open source, Part 1: Why all the attention, and why now?

SCOTT CRAWFORD

17 JULY 2018

Open source adoption has exploded, and with it come new risks. High-profile security incidents like Heartbleed, as well as the Equifax breach, have highlighted their impact. We look at how these risks have affected application and software security and at technology specifically intended to address them: software composition analysis.

In 2011, venture investor and Web pioneer Marc Andreessen asserted that 'software is eating the world.' Businesses at the time were moving wholesale toward running on software and delivery as online services. Five years later, when Andreessen Horowitz raised its fifth fund, it evolved that assertion to 'software will program the world.' Datacenters have largely been overtaken by the ongoing evolution of virtualization, which has progressed from virtual machines to containers to 'serverless' concepts – yet what these all have in common is that they are 'infrastructure as code': the implementation of functionality formerly seen as inextricably tied to the underlying physical platform now rendered as software. Add to this the fact that as the costs of computing and storage continue to fall, programmable functionality becomes far more accessible and pervasive and the result is a world where technology is on the verge of penetrating nearly every aspect of life from the home to multinational commerce.

These new capabilities all rely on one key element: the software necessary to make them run. As a result, the explosion in pervasive technology has led to a demand for software as never before – a demand being met in no small measure by open source software (OSS).

THE 451 TAKE

Why open source? For one thing, the sheer scale of demand requires many more minds focused on many more opportunities and problems that only software can address. These factors have also sped up the pace of development. Open source presents a unique opportunity to tackle both these issues. It encourages wide participation in projects, which, in turn, stands to yield more innovative ideas – often from unexpected sources – than closed initiatives. This can help accelerate innovation. It can also help to reveal what doesn't work, which can help make development more efficient than efforts not open to such broad scrutiny. Security, for example, benefits from having many eyes on a project, given that the universe of those able to find vulnerabilities in software may extend well beyond the expertise and insight of any closed group. But this openness to discovery also means that open source vulnerabilities can be just as widely exposed – and given the unique dynamics of the open source universe, these exposures can be just as uniquely challenging to resolve.

In this report, we look at how the boom in OSS adoption has also led to an increase in awareness of open source risks, from licensing issues to security – and the measures required to protect organizations against those risks. We examine two incidents in particular – the Heartbleed vulnerability and the 2017 Equifax data breach – and how those events have shone a glaring light on the gaps organizations face in protecting themselves. We conclude with an introduction of a technology segment focused on specific aspects of these concerns – software composition analysis (SCA) – with a look ahead toward our next report on this topic.

THE 'HOCKEY STICK' OF OSS ADOPTION

The many values of OSS combined with the boom in software-hungry technology have led to a remarkable rate of OSS adoption in just the past year alone. In its 2018 Report on Open Source Security and Risk Analysis (OSSRA), the Synopsys Center for Open Source Research and Innovation found that the average percentage of OSS in the codebase scanned by the company's Black Duck On-Demand service was 57% – an increase from 36% in 2017. Says the report, 'many applications now contain more open source than proprietary code.' It's not just consumption that's on the rise. Trackers such as Modulecounts.com see a steady increase in OSS production as well.

This increase in adoption has also introduced a new range of challenges as well as opportunities for managing OSS. For example, sharing, access control, collaboration and version control for open source components via readily accessible repositories has grown to the point that major software companies that had taken a notably anti-open source stance in the past have not only recognized the OSS boom but have actively embraced it. In early June, Microsoft hung a sizable number on its interest: $7.5bn in Microsoft stock, to be paid for GitHub, used by 28 million developers and hosting 85 million public and private code repositories (and delivering a nice return to GitHub investors including Andreesen Horowitz, further confirming the VC's expectation of software consuming – and powering – the world).

OPEN SOURCE RISKS

As with any technology, OSS presents many unique advantages – and also has its own set of risks. Licensing, for example, can be a common issue because there is a dizzying array of licenses under which OSS can be used and distributed (no fewer than 83 are listed by the Open Source Initiative on its website). Even the definition of 'free' software may be unclear because various groups may have differing interpretations. Terms may vary regarding issues such as distribution and modification, patent or trademark grants, whether modifications made by any one developer can be kept private within a specific organization or must be shared with the community, and where issues such as sublicensing (as when OSS is licensed under a copyright) must prevail to preserve the license under which the original code was provided. Many of these issues have arisen as private concerns wrestle with the opportunity – or, viewed from another angle, temptation – to profit from open source, as we described in this report.

Even the linking of code to that governed by a different license can be sticky, as when software is provided as a library to which other code is linked in operations. Interactions among interested groups in determining the terms of licenses further complicate the picture. No matter how complex and involved the licensing that governs a particular deployment may be, violations can lead to a loss of interest in or control over an investment in intellectual property or other legal liabilities which may not be clear unless these complex licensing issues are well understood.

OSS AND SECURITY

Even more serious risks may face organizations using OSS – which at least touches virtually every software-using organization in the world – when it comes to security. As with commercially produced software maintained by the developer, the code developed by third parties may often include security vulnerabilities and the using organization is dependent on the supplier to resolve them. Unlike commercial software, the 'supplier' in this case is the community that developed the OSS – and communities and their participants may have varying degrees of motivation or timelines for resolving such issues.

An additional complication arises when further software development is based on open source packages. Regardless whether developed by bespoke efforts or new OSS projects built on prior work, this introduces dependencies on the underlying packages. When security vulnerabilities are discovered in those dependencies, resolving them can have a 'downstream' impact that can slow or hamper the measures necessary to deal with the exposure.

CHANGING THE GAME: ENTER HEARTBLEED

These issues came to prominence with the Heartbleed vulnerability that came to light in 2014, one of the first vulnerabilities to be given a name thanks in part to its prevalence and impact. Heartbleed arose from a software 'bug' – a defect in implementation – introduced in the 'heartbeat' feature of OpenSSL, the popular open source encryption and communications privacy package (hence the name 'Heartbleed,' first given by an engineer at Codenomicon, a security firm acquired in 2015 by Synopsys). A heartbeat is used by systems to keep components aware of the presence or state of other system elements; in this case, it was used to keep communications encrypted with OpenSSL's Transport Layer Security (TLS) implementation open without the computationally expensive and time-consuming need to renegotiate connections that could appear to be dormant, which could affect the performance and availability of relying systems.

Inadequate input validation in the implementation led to a situation where more data could be read from the memory of affected systems than the developers intended. Because the heartbeat feature in affected versions of OpenSSL was enabled by default, large numbers of systems faced exposure of encryption keys, Web session cookies and passwords, the security of digital certificates used in affected cases and, with all these, the sensitive content believed encrypted by vulnerable versions of OpenSSL.

The Heartbleed vulnerability was not detected when first introduced into OpenSSL version 1.0.1 in March 2012. It was not discovered and publicly disclosed by security researchers until April 2014. By that time, Internet watchers such as Netcraft estimated that about 17% of secure websites on the Internet using roughly a half-million digital certificates issued by trusted certificate authorities were affected – no small matter considering that Web application attacks have been repeatedly identified as a leading factor in security incidents by researchers such as Verizon's Data Breach Investigations Report team.

As badly as it affected the Web, the impact of Heartbleed went far beyond websites because vulnerable OpenSSL had been widely implemented on everything from enterprise network infrastructure (Cisco identified at least 78 products affected within days of Heartbleed's discovery) to personal devices. Heartbleed thus demonstrated how widely just one OSS package – indeed, a single version of that package and its risks – can penetrate. More pointedly, it also became the first definitive example of how undiscovered security vulnerabilities in OSS can propagate throughout a remarkably broad scope of technologies, leading to problems or even outright crises where dependencies and interrelationships can be extraordinarily complex to resolve.

EQUIFAX 'STRUTS' SOME (UNFORTUNATE) STUFF

Another incident highlighting OSS security risks emerged in the wake of a vulnerability in Apache Struts (specifically, Struts 2) that led to headlines throughout 2017. Struts is a software package designed to extend the Java Servlet API for enabling the management and delivery of dynamic Web content, allowing websites to use Java to enhance and extend their server-side capabilities. Because Struts 2 uses functionality such as the Open-Graph Navigation Language (OGNL) to further these extensions through programmable interaction, it has risks of remote code execution – risks which can be mitigated and managed but when exposed and unresolved can introduce the possibility that attackers can remotely execute malicious code to exploit server-side functionality.

Just such a vulnerability was disclosed by researchers in March 2017 in the Jakarta Multi-Part Processor incorporated in Struts 2 that handles OGNL in form data uploaded to a website. (Jakarta is the name of an umbrella project, now retired, for open source development of Java enhancements under the auspices of the Apache Software Foundation – Apache, in turn, being the proponent of several OSS efforts including the widely adopted Apache HTTPD web server. See how intricate the interrelationships among OSS projects can become?)

The upshot of the vulnerability was that attackers could include malicious commands in content uploaded to a form on a server employing the vulnerable Struts 2 functionality. Researchers such as Sophos' Paul Ducklin described how the vulnerability could be exploited without needing to log into a vulnerable site, retrieving a given web form, or even having any legitimate form data to upload. While knowledge of how to craft specific syntax was required to exploit the vulnerability, that level of detail quickly became widely available, which meant that attackers often needed little technical knowledge or skill to capitalize on the issue.

This was the vulnerability identified in a September 15, 2017 press release issued by credit reporting provider Equifax that was exploited in the breach of its systems that exposed the personal information of more than 143 million individuals (a number since adjusted upwards to more than 146 million). Although Equifax first announced the breach a week before on September 7, the September 15 statement noted that it first discovered indicators of the breach the previous July 29 – more than four months after the vulnerability was first published with its unique identifier (CVE-2017-5638) by the US National Vulnerability Database on March 10.

UNDERESTIMATING THE IMPACT OF MITIGATION

These incidents highlight several key factors implicated in open source vulnerability management and the difficulties faced in remediation that can dramatically extend the time and extent to which organizations may be exposed to security risks:

- Any one application, server or other software functionality may be composed of multiple OSS projects, with dependencies upon dependencies that must be teased out during resolution.
- Even within a single OSS project, multiple versions of the project may have different vulnerabilities, each requiring its own remediation.
- Because open source often requires the deploying organization to do its own compiling or building of complex software, resolution may necessarily involve recompiling or otherwise rebuilding that software. Web applications vulnerable to CVE-2017-5638 in Struts 2, for example, may have required recompilation to eliminate the exposure to remote code execution.
- In addition, any recompile or rebuild may be subject to testing or other requirements to demonstrate its reliability and performance as well as its security.
- These factors can exacerbate what may already be long cycles of remediation for even the most severe vulnerabilities in application software. According to CA Technologies Veracode's 2017 State of Software Security report, only 14% of very high severity flaws were closed in 30 days or less; 25% of sites in the study were running on web servers having at least one high-severity vulnerability.
- Remediating vulnerabilities in OSS in particular is even more problematic, with many components remaining unpatched once they're built into software. The 2017 CA Veracode report goes on to note that 88% of Java applications in their study had at least one flaw in a component.
- For an organization that supports multiple products incorporating OSS, these dynamics can further draw out resolution at substantial cost.
- These factors are in play when OSS is supported. When an organization depends on OSS that is no longer supported, resolution can enter an entirely different dimension of frustration.
- Critical dependencies the business may have on the implicated functionality can throw up additional roadblocks to the system downtime required to remediate, no matter how severe the vulnerability. This, in turn, requires businesses to be responsive in assessing their risk and exposure, and taking action which, although possibly damaging to the business, can head off more serious consequences.
- This level of maturity in technology risk management is often beyond that of many organizations. An effective response to such issues requires engagement with business stakeholders who can make the timely decisions necessary to protect the business at an appropriate level of responsibility. Compare that with statements made to Congress in October 2017 by the (now) former CEO of Equifax, who claimed an 'individual' in Equifax's technology department was responsible for failing to respond to security warnings and implementing necessary remediation. This suggests how even organizations responsible for the most sensitive information may be challenged in achieving the level of maturity needed to grasp the impact of risks and respond to complex security issues affecting the business beyond just its technology assets.

SOFTWARE COMPOSITION ANALYSIS COMES INTO ITS OWN

For these reasons and more, organizations have been embracing technologies like SCA that help address challenges such as:

- Identifying and clarifying OSS licensing issues before they have an impact on an organization's investment in technology – or in investments such as mergers and acquisitions on which technology can have a substantial bearing.
- Discovering and tracking security issues in OSS, with an emphasis on early identification of known issues, to help avoid incorporating vulnerabilities, vulnerable versions of OSS and unsupported releases into software projects.
- Helping to integrate these measures more seamlessly into software development processes, such as DevOps tools and techniques that increasingly characterize agile shops in implementing technology, from concept to the delivery of production functionality, including 'infrastructure as code.'
- Because of these values, SCA has played a highly visible role in recent market moves, including acquisitions among major vendors carving out a strategic role in shaping what application and software security is becoming. In part 2 of this report, we'll take a closer look at SCA, its distinguishing characteristics, and a sampling of players and recent activity in the field.