SYNOPSYS®

2019 OPEN SOURCE SECURITY AND RISK ANALYSIS

Synopsys Cybersecurity Research Center

CAN OUR READERS SAY WITH CONFIDENCE THAT THE OPEN SOURCE COMPONENTS USED IN THEIR APPLICATIONS ARE UP-TO-DATE WITH ALL CRUCIAL PATCHES APPLIED?

IT'S IMPOSSIBLE TO PATCH SOFTWARE WHEN YOU DON'T KNOW YOU'RE USING IT.

TABLE OF CONTENTS

ABOUT THIS REPORT

The Open Source Security and Risk Analysis (OSSRA) report provides an in-depth look at the state of open source security, compliance, and code quality risk in commercial software.

For over 15 years, security, development, and legal teams around the globe have relied on Black Duck® solutions to help them manage the risks that come with the use of open source. Built on the Black Duck KnowledgeBase™—the most comprehensive database of open source component, vulnerability, and license information available—Black Duck software composition analysis solutions and open source audits give organizations the insight they need to track open source in code, mitigate security and license compliance risks, and automatically enforce open source policies using existing DevOps tools and processes.

Each year, the Black Duck Audit Services team at Synopsys conducts open source audits on thousands of codebases for its customers, primarily in conjunction with merger and acquisition transactions. These audits are the primary source of data for the OSSRA report. This year's analysis examines findings from the anonymized data of over 1,200 commercial codebases audited in 2018.

The analysis of the 2018 data took place at the Synopsys Cybersecurity Research Center (CyRC). CyRC's global research labs include locations in Boston, Belfast, Calgary, and Oulu, Finland.

The Boston CyRC big data research team maintains the Black Duck KnowledgeBase. This team analyzes and refines open source activity from thousands of data sources to identify the most significant open source projects in use.

Our Belfast team identifies the impact of open source vulnerabilities and their exploitability. Their work forms the basis of Black Duck Security Advisories, which offer deep-sourced vulnerability data that the team discovers, curates, analyzes, and publishes hourly.

The Calgary group works to identify coding patterns contributing to software vulnerabilities.

Our researchers in Oulu identified the OpenSSL vulnerability known as Heartbleed and continue to perform protocol-based research.

This year, the CyRC Belfast team examined findings from the anonymized data of over 1,200 commercial codebases reviewed by the Black Duck Audit Services team in 2018. The 17 industries represented in the report range from aerospace to virtual reality (see the next page for a full list). The audit services team reviewed an average of 71 codebases per industry during 2018.

Operating within the Synopsys charter of making software secure and high-quality, CyRC publishes research such as the annual OSSRA report to support strong cyber security practices.

The 2019 OSSRA report includes insights and recommendations intended to help organizations and security, risk, legal, and development teams better understand the open source security and license risk landscape as they strive to improve their application risk management processes.

INDUSTRIES REPRESENTED IN THE 2019 OSSRA REPORT

- AEROSPACE, AVIATION, AUTOMOTIVE, LOGISTICS, TRANSPORTATION
- BIG DATA, AI, BI, MACHINE LEARNING
- COMPUTER HARDWARE & SEMICONDUCTORS
- CYBERSECURITY
- EDTECH
- ENTERPRISE SOFTWARE/SAAS
- ENERGY & CLEANTECH
- FINANCIAL SERVICES & FINTECH
- HEALTHCARE, HEALTH TECH, LIFE SCIENCES
- INTERNET OF THINGS
- INTERNET & MOBILE APPS
- INTERNET & SOFTWARE INFRASTRUCTURE
- MANUFACTURING, INDUSTRIALS, ROBOTICS
- MARKETING TECH
- RETAIL & E-COMMERCE
- TELECOMMUNICATIONS & WIRELESS
- VIRTUAL REALITY, GAMING, ENTERTAINMENT, MEDIA

This year's analysis examines findings from the anonymized data of over 1,200 commercial codebases audited in 2018.

2019 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT

As part of Synopsys' software composition analysis offerings, the Black Duck Audit Services team performs open source audits for organizations looking to assess license compliance and security risks in an application or codebase. One of the outputs of these audits is a complete listing of the open source components in use, referred to as a bill of materials. The BoM is cross-referenced with data contained in the Black Duck KnowledgeBase to identify potential license compliance and security risks. The KnowledgeBase currently identifies open source activity from over 16,000 sources worldwide, making it an authoritative source for open source projects and components.

As the report details, open source components and libraries form the backbone of nearly every application in every industry. Use of open source lowers costs and speeds development—both critical in today's agile software world.

Open source composition of scanned codebases

Black Duck Audits found open source in over 96% of codebases scanned in 2018, a percentage similar to the figures from the last two OSSRA reports. It's worth noting that most of the codebases found to have no open source consisted of fewer than 1,000 files. More than 99% of the codebases scanned in 2018 with over 1,000 files contained open source components.

In most industries, the year-to-year difference in the percentage of codebases containing open source was negligible (see Table 1).

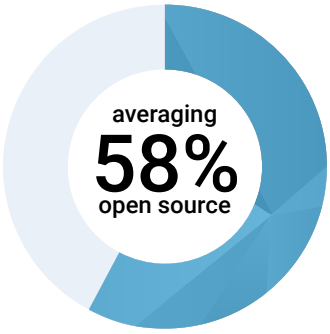
On average, Black Duck Audits identified 298 open source components per codebase in 2018 versus 257 in 2017. Open source represented 60% of the code analyzed in 2018, up from 57% in 2017. Reflected in this percentage is the fact that the audited codebases were generally from companies whose business is building software, versus enterprises for whom software supports their main business. Often, the value of these companies is in their proprietary code, so the ratio of open source to proprietary code in their codebases is lower. Analysts such as Forrester and Gartner, who tend to look at IT groups in enterprises, have noted that over 90% of IT organizations use open source software in mission-critical workloads[1] and that open source composes up to 90% of new codebases.[2]

Open source represented 60% of the code analyzed in 2018, up from 57% in 2017.

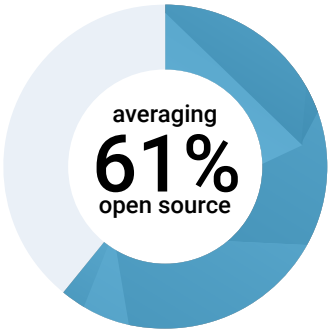
Table 1: Percentage of audited codebases containing open source components by industry (2018 & 2017)

Industry	2018	2017
Energy & CleanTech	100%	100%
Internet & Software Infrastructure	100%	100%
Virtual Reality, Gaming, Entertainment, Media	100%	100%
Financial Services & FinTech	100%	99%
Retail & E-Commerce	99%	99%
Telecommunications & Wireless	98%	97%
Cybersecurity	96%	100%
Big Data, AI, BI, Machine Learning	97%	92%
Internet & Mobile Apps	97%	100%
Healthcare, Health Tech, Life Sciences	96%	96%
Aerospace, Aviation, Automotive, Transportation, Logistics	96%	94%
Enterprise Software/SaaS	95%	97%
Marketing Tech	93%	100%
EdTech	92%	95%
Manufacturing, Industrials, Robotics	92%	100%
Internet of Things	91%	100%
Computer Hardware & Semiconductors	80%	100%

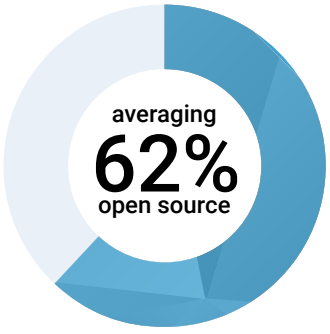
Also of significance is the number of the audited codebases that contained more open source than proprietary code. In 13 out of 17 industries, more than 50% of the average codebase comprised open source (see Table 2).



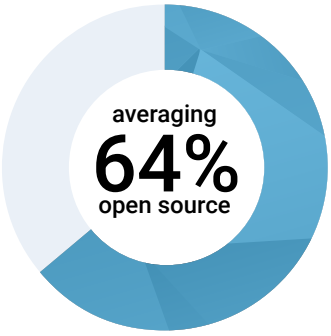
Enterprise Software/SaaS; Virtual Reality, Gaming, Entertainment, Media



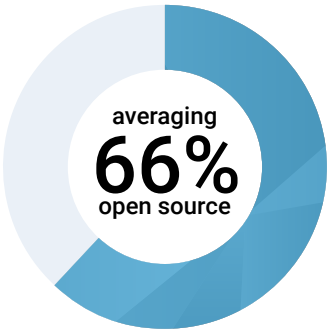
Internet & Software Infrastructure; Computer Hardware & Semiconductors



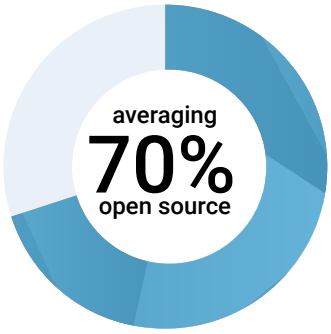
Retail & E-Commerce



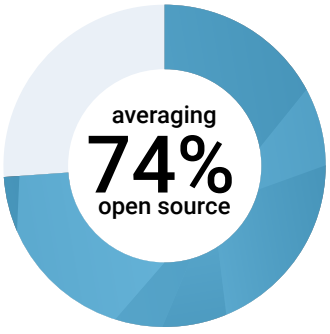
Financial Services & FinTech; Big Data, AI, BI, Machine Learning; Healthcare, Health Tech, Life Sciences; Energy & CleanTech



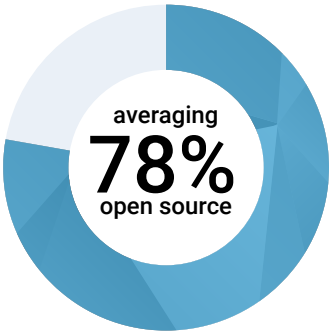
Internet of Things



Cybersecurity



Internet & Mobile Apps



Marketing Tech

A number of components were commonly used across different codebases. jQuery, open source software using the permissive MIT License, was found in 56% of the scanned codebases and in virtually every industry covered in the OSSRA report. Other notable open source components found in the scans include Bootstrap, an open source front-end web framework; jQuery UI, a curated set of user interface interactions, effects, widgets, and themes built on the jQuery JavaScript Library; and Font Awesome, an open source font and icon toolkit based on CSS and LESS (see Table 3).

jQuery, open source software using the permissive MIT License, was found in 56% of the scanned codebases and in virtually every industry.

Despite using so much open source, few companies accurately track the components they use in their code. Most lack the policies, processes, and tools to keep up with the choices made by their developers. As a consequence, all the good functionality that comes with open source also brings along a variety of risks.

Table 2: Average percentage of open source in each audited codebase by industry

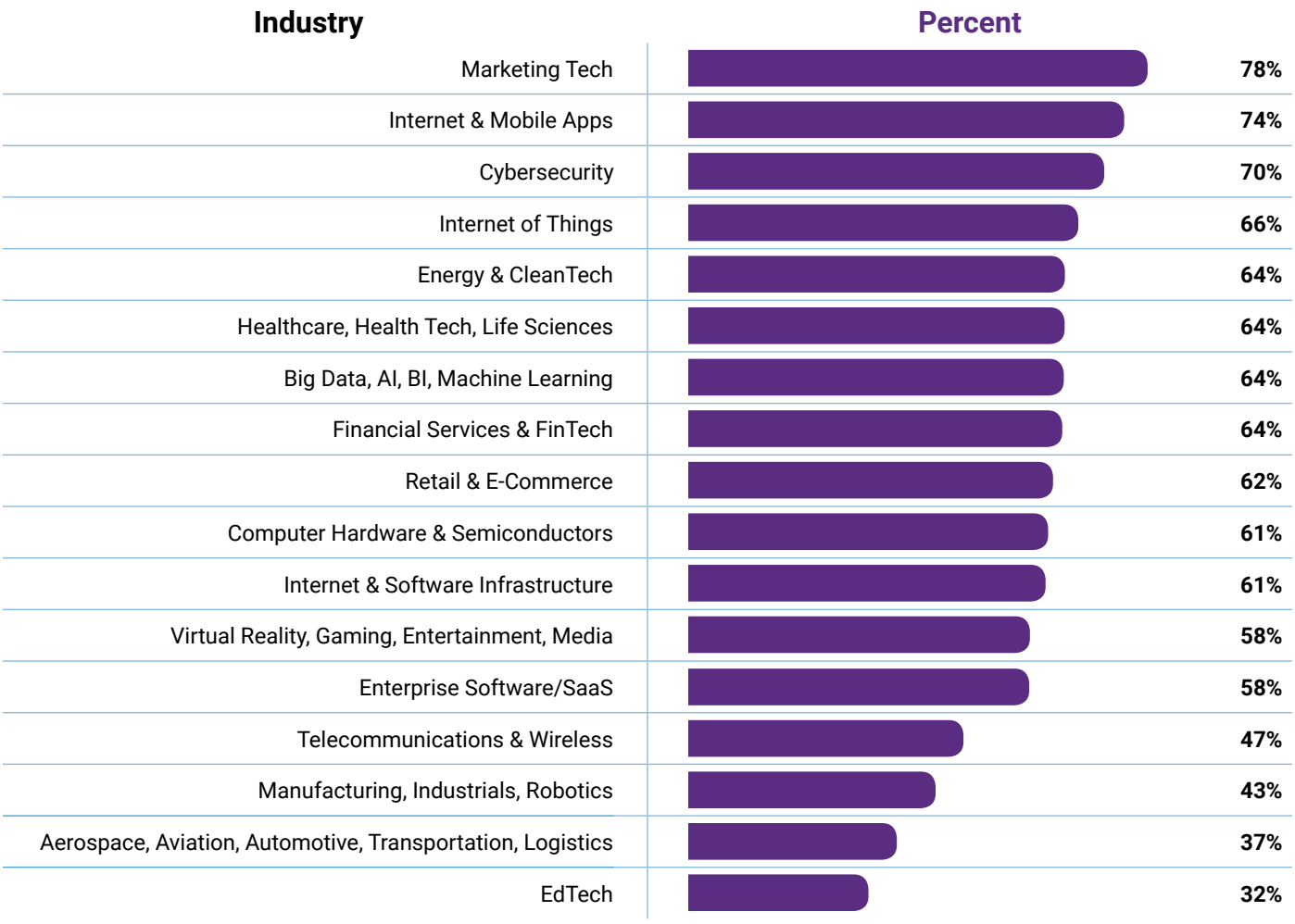
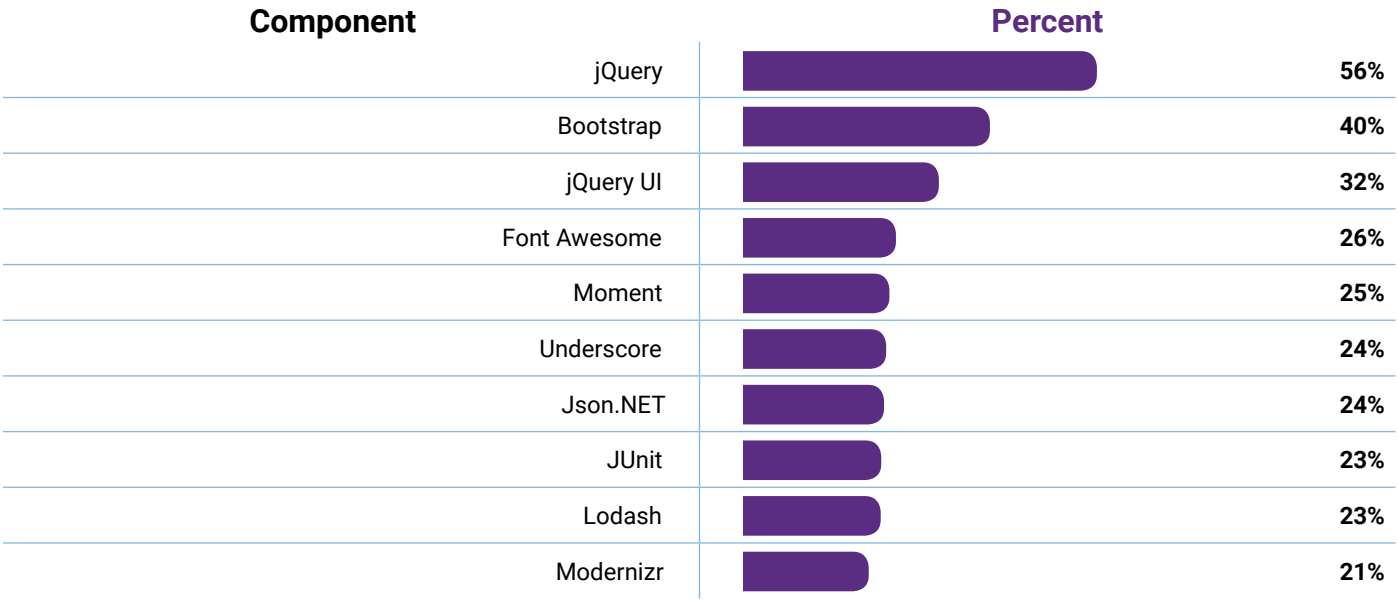


Table 3: Percentage of audited codebases containing the top 10 open source components



Open source security: The risk issue is unpatched software, not open source use

When a report on open source risk such as the OSSRA report is released, a few critics mischaracterize the findings as an attack on the use of open source technology itself. The reality is that open source is not less secure than proprietary code. But neither is it more secure. All software, be it proprietary or open source, has weaknesses that might become vulnerabilities, which organizations must identify and patch. Most organizations have thousands of different pieces of software ranging from mobile apps to cloud-based systems to legacy systems running on-premises. That software is typically a mix of commercial off-the-shelf packages, open source software, and custom-built codebases. Vulnerabilities affect all of them.

The open source community does an exemplary job of issuing patches, often at a much faster pace than their proprietary counterparts. But whether companies are using proprietary or open source software, an alarming number of them don't apply patches, opening themselves to risk.

Unpatched software vulnerabilities are one of the biggest cyberthreats organizations face, and unpatched open source components in software add to security risk. Certain characteristics of open source make vulnerabilities in popular components attractive to attackers. For example, unlike commercial software, whose publishers can automatically push fixes, patches, and updates to users, open source has a pull support model. In other words, ***you*** are responsible for keeping track of both vulnerabilities and fixes for the open source you use. Because open source is so pervasive, manually tracking components, their versions, and their vulnerabilities is a task far beyond many organizations' capabilities. An automated solution for identifying and patching known vulnerabilities in open source components can help these organizations manage vulnerability risks much more effectively.

The ubiquity of open source in both commercial and internal applications provides attackers with a target-rich environment as vulnerabilities are disclosed through sources such as the National Vulnerability Database (NVD), mailing lists, GitHub issues, and project homepages. The widespread use of open source can lead to another issue: Many organizations don't keep accurate, comprehensive, and up-to-date inventories of the open source components used in their applications. Even if you have an ongoing initiative to develop a comprehensive inventory of your IT systems, including all the software components for each system, your inventory might not be complete or current at any given time. An incomplete inventory makes it extremely difficult to maintain adequate software asset management procedures. For example, a staff report by the U.S. Senate Permanent Subcommittee on Investigations noted that Equifax's lack of a complete software inventory was a contributing factor to its massive 2017 data breach.[3]

Can our readers say with confidence that the open source components used in their public and internal applications are up-to-date with all crucial patches applied? If you can't answer that question and can't produce a full and accurate inventory of the open source used in your applications, it's time to create a bill of materials for your open source. It's impossible to patch software when you don't know you're using it.

Unpatched software vulnerabilities are one of the biggest cyberthreats organizations face, and unpatched open source components in software add to security risk.

CONSIDER THE FOLLOWING HEADLINES RANGING FROM 2009 INTO 2018:

"Unpatched Vulnerabilities the Source of Most Data Breaches."

—DARK READING

"Unpatched Applications Are #1 Cyber Security Risk."

—PC WORLD

"Unpatched Software Vulnerabilities a Growing Problem."

—OPSWAT

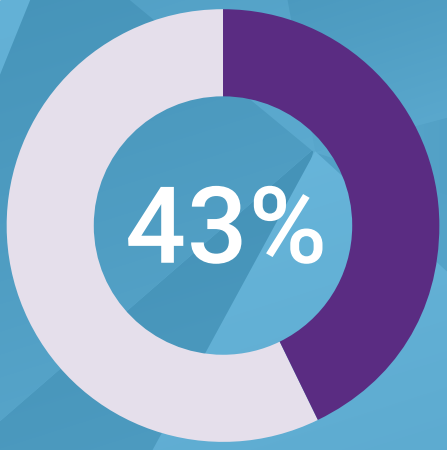
"Outdated, Unpatched Software Rampant in Businesses."

—THREAT POST

"Thousands of Companies Are Still Downloading the Vulnerability That Wrecked Equifax."

—FORTUNE

60% of the codebases audited in 2018 contained at least one vulnerability—still significant, but far better than the figure of 78% from 2017.



of the codebases scanned in 2018 contained vulnerabilities over 10 years old.

Another record year for disclosed vulnerabilities

The year 2018 was another record year for disclosed vulnerabilities. The NVD listed over 16,500 new vulnerabilities, and we added 7,393 open source vulnerabilities to the Black Duck KnowledgeBase. By contrast, approximately 4,800 open source vulnerabilities were reported in 2017. Well over 50,000 open source vulnerabilities have been reported in the past two decades.

Efforts such as the OSSRA report publicizing the urgency of identifying and patching known open source vulnerabilities seem to have had some impact in 2018. Sixty percent of the codebases audited in 2018 contained at least one vulnerability—still significant, but much better than the figure of 78% from 2017.

The average age of vulnerabilities identified in 2018 Black Duck Audits was 6.6 years, slightly higher than 2017—suggesting remediation efforts haven't improved significantly. The oldest vulnerability identified in the scans is probably older than some of our readers: CVE-2000-0388, a 28-year-old high-risk vulnerability in FreeBSD first disclosed in 1990. Indeed, 43% of the codebases scanned in 2018 contained vulnerabilities over 10 years old.

CVE-2000-0388, the oldest vulnerability identified in the 2018 scans, is probably older than some of our readers.

Vulnerability breakdown

Over 40% of the audited codebases contained high-risk vulnerabilities, including CVE-2018-7489, a critical FasterXML jackson-databind vulnerability; CVE-2017-15095, a critical deserialization flaw in jackson-databind; CVE-2014-0050, a high-risk vulnerability affecting Apache Tomcat, JBoss Web, and other products; and CVE-2017-15708, a critical vulnerability allowing remote code execution attacks in Apache Synapse (see Table 4).

The most common vulnerability found was CVE-2012-6708, a medium-severity vulnerability making versions of jQuery before 1.9.0 vulnerable to cross-site scripting (XSS) attacks. jQuery was also the component most likely to have identified vulnerabilities, with 14% of the codebases containing some type of jQuery vulnerability. This isn't surprising, as jQuery was the most common open source component found in the scanned codebases.

jQuery, the most common open source component found in the scanned codebases, was also the component most likely to have identified vulnerabilities.

Table 4: Percentage of audited codebases containing the top 10 high-risk vulnerabilities

CVE	Percent
CVE-2018-7489	12%
CVE-2017-7525	11%
CVE-2017-15095	11%
CVE-2015-6420	10%
CVE-2014-0050	9%
CVE-2017-15708	9%
CVE-2014-0107	9%
CVE-2016-3092	6%
CVE-2016-8735	5%
CVE-2014-3567	5%

The OSI notes the following open source licenses (in alphabetical order) as popular, widely used, or having strong communities:

2-Clause BSD License (a.k.a. Simplified BSD or FreeBSD License)

3-Clause BSD License (a.k.a. New or Revised BSD License)

Apache License 2.0

Common Development and Distribution License

Eclipse Public License

GNU General Public License (GPL)

GNU Library (or Lesser) General Public License (LGPL)

MIT License

Mozilla Public License 2.0

License risk in open source components

Sixty-eight percent of the 2018 audited codebases contained components with license conflicts. Most common were GNU General Public License (GPL) license violations, with 61% of the codebases having some form of GPL conflict. This makes sense, as GPL is one of the most common licenses and one of the most likely to conflict.

Different organizations define the terms "free software" and "open source software" and their variants (FOSS, FLOSS, "libre software," etc.) differently. At a minimum, all definitions of open source or free software include these essential characteristics: The source code is made available for inspection and modification, and it may be freely distributed.

An open source license is a type of license that allows the source code to be used, modified, or shared under defined terms and conditions. The Open Source Initiative (OSI), a nonprofit corporation that promotes the use of open source software in the commercial world, defines open source with 10 criteria and lists 82 OSI-approved licenses, with nine being "popular, widely used, or having strong communities" (see sidebar). By contrast, the Software Package Data Exchange® (SPDX®), which doesn't require strict compliance with all elements of the Open Source Definition, lists some 350-odd commonly found open source licenses and includes the concept of deprecated licenses.

The Black Duck KnowledgeBase lists over 2,500 licenses associated with software whose source is freely available on the internet. Most of these licenses don't meet the strict OSI and SPDX definitions of "open source," and while many are acknowledgeable as one-offs, all specify rights, and most have obligations that users must attend to.

Black Duck scans indicate that the 20 most popular licenses cover approximately 98% of the open source in use. The OSI and the open source community have done a commendable job in reducing the multitude of open source licenses seen a decade ago. But the fact remains that if you're using an open source component, the license the author applied to it matters, whether sanctioned by the OSI or not.

Types of open source licenses

Open source licenses generally fall into one of two categories: permissive and copyleft. Permissive licenses are sometimes referred to as "attribution style," and copyleft licenses are also known as reciprocal and viral licenses.

The permissive license is the most basic type of open source license. In most cases, it allows you to do whatever you want with the code as long as you acknowledge the authors of the code and follow other obligations such as redistribution and documentation requirements.

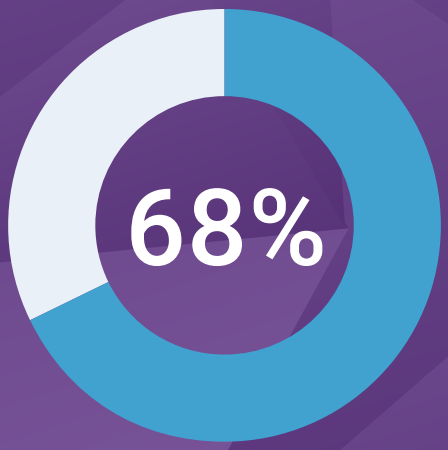
Copyleft licenses add further requirements to the permissive license. For example, if you distribute binaries, you must make the source code for those binaries available. You must make the source code available under the same copyleft terms as the original code. You can't place additional restrictions on the licensee's exercise of the license.

One common type of license variant comes from the addition of new restrictions to another license. For example, the JSON License is essentially the permissive MIT License with the addition "The Software shall be used for Good, not Evil." Such license additions are often well-intended but can still raise concerns, especially in merger and acquisition transactions, with lawyers needing to interpret the impact and risks of such modifications. In recent years, owners of many popular projects have removed code under the JSON License because of the license's moral ambiguity.

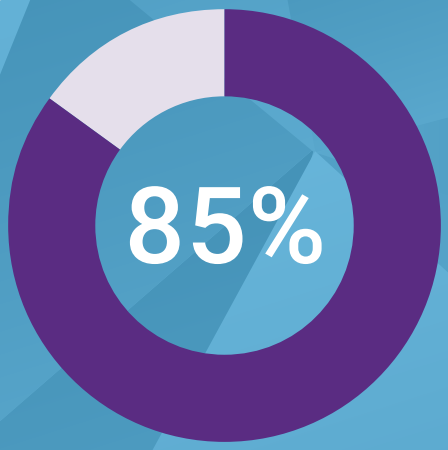
Whether open source licenses are permissive, copyleft, or some variation, organizations can manage and comply with their requirements only after identifying the open source components governed by those licenses. Failure to comply with open source licenses can put businesses at significant risk of litigation and compromise of intellectual property (IP). In their experience performing code audits for M&A due diligence, the Black Duck Audit Services team has found that 95% of scans reveal open source that the target didn't know was there.

Open source components with no or custom licenses

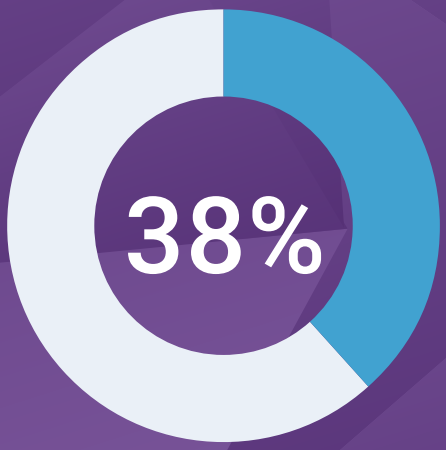
Open source components with no identifiable license terms also can be problematic. In the U.S. and many other



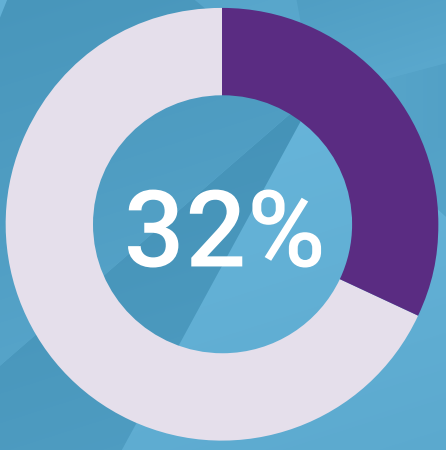
of the 2018 audited codebases contained components with license conflicts.



of audited codebases contained components that were more than four years out-of-date or had no development activity in the last two years.



of audited codebases contained components that were "not licensed."



of audited codebases contained custom licenses that had the potential to cause conflict or needed legal review.

jurisdictions, creative work is under exclusive copyright by default—and that includes code. Unless a license specifies otherwise (or the creators grant permission), nobody else can use, copy, distribute, or modify that work without incurring the risk of litigation. Organizations that use open source code that lacks clear statements of rights and obligations are at greater risk of violating copyright law.

Not all teams publishing free software assign a license to their projects. GitHub, the most popular repository of open source on the internet, introduced the requirement to attach a license to a project only a few years ago. Example code available from Stack Overflow and other developer forums is also a common source of software components without clear licensing.

Black Duck Audits designate a component as "not licensed" when the author has given no clear grant of license or terms of use in the code or associated files or on the site where it's hosted. Thirty-eight percent of the codebases audited contained components that were "not licensed."

Custom licenses, on the other hand, are software components whose license language seems to have been invented by the developer writing the code. In other words, the developer has assigned their own license language to the component instead of using a standard license. In all cases, organizations using this code must review and interpret these custom licenses to assess their risk. In 32% of the codebases audited, Black Duck Audits found custom licenses that had the potential to cause conflict, or at least needed legal review to determine the risk.

Open source license risk across verticals

Compared to the 2017 data, license conflicts in codebases audited in 2018 shrank a bit in most industries (see Table 5). While this trend is encouraging, the percentage of codebases with license conflicts (52–79%) is still too large for most companies (and their legal advisors) to feel comfortable.

Based on the audit data, organizations in all verticals should still be concerned with open source licensing and the potential risk of litigation or compromise of their IP because of their failure to comply with an open source license.

Operational factors in open source use

In addition to security and license risk, operational risk is a less acute but still important consequence of open source use. Many open source components in use are abandoned. In other words, they don't have a community of developers contributing to, patching, or improving them. If a component is inactive and no one is maintaining it, that means no one is addressing its potential vulnerabilities. Therefore, organizations determining the risks in a codebase must also consider the operational factors in open source components. Black Duck Audits found that 85% of codebases contained components that were more than four years out-of-date or had no development activity in the last two years.

Table 5: Percentage of audited codebases containing license conflicts by industry (2018 & 2017)

Industry	2018	2017
Marketing Tech	79%	77%
Retail & E-Commerce	74%	61%
Internet & Software Infrastructure	74%	78%
Big Data, AI, BI, Machine Learning	72%	72%
Internet of Things	72%	75%
EdTech	70%	77%
Healthcare, Health Tech, Life Sciences	70%	71%
Cybersecurity	68%	76%
Telecommunications & Wireless	68%	100%
Aerospace, Aviation, Automotive, Transportation, Logistics	68%	78%
Energy & CleanTech	67%	78%
Financial Services & FinTech	65%	78%
Enterprise Software/SaaS	65%	83%
Manufacturing, Industrials, Robotics	64%	91%
Virtual Reality, Gaming, Entertainment, Media	63%	92%
Internet & Mobile Apps	59%	64%
Computer Hardware & Semiconductors	52%	72%

CONCLUSION

Today's software is built on a foundation of reusable components—primarily open source. As this report notes, 99% of codebases with over 1,000 files—in other words, most of them—are likely to contain open source components. The 1,200+ codebases reviewed by the Black Duck Audit Services team in 2018 contained 298 open source components on average.

Those using open source often overlook associated security and license risks. Software developers routinely take code from open source repositories to embed in their company's products to speed up the development process. While the efficiencies and cost savings of code reuse are clear, organizations rarely review the incoming code regularly for potential security and legal issues. Too few companies manage their developers' use of open source, and too few can produce an accurate inventory of open source components, licenses, versions, and patch status. Thus, they remain uninformed about their open source risks and obligations.

While the number of vulnerabilities in open source is small compared to proprietary software, over 7,000 open source vulnerabilities were discovered in 2018 alone. Over 50,000 have emerged over the past two decades.

Of the codebases reviewed by the Black Duck Audit Services team in 2018, 60% contained at least one open source vulnerability. Over 40% contained high-risk vulnerabilities, and 68% contained components with license conflicts.

Only a handful of open source vulnerabilities—such as those infamously affecting Apache Struts or OpenSSL—are likely to be widely exploited. With that in mind, organizations should focus their open source vulnerability management and mitigation efforts on CVSS (Common Vulnerability Scoring System) scores and the availability of exploits, not only on "day zero" of a vulnerability disclosure but over the life cycle of the open source component.

Patch priorities should align with the business importance of the system, the criticality of the asset, and the risk of exploitation. Organizations should design patch policies with the understanding that patches for commercial software and open source components differ. Open source patches must originate from either the root project or the distribution channel where they obtained the component. But while commercial vendors can push updates and security information, consumers of an open source component are not likely to be aware of updates, patches, or security best practices unless they're engaged in the community supporting that component.

Proper management of open source software isn't only about security; it's also about license management. There are thousands of open source licenses, with more than 2,500 listed in the Black Duck KnowledgeBase. Whether a license is one of the most popular OSI-approved licenses or a one-off variant, unless organizations are aware of the rights, obligations, and restrictions of using a specific open source component, they can't be sure whether they comply with those obligations. Noncompliant organizations could theoretically lose rights to their proprietary code or call into question the ownership of their IP.

Recommendations

Use of open source itself is not risky; ***unmanaged*** use of open source is. To defend against open source security and compliance risks, we recommend organizations take these steps:

- **CREATE AND ENFORCE OPEN SOURCE RISK POLICIES AND PROCESSES.** Educate developers about the need for managed use of open source. Put in place an automated process that tracks the open source components in a codebase and their known security vulnerabilities, as well as operational risks such as versioning and duplications, and prioritizes issues based on their severity.

- **PERFORM A FULL INVENTORY OF THEIR OPEN SOURCE SOFTWARE.** Organizations can't defend against threats that they don't know exist. It's essential to obtain a full, accurate, and timely inventory of the open source used in their codebases. The inventory should cover both source code and information on how open source is used in any commercial software or binary deployed in production or used as a library in an application.

- **MAP OPEN SOURCE TO KNOWN VULNERABILITIES.** Public sources, such as the NVD, are a good first stop for information on publicly disclosed vulnerabilities in open source software. Keep in mind, however, that over 90 organizations contribute entries to the NVD. Not only does the NVD reflect their priorities, but there can be significant lags in data reporting, scoring, and actionability of the data in a CVE entry.

  Don't rely solely on the NVD for vulnerability information. Instead, look to a secondary source that provides earlier notification of vulnerabilities affecting your codebase and, ideally, delivers security insight, technical details, and upgrade and patch guidance.

- **CONTINUALLY MONITOR FOR NEW SECURITY THREATS.** The job of tracking vulnerabilities doesn't end when applications leave development. Organizations need to continuously monitor for new threats for as long as their applications remain in service. Importantly, any continuous monitoring strategy must take into account the composition of the software under attack, lest it become overwhelmed by the volume of vulnerability disclosures we experienced in 2018.

- **IDENTIFY LICENSING RISKS.** Failure to comply with open source licenses can put organizations at significant risk of litigation and compromise of IP. Educate developers on open source licenses and their obligations. Involve your legal advisors in the education process and, of course, in reviewing licenses and compliance with legal obligations.

- **MAKE SURE OPEN SOURCE IS PART OF M&A DUE DILIGENCE.** If you're engaged in an acquisition, understand that the target company is using open source—and likely not managing it well. Don't hesitate to ask questions about their open source use and management. If the software assets are a significant part of the valuation of the company, have a third party audit the code for open source.

Open source offers a plethora of benefits to organizations that use it—but only if they track what open source components they're using and identify any related security and legal compliance issues.

References

1. Mark Driver, What Every CIO Must Know About Open-Source Software, Gartner, March 30, 2017.
2. Amy DeMartine, The Forrester Wave™: Software Composition Analysis, Q1 2017, Forrester, Feb. 23, 2017.
3. Permanent Subcommittee on Investigations, How Equifax Neglected Cybersecurity and Suffered a Devastating Data Breach, Committee on Homeland Security and Governmental Affairs, U.S. Senate, accessed March 29, 2019.

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.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

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

SYNOPSYS®

04/22/19. rep-OSSRA-19.