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.

REPORT REPRINT

Securing open source, Part 2: Software composition analysis comes into its own

SCOTT CRAWFORD, JAY LYMAN, DANIEL KENNEDY

7 SEP 2018

In part 1 of this report, we took a look at the factors driving the recent upsurge of attention given to open source security. In part 2, we look specifically at software composition analysis, a technology segment that has arisen to deal specifically with managing some of open source's most serious risks.

 |

In part 1 of this report, we examined the drivers behind the adoption of open source software, which is redefining the nature of software development and operational deployment technologies such as containers and container orchestration. We identified a number of the risks and implications of open source adoption, from security and licensing to the complexity of management inherent in these issues.

In part 2, we take a look at the attributes of software composition analysis (SCA), the technology segment that has arisen to address a number of these risks for organizations embracing the open source revolution.

THE 451 TAKE

Software composition analysis is becoming an increasingly visible aspect of due care for security and license risks in open source software. It is also a recognition of the complexity organizations face in getting a handle on these challenges – and an acknowledgement that a need exists for tools and technologies to help. In 451 Research's 2018 Voice of the Enterprise – Information Security: Workloads and Key Projects survey, nearly half (44%) of all respondents were either using software composition analysis (in a pilot or proof of concept of SCA) or planning to deploy it in the next 12-24 months. Of these, 42% expect to increase their spend on SCA in the next 12 months; one-sixth of this group expects that increase to be significant.

In this report, we explore the key values of managing security vulnerabilities, license complications and administrative complexity that are driving the adoption of SCA, as well as the increasing breadth of the technology – in support for languages and implementations, in the scope of coverage from development to operations, and in the range of integrated tools across the DevOps spectrum. We conclude with a look at example vendors, and what we expect in SCA going forward.

THE RISE OF SCA

The history of SCA traces back to a number of threads – from the lessons learned by private companies and their founders, who personally endured the pain of manually accumulating an inventory of software components and their liabilities, to OWASP (the Open Web Application Security Project, administered by The OWASP Foundation community), which in 2013 added 'using components with known vulnerabilities' to the OWASP Top 10 list of security issues facing web applications.

This addition was significant in that many policy and enforcement initiatives either directly require or indirectly influence organizations to address the OWASP Top 10 as part of their fundamental security measures. Nothing, however, has driven the rise of SCA like the emergence of recent high-profile security incidents related to open source vulnerabilities, from Heartbleed to the Equifax breach of 2017.

The integration of SCA capabilities with widely adopted repositories helped spur its adoption in some quarters; for others, the value of license compliance remains strong. More recently, a focus on security has shaped early players, as well as more recent entrants, many of whom also emphasize a more seamless approach to integration with popular DevOps practices and tools.

Today, SCA revolves around three fundamental realms of capability:

- Identifying and resolving security vulnerabilities in the open source components on which software is increasingly built.
- Addressing the impact of open source licenses on software projects.
- Managing the range and complexity of SCA involvement across the software spectrum, from mitigating issues in development to integrating with a wide range of modern development and IT operations toolsets and resolving exposures in operational environments. An emphasis on automation in concert with these toolsets adds to current market drivers.

INVENTORY, INTEGRATION, ADMINISTRATION AND COVERAGE

One of the first and most fundamental values provided by SCA is the ability to identify and track all the open source software (OSS) in your portfolio. The accessibility of OSS is one of its primary values, but that same accessibility comes with risks. Licensing issues and exposure to security vulnerabilities inherent in that software are two of the best known (we will get to those in more detail shortly). But before any organization can assess these exposures, it needs to know just how and where these exposures may exist – and that requires an accurate and up-to-date inventory of OSS assets.

Software composition analysis can gather details on component inventory from a wide variety of sources to develop a bill of materials (BOM) for specific projects, or to provide a comprehensive view across a number of projects to help organizations manage multiple (sometimes interrelated) and often complex software efforts.

Repositories are among the most common sources of this data, with SCA tools often connecting directly to popular resources, including GitHub, package managers such as npm, open source projects such as Apache Maven and private repositories. Git and GitHub, in particular, have become an increasing focus of such efforts, representing an aspect of the larger trend toward 'GitOps,' which refers to using Git and GitHub pull requests as a management tool for software deployment and infrastructure provisioning. SCA can also police check-in/check-out processes to monitor the integrity of components and status of policy compliance. It can also enforce policy on the incorporation into projects of components that don't comply.

SCA is also becoming an aspect of capabilities being embraced by vendors that are advancing wider initiatives to coordinate pipelines of tools – both open source and proprietary – in a more coherent approach to DevOps automation and process management. Development projects and individual organizations may vary widely in the range of languages, implementations and toolsets they employ. To keep up with these dynamics, vendors increasingly widen the aperture of support. Even if known as a particularly recognized advocate of ecosystems, such as Java, many vendors today tout their support not only for multiple languages, but for multiple tools across the automation spectrum that reflect the distinctive preferences of each development and operational team.

SCA tools can deliver findings at multiple stages of a project – from initial source development to compiled binaries; output artifacts; and even operational deployments of containers, microservices and 'serverless' environments. While integration with specific tools is common – for example, integration with Jenkins helps prevent the incorporation of problems at build time – an emphasis on integrating seamlessly with tools throughout the DevOps spectrum has become increasingly evident.

At the development end of the spectrum, this reflects a 'shift left' trend that seeks to empower developers more directly for better software security and quality outcomes. These efforts align with closely related integrations, such as application and software security testing, as well as with source code analysis available to the individual developer that provides the ability to check one's own code before adding it to or changing a specific project. Here, SCA can, for example, block the incorporation of vulnerable components into a development project; prevent their release into source code management when not compliant with, for example, version control policy; prevent or alert on builds before they are executed; or restrict the inclusion of noncompliant components into operations.

As these examples suggest, 'shifting left' doesn't mean that SCA is exclusive of the operational end of the DevOps spectrum. Coverage of containers has also become a more frequent aspect of SCA, given the potential to include exposures introduced from components that go into the makeup of applications and microservices, and the highly portable software platforms on which they are increasingly deployed. Guarding against building known vulnerabilities into these environments is only part of the use case. When newly discovered vulnerabilities affect a given composition of containers or microservices, organizations will want to be aware of how they affect their own deployments, so they can remediate the existing resource complement and take appropriate action in the operational environment.

SCA is also now looking toward 'serverless' environments, where providers enable organizations to expose only the functionality specific to an event-driven service, without the need to provision or manage the underlying platform. The software that goes into this functionality may be composed of underlying components beyond custom development – and those components fall within the scope of SCA.

When SCA technologies present their insights, they must do so in a meaningful way. Among the questions they must answer: Do identified issues pose significant risk? Can these issues be ranked according to severity or impact in such a way as to facilitate more systematic management? Many tools today not only seek to answer these questions, but also to provide guidance on how best to answer them. SCA allows organizations to define policy and policy templates for specific issues, including the ability to whitelist acceptable components and software versions; alert or respond to security issues based on impact; and address contractual, licensing or other considerations.

Insights are not always presented only to human users alone, however. Feeding the findings of SCA directly into automation and integration with broader toolsets – from IDEs to operations management – is part of the overall trend to increase the level of automation in DevOps pipelines and further accelerate agile development and operations.

SECURITY VULNERABILITY RECOGNITION AND REMEDIATION

One of the key values of SCA – and arguably the one most directly relevant to software security – is the ability to identify vulnerabilities in software components, modules and projects on which development efforts are often built or based. While these components may include proprietary or closed elements, the emphasis of the SCA market is largely on open source, given the focal role it plays in modern software, and the visibility and impact of open source vulnerabilities.

Vulnerability inventory has long played a role in security, such as in the development of the US National Vulnerability Database, which provides a central clearinghouse for identifying specific vulnerabilities and tracking their severity, exploitability and impact. But to be listed in the NVD, a vulnerability must have already been researched and analyzed. Attackers, of course, aren't limited to the exploit of these issues – a 'zero day' vulnerability is, by definition, one that only becomes known when an attack for it appears. Independent research into software vulnerabilities thus becomes a differentiator for SCA providers that can give their customers early and actionable warning of such exposures.

Increasingly, this insight is also accompanied by guidance on how to resolve the exposure, whether through upgrading to a version in which the issue is resolved or illustrating how an organization can modify its software, if an upgrade is impractical or infeasible. This guidance can include specific implementations of source code, but these are often limited to the most serious issues, given the sheer scale of potential vulnerabilities across the breadth of modern software.

One of the ways that vulnerabilities often appear is through software dependencies, and these dependencies may appear at multiple levels of depth, given how one project may be built on another, which in turn is built on others, and so on. Dependency analysis has recently become more visible, and not only among SCA players per se. GitHub itself, which in early 2018 'acqui-hired' the Y Combinator-backed AppCanary vulnerability analysis team, introduced the Dependency Graph in late 2017, which provides the ability to track dependencies throughout software projects. Shortly thereafter, GitHub also introduced security alerts for owners and administrators of public repositories (and private repositories for those who have opted in to vulnerability detection) concerning public vulnerabilities in components such as Ruby gems, npm and Python packages.

One of the advantages of dependency tracking is the ability to locate and identify vulnerabilities, even if organizations are not aware that vulnerable components are part of their software landscape. This can help to prioritize risk and exposure, as well as help determine a proper approach to remediation, whether via upgrade, modifying existing sources, or configuring access and use. Automating the kickoff of processes such as policy exception and approval workflows are other ways in which SCA can help organizations resolve issues without disrupting the development or operation of high-priority business functionality.

Another aspect of vulnerability management in SCA has appeared in a discernment of effective exposure. A component vulnerability that isn't actually used in the implementation of a dependent project may be effectively dormant – present, but not posing a risk since it cannot be accessed, or an exploit would not lead to compromise since the functionality is not in use. These distinctions must be reasonably sure that exploit would, in fact, be a dead end, and not inadvertently reveal an otherwise unsuspected exposure.

This capability has become more evident in SCA – and not just as a means to help rein in false positives and achieve the always sought-after goal of identifying exposures that most urgently require action. One of the oft-touted values of SCA is the ability to halt a process that would result in the incorporation of vulnerabilities or other issues that would violate policy. But halting an agile pipeline can be one of the worst things that could happen to a modern organization – it could result in disruption of downstream expectations, significant cost and lost business opportunity. Issues that require immediate resolution must therefore be distinguished from those that do not necessarily threaten software production and operations, or the business itself. SCA can also automate the initiation of less disruptive processes to resolve issues as they arise.

RESOLVING LICENSING CONCERNS

The spectrum of licenses encumbering software can be much broader – and their intersections, conflicts and overlaps more complicated – than many organizations suspect. In part 1 of this report, we noted that the Open Source Initiative listed no fewer than 83 open source licenses on its website. Organizations and developers alike may assume – or declare – that software conforms to a particular license. In fact, it may be governed by additional licenses, or another license altogether, and there may be not only conflicts among these licenses, but disconnects between the declared license(s) and the way the software is actually used.

As if that weren't complication enough, software products may in turn be utilized by additional organizations in subsequent developments – both inside an enterprise and third parties. Licensing conflicts and issues may dog software well beyond an initial implementation – and may come back to haunt any organization in the sequence of development that built on erroneous assumptions. This can be a particular bedevilment in cases such as mergers and acquisitions, where an acquiring organization may assume the obligations of the target – including proper licensing of software. SCA thus becomes part of due diligence to deliver a thorough inventory of the license conditions and obligations associated with the intellectual property of an acquisition, which factors into the liabilities, assets and opportunities acquired.

SCA can confirm that software use agrees with the declared license, and identify when use varies from that license. It can analyze licensing across all components of software to identify the scope of licenses that apply; identify conflicts; and (as with other aspects of policy noncompliance) alert on issues, recommend remediation or automate response.

EXAMPLE VENDORS

The following is a sampling of vendors offering software composition analysis, listed in alphabetical order. In addition to pure plays, these example vendors include those that have developed or acquired SCA to expand software supply chain management or an application and software security portfolio. In the latter case, SCA particularly aligns with static analysis of source code.

- CA Technologies acquired SourceClear in early 2018 to anchor its SCA strategy and align with the assets of its 2017 Veracode acquisition in application security – particularly its static analysis capabilities for source code testing. That alignment is further reinforced through SourceClear's founder, Mark Curphey, who was also the founder of OWASP. SourceClear emphasizes its investment in analytics, pointing to its Security Graph Language, which it claims is the industry's first domain-specific language designed to identify security issues in open source code. SourceClear uses a call-graph to trace a project's use of an OSS library to a specific line of code, which helps it identify when software is invoking a specific vulnerability. SourceClear is a fully SaaS offering, supporting multiple languages and integrating with a number of package managers and CI/CD tools. Features include license risk analysis, real-time vulnerability detection, analysis of vulnerable methods, a library catalog and a CI/CD agent. Tight coupling with DevOps toolchains enables SourceClear to control pipelines. Its analysis of vulnerability impact enables organizations to consider alternative approaches to remediation that don't require failing a build, which helps improve security's support for agile objectives.
- Flexera (fka Palamida) was one of the original companies to provide software license and compliance scanning for open source software. When it was Palamida, the company was primarily focused on code scanning and management, with some M&A vetting business, as well. Prior to its acquisition by Flexera in 2016, the company had more formally made security a focus by specializing in scanning open source software for vulnerabilities, in addition to license and other information. Flexera still offers software composition analysis and license optimization, but its focus goes well beyond open source software to include management of cloud, vulnerabilities, patches and edge computing. Flexera was among the first code management vendors to embrace security aspects of software releases, and the company still offers application security software. Flexera sells software and services, including consulting and training, and specializes in software supply chains for certain industries, including financial services, education, government, manufacturing and utilities. In 2015, Flexera acquired Secunia Research, which specializes in vulnerability intelligence, provides technology for vulnerability assessment across multiple platforms and helps organizations remediate software vulnerabilities. Secunia provides verification, testing and validation for Flexera's software vulnerability research.
- Snyk is a relative newcomer to SCA, but has made a splash based in part on its approach to reaching the open source development community, as well as its approach to vulnerability remediation. Snyk's offering is directly accessible from community repositories such as GitHub or npm, via API or from the Snyk website. Assessment may be as simple as validation of every GitHub pull request for newly introduced vulnerable packages, continuous scanning of repositories for known vulnerabilities, scanning a local project using a simple command line interface, or as part of continuous integration. Vulnerabilities are correlated to dependencies mapped by Snyk, as well as remediation guidance, which may suggest available upgrades or patches when upgrade is infeasible. Snyk for Enterprise offers these capabilities, plus dashboards for reports across an organization, license management, groups (including single-sign-on support) and issue-tracking through widely accepted tools such as Jira and Slack, as well as API-based access for custom enhancements and automation. Founded in 2015, the company is based in London with offices in Tel Aviv and Boston.
- Sonatype is among the early entrants in SCA. As an original core contributor to Apache Maven and curator of the Maven Central Repository, the company has strong ties to developers in the open source community. Today, Sonatype's products extend across the entire software supply chain and enable organizations to implement different levels of automated open source governance. Founded in 2008, the Fulton, Maryland-based company is now known for its support of all popular OSS languages and integration into leading DevOps tools. The Sonatype Nexus portfolio consists of three key anchors: Nexus Lifecycle, which helps organizations integrate intelligence, policy and control across a wide range of DevOps automation tools and processes; Nexus Firewall, which automatically enforces OSS policies by blocking undesirable OSS components at the earliest point in the development lifecycle; and Nexus Repository, which provides centralized management for OSS components, containers, build artifacts and release candidates.

- Synopsys gained a substantial share of the SCA market when it acquired Black Duck Software in 2017 to bolster its Software Integrity Group's application security portfolio. Prior to the addition of Black Duck, Synopsys' SCA offering, formerly known as Protecode, was somewhat limited, but maintained differentiation through its ability to analyze binaries, which is valued by organizations that don't have access to source code or may only see binaries late in the software supply chain. Black Duck, however, has broad penetration of SCA overall. Today, Black Duck by Synopsys includes the Protecode binary analysis technology and offers a broad range of capabilities, including open source software inventory, vulnerability mapping, identification and management of license and quality risks, open source risk policy management, and proactive alerting for newly detected security threats. Black Duck Hub provides centralized management for these capabilities, while OpsSight reflects a recent expansion of capability for assessing containers and alerting on newly discovered container vulnerabilities. Black Duck provides coverage for multiple languages and integrates with a wide range of DevOps tools, such as IDEs, CI tools and code repositories, as well as the Kubernetes and OpenShift container management platforms.
- WhiteHat Security acquired Infrared Security's static application security testing technology in 2011, and today offers both static (SAST) and dynamic (DAST) application security testing. Noting that an application is only as secure as its weakest elements, WhiteHat sees the high proportion of open source components in modern applications, and offers SCA integrated with SAST in its WhiteHat Sentinel Source offering, with SCA made available to Sentinel Source customers. Both types of tests are invoked from, and results delivered to, the WhiteHat Sentinel platform, which unifies assessment and analysis. The SCA functionality offers reporting, targeted at developers, who require detail on affected components for identification of vulnerabilities and appropriate remediation, in addition to high-level trend reports that cater to senior IT and security executives, as well as to software project managers.
- WhiteSource became an early entrant in SCA – advancing a concept of continuous open source component management and real-time detection of vulnerability and licensing issues whenever code is built, committed or utilized – in order to overcome then-current limitations of approaches such as periodic (and often incomplete) scanning. Today, WhiteSource Open Source Management offers solutions for both governance (policy enforcement, reporting, etc.) and developers that fit into the developers' natural ecosystem – from IDEs to the browser (often used to discover code examples or development guidance, for example) – to provide actionable insight into vulnerabilities and licensing issues, as well as pragmatic guidance for remediation. More recently, WhiteSource introduced effective usage analysis to provide detail on how components are used in software, which helps to prioritize vulnerability identification and remediation. Coverage includes support for container assessment – a growing enterprise demand – as well as for open source applications, modules and projects. WhiteSource offers extensive language coverage and has technology partnerships with other recognized names in application and software security, including Checkmarx and IBM, and offers an integration with Micro Focus (formerly HP) Fortify.

TOWARD THE FUTURE

There is somewhat of a parallel between the open source boom and the early days of SaaS and the public cloud, when the adoption of these new and disruptive resources was as simple as providing a credit card for whatever a development or business team needed. In the case of open source, that model extends from inexpensive to free. Open source projects abound and are often the seeds of new efforts for individual vendors, as well as for corporate development. Given the universal and seemingly unquenchable thirst for new technology, this model is one of the primary drivers of the open source revolution.

But such a disruptive model introduces its own risks. In the cloud, the accumulation of just a few dollars here and there has led to the Jevons paradox. With open source, this has led, for example, to the fact that there may be no central authority or agency of any sort responsible for assuring the mitigations of risk that may be introduced by any one developer or component, which can drive cost, complexity and risk beyond expectations.

This openness, however, introduced an opportunity for SCA to thrive. SCA vendors are now embracing the factors that have driven open source adoption itself. Accessing tools for dependency and vulnerability analysis from springboards such as GitHub or exposing their own API has become a market-penetration vehicle for some, giving a wide number of users a taste for SCA's benefits, and whetting an appetite for enterprise-class functionality.

As the demand for new technology parallels a growing awareness of its risks, we expect SCA to continue to prosper. We expect further acquisitions of players in the space, particularly among those that most closely reflect the values of open source adoption and practice, as well as those that need to invest in thought leadership to refresh portfolios that may be showing their age with respect to today's realities of development and IT operations.

We further expect SCA to reflect a growing awareness of the need to apply advances in automation and analytics to solve some of security's most long-standing problems with vulnerability awareness and remediation. An advantage of SCA is the ability to address many of these issues before they become incorporated in software. Uniting these efforts with advances in operational vulnerability awareness and mitigation is desperately needed, before a profusion of technologies permeating every aspect of everyday life poses threats as yet unimagined.