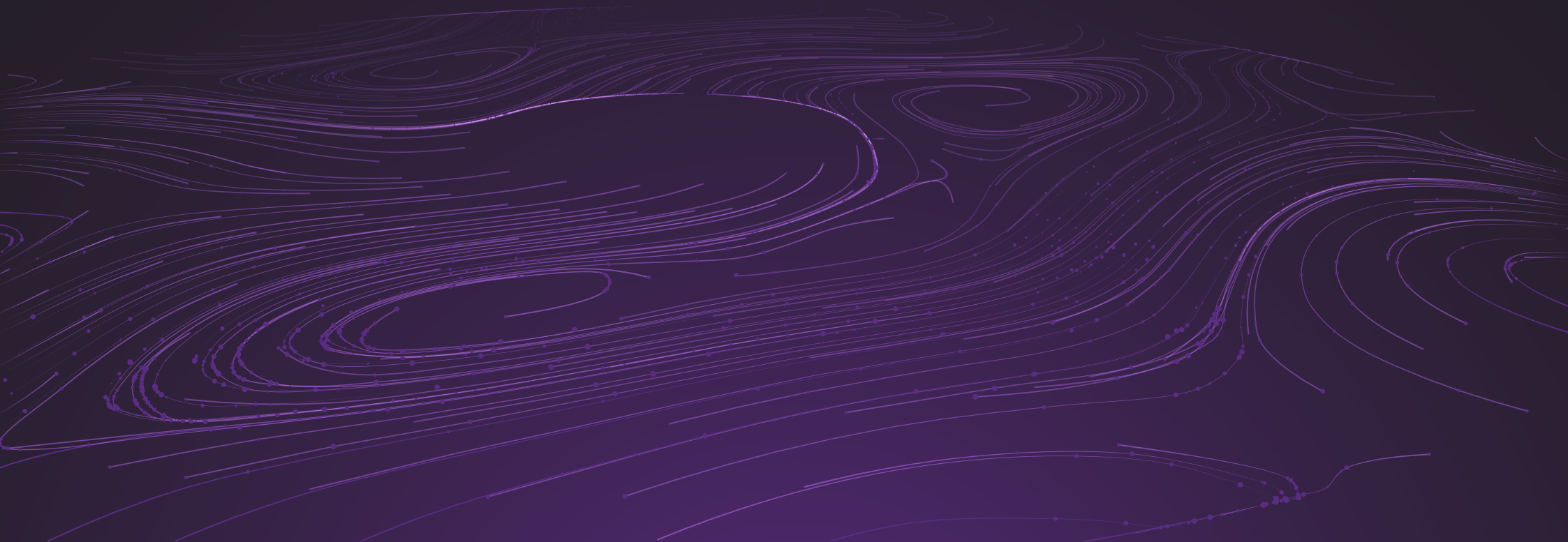
SYNOPSYS®

2021 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT

TABLE OF CONTENTS

INTRODUCTION



ABOUT THE CYRC AND THE 2021 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT

The Synopsys Cybersecurity Research Center's (CyRC) mission is to publish security advisories and research to help organizations better develop and consume secure, high-quality software. Our most recent security and software quality reports include "Peril in a Pandemic: The State of Mobile Application Security," an analysis of the most popular Android apps used during the COVID-19 pandemic; and "DevSecOps Practices and Open Source Management in 2020," a survey of software professionals on open source management and DevSecOps.

This research, the CyRC's annual "Open Source Security and Risk Analysis" (OSSRA) report, provides an in-depth snapshot of the current state of open source security, compliance, licensing, and code quality risk in commercial software.

For over 17 years, security, development, and legal teams around the globe have relied on Black Duck® software composition analysis (SCA) solutions and Audit Services. Our SCA solutions help organizations identify and track open source code and automate the enforcement of open source policies through integration with currently used DevOps tools and processes. Our Audit Services team conducts audits on thousands of codebases for customers each year, both to support merger and acquisition (M&A) transactions and to provide customers with a comprehensive, up-to-date Bill of Materials of the open source, third-party code, web services, and APIs used in their applications.

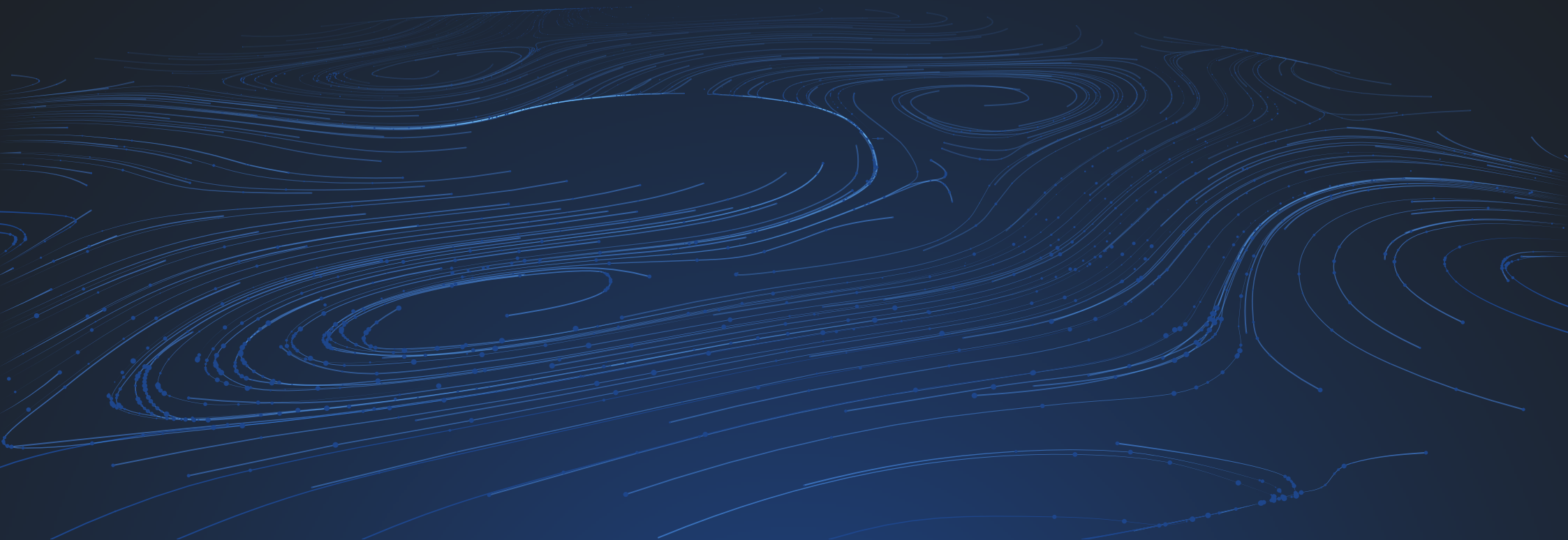
The audit data is cross-referenced with the Black Duck KnowledgeBase™ to identify potential license compliance and security risks as well as other factors that may affect the overall codebase. Curated by the CyRC, the KnowledgeBase houses data on millions of open source libraries from over 24,000 forges and repositories.

Audits are the primary source of data for the 2021 OSSRA report. Additional data used in the report (specifically in the "Parallels between the 'State of Mobile Application Security' and OSSRA reports" section and the conclusion) comes from Black Duck Binary Analyses and Coverity Scan®. The 2020 audit data analysis used in this report was conducted by the CyRC's Belfast and Boston teams. In addition to validating data used in the OSSRA, the Belfast team's work forms the basis of Black Duck Security Advisories (BDSAs), which offer enhanced vulnerability information that the team publishes as a service to commercial Black Duck customers.

This year, the CyRC teams examined anonymized audit findings from over 1,500 commercial codebases in 17 industries. You need look no further than the pages of this report to see that open source libraries are the foundation for literally every application in every industry. But paralleling the popularity of open source is a growth in risk—specifically around open source licensing, security, code quality, and maintenance.

This sixth edition of our report, the 2021 OSSRA, includes recommendations to help open source developers and consumers better understand the software ecosystem they are part of, as well as the risks that come with unmanaged open source development and use.

OVERVIEW



1,546
CODEBASES
AUDITED IN 2020

17 INDUSTRIES REPRESENTED

98
PERCENT OF
CODEBASES CONTAINED OPEN SOURCE

84%
OF CODEBASES HAD AT
LEAST ONE VULNERABILITY
WITH AN AVERAGE OF

158
PER CODEBASE

65%
OF CODEBASES
HAD LICENSE
CONFLICTS

THE AVERAGE
VULNERABILITY
FOUND WAS

2.2
YEARS OLD

75%
OF ALL CODEBASES WERE
COMPOSED OF OPEN SOURCE

TERMINOLOGY USED IN THIS REPORT

Codebase
The code and associated libraries that make up an application or service.

Binary analysis
A type of static analysis that examines the software of an application when access to the source code isn't possible.

Black Duck Security Advisory (BDSA)
A classification of open source vulnerabilities identified by the CyRC security research team. BDSAs provide Synopsys customers with early and/or supplemental notification of open source vulnerabilities and upgrade/patch guidance.

Software library
Prewritten code that developers can add to their software. A software library might be a utility, such as a calendar function, or a comprehensive software framework supporting an entire application.

Dependency
A software library becomes a dependency when other software uses it—that is, when software becomes dependent on that library. Any given application or service may have many dependencies, which themselves may be dependent on other libraries.

Open source license
A set of terms and conditions stating end-user obligations when an open source library is used in software, including how the library may be used and redistributed. Most open source licenses fall into one of two categories:

- **Permissive license**
  A permissive license allows use with few restrictions. Generally, the main requirement of this type of license is to include attribution of the original code to the original developers.
- **Copyleft license**
  This type of license generally includes a reciprocity obligation stating that modified and extended versions are released under the same terms and conditions as the original code. Commercial entities are wary of including open source with copyleft licenses in their software, as its use can call the overall codebase's intellectual property (IP) into question.

Bill of Materials (BOM)
A comprehensive inventory of the open source dependencies in a codebase, often generated by a software composition analysis tool. A BOM lists all the open source, associated licenses, versions in use, download locations for libraries/dependencies, and subdependencies the dependencies link to.

Software composition analysis (SCA)
A type of application security tool used to automate the process of open source software management. SCA tools identify the open source used in a codebase, provide risk management and mitigation recommendations, and perform license compliance verification.

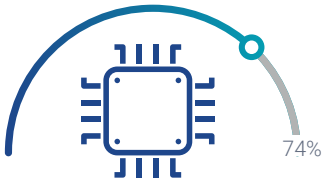
Static analysis
Also referred to as static application security testing (SAST), automated static analysis is used to identify coding flaws within nonrunning (static) code. Static analysis is an important part of the software development life cycle (SDLC) and is commonly used by most software development teams.

INDUSTRY SECTORS AND OPEN SOURCE

Percentage of Codebases That Contain Open Source, by Industry

Aerospace, Aviation, Auto, Transportation, Logistics

Big Data, AI, BI, Machine Learning

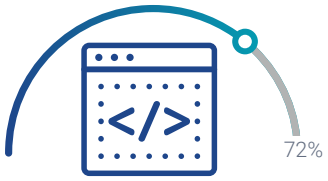


Computer Hardware and Semiconductors

Cyber Security

Ed Tech

Energy and Clean Tech



Enterprise Software/SaaS

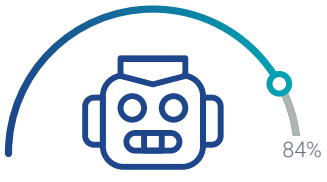
Financial Services and FinTech

Healthcare, Health Tech, Life Sciences

Internet and Mobile Apps

Internet and Software Infrastructure

Internet of Things



Manufacturing, Industrials, Robotics

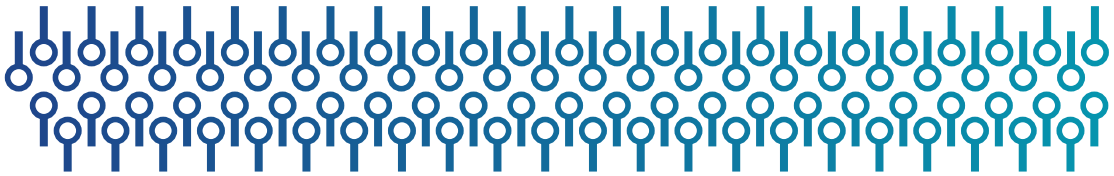
Marketing Tech

Retail and E-Commerce

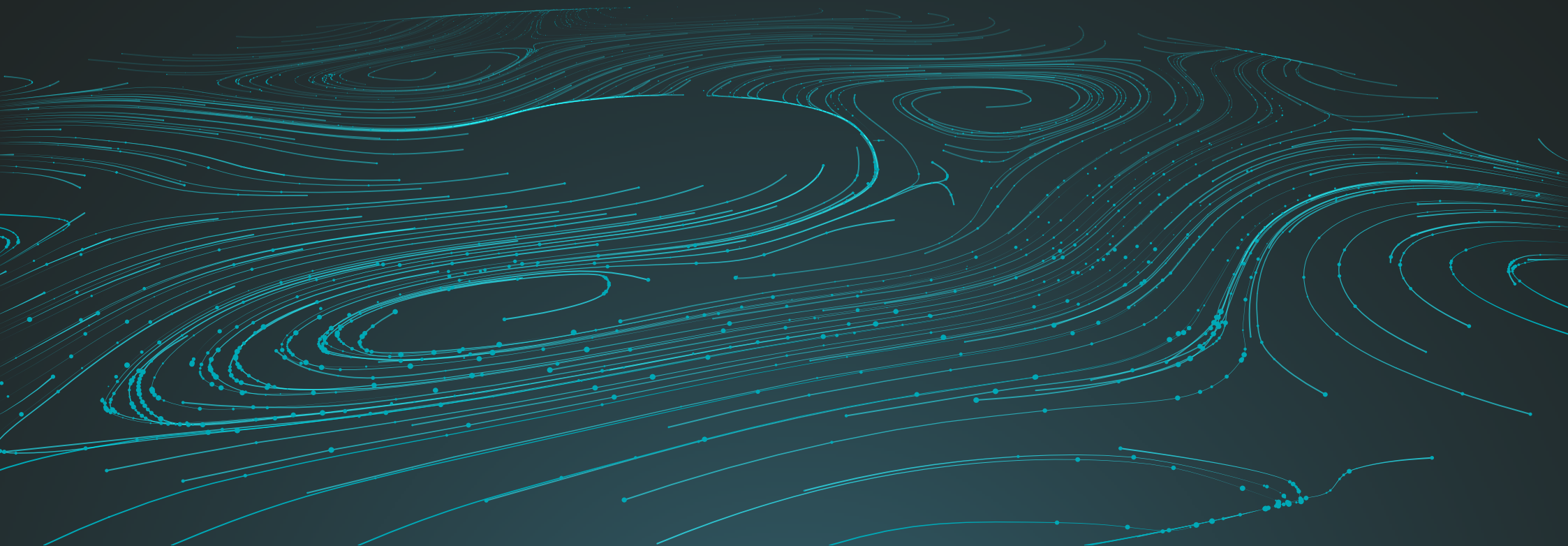
Telecommunications and Wireless

Virtual Reality, Gaming, Entertainment, Media

Average of 91 Codebases Audited per Industry



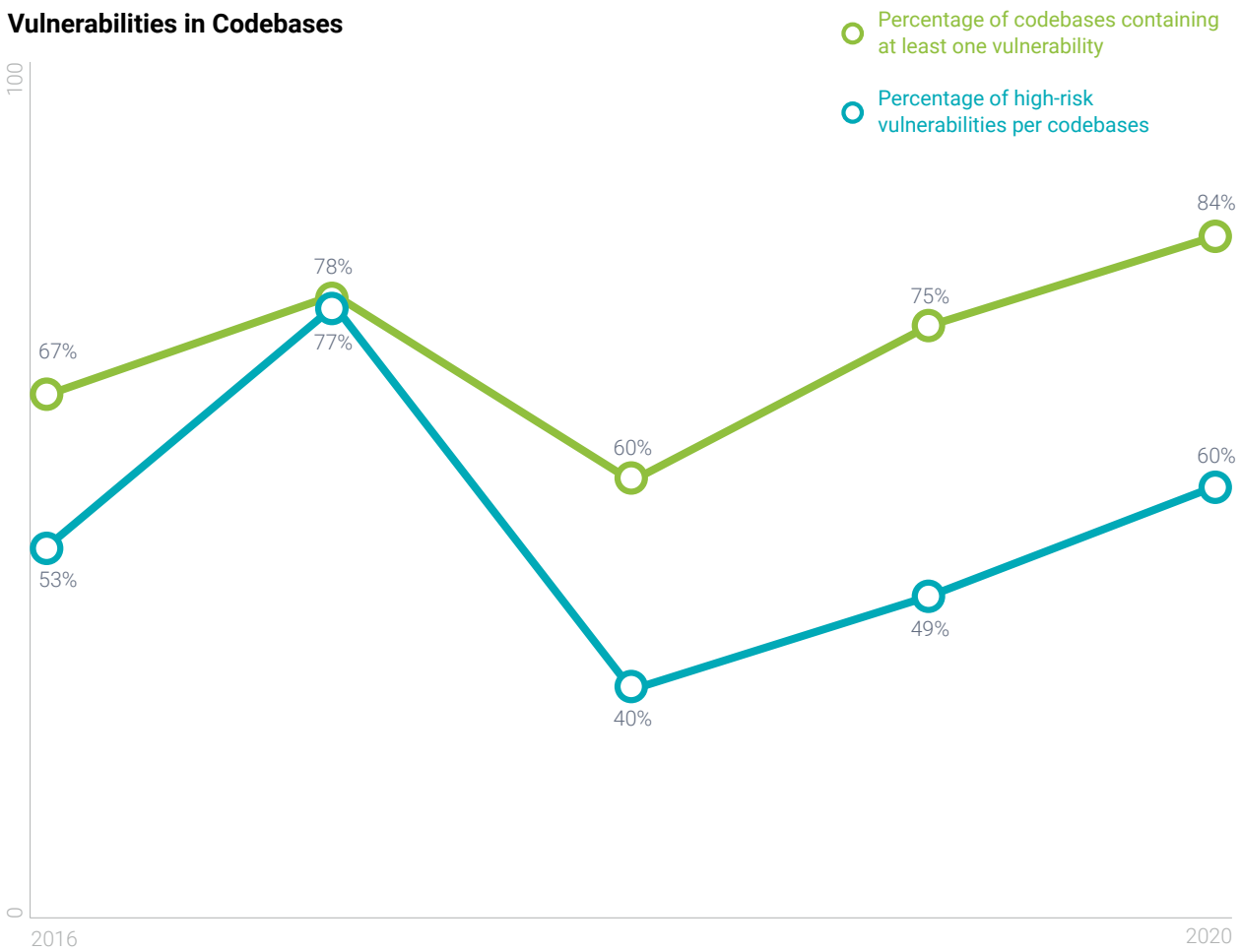
VULNERABILITIES AND SECURITY



OPEN SOURCE VULNERABILITIES AND SECURITY

A full 84% of the 1,500+ codebases Black Duck Audit Services audited in 2020 contained at least one public open source vulnerability—a 9% increase from the 75% of 2019 and the second-highest increase seen since 2017. Similarly, the percentage of codebases containing high-risk open source vulnerabilities increased to 60% in 2020, a dramatic 11% increase from the 49% of the 2019 audits. "High-risk" indicates that a vulnerability has been actively exploited, has documented proof-of-concept exploits, or has been classified as a remote code execution vulnerability. Several of the top 10 open source vulnerabilities that were found in codebases in 2019 reappeared in the 2020 audits, all with significant percentage increases.

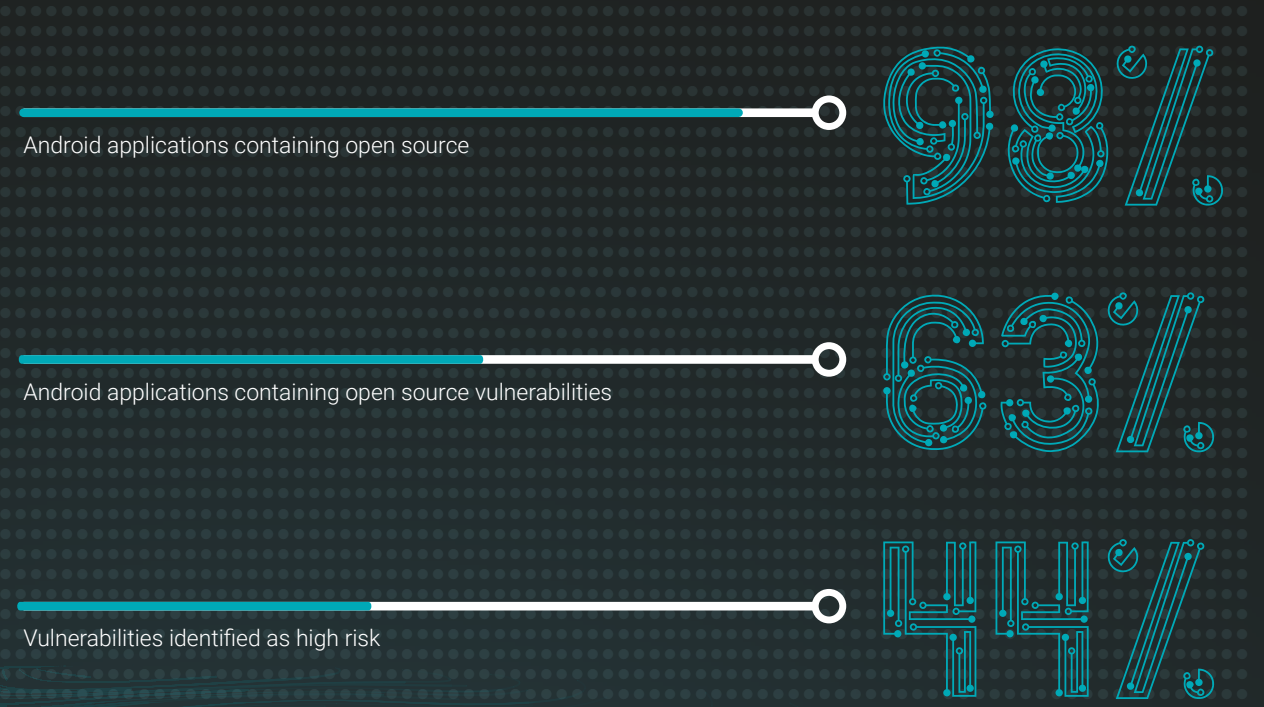
Vulnerabilities in Codebases



PARALLELS BETWEEN THE 'STATE OF MOBILE APPLICATION SECURITY' AND OSSRA REPORTS

The OSSRA results parallel the findings of the CyRC's 2021 "Peril in a Pandemic: State of Mobile Application Security" report. For that report, CyRC researchers used binary analysis to scan over 3,000 of the most popular Android applications in the Google Play Store. Over 98% of those applications contained open source—and 63% contained vulnerable open source libraries. Nearly half of the open source vulnerabilities found in that report were identified as high risk.

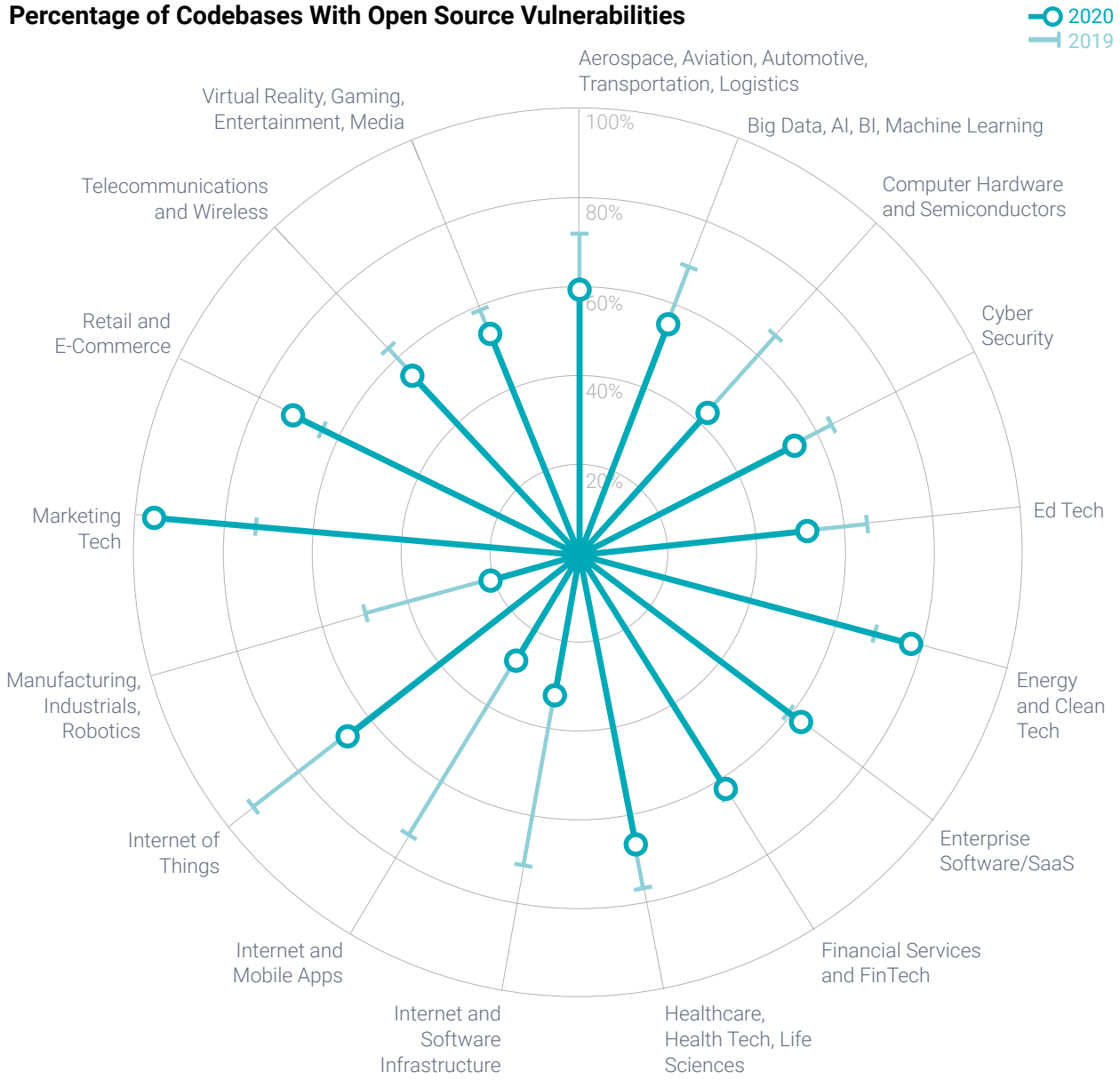
The "Peril in a Pandemic: State of Mobile Application Security" report shows the clear impact the COVID-19 pandemic has had on the growth of app downloads, as well as a corresponding likelihood that open source vulnerabilities are present in those apps. Similarly, the number of open source vulnerabilities increased in the audits reported in the 2021 OSSRA, and that increase is especially pronounced when looking at industry breakdowns.



Despite lockdowns and work-from-home policies, businesses still need to seek prospects, close deals, communicate with and support customers—all of which engendered a significant increase in the use of customer relationship technologies during 2020. Veeva Systems, a cloud computing company serving the healthcare sector, noted that it experienced 10 times more usage of its customer relationship management products during the pandemic. The videoconferencing company Zoom emerged as one of the corporate success stories of 2020, as video meetings became an essential part of work and school. And retailer L.L. Bean saw its best revenue growth since 2011 and added 1 million new customers thanks to two hot retail segments fueled by the pandemic—comfort clothing and outdoor gear.

The OSSRA data notes that 100% of the companies audited in the marketing tech category—which includes lead-generation, CRM, and social media—contained open source in their codebases. Ninety-five percent of the marketing tech codebases also contained open source vulnerabilities. Seventy-one percent of the audited retail and e-commerce codebases contained vulnerabilities. Both the financial services/fintech and the healthcare industry sectors had codebases with open source vulnerabilities exceeding 60%.

Percentage of Codebases With Open Source Vulnerabilities

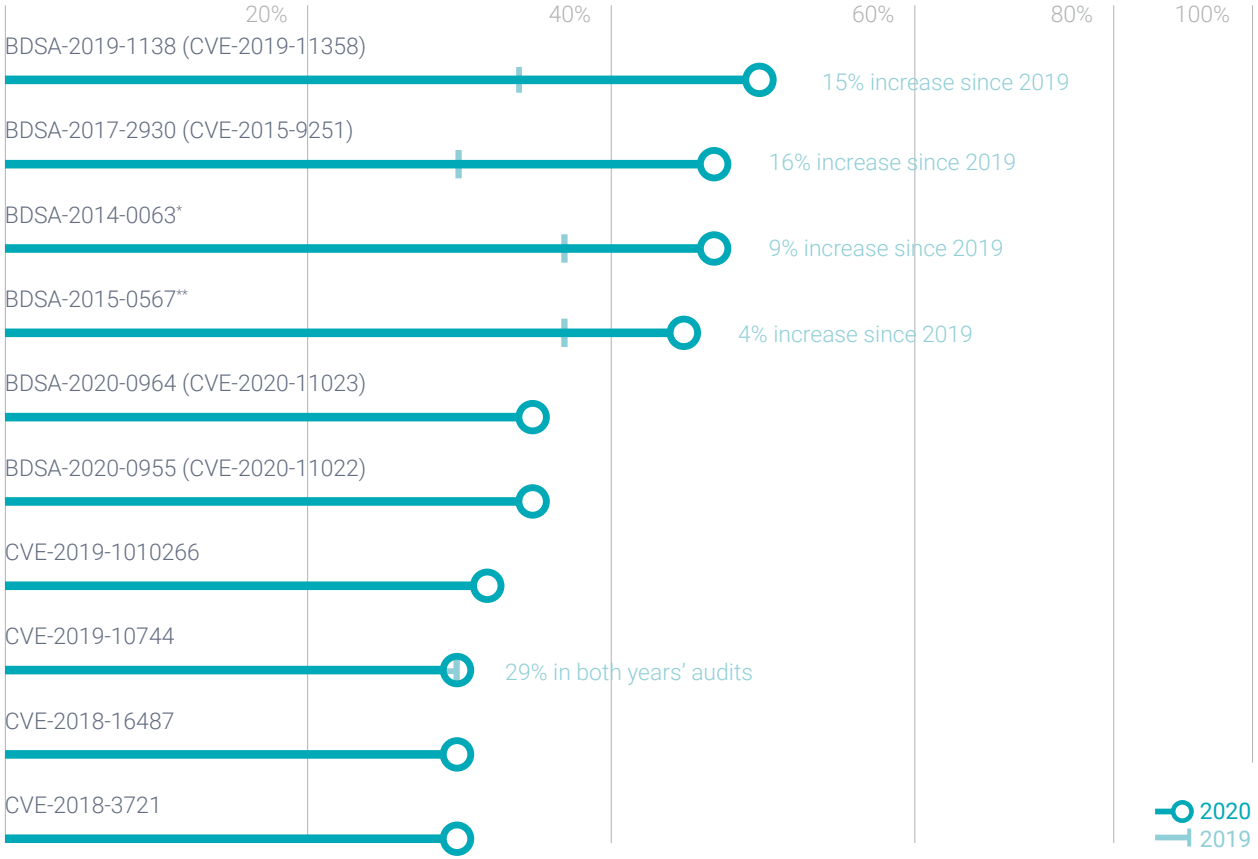


The top 10 vulnerabilities

Several of the top 10 open source vulnerabilities—including one that is a high-risk vulnerability—appearing in the 2019 codebases reappeared in the 2020 audits, some with significant increases in percentages. CVE-2019-10744, a lodash vulnerability rated by the National Vulnerability Database (NVD) as "critical" and affecting all versions of the popular JavaScript library prior to 4.17.12, appeared in 29% of both years' codebase audits.

Development teams appear to be struggling with the dynamic nature of open source security risk, especially with the increase in open source use. An open source library with no vulnerabilities doesn't necessarily stay that way a year or a month—sometimes not even a week—later. Access to reliable and diverse sources of vulnerability data is critical to staying atop of open source risk. Ideally, vulnerability information should be pushed to developer or security teams via the alert systems they already use, such as email, Slack, and Microsoft Teams.

Percentage of Codebases With Top 10 CVEs/BDSAs



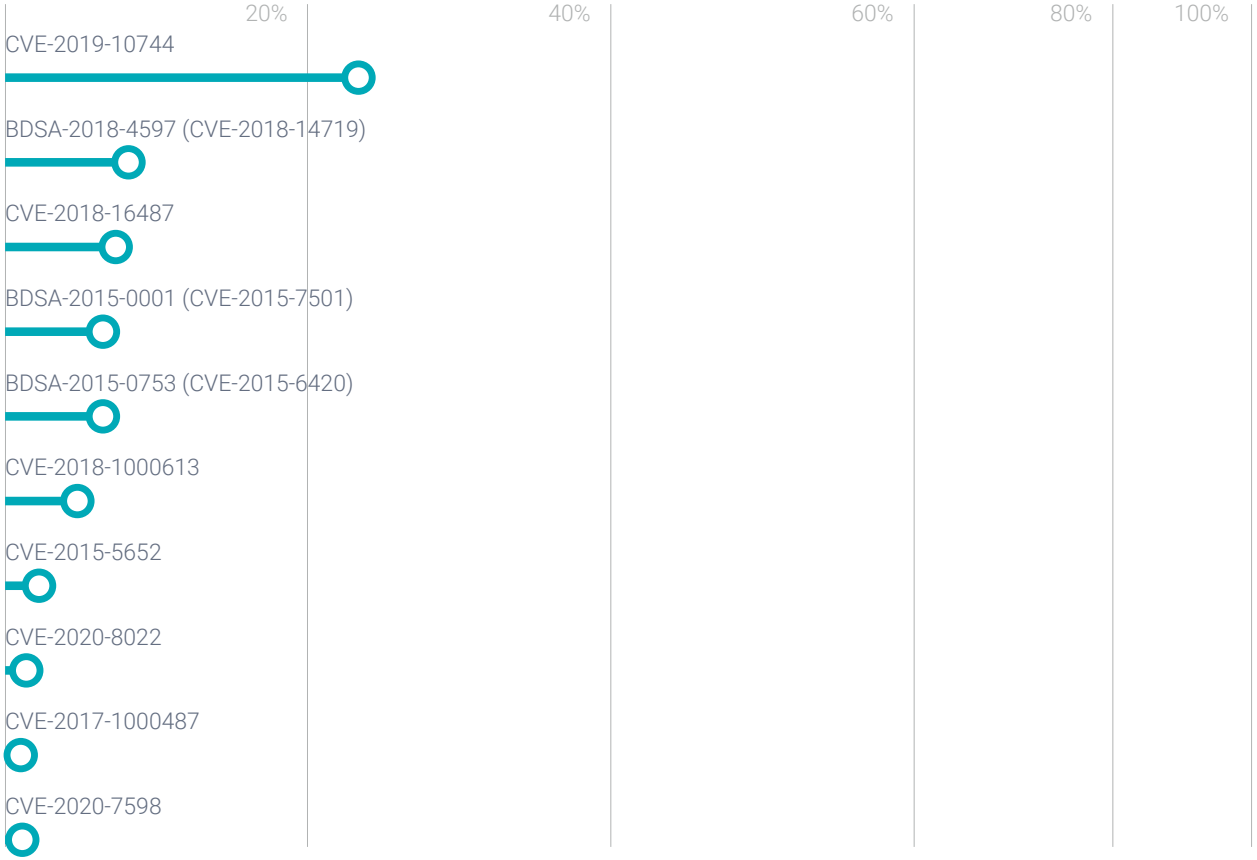
* BDSA-2014-0063 is a high-severity vulnerability in which jQuery is vulnerable to cross-site scripting (XSS) due to lack of validation of user-supplied input. A fix is available.

** BDSA- 2015-0567 affects all jQuery versions that use an unpatched UglifyJS parser, opening them to arbitrary code execution through crafted JavaScript files. A fix is available.

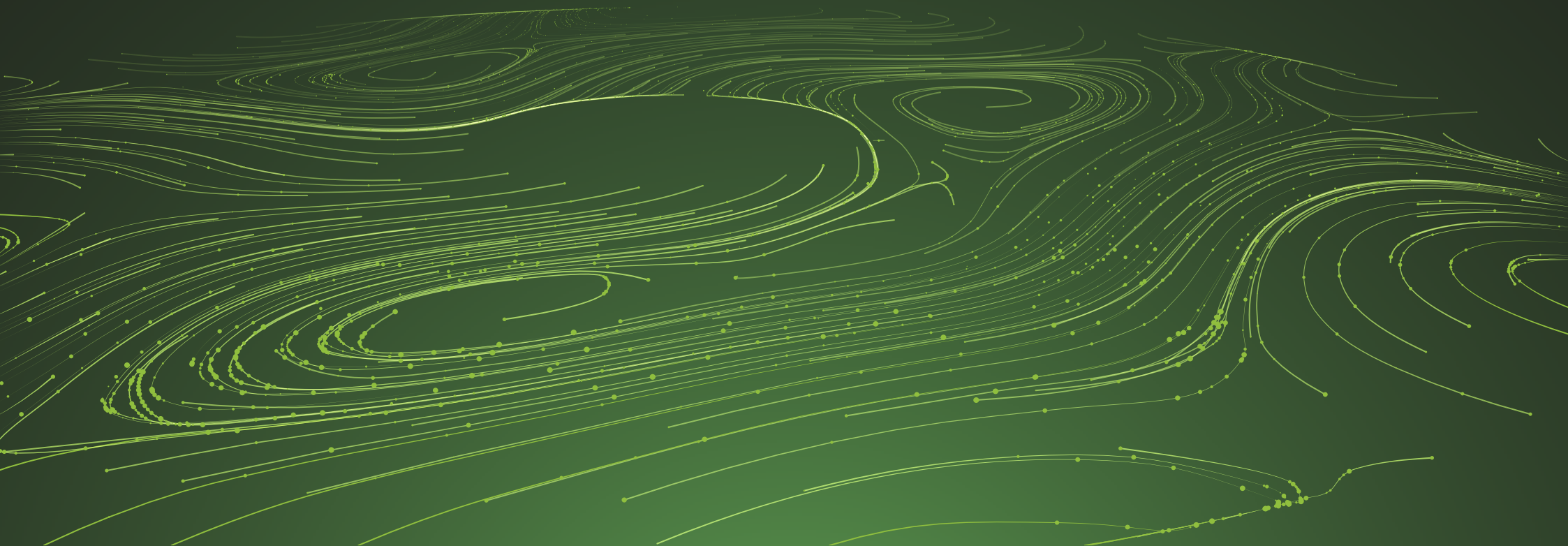
It's also possible that the knowledge that the codebase was dependent on a vulnerable open source library was buried somewhere inside the collective memory of the development team—possibly forgotten, possibly not documented at all. To fix an open source vulnerability, you first have to know the vulnerability is there. Pinpointing vulnerable open source depends on identifying and inventorying all open source you're using.

Most applications are dependent on hundreds of open source libraries—the average number of libraries found in the 2020 audits was 528 per codebase. An open source inventory or Bill of Materials automatically generated by a software composition analysis tool can provide the comprehensive information needed to address security risk.

Percentage of Codebases With Top 10 High-Risk CVEs/BDSAs



LICENSING



OPEN SOURCE LICENSING

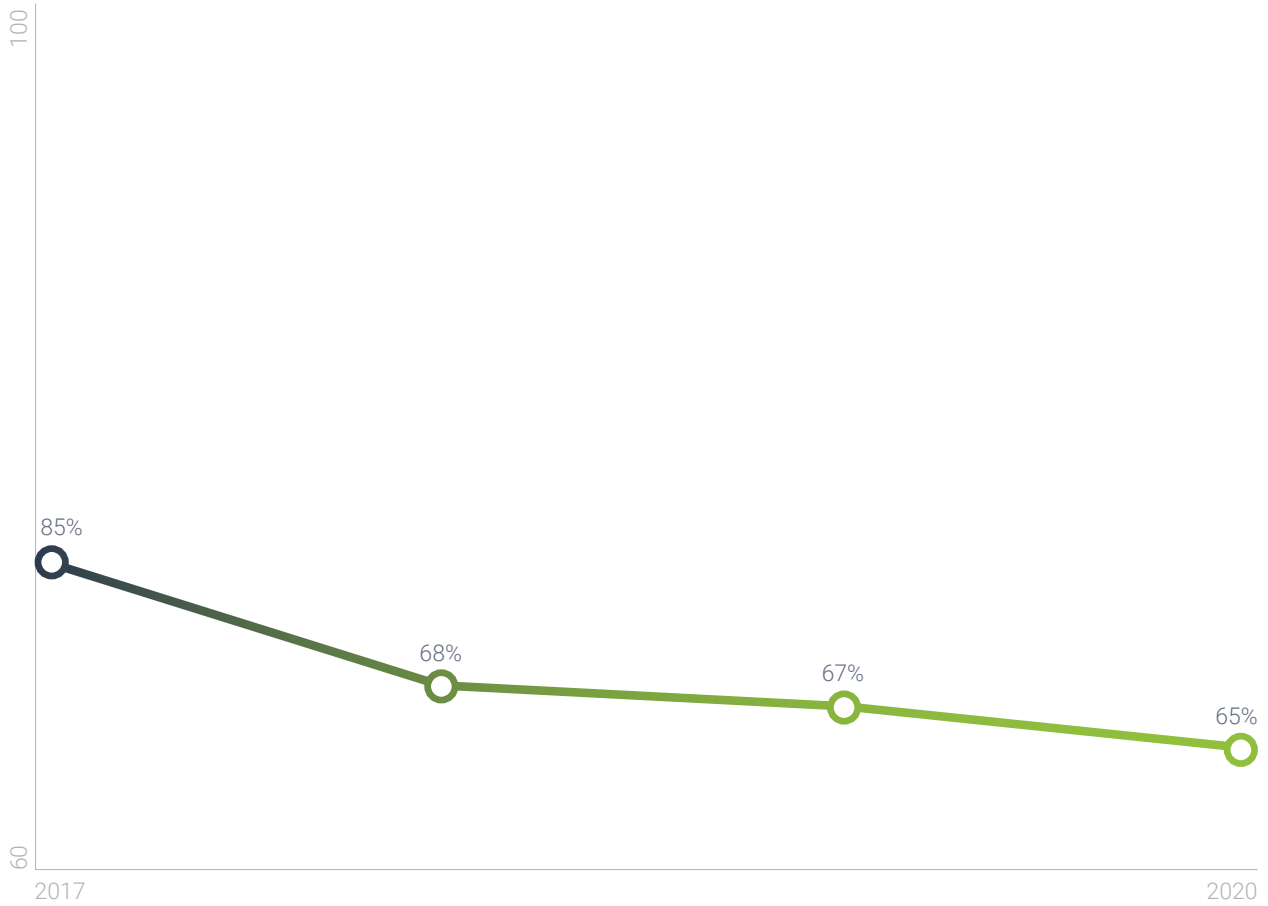
Black Duck Audit Services found that 65% of the 2020 audited codebases contained open source with license conflicts, a slight decrease from 2019. Nearly three-quarters of the codebases with a license conflict were specifically in conflict with one version or another of the GNU General Public License.

Twenty-six percent of the codebases were found to be using open source with no license or a customized license. Codebases with customized open source licenses need to be evaluated for possible IP and other legal issues. The JSON license, for example, essentially uses the permissive MIT license with the addition, "The Software shall be used for Good, not Evil." Owners of many popular projects—notably, Apache Foundation projects—have removed code using the JSON license because of the license's ambiguity; that is, although "software" is a defined term, "good" and "evil" are open to arguable interpretation.

Codebases that include open source dependencies with no discernable license also may require a decision about whether to replace those dependencies altogether.

Broken down by industry, the sectors with the highest percentage of codebases that contained open source license conflicts (86%) were the energy and clean tech sector and the manufacturing, industrials, robotics sector. The retail and e-commerce sector had the lowest percentage of codebases with open source license conflicts at 47%.

Percentage of Codebases With License Conflicts

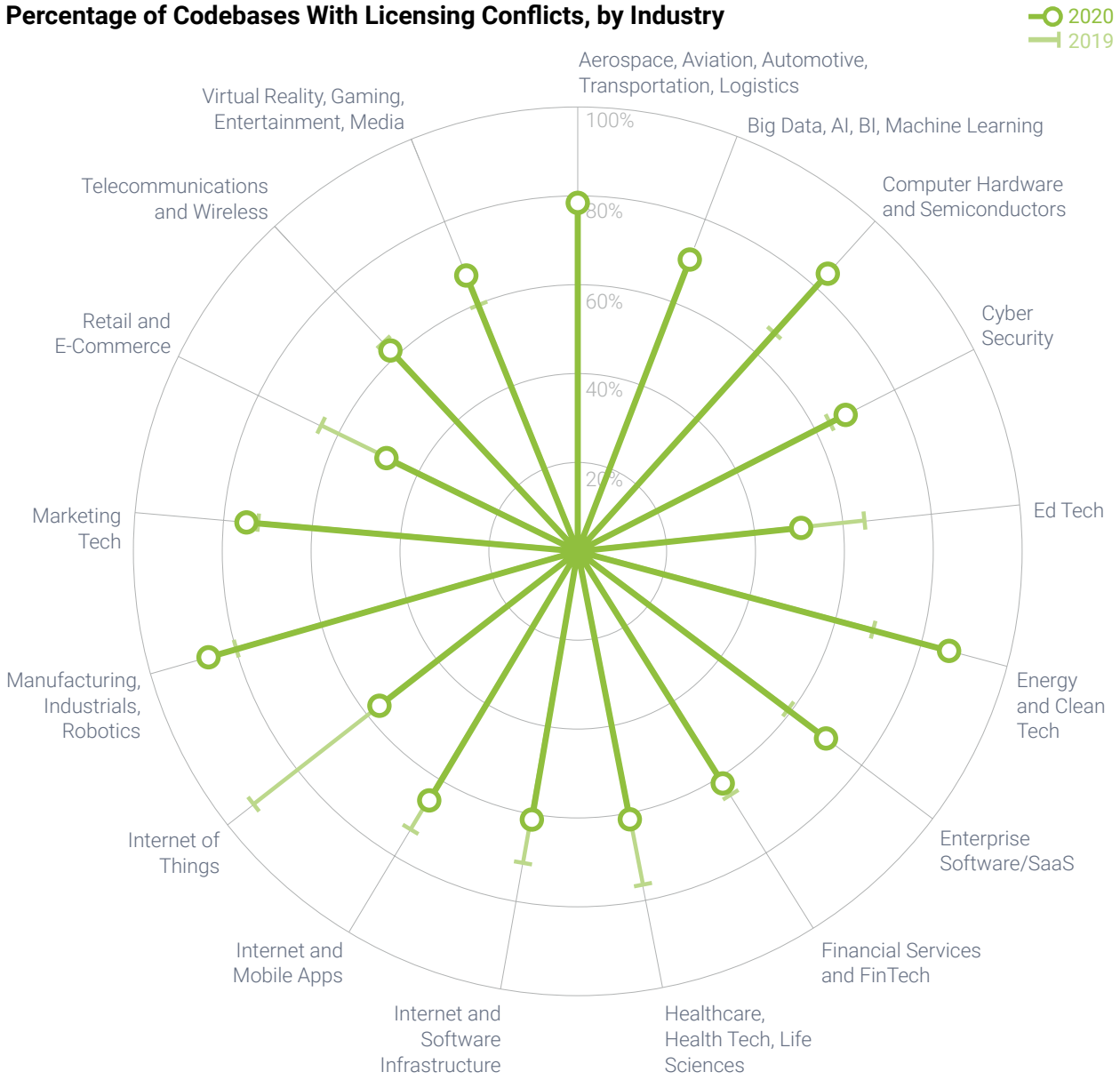


Understanding license risk

According to copyright law, using software in any way requires permission in the form of a license describing the rights conveyed to users and the obligations users must meet. Even the friendliest open source licenses include obligations the user takes on in return for use of the software.

Open source license litigation (including those for copyright, contract, antitrust, patenting, and fair use) is on the rise around the world. Potential license risk arises when a codebase includes open source with licenses that appear to conflict with the overall license of the codebase. The most common example of this is open source code under the GNU General Public License v2.0 (GPLv2), which often creates a conflict when compiled into a distributed piece of commercial software. But the same code isn't a problem in software considered software as a service (SaaS), because the GPL doesn't consider SaaS code to be "distributed." This isn't to imply that SaaS software is immune from license conflicts; some licenses can be problematic for SaaS as well.

Percentage of Codebases With Licensing Conflicts, by Industry

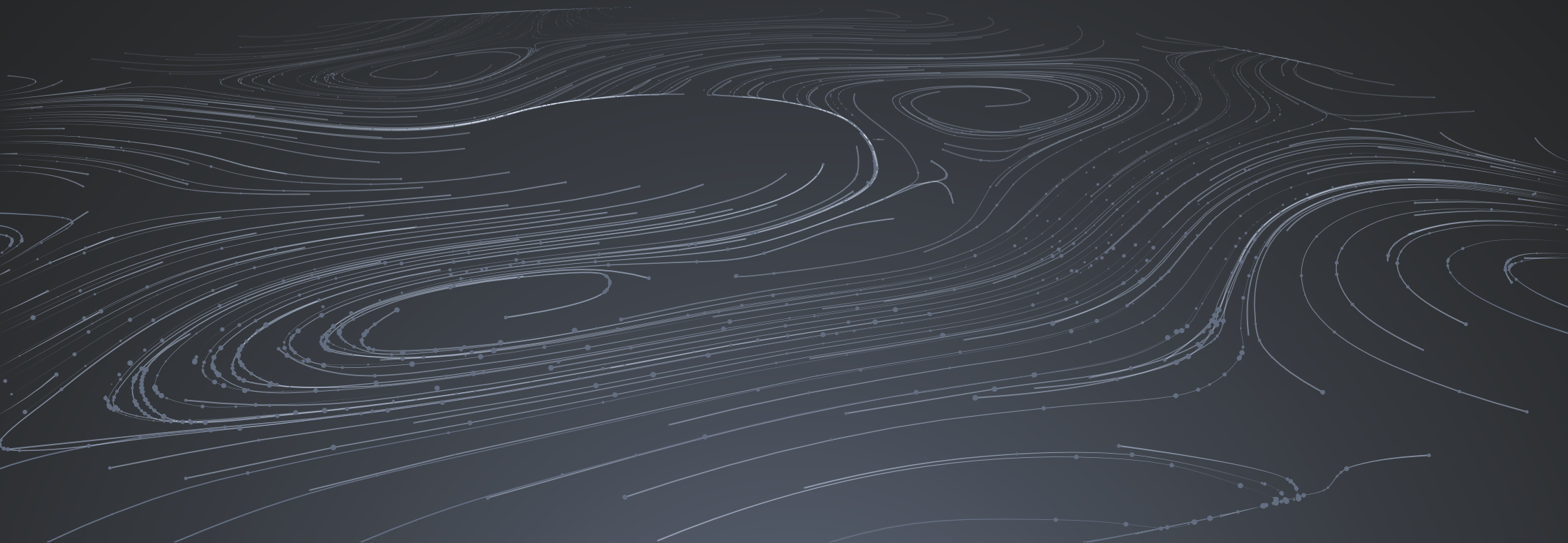


Sometimes an open source component has a so-called "custom license" in which the developer used their own licensing language or added language to a standard license. Such license additions are often well-intentioned but can raise concerns, especially in merger and acquisition transactions.

Whether open source or not, if third-party code doesn't have a license, serious legal issues can arise. In the U.S. and many other jurisdictions, creative work—including software—is placed under exclusive copyright by default. Unless there's a license that specifies otherwise (or the copyright holders grant permission), no other party can use, copy, distribute, or modify the software without the risk of litigation.

Percentage of Codebases Containing Open Source With No License or Custom License

SUSTAINABILITY

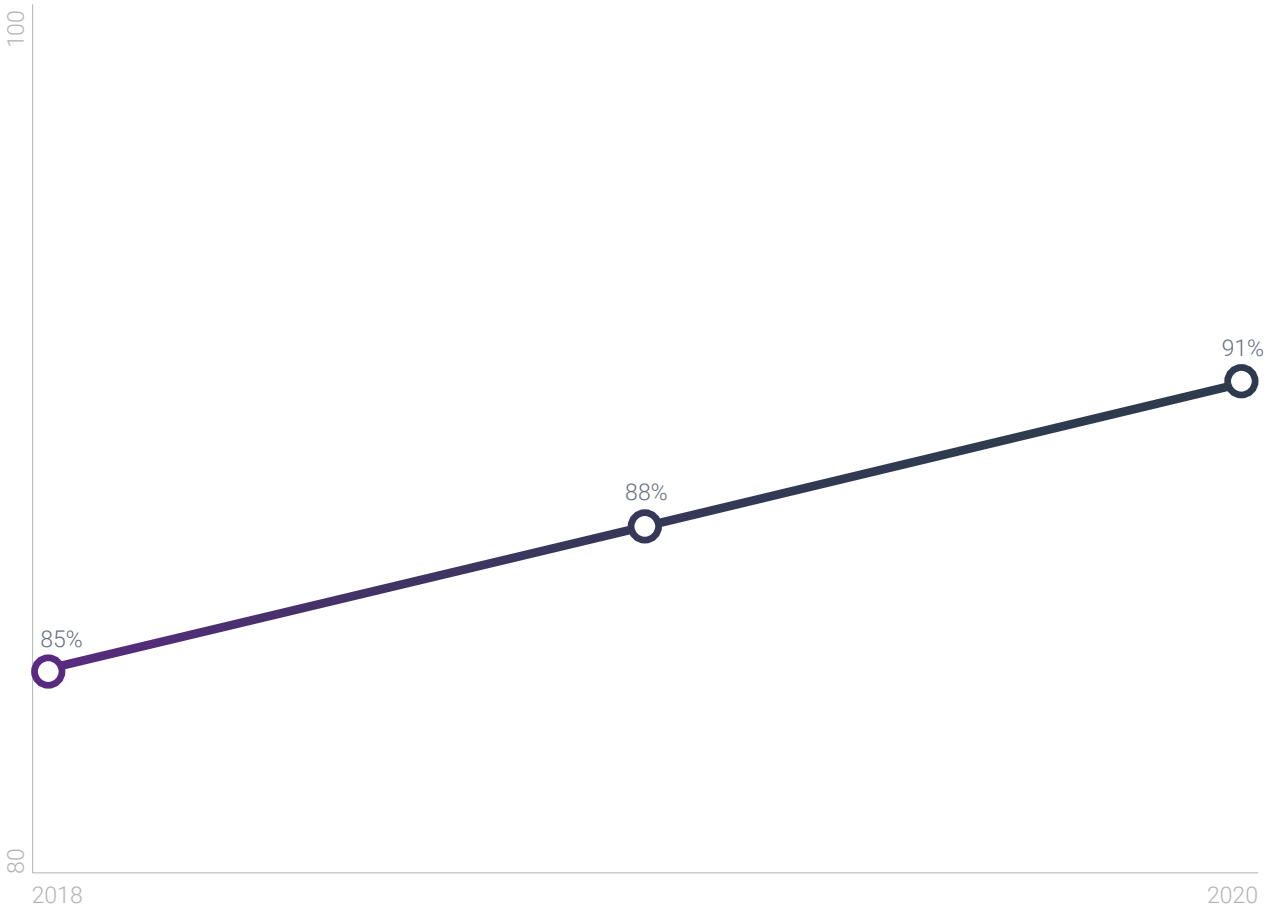


OPEN SOURCE SUSTAINABILITY

Of the 1,500+ codebases examined by Black Duck Audit Services in 2020, a staggering 91% contained open source dependencies that had had no development activity in the last two years. That figure means 91% of the codebases audited contained dependencies with no feature upgrades, no code improvements, and no security issues fixed over the past two years.

One of the reasons behind the popularity of open source is the volunteer communities continuously updating code and addressing vulnerabilities. Software developer and author Eric Raymond calls this Linus's Law in action: with many eyes looking at code, "all bugs become shallow." A Purdue University study showed that Linus's Law does work—open source communities regularly issue patches faster than their proprietary software counterparts. But there's no guarantee that the volunteer community behind any given open source project will continue maintaining the code indefinitely or that the community will continue to have members who are knowledgeable about the project's code.

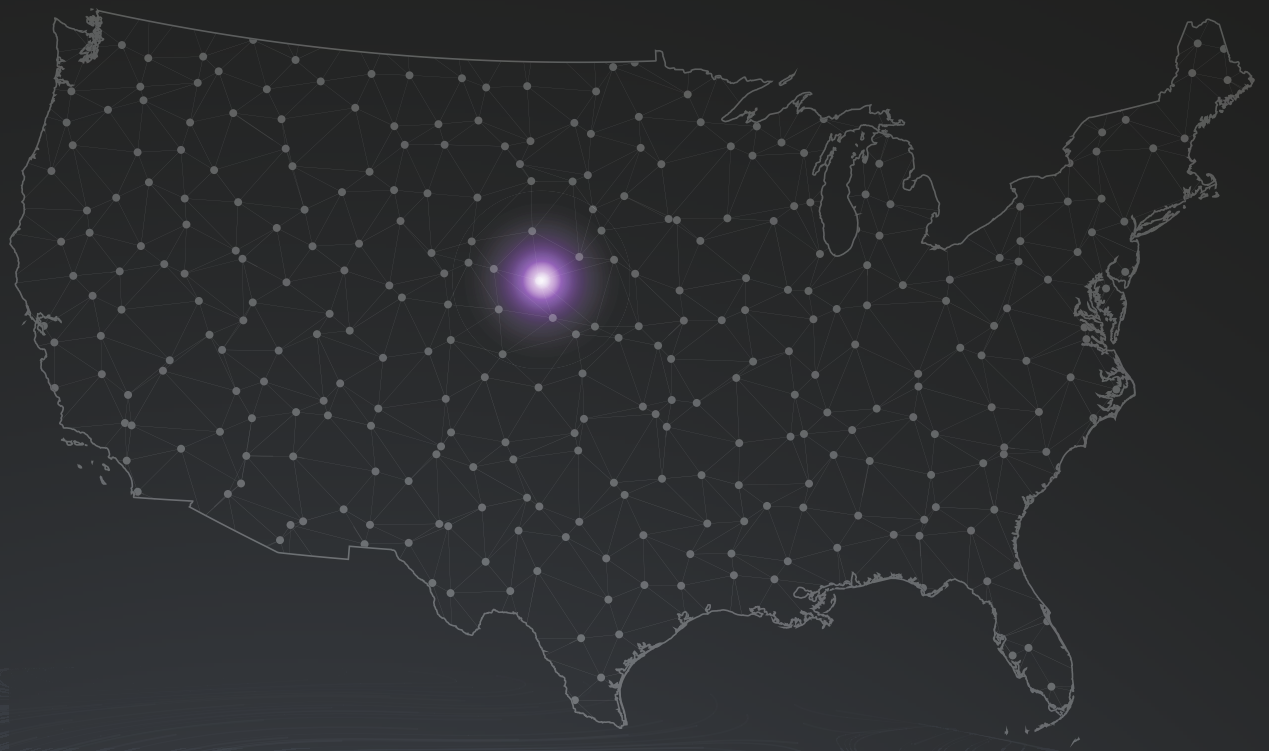
Percentage of Codebases Containing Components That Have Had No Development Activity Within the Past Two Years



THE PRICE OF POPULARITY

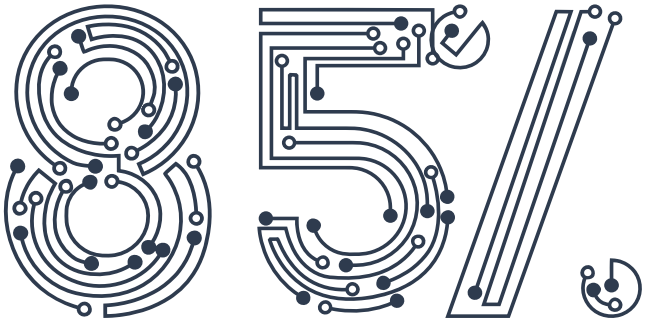
When an open source library becomes popular, the price is increased pressure on its (usually unpaid) maintainers—the people who handle bug reports, feature requests, code reviews, and code commits for the free software. It's not unusual for the maintainers to be a solo developer—a "random person from Nebraska," as the popular xkcd internet comic has it.

That random person is often the only bulwark supporting their piece of the open source infrastructure that modern software depends on. As an open source project grows in popularity—with no corresponding growth in people maintaining the project—the consequence is often developer burnout, and many open source projects are abandoned. The problem is severe enough that The Core Infrastructure Initiative was created to enable technology companies to collaboratively identify and fund open source projects that are in need of assistance, while still allowing developers to continue their work under the community norms that have made open source so successful.



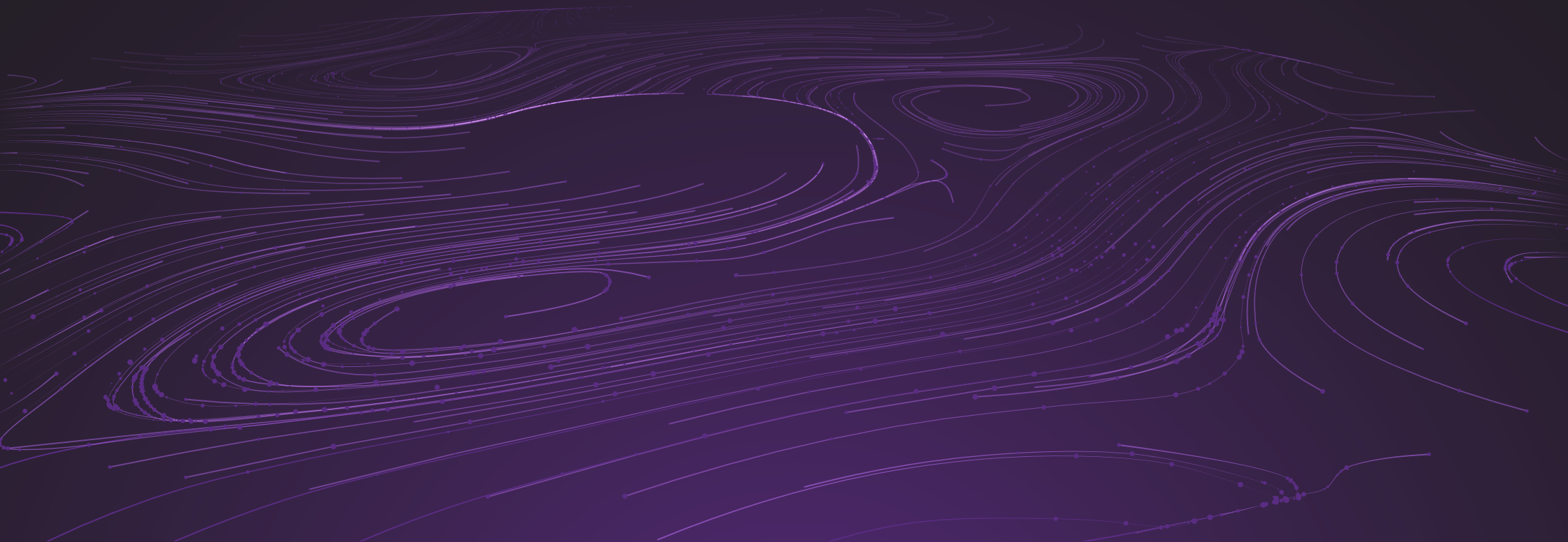
Eighty-five percent of the codebases Black Duck Audit Services examined in 2020 had open source dependencies that were more than four years out-of-date. That is, the codebases were using an open source library with newer versions available—often with many newer versions available. As noted earlier, it's clear that development teams are struggling to keep their open source dependencies up-to-date.

Returning to the lodash vulnerability mentioned in the "Top 10 vulnerabilities" section, 29% of the codebases in both the 2020 and 2019 audits contained CVE-2019-10744, even though an upgrade of the library that addresses the vulnerability has been available since July 2019. Did the developers determine that the risk was low enough to put off an update? Was their thinking "if it ain't broke, don't fix it"? Were they even aware of the version of the dependencies their applications call on? Postponing a dependency update for six months to a year may be justifiable, but what are we to make of the 85% of audited codebases with open source libraries more than four years behind the latest versions?



OF AUDITED CODEBASES CONTAINED OPEN SOURCE COMPONENTS THAT WERE MORE THAN FOUR YEARS OUT-OF-DATE

CONCLUSION



THE PETER PARKER PRINCIPLE

"With great power comes great responsibility."
—anon., often attributed to Stan Lee

How does it feel to be part of a revolution? As the data in the 2021 OSSRA report demonstrates, it's rare today to find an application that isn't dependent on the power of open source. Not only is more open source in use, but more developers are writing open source. The 2020 FOSS Contributor Report sponsored by the Linux Foundation notes that nearly half of respondents to its survey were being paid by their organizations to contribute to open source projects. A CyRC survey ("DevSecOps Practices and Open Source Management in 2020") indicates that the majority of organizations in the business of building software—65%—have policies in place allowing their developers to contribute to open source projects.

As this report has stated, paralleling the growth of open source is a growth in risk—specifically around open source security, code quality, and sustainability. Part of the reason is that the increased use of open source makes managing a dynamic, changing risk landscape more difficult. To meet the challenge, development teams need to have reliable and timely vulnerability information, a comprehensive inventory of the open source dependencies their software uses, accurate guidance on vulnerability severity and exploitability, and clear direction on how to patch the affected open source.

Does your organization have a published policy for its developers to make open source contributions?

("DevSecOps Practices and Open Source Management in 2020" survey)

Mistakes versus malice

Although malicious attacks tend to steal the spotlight in the media, code flaws introduced by mistake can be just as disruptive and are much more likely to impact open source projects. According to the "2020 State of the Octoverse" report, 83% of the vulnerabilities that GitHub sent alerts on from 2019 through 2020 were due to coding errors rather than malicious intent.

If most attacks exploit unintentional vulnerabilities in code, preventing these unintentional vulnerabilities becomes all the more crucial. One strategy is to educate developers on secure software development. Free courses from OpenSSF are available on edX, and many software security companies such as Synopsys offer commercial application security eLearning courses.

Encouraging the use of detection tools such as static analysis before code commit is another means to reduce open source coding errors. Static analysis examines source code against a set of coding rules to uncover common coding errors. Synopsys offers a free static analysis service for open source developers who have registered their projects with scan.coverity.com. Coverity Scan is powered by the same engine used by Synopsys' commercial Coverity static analysis tool to help identify code defects for fast and easy remediation. Respondents to the Linux Foundation survey "overwhelmingly cited Coverity Scan and clang security checkers" as the primary static analysis tools they use. The next page details a case study on how Coverity Scan helps ensure code quality and security for NGINX Open Source.

COVERITY SCAN DATA

2,182 ACTIVE PROJECTS SCANNED

31.7 BILLION LINES OF CODE SCANNED

Top 10 defects/vulnerabilities found in the scans

Resource leaks	Uninitialized variables
Null pointer dereferences	Cross-site scripting
Memory corruptions	Extra argument error in call
Error handling issues	Insecure data handling
Control flow issues	Uncaught exception

NGINX OPEN SOURCE

A Coverity Scan Case Study

One of the world's most widely used web servers—powering sites such as Netflix, Hulu, Pinterest, and GitHub—NGINX Open Source (pronounced "engine x") is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. Other members of the NGINX Open Source family include NGINX JavaScript (njs), a module adding JavaScript support to NGINX; and NGINX Unit, a dynamic application server supporting applications written in Perl, Python, Ruby, Node.js, Go, Java, and PHP.njs.

Developers for all three NGINX Open Source projects use Coverity Scan® to find and fix defects in their code. A free online service provided by Synopsys and powered by the same engine used by Synopsys' commercial Coverity static analysis tool, Scan helps open source developers identify code defects for fast and easy remediation.

"I have a strong belief in the power of open source," said Igor Sysoev, the software's author and cofounder of NGINX in a 2014 interview. "NGINX was an experiment focused on a very specific problem—how to handle more customers on a single, existing server. It turned out to be a universal problem. As soon as I realized NGINX really helps to improve web performance, I wanted people to use it, so I made it open source."

A web server that can also be used as a reverse proxy, load balancer, mail proxy, and HTTP cache, the open source version of NGINX powers more than 400 million websites. Sysoev cofounded NGINX in 2011 to provide formal support for NGINX Open Source and to offer a commercial version, NGINX Plus, which adds enterprise-grade features to NGINX Open Source.

NGINX was acquired by F5 Networks, an application security and delivery company, in 2019. Today, the NGINX family of open source projects include njs, a module adding JavaScript support to NGINX and NGINX Unit, a dynamic application server.

The problem: Ensuring open source code quality and security

"We integrated Coverity Scan into our CI/CD pipeline soon after establishing NGINX," said Maxim Konovalov, one of the company's cofounders and now VP of engineering. "We've been submitting NGINX build artifacts daily since 2012."

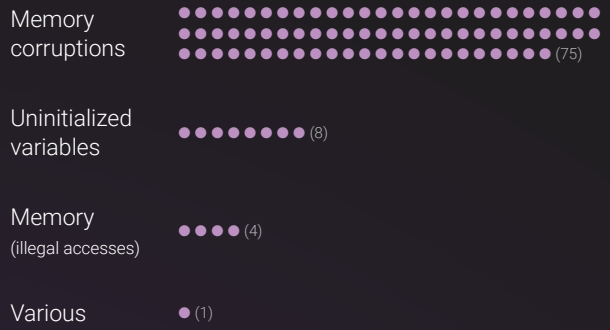
"In many cases, NGINX acts as an internet front end," continued Konovalov. "Its security and stability are essential to its users. My team is passionate about code quality and are always looking for best practices and tools to help us improve it. Static code analyzers such as Coverity Scan provide a great help to us."

NGINX takes its role as a foundational technology to millions of apps and websites very seriously. Code quality and security are part of its ethos, and the tools that help support that mission are integral to its development practices.

The solution: Static code analysis with Coverity Scan

Contrary to popular opinion, most software vulnerabilities are the result of coding mistakes, not malicious attacks. According to the "2020 State of the Octoverse" security report, 83% of the vulnerabilities that GitHub sent alerts on from 2019 through 2020 were due to coding errors rather than malicious intent.

"High-impact" outstanding defects



But malicious attacks do exploit flaws in code, and developers need to embrace proactive detection tools to uncover bugs in the code they write. Static analysis examines source code against a set of coding rules to uncover common coding errors.

The results: 658,000 lines of code scanned with a defect density of 0.02%

In the January 2021 Coverity Scan of a NGINX build, 658,665 lines of code were analyzed, and various code defects uncovered, including two CWE Top 25 defects. Thanks to F5's regular use of Coverity Scan, the NGINX project has a defect density (number of defects per 1,000 lines of code) of only 0.02%.

"Coverity Scan provides an invaluable service to us," says Maxim Konovalov. "I regularly recommend Coverity Scan and its ability to provide specific defect IDs in code commits. And in fact, I'm a member of the FreeBSD committers group, and we use Coverity Scan for code analyses of FreeBSD as well."

Create demand for a Bill of Materials

The concept of a software Bill of Materials (BOM) comes from manufacturing, where the classic BOM is an inventory detailing the items included in a product. When a defective part is discovered, the manufacturer knows precisely what product is affected and can begin the process of repair or replacement.

While still a new concept to many, the demand for open source BOMs is growing. In its 2020 Magic Quadrant for Application Security Testing, Gartner predicted, "By 2024, the provision of a detailed, regularly updated software Bill of Materials by software vendors will be a non-negotiable requirement for at least half of enterprise software buyers, up from less than 5% in 2019."

If software vendors can anticipate that their enterprise customers will require a software BOM, it's fair to anticipate the same for the open source projects the software depends upon. For many projects this can be done, at least in part, by package management information that identifies direct and indirect dependencies. Software composition analysis tools can use this information to create more complete (specific versions, license, etc.) BOM information.

Open source consumers should expect that many projects—especially those with few active contributors—won't be ready to provide BOMs. This may be the perfect opportunity for companies that depend on open source projects to help develop and maintain the project's BOM.

Coda

Whether you believe it was Voltaire or Peter Parker's Uncle Ben who first said, "with great power comes great responsibility," you can't deny the proverb's accuracy. As part of the open source ecosystem we all share in its power—and we all share responsibility to keep open source safe and secure. It's time we exercise our power as developers and consumers of open source and take on the shared responsibility of maintaining open source quality and security.

FURTHER READING

- Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks
- Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies
- DevSecOps Practices and Open Source Management in 2020
- Finding Critical Open Source Projects (Google blog)
  - Related: Finding Critical Open Source Projects (Top 10 list)
- Get earlier, actionable vulnerability insights from Black Duck Security Advisories
- How the Linux Foundation's Software Signing Combats Supply Chain Attacks
  - Related: What is sigstore?
- Know, Prevent, Fix: A framework for shifting the discussion around vulnerabilities in open source
- Open source licenses: No license, no problem? Or … not?
- OpenSSF: Secure Software Development Fundamentals Courses
- Peril in a Pandemic: The State of Mobile Application Security
- Preventing Supply Chain Attacks like SolarWinds
- TANSTAAFL! The tragedy of the commons meets open source software
- What is a software bill of materials?

REFERENCES

1. Tyler Clifford, Veeva Systems sees product usage increase tenfold as biotech companies race to find COVID-19 cure, CNBC, 3/26/2020.
2. David Sharp, L.L. Bean Sees Sales Boom Amid Pandemic's Push to Outdoors, U.S. News, 3/19/2021.
3. Wikipedia, Open source license litigation, accessed 3/26/2021.
4. Kemal Altinkemer, Jackie Rees, and Sanjay Sridhar; Vulnerabilities and Patches of Open Source Software: An Empirical Study, Krannert Graduate School of Management, The Center for Education and Research in Information Assurance and Security, Purdue University; 1/2005.
5. Explain xkcd, 2347: Dependency, 8/17/2020.
6. Frank Nagle et al, Report on the 2020 FOSS Contributor Survey, The Linux Foundation, 12/8/2020.
7. GitHub, Nicole Forsgren et al, The 2020 State of the Octoverse, 2020.
8. Frank Nagle et al, Report on the 2020 FOSS Contributor Survey, The Linux Foundation, 12/8/2020.
9. Mark Horvath, Dionisio Zumerle, and Dale Gardner, Magic Quadrant for Application Security Testing, Gartner, 4/29/2020.

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

About CyRC

The Synopsys Cybersecurity Research Center (CyRC) works to accelerate access to information around the identification, severity, exploitation, mitigation, and defense against software vulnerabilities. Operating within the greater Synopsys mission of making the software that powers our lives safer and of the highest quality, CyRC helps increase awareness of issues by publishing research supporting strong cybersecurity practices. For more information, go to www.synopsys.com/software.

Synopsys, Inc.
690 E Middlefield Rd,
Mountain View, CA 94043 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com