synopsys®

2022 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT

TABLE OF CONTENTS

INTRODUCTION

ABOUT THE 2022 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT AND THE CYRC

Welcome to the 2022 Open Source Security and Risk Analysis (OSSRA) report. The 7th edition of OSSRA delivers our annual in-depth look at the current state of open source security, compliance, licensing, and code quality risks in commercial software. Synopsys shares these findings to help security, legal, risk, and development teams better understand the security and license risk landscape. The data in this report is possible thanks to the Synopsys Cybersecurity Research Center (CyRC), whose mission is the publication of security advisories and research that help organizations better develop and consume secure, high-quality software.

This year, CyRC researchers examined anonymized findings from over 2,400 commercial codebases across 17 industries. The growth in the number of audited codebases—64% larger than last year's—reflects the significant increase in merger and acquisition (M&A) transactions throughout 2021. According to Morgan Stanley, 2021 saw a record number of M&A deals, with a total value of more than $4.9 trillion.[1] The growth in audits can also be attributed to a recognition that software is often a key element of a company's intellectual property (IP). Consequently, acquirers in M&A deals want to understand what risk may be associated with the software they're acquiring—specifically risk around licensing, security, and the quality of the open source used in that software.

For nearly 20 years, development, security, and legal teams around the world have placed their trust in Black Duck® software composition analysis (SCA) solutions and audit services. Our SCA offerings help organizations effectively identify and track open source code and automate open source policy enforcement across development environments.

Each year, our Audit Services team audits thousands of codebases for our customers, mainly to identify a range of software risks in M&A transactions. Black Duck audits provide a comprehensive and up-to-date software Bill of Materials (SBOM) covering the open source, third-party code, web services, and APIs in an application. The Audit Services team relies on data from the Black Duck KnowledgeBase™ to identify potential license compliance and security risks. The KnowledgeBase contains information for nearly 200 million versions of over 5.1 million open source components that use data from more than 26,000 unique sources. This data is curated and validated by the CyRC.

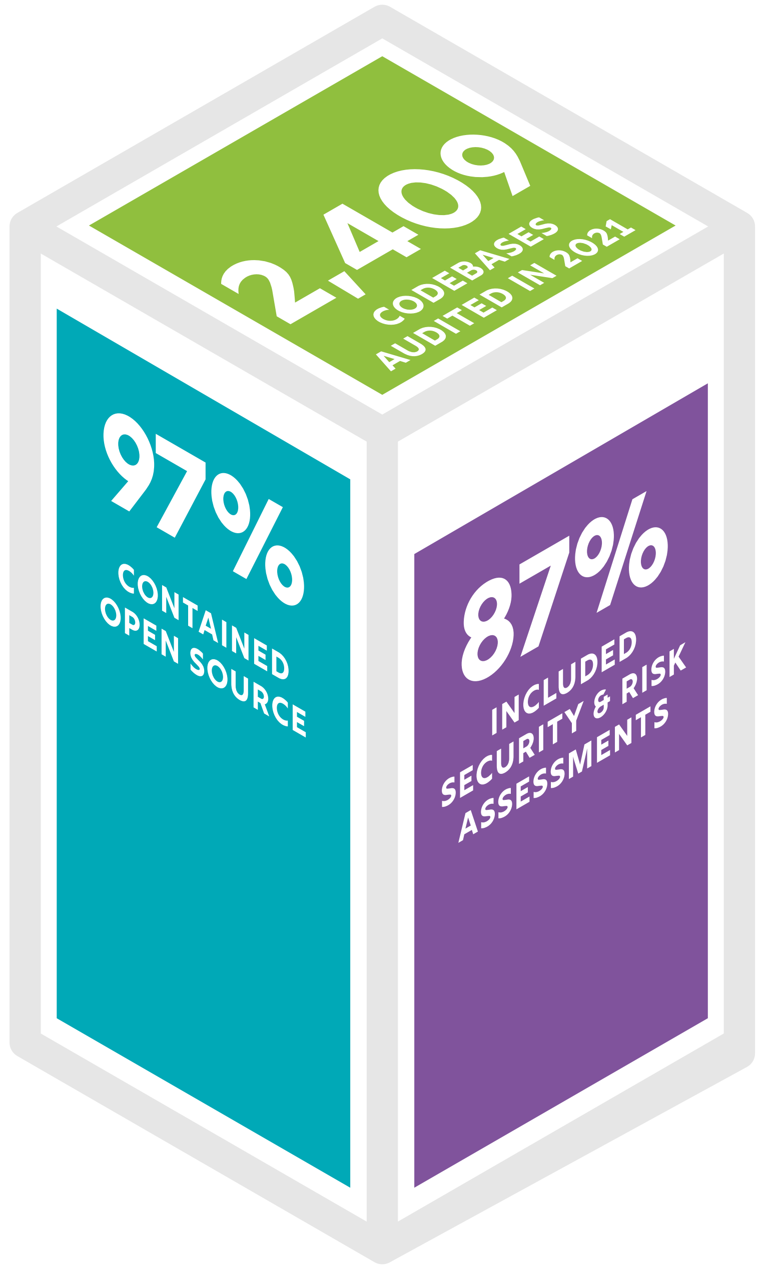
This analysis of 2021 audit data was conducted by the CyRC's Belfast team. In addition to their role in collecting and analyzing the data used for this report, the team issues Synopsys Black Duck Security Advisories. These detailed notifications deliver enhanced vulnerability information directly to commercial Black Duck customers.

Whatever industry you're in, OSSRA data indicates that it's prudent to assume open source will be part of the software your business builds and uses. As our findings underscore, open source is everywhere, as is the need to properly manage its use. Open source is the foundation for every application we rely on today. Identifying, tracking, and managing open source is critical for effective software security. This report offers key recommendations to help developers and consumers better understand the open source ecosystem and manage open source responsibly.

OPEN SOURCE IS EVERYWHERE, AS IS THE NEED TO PROPERLY MANAGE ITS USE

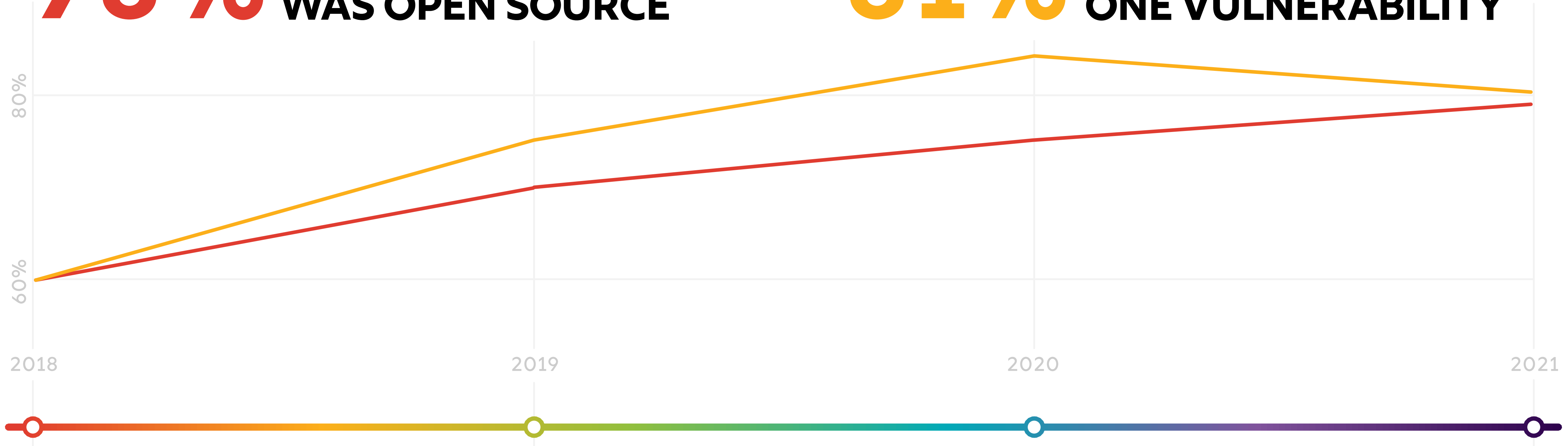
OVERVIEW

2022 IN REVIEW


2,409
CODEBASES AUDITED IN 2021
97%
CONTAINED OPEN SOURCE
87%
INCLUDED SECURITY & RISK ASSESSMENTS

78% OF CODE IN CODEBASES WAS OPEN SOURCE

81% CONTAINED AT LEAST ONE VULNERABILITY

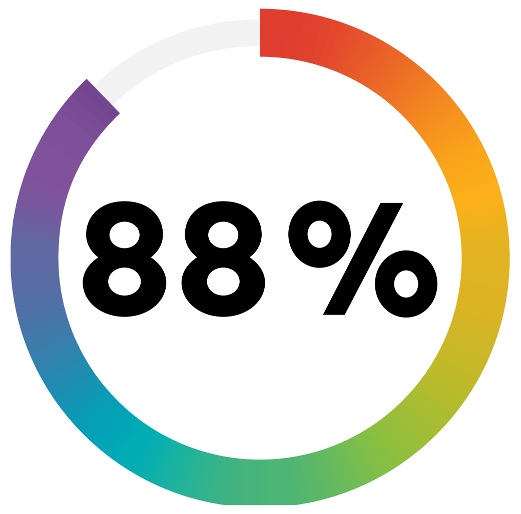


88% CONTAINED COMPONENTS THAT HAD NO NEW DEVELOPMENT IN TWO YEARS

85% CONTAINED OPEN SOURCE THAT WAS MORE THAN FOUR YEARS OUT-OF-DATE

OF AUDITED CODEBASES HAD LICENSE CONFLICTS

CONTAINED OPEN SOURCE WITH NO LICENSE OR CUSTOM LICENSE



UTILIZED COMPONENTS THAT WERE NOT THE LATEST VERSION

OVERVIEW

TERMINOLOGY

CODEBASE

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

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 COMPONENT

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

DEPENDENCY

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

OPEN SOURCE LICENSE

A set of terms and conditions stating end-user obligations when an open source component (or a snippet of a component's code) is used in software, including how the component 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 disclaim any liability on the part of the original developer and provide attribution of the original code to the original developers.

COPYLEFT LICENSE

This type of license generally includes a reciprocity obligation stating that derivative works based on original code provided under a copyleft license are released under the same terms and conditions as the original code, and that the source code containing changes must be available or provided upon request. Commercial entities are wary of including open source with copyleft licenses in their software, as its use can call the rights, ownership and control of the codebase subject to the copyleft license into question.

SOFTWARE BILL OF MATERIALS (SBOM)

A comprehensive inventory of the open source dependencies in a codebase, often generated by a software composition analysis tool. An SBOM lists all the open source, proprietary code, associated licenses, versions in use, download locations for components/dependencies, and subdependencies the dependencies link to. Since SBOMs are intended to be shared across companies and communities, having a consistent format (that is both human- and machine-readable) with consistent content is critical. National Institute of Standards and Technology guidelines currently specify three standards as approved formats: SPDX, CycloneDX, and SWID.

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.

APACHE LOG4J2 VULNERABILITIES (BDSA-2021-3887, CVE-2021-44228, ET AL.)

The open source component Apache Log4j2 (commonly known as Log4j) is broadly used within the Java community to implement application logging. Several vulnerabilities have been identified in Log4j, including remote code execution, denial of service, and LDAP vulnerabilities.

EXECUTIVE ORDER 14028

In May 2021, U.S. President Biden issued an order titled "Improving the Nation's Cybersecurity" instructing various agencies of the federal government to create software security guidelines for companies doing business with the federal government. This order includes a timeline for activities that, as of the writing of this report, do not mandate contractual obligations. However, despite the lack of hard requirements, the order has prompted many organizations to re-examine their security practices and scrutinize their level of software security risk. The use of a software Bill of Materials is a key element promoted by Executive Order 14028, as it facilitates the communication of software supply chain information between producers and consumers of software.

OVERVIEW

PERCENTAGE OF SCANNED CODEBASES CONTAINING OPEN SOURCE

100%	Computer Hardware and Semiconductors
100%	Cybersecurity
100%	Energy and Clean Tech
100%	Internet of Things
99%	Internet and Mobile Apps
99%	Marketing Tech
99%	Retail and eCommerce
98%	Internet and Software Infrastructure
98%	Virtual Reality, Gaming, Entertainment, Media
97%	Aerospace, Aviation, Auto, Transportation, Logistics
97%	Financial Services and FinTech
97%	Manufacturing, Industrials, Robotics
96%	Enterprise Software/ SaaS
95%	Telecommunications and Wireless
94%	Big Data, AI, BI, Machine Learning
94%	Ed Tech
93%	Healthcare, Health Tech, Life Sciences

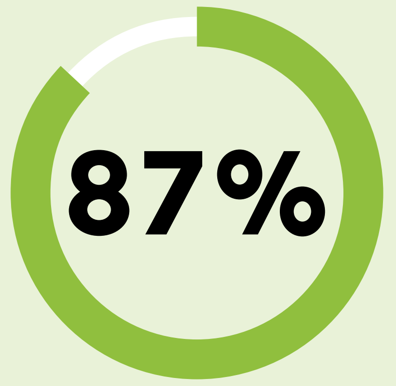
VULNERABILITIES AND SECURITY

OPEN SOURCE VULNERABILITIES AND SECURITY

Of the 2,409 codebases analyzed by Black Duck Audit Services for this year's report, 97% contained open source. Eighty-one percent contained at least one known open source vulnerability, a minimal decrease of 3% from the findings of the 2021 OSSRA.

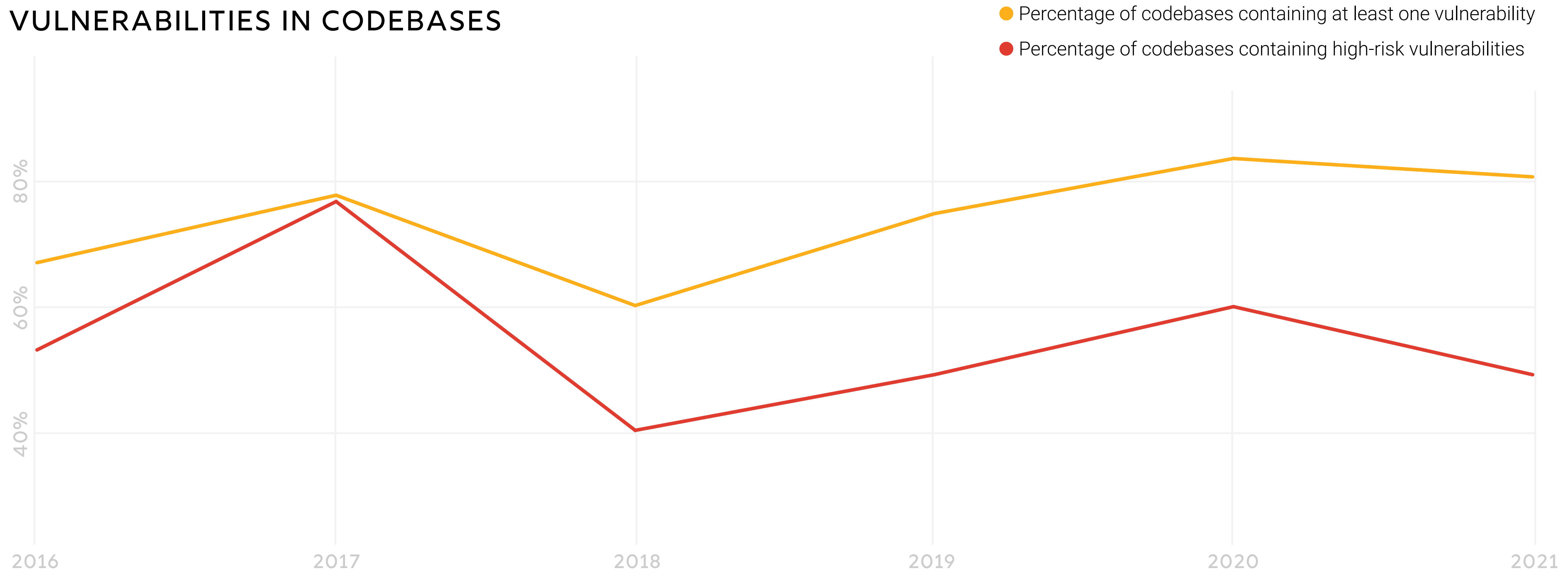
We found a more dramatic decrease in the number of codebases containing at least one high-risk open source vulnerability; only 49% of this year's audited codebases contained at least one high-risk vulnerability, compared to 60% last year. High-risk vulnerabilities are those that have been actively exploited and already have either a documented proof-of concept exploit or classification as a remote code execution vulnerability.

All Black Duck audits examine open source license compliance, but customers can opt out of the vulnerability/operational risk assessment portion of the audit at their discretion. In 2021, the Audit Services team conducted a total of 2,409 audits. Of those audits, 13% (312) opted out of a security and operational risk assessment. The data in the Open Source Vulnerabilities and Security and Open Source Maintenance sections of the 2022 OSSRA report is based on the 2,097 codebases that included risk assessments, whereas the data in the Licensing section is based on all 2,409 codebases.

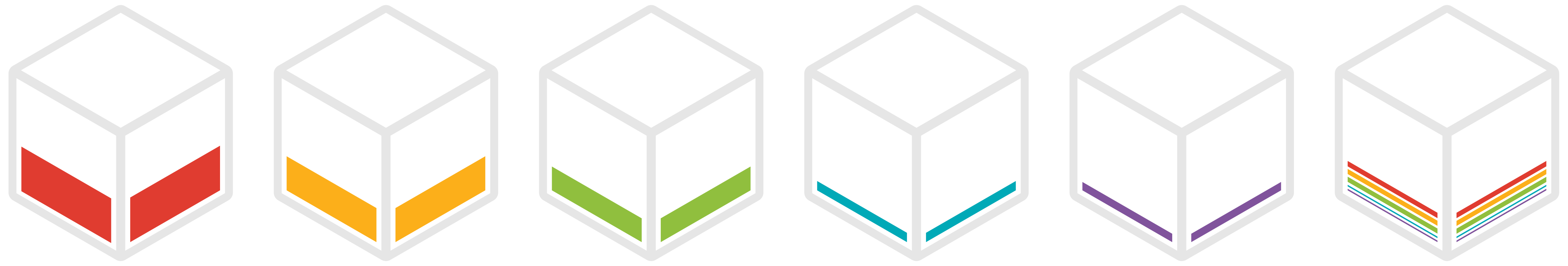


OF THE CODEBASES HAD RISK ASSESSMENTS

VULNERABILITIES IN CODEBASES



PERCENTAGE OF CODEBASES CONTAINING VULNERABLE COMPONENTS



43%	jQuery
33%	Lodash
23%	Bootstrap (Twitter)
9%	jackson-databind
8%	Spring Framework
5%	Netty Project
4%	Apache Tomcat
4%	OpenSSL
<1%	Ruby
<1%	PHP

2021: THE YEAR OF OPEN SOURCE

Although the decrease in high-risk vulnerabilities found in the audits was encouraging, 2021 was still a year filled with open source issues including supply chain attacks,[2] hacker exploits of Docker images,[3] and a developer sabotaging their own open source libraries and breaking thousands of dependent applications in the process.[4] Most notably, 2021 ended with a zero-day vulnerability in the popular Apache Log4j utility. The primary Log4j vulnerability, known as Log4Shell (CVE-2021-44228), allowed attackers to execute arbitrary code on vulnerable servers. As the story unfolded, the potential severity of this vulnerability became clear.

What's most notable about Log4Shell, however, is not its ubiquity but the realizations it spurred. In the wake of its discovery, businesses and government agencies were compelled to re-examine how they use and secure open source software created and maintained largely by unpaid volunteers, not commercial vendors. What also came to light was that many organizations are simply unaware of the amount of open source used in their software.

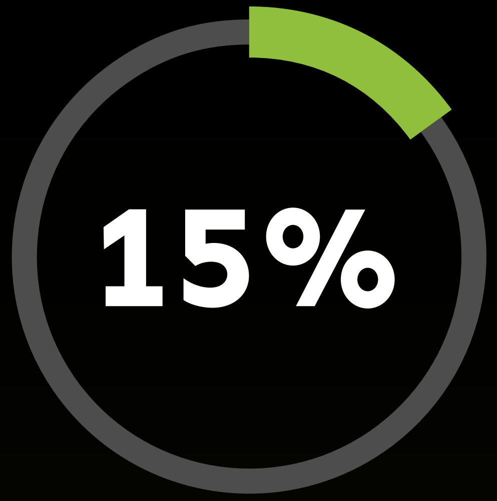
Also unveiled by the Log4j incident is the inherent trust organizations place in open source; most development teams use it without performing the same security reviews required for commercial or proprietary software.

Further complicating the situation is the variety of open source code. For example, GitHub has millions of projects in which the number of developers is in the single digits. But in popular open source projects like Kubernetes, large numbers of volunteer developers work to maintain the code. Some of those maintainers are employed by companies that use Kubernetes and therefore have a vested interest in its maintenance.

One of the takeaways from Log4Shell's discovery should be the need to create a path to mitigate the business risk associated with using open source software. The important distinction here is that *open source itself doesn't create business risk, but its mismanagement does.*

The first step toward squashing business risk should involve a comprehensive inventory of all software a business uses, regardless of where it came from or how it was acquired. Only with this complete inventory, known as a software Bill of Materials (SBOM), can teams identify which components are used by which asset. This level of information, provided by a software composition analysis tool, enables security teams to chart a path forward, with plans in place to address risk stemming from new security disclosures like Log4Shell.

Put simply, it's awfully hard to fix something you don't know about or can't find.



PERCENTAGE OF AUDITED JAVA CODEBASES THAT CONTAINED VULNERABLE LOG4J COMPONENT

VULNERABILITIES IN INDUSTRIES

This year, we found that 4 of the 17 industry sectors represented in this report—Computer Hardware and Semiconductors, Cybersecurity, Energy and Clean Tech, and Internet of Things—contained open source in **100% of their codebases**. The remaining verticals had open source in 93% to 99% of their codebases. Even the sector with the lowest percentage—Healthcare, Health Tech, Life Sciences—had 93%, which is still very high. It's clear that open source really is everywhere. And this fact did not go unnoticed by the U.S. government. A January 2022 White House briefing statement described software as "ubiquitous across every sector of our economy and foundational to the products and services Americans use every day. Most major software packages include open source software... [which] brings unique value but has unique challenges."[5]

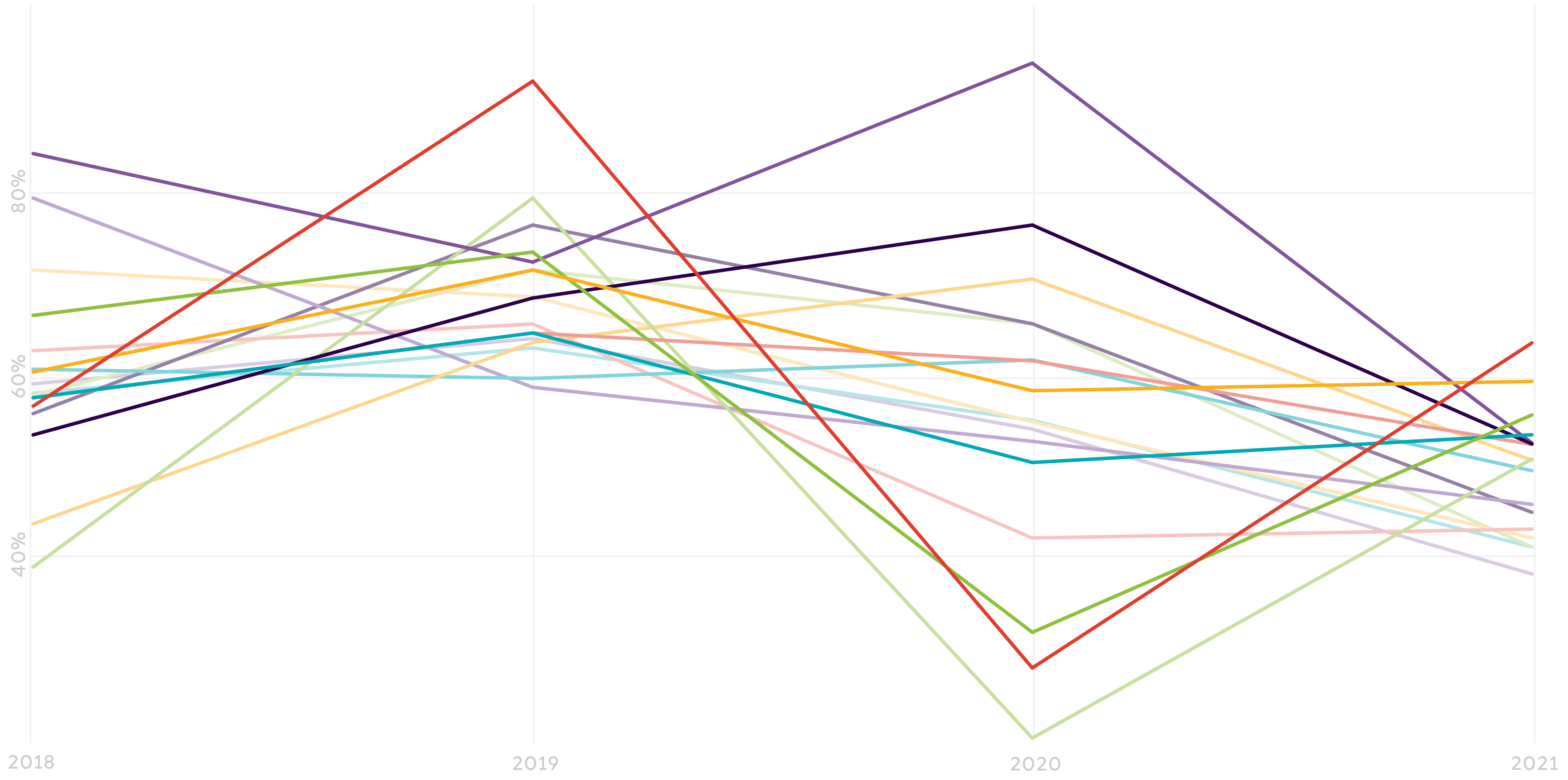
A layer deeper, the *amount* of open source in codebases was also high. For example, 100% of codebases in the IoT sector contained open source, and an astounding 92% of the audited code in this sector **was** open source. Troublingly, 64% of the IoT codebases also contained vulnerabilities.

Similarly, the Aerospace, Aviation, Automotive, Transportation, and Logistics sector had open source in 97% of its codebases, and 60% of the total code was composed of open source. The real revelation came when we looked at open source vulnerabilities: 60% of this sector's codebases had open source vulnerabilities.

We found more of the same in the Internet and Mobile Apps sector; 99% of codebases **contained** open source, and 80% of the codebases were **composed** of open source. Fifty-six percent of the codebases contained open source vulnerabilities.

This story was echoed across all industry sectors; open source was in almost everything we scanned. Open source components made up the large majority of codebases, and much of those codebases were vulnerable to exploit and attack.

PERCENTAGE OF CODEBASES CONTAINING OPEN SOURCE VULNERABILITIES, BY INDUSTRY



2021

- **64%** Internet of Things
- **60%** Aerospace, Aviation, Automotive, Transportation, Logistics
- **56%** Internet and Mobile Apps
- **54%** Ed Tech
- **53%** Marketing Tech
- **53%** Energy and Clean Tech
- **53%** Financial Services and FinTech
- **51%** Retail and eCommerce
- **51%** Manufacturing, Industrials, Robotics
- **50%** Enterprise Software/SaaS
- **46%** Virtual Reality, Gaming, Entertainment, Media
- **45%** Healthcare, Health Tech, Life Sciences
- **43%** Computer Hardware and Semiconductors
- **42%** Big Data, AI, BI, Machine Learning
- **41%** Internet and Software Infrastructure
- **41%** Telecommunications and Wireless
- **38%** Cybersecurity

THE EXECUTIVE ORDER AND SUPPLY CHAIN SECURITY

In light of the uptick in security breaches this year, President Biden issued Executive Order 14028, outlining how companies doing business with the federal government should secure their software. While Biden's aim was to help bolster the United States' cybersecurity profile, the order prompted an analysis of security practices by industries and organizations nationwide.

Looking specifically at open source security in the context of software supply chains, it's important to first acknowledge that open source software, just like commercial software, is made up of many components, which themselves may utilize a large number of subcomponents, or "dependencies."

This is true for most software, whether it's for mobile applications, IoT firmware, business logic functions, or any other use. Each element has dependencies that are required for the software to function properly. The dependencies used within any given application represent the suppliers within this software supply chain. Some of those suppliers may be commercial entities, like those supplying custom SDKs, but as we've seen with open source usage, a majority of the dependencies are open source. These dependencies are where the greatest risk exposure exists within a software supply chain.

The only way to minimize this risk is with a comprehensive and exhaustive SBOM that tracks dependencies and their associated risk, allowing you to take prioritized and informed action when needed.

THE TOP 10 VULNERABILITIES

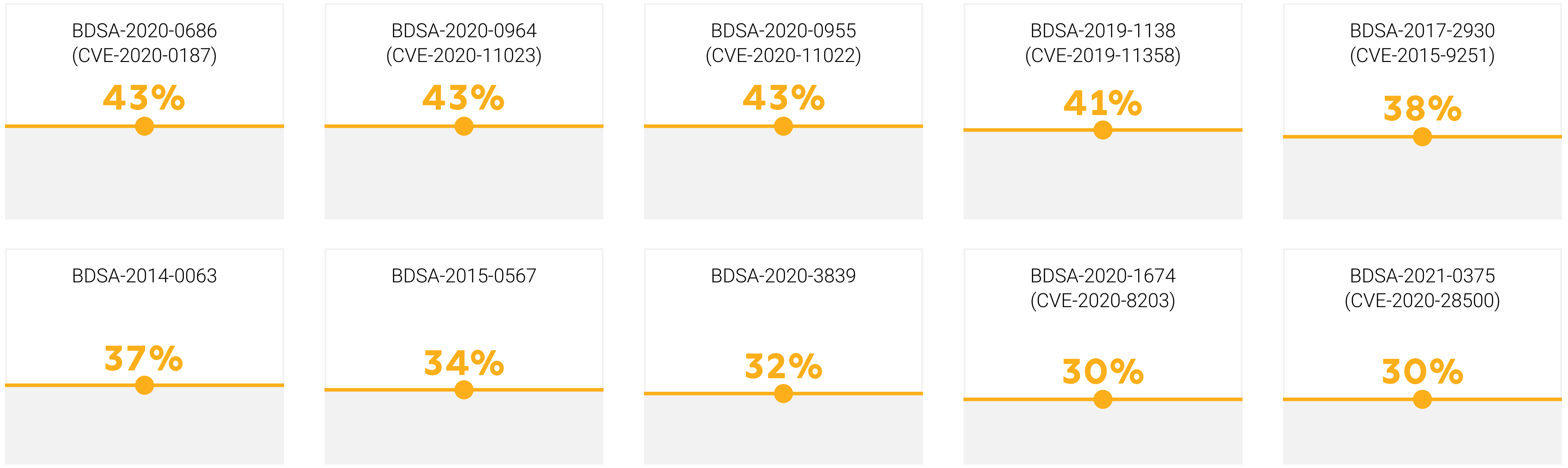
Several vulnerabilities discovered in last year's audits surfaced again this year, with some concerning increases. CVE-2020-11023 and CVE-2020-11022 were found in 37% of codebases last year. This year the percentages for both increased to 43%. Rated as Medium severity by the National Vulnerability Database (NVD),[6] both CVEs are found in versions of jQuery. Our audits showed that jQuery was the #1 component containing vulnerabilities. Forty-three percent of the audited codebases contained the jQuery component.

When the percentage of a given vulnerability remains constant or increases year over year, a conclusion could be drawn that some DevSecOps teams are struggling to stay on top of open source risk.

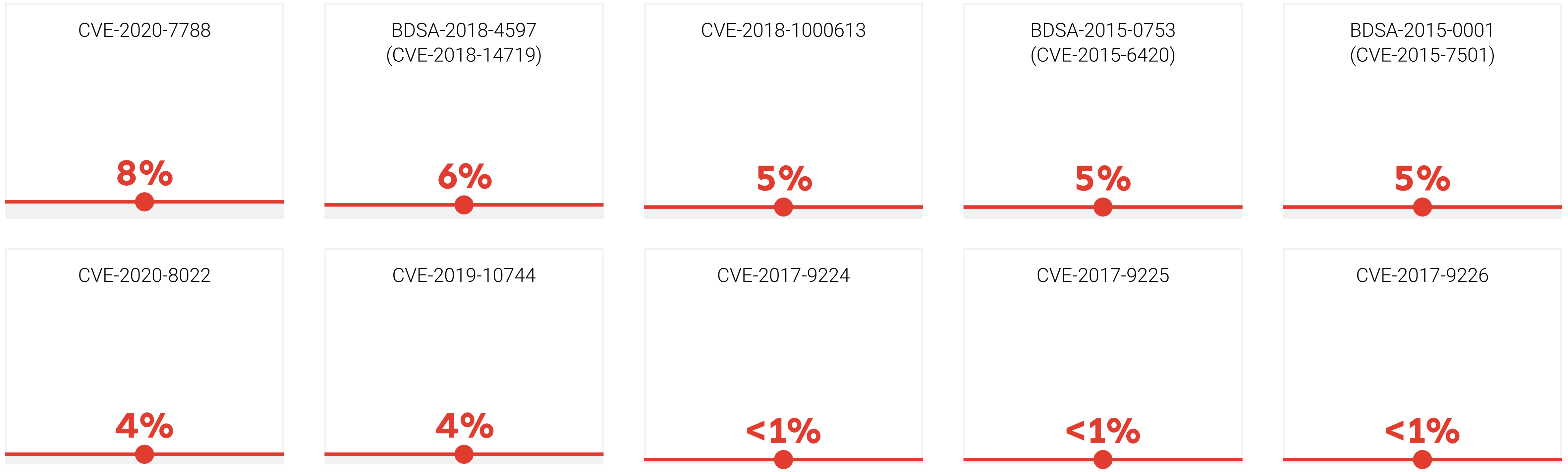
On the other hand, we saw promising improvements this year in the number of high-risk CVEs/BDSAs present within the audited codebases. The top vulnerability last year was present in 29% of codebases. This year, the most prevalent high-risk vulnerability, CVE-2020-7788, was identified in only 8% of codebases. All reoccurring high-risk vulnerabilities saw significant decreases.

Prompt identification, prioritization, and mitigation of high-risk vulnerabilities can help teams address the risks that pose the greatest threat to their organizations.

PERCENTAGE OF CODEBASES WITH TOP 10 CVEs/BDSAs



PERCENTAGE OF CODEBASES WITH TOP 10 HIGH-RISK CVEs/BDSAs



LICENSING

OPEN SOURCE LICENSING

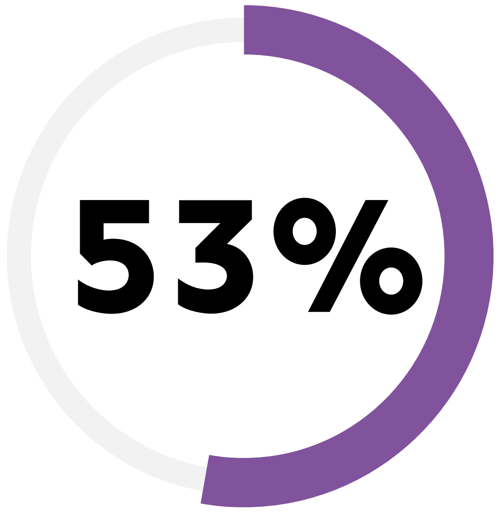
Black Duck Audit Services found that 53% of the 2021 audited codebases contained open source with license conflicts, a dramatic decrease from the 65% seen in 2020. Generally speaking, license conflicts decreased across the board between 2020 and 2021.

From a specific license standpoint, one increase we saw in 2021 concerned the Creative Commons ShareAlike 3.0 license. In 2021, 17% of audited codebases were found to have some form of conflict with that license, versus 15% the year before.

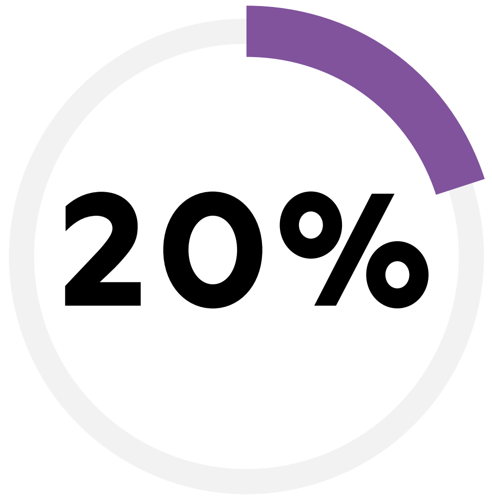
The Creative Commons ShareAlike 3.0 license conflict numbers illustrate an often overlooked issue when it comes to open source licenses. Both commercial and open source developers can introduce code snippets, functions, methods, and operational pieces of code into their software, generally termed *dependencies*, as the overarching software is dependent on that code. **Therefore, software, including open source projects, often contain more terms and conditions than simply the license that governs the project itself.**

The popular node.js platform is a great example. Versions up to 0.64.0 node.js often include a component named react-native that leverages code published on Stack Overflow and licensed under Creative Commons Attribution ShareAlike3.0. This introduces the potential for a license conflict, as the license requirements outlined in Creative Commons Attribution ShareAlike 3.0 become an inextricable part of the react-native component. The issue is explored in more detail in an article by Synopsys CyRC researchers Gary Armstrong and Rich Kosinski.[7]

As noted in the introduction of this report, acquirers in M&A deals have become more sensitive to potential risks stemming from software they're acquiring—specifically risk around licensing, security, and the quality of the open source used within the software. Our 2021 audit numbers indicate that potential sellers have also become more sensitive to potential license conflicts in their software that might undermine a deal, driving them to take a proactive stance toward mitigating possible license issues *before* the M&A is underway.



OF AUDITED CODEBASES CONTAINED LICENSE CONFLICTS



CONTAINED OPEN SOURCE WITH NO LICENSE OR A CUSTOM LICENSE

PERCENTAGE OF CODEBASES CONTAINING TOP 10 LICENSES WITH CONFLICTS

17%	Creative Commons Attribution ShareAlike 3.0
13%	GNU Lesser General Public License v2.1 or later
13%	Creative Commons Attribution ShareAlike 4.0
11%	GNU General Public License v2.0 or later
8%	GNU General Public License v3.0 or later
7%	Apache License 2.0
7%	Creative Commons Attribution 3.0
6%	GNU Lesser General Public License v3.0 or later
4%	Eclipse Public License 1.0
4%	Mozilla Public License 2.0

By industry, the Computer Hardware and Semiconductors sector had open source license conflicts in a shocking 93% of its codebases. Slightly better was the IoT sector, which had license conflicts in 83% of its codebases. The Healthcare, Health Tech, and Life Sciences industry had the lowest percentage of license conflicts, with only 41%.

As discussed earlier in this report, the majority of audited codebases *contained* open source, were often largely *composed of* open source, and contained a large number of open source *vulnerabilities*.

UNDERSTANDING LICENSE RISK

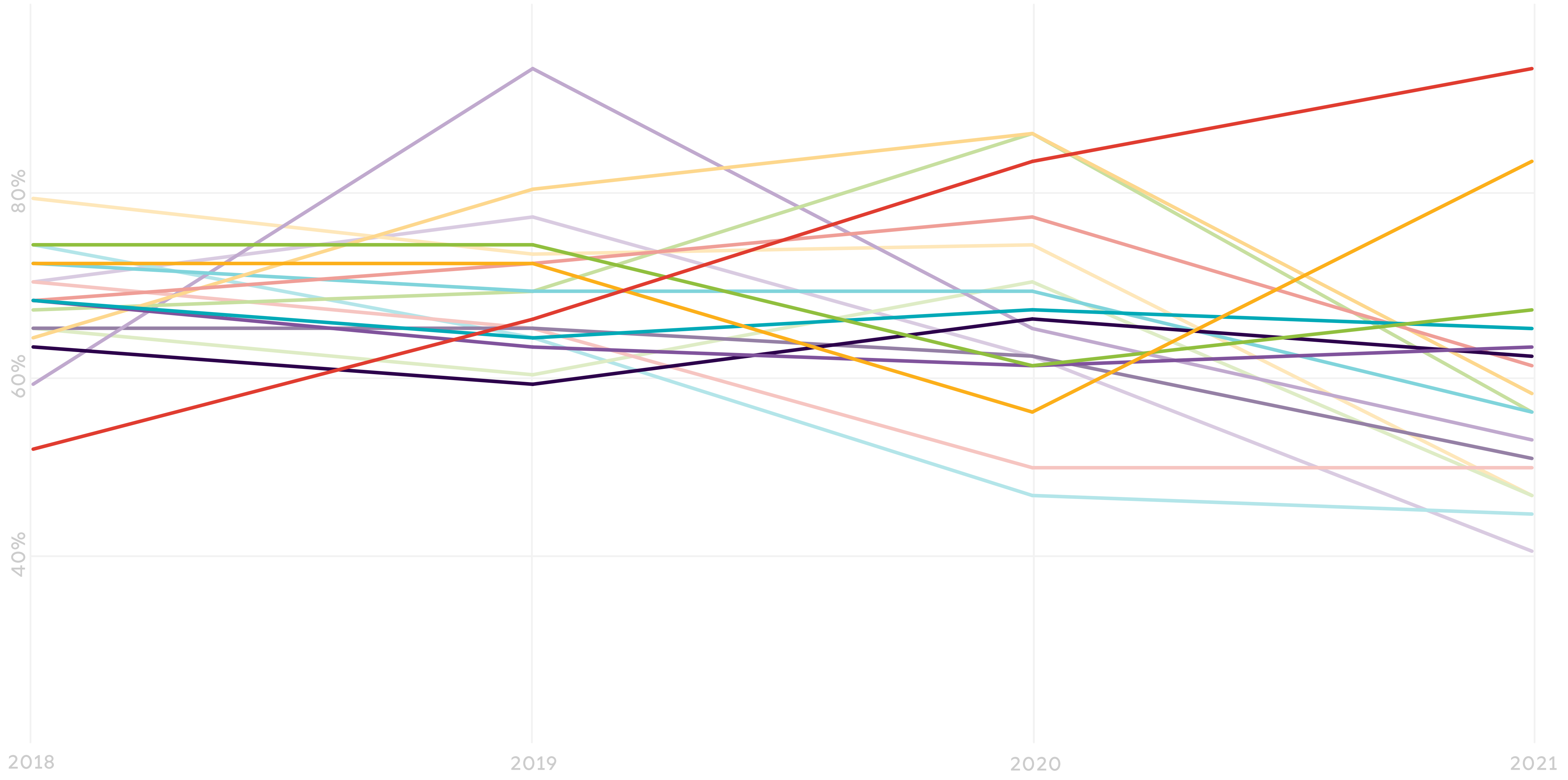
In the U.S. and many other places, creative work (including software) is protected by exclusive copyright by default. No one can legally use, copy, distribute, or modify software without explicit permission from the creator/author in the form of a license that grants the right to do so. Even the most permissive open source licenses include obligations the user takes on in return for use of that software.

Potential license risk arises when a codebase includes open source with licenses that appear to conflict with the overall license of the codebase. For example, the GNU General Public License (GPL) often governs the use of open source used in commercial software. But commercial software vendors may overlook the requirements of the GPL license and create a conflict with that license.

Customized open source licenses might place undesirable requirements on the licensee and will often require legal evaluation for possible IP issues or other implications. For example, the JSON license is based on the permissive MIT license, but the JSON license adds the distinction that "The software shall be used for good, not evil."[8] The ambiguity of this statement leaves its meaning up to interpretation, posing a particular concern in M&A scenarios where acquirers are hesitant to inherit this type of indistinct legal risk.

Codebases that contain open source components with no discernible license or a customized license have an additional layer of risk. Twenty percent of the audited codebases contained open source with no license or a custom license.

PERCENTAGE OF CODEBASES WITH LICENSING CONFLICTS, BY INDUSTRY



2021:

93%	Computer Hardware and Semiconductors
83%	Internet of Things
67%	Internet and Software Infrastructure
65%	Cybersecurity
63%	Telecommunications and Wireless
62%	Virtual Reality, Gaming, Entertainment, Media
61%	Aerospace, Aviation, Automotive, Transportation, Logistics
58%	Manufacturing, Industrials, Robotics
56%	Energy and Clean Tech
56%	Big Data, AI, BI, Machine Learning
53%	Internet and Mobile Apps
51%	Financial Services and FinTech
50%	Ed Tech
47%	Marketing Tech
47%	Enterprise Software/SaaS
45%	Retail and eCommerce
41%	Healthcare, Health Tech, Life Sciences

OPEN SOURCE MAINTENANCE

OPEN SOURCE MAINTENANCE

MAINTENANCE BY OPEN SOURCE DEVELOPERS

Of the more than 2,000 codebases examined by Black Duck Audit Services that included risk assessments, 88% contained open source that had no development activity in the last two years—no feature upgrades, no code improvements, and no security issues fixed over the past 24 months. This could mean that the project participants were satisfied with their work and saw no need for new features or improvement. More likely it means that the project is no longer maintained.

The recent Census II study,[9] produced by the Linux Foundation and the Laboratory for Innovation Science at Harvard, found that almost all the most widely used open source is developed and maintained by only a handful of contributors. When examining the top 50 non-npm projects, the study found that 23% had only one developer accounting for more than 80% of the lines of code, a perfect example of the so-called "80-20" rule.[10]

Ninety-four percent of the projects had fewer than 10 developers accounting for more than 90% of the lines of code. As the Census II study concluded, "these findings are counter to the typically held belief that thousands or millions of developers are responsible for developing and maintaining [free and open source software]."

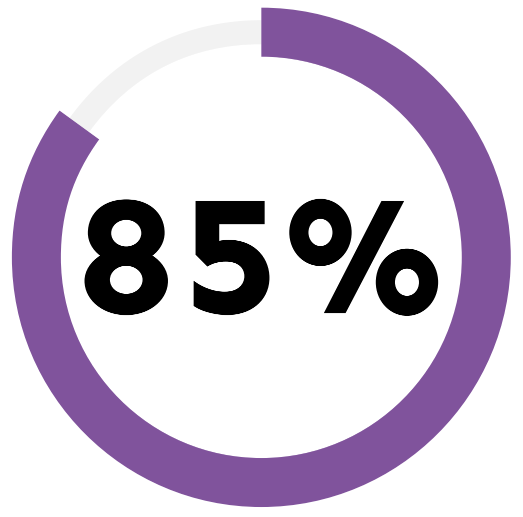
Open source projects popular enough to become an industry standard, like Kubernetes, have a large number of volunteer developers working on the code. Some of these developers are even employed by companies that depend on Kubernetes, giving those organizations a vested interest in supporting and encouraging their employees' work on Kubernetes. This translates to resiliency for the Kubernetes ecosystem. The departure of even key team members on popular projects like this is something that can usually be handled with minimal disruption to the overall project.

The same can't be said for smaller projects. GitHub has millions of projects in which the number of developers is in the single digits. The departure of a single developer often means losing the only person who understands exactly how and why the code was written. Incidentally, these smaller projects are often one of an application's most common dependencies, as they often perform basic tasks like keeping log data—as is the case with Log4j.

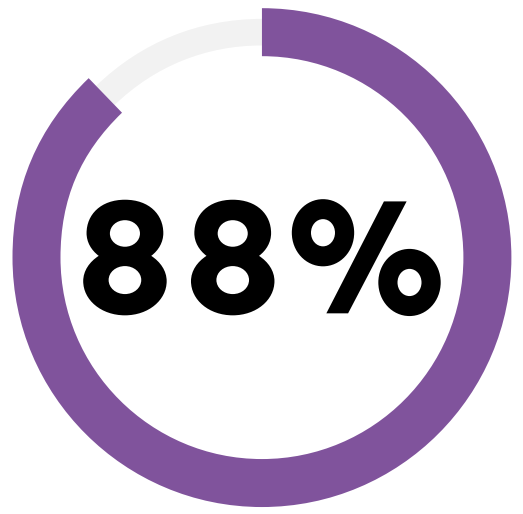
IS YOUR ORGANIZATION SUPPORTING OPEN SOURCE?

If your organization's software relies on the security and stability of an open source project, it should be a standard practice for your organization to support that project, either through developer contribution, monetary aid, or by other means. More organizations are heeding this call each year.

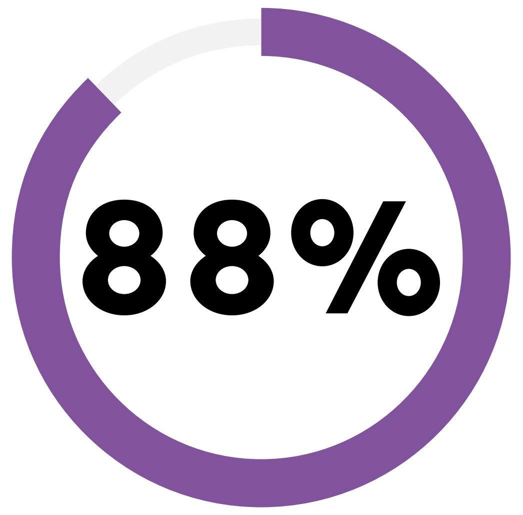
The 2020 FOSS Contributor Report, sponsored by the Linux Foundation,[11] found that nearly half of respondents to its survey are paid by their organizations to contribute to open source projects. A CyRC survey report[12] found that the majority (65%) of organizations in the business of building software have policies in place allowing their developers to contribute to open source projects. It's a trend that the open source community can hope will continue.



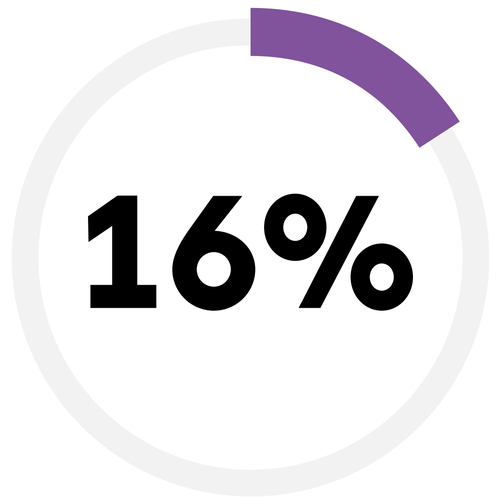
CONTAINED OPEN SOURCE MORE THAN FOUR YEARS OUT-OF-DATE



CONTAINED COMPONENTS THAT HAD NO NEW DEVELOPMENT IN THE PAST TWO YEARS



CONTAINED COMPONENTS WITH OUTDATED VERSIONS



CONTAINED COMPONENTS THAT WERE A YEAR OR MORE BEHIND ON MAINTAINER UPDATES

MAINTENANCE BY OPEN SOURCE CONSUMERS

Of the more than 2,000 codebases examined by Black Duck Audit Services that included risk assessments, 88% contained outdated versions of open source components. That is, an update or patch was available but had not been applied.

There are justifiable reasons for not keeping software up-to-date. A DevSecOps team might determine that the risk of unintended consequences outweighs whatever benefit would come from applying the newer version. Embedded software may be at minimal risk from vulnerabilities that can only be introduced from an external source.

Or it could be a time/resources issue. With many teams already stretched to the limit building and testing new code, updates to existing software can become a lower priority, aside from the most critical issues.

But it's highly possible that a large percentage of that 88% is due to the DevSecOps team being unaware that a newer version of the open source component is available—if they are aware of the component at all. As noted in earlier editions of the OSSRA report, open source *is* different from commercial software—not worse, not better, but different—and thus requires different techniques to manage.

For example, procurement and patches are handled differently for commercial and open source software. The purchase of commercial software usually requires the involvement of a procurement department, as well as review standards that are part of a vendor risk management program. Open source may simply have been downloaded and used at the developer's discretion. There may be some organizational guardrails for its use—*only code with permissive licenses allowed,* for instance—but in many cases, not even this guidance exists.

Unless developers keep an accurate and up-to-date inventory of the open source they introduce into their code, that knowledge may be lost when they move on to other projects or leave the organization altogether. The open source component becomes forgotten and ignored. Until, of course, the component breaks or becomes vulnerable to a high-risk exploit, and then the scramble to update is on. Which is precisely what occurred with Log4Shell.

Similarly, all organizations that use commercial software are familiar with patches and updates being pushed to their software. That's seldom the case with open source, where the user is expected to be aware of a component's security and stability status and apply new versions as they become available.

If your organization uses software—and what organization today doesn't?—that software almost certainly includes numerous open source components. The data makes it clear: You need an accurate, comprehensive inventory of the open source in your software, as well as processes and policies in place to monitor vulnerabilities, upgrades, and the overall health of the open source you use.

TWENTY-THREE PERCENT OF OPEN SOURCE PROJECTS HAVE ONLY ONE DEVELOPER CONTRIBUTING THE BULK OF CODE. **NINETY-FOUR PERCENT** OF THE PROJECTS HAVE FEWER THAN 10 DEVELOPERS ACCOUNTING FOR MORE THAN 90% OF THE LINES OF CODE.

"THESE FINDINGS ARE COUNTER TO THE TYPICALLY HELD BELIEF THAT THOUSANDS OR MILLIONS OF DEVELOPERS ARE RESPONSIBLE FOR DEVELOPING AND MAINTAINING [FREE AND OPEN SOURCE SOFTWARE]."

— LINUX FOUNDATION, CENSUS II

CONCLUSION

A PRESCRIPTION FOR THE "WITCHES' BREW" OF OPEN SOURCE

EYE OF NEWT AND TOE OF FROG,
WOOL OF BAT AND TONGUE OF DOG,
ADDER'S FORK AND BLIND-WORM'S STING,
LIZARD'S LEG AND OWLET'S WING,
FOR A CHARM OF POWERFUL TROUBLE.

(WM. SHAKESPEARE MACBETH, ACT IV)

Although the decrease in high-risk vulnerabilities found in our audits was encouraging, 2021 was still a year of open source vulnerabilities and exploits—as is nearly every year in these modern times. The newest addition to the high-profile vulnerability pack, the Log4Shell vulnerability, caused the biggest stir of 2021.

Based on download volume, Log4j is one of the most popular open source components in use and a dependency in more than 7,000 other open source projects. The Log4Shell vulnerability is considered dangerous enough to earn it the highest score possible on the CVSS severity scale issued by the NVD—a 10 out of 10. In a flurry of reports, we closed out 2021 with continued warnings of attempts to scan and attack systems vulnerable to Log4Shell.

ARE WE VULNERABLE? ARE OUR CUSTOMERS VULNERABLE? WILL WE BE HELD ACCOUNTABLE?

According to the Federal Trade Commission (FTC), the original Log4j vulnerability is being widely exploited in the wild and poses a "a severe risk to millions of consumer products" including enterprise software and web applications. In fact, the FTC finds the vulnerability dangerous enough that it has issued a statement that it "intends to use its full legal authority to pursue companies that fail to take reasonable steps to protect consumer data from exposure as a result of Log4j, or similar known vulnerabilities in the future."[13]

Largely thanks to the heroic efforts of DevSecOps teams at countless organizations, many of whom worked tirelessly through the holidays, Log4Shell's threat to their organization appears to have been essentially mitigated. But somewhat lost in the uproar was the fact that the need for this round-the-clock remediation effort was often a result of organizations not knowing where Log4j was located within their systems and applications, or in fact, if it was there at all. The identification problem was multiplied across thousands of IT groups, which all scrambled to answer questions like "Are we vulnerable to Log4Shell? Is our vendors' software vulnerable? Are the customers using our software vulnerable?"

A key tenet of the OSSRA report is to highlight the risks that can stem from *unmanaged* open source use. As we've said before, it's important to distinguish between a lack of open source management and the fact that *open source itself is not the problem*.

In earlier times, the open source community had to endure disparaging comments from the commercial software world, usually with the implication that open source was dangerous to use. Remarks ranged from open source being called "snake oil" (Ken Olsen, one-time CEO of Digital Equipment Corporation, 1987) to "a cancer that attaches itself in an intellectual property sense to everything it touches" (Steve Ballmer, one-time CEO of Microsoft Corporation, 2001).

The CEOs' criticism was neither accurate nor enduring. In fact, open source now serves as the foundation of commercial software, with 97% of commercial code containing open source, as noted in this year's OSSRA report.

Yet as universal as it's become, the misperception that open source is somehow inherently dangerous persists. As recently as last year, Anne Neuberger, the U.S. Deputy National Security Advisor for Cybersecurity, characterized open source as a "witches' brew" while discussing the Log4Shell vulnerability.[14]

OPEN SOURCE: A CHARM OF POWERFUL TROUBLE

Tracing the etymology back to its sixteenth-century origins, the term "Witches' brew" has carried various interpretations, all of which center around a meaning of "a mixture of unknown and potentially dangerous ingredients," which is the point Ms. Neuberger seemed to want to make about open source.

A core principle to any security program is the need to know what's in the code (the "brew") you build or use. Without this information, you're left in the dark. Effective management of open source begins with the *identification* of open source.

A SOFTWARE BILL OF MATERIALS

The concept of a software Bill of Materials (SBOM) derives from manufacturing, where the classic Bill of Materials is an inventory detailing all the items included in a product. When a defective part is discovered, the manufacturer knows which of its products is affected and can begin the process of repair or replacement. Similarly, maintaining an accurate, up-to-date SBOM that inventories open source components is necessary to ensure that code remains high-quality, compliant, and secure. As in manufacturing, an SBOM of open source components allows you to pinpoint at-risk components quickly and prioritize remediation appropriately. A comprehensive SBOM lists all open source components in an applications as well as those components' licenses, versions, and patch status.

In the world of 2022, where 97% of commercial code contains open source, visibility into the open source components used in an application needs to be considered a mandatory and minimum requirement for any effective DevSecOps or AppSec effort. Without this information, business risk increases. The issue is preventable with a comprehensive view of the open source powering your business.

CONCLUSION

ADDITIONAL READING

NTIA Multistakeholder Process on Software Component Transparency

Executive Order on Improving the Nation's Cybersecurity

Census II of Free and Open Source Software—Application Libraries

Log4Shell: A Case for Trusting Open Source—With Guardrails

REFERENCES

1. Ian Forsyth, Global M&A momentum to keep going like a train, the Press and Journal, 2/25/2022.
2. Eran Orzel, 2021 Software Supply Chain Security Report, Argon Security, 2021.
3. CVEdetails.com, Docker security vulnerabilities, 2021.
4. Dominick Reuter, A developer sabotaged their own open-source libraries, breaking thousands of apps, in apparent protest of mega-corporations, Business Insider, 1/10/2022.
5. Jen Psaki, Readout of White House Meeting on Software Security, whitehouse.gov, 1/13/2022.
6. National Vulnerability Database, CVE-2020-11023 Detail, 2/7/2022.
7. Gary Armstrong and Rich Kosinski, The license and security risks of using Node.js, synopsys.com, 8/13/2019.
8. The JSON License, json.org, 2002.
9. The Linux Foundation and the Laboratory for Innovation Science at Harvard, Census II of Free and Open Source Software—Application Libraries, linuxfoundation.org, 1/2022.
10. Vincent Tabora, The Pareto Principle In Software Engineering —Applying the 80/20 Rule, medium.com/0xcode, 8/30/2021.
11. The Linux Foundation and the Laboratory for Innovation Science at Harvard, Report on the 2020 FOSS Contributor Survey, linuxfoundation.org, 12/10/2020.
12. Synopsys Cybersecurity Research Center, DevSecOps Practices and Open Source Management in 2020, synopsys.com, 2020.
13. Federal Trade Commission, FTC warns companies to remediate Log4j security vulnerability, ftc.gov, 1/4/2022.
14. Jack Gillum and Jennifer Jacobs, Some Federal Systems Affected by Software Flaw, Official Says, Bloomberg, 12/16/2021.

THE SYNOPSYS DIFFERENCE

Synopsys Software Integrity Group provides integrated solutions that transform the way development teams build and deliver software, accelerating innovation while addressing business risk. Our industry-leading portfolio of software security products and services is the most comprehensive in the world and interoperates with third-party and open source tools, allowing organizations to leverage existing investments to build the security program that's best for them. Only Synopsys offers everything you need to build trust in your software.

ABOUT THE 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, the 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 Road
Mountain View, CA 94043 USA

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