

9th Annual

# State of the Software Supply Chain

Sonatype's industry-defining research on the rapidly changing landscape of open source, software development, and software supply chain security

PRESENTED BY



# Contents

**Introduction**.....3

**State of the Software Supply Chain by the numbers**.....4

## CHAPTER 1

**Open Source Supply, Demand, and Security**.....5

Open source supply sees a resurgence .....7

Open source consumption is decelerating..... 8

Individual ecosystem analysis .....10

Open source software security concerns see no sign of slowing .....10

Notable malicious packages and vulnerabilities .....11

Differentiating software vulnerabilities and malware.....12

Consumption behavior contributing to security concerns.....12

A global view of vulnerable open source downloads .....16

## CHAPTER 2

**Open Source Security Practices** .....17

The state of scorecard check scores.....18

The Importance of Maintenance .....21

Year-Over-Year Differences .....21

Conclusion..... 22

## CHAPTER 3

**Modernizing Open Source Dependency Management**.....23

Open source consumers are not paying attention .....24

What makes an optimal open source component? ..... 26

Understanding component upgrade urgency.....27

Downloads by upgrade urgency..... 29

Global patterns of open source consumption behavior .....31

## CHAPTER 4

**Software Supply Chain Maturity**.....34

Software Supply Chain Maturity — Peer Insights .....35

How mature are today’s software supply chains?.....36

How have software supply chain management trends changed?.....37

## CHAPTER 5

**Establishment and Expansion of Software Supply Chain Regulations and Standards**.....42

United States .....43

European Union.....46

Canada .....47

Global partnerships .....47

Are software supply chain regulations working? .....48

Navigating the policy frontier: Global cybersecurity regulations.....49

## CHAPTER 6

**AI in Software Development**.....51

Sonatype’s risks & rewards of AI survey ..... 52

The intersection of AI and software development ..... 52

Navigating concern .....53

AI’s open source toolkit: Components and models in focus..... 55

Conclusion: A collaborative future..... 59

**About the Analysis**.....60

**Acknowledgements**..... 61

# Introduction

In today's fast-paced world, the pursuit of excellence is a relentless journey. We all understand the significance of innovation, efficiency, and the individuals at the core of it all: developers. From our past eight State of the Software Supply Chain reports, we know that developer productivity soars when they have access to superior tools and better open source components, making them the driving force behind better security and better products.

But what exactly does “better” mean in the ever-expanding landscape of technology choices? This question, in part, is the reason we've been studying the software supply chain for the last nine years. As we deliver the 9th version of this State of the Software Supply Chain report, that question is more paramount than ever. As we explore how to make “better” software, it's not just about the introduction of AI or cutting-edge technologies; it's about addressing fundamental issues that, in many ways, have not changed in nine years. It's about the often-overlooked, yet vital, element that lies within our software supply chain: open source consumption behavior.

We aim to sift through the labyrinthine market of software components, not to add to the cacophony of choices but to streamline it. Why? Because choice is a double-edged sword. The consequences of choosing poorly are far-reaching.

Consider this — last year, we revealed that a staggering 85% of projects in Maven Central — the largest public repository for Java open source components — are inactive. In other words, developers are faced with a perplexing array of choices, with only a fraction of them leading to active, well-maintained projects. Yet, we also found, and re-affirmed this year, that 96% of all vulnerable downloads from Maven Central, had known fixes available. There are so many choices to make, and only with the right tools, the right automation, can developers truly be set up for success.

As we dissect the intricacies of open source adoption and consumption, we validate a frustrating truth — development practices remain rife with inconsistency. When choices are made poorly, this inconsistency translates into increased risks, discontent among developers, and, perhaps most significantly, a loss of both time and money.

The State of the Software Supply Chain report each year isn't just a cautionary tale, but a call to action. It is a response to the pressing need to redefine our priorities and a testament to our willingness to evolve. We find ourselves in a period of revolution. Modernization is our ally. With regulations a focus in nearly every region, an uncertain economic climate demanding cost savings and efficiencies, and malicious activity more prominent than ever, it's time for change.

In the following pages, we provide you with an in-depth update on open source usage trends and security practices. We continue to draw from public and proprietary data sources to illustrate a host of issues with effective supply chain management. We'll look at:

- ▶ Ongoing growth of the software supply chain, as well as persistent security concerns
- ▶ The advantages of using well-maintained open source packages
- ▶ Open source consumption and trends in upgrade urgency of components
- ▶ Peer insights into the use of SBOMs and mature software supply chain management
- ▶ The rise of open source and software supply chain regulations
- ▶ What role AI and ML play in assisting developers, and the challenges that AI practitioners face in developing AI products

We also look at what it really means to have a Software Bill of Materials (SBOM) and a Software Composition Analysis (SCA) program, and ultimately shed light on the path to a more efficient, cost-effective, and secure development.

# State of the Software Supply Chain by the numbers

**1 in 8** open source downloads have known risk

**245,000** malicious packages discovered—2X all previous years combined

**18.6%** of open source projects across Java and JavaScript that were maintained in 2022, are no longer maintained today

**96%** of vulnerable downloaded releases had a fixed version available

**10** superior versions of components are typically available, for every nonoptimal component upgrade made

**2x** When paired with optimal upgrades, good data saves you twice as much time or nearly 1.5 months of time per application, per year when upgrading components.

**135%** increase in the adoption of AI and ML components within corporate environments, over the last year

**67%** of survey respondents feel confident that their applications do not rely on known vulnerable libraries, despite 10% of respondents reporting their organizations had security breaches due to open source vulnerabilities in the last 12 months



CHAPTER 1

# Open Source Supply, Demand, and Security



A monk by the name of John of Salisbury wrote a famous phrase in a 12th century [manuscript](#), borrowed by Sir Isaac Newton and hundreds of others since:

*“We are like dwarfs sitting on the shoulders of giants. We see more, and things that are more distant, than they did, not because our sight is superior or because we are taller than they, but because they raise us up, and by their great stature add to ours.”*

The meaning of the passage is simple: The progress we make only happens because of the

progress in learning and understanding others have made before us.

Nowhere else is this seen more than in the adoption of open source. Nearly all of the software shipped today relies on previous innovation that is distributed freely on scaffolding built by the utmost experts in the world, available to all developers free of charge.

In past State of the Software Supply Chain reports, we’ve estimated that up to 90% of the code we run in production is of [open source origin](#). Therefore, the economics of open source

are good indicators of trends and challenges in the wider software market.

For the 9th consecutive year, we continue to track the growth of open source adoption across the top four major open source ecosystems. These collectively account for four of the top five languages in [GitHub](#), and a 60% share of the most popular programming languages according to PYPL Language popularity index<sup>1</sup>.

Leveraging our continued monitoring, we present the combined statistics of each ecosystem in the table below.

**FIGURE 1.1**  
**OPEN SOURCE ADOPTION AS PROJECTED FOR 2023**

Ecosystem	Total Projects	Total Project Versions	2023 Annual Request Volume Estimate	YoY Project Growth	YoY Download Growth Estimate	Average Versions Released per Project
Java (Maven)	557k	12.2M	1.0T	28%	25%	22
JavaScript (npm)	2.5M	37M	2.6T <sup>2</sup>	27%	18%	15
Python (PyPI)	475k	4.8M	261B <sup>3</sup>	28%	31%	10
.NET (NuGet Gallery)	367k	6M	162B <sup>4</sup>	28%	43%	17
<b>Totals / Avgs</b>	<b>3.9M</b>	<b>60M</b>	<b>4T</b>	<b>29%</b>	<b>33%</b>	<b>15</b>

<sup>1</sup> <https://pypl.github.io/PYPL.html> Accessed August 2023

<sup>2</sup> Figure estimated using npm download counts to from January to August 2023 as queried from <https://github.com/npm/registry/blob/master/docs/download-counts.md>

<sup>3</sup> YoY growth estimated based on known PyPI downloads from January to August 2023 as queried from <https://console.cloud.google.com/marketplace/product/gcp-public-data-pypi/>

<sup>4</sup> YoY growth estimated based on known NuGet Gallery downloads from January to August 2023 as queried from <https://www.nuget.org/stats>

# Open source supply sees a resurgence

The supply side of open source is an interesting metric to gauge the pace and scale of innovation that occurs in a given ecosystem. The more open source projects are published every year, the more innovation occurs in a given ecosystem.

New open source projects across the monitored ecosystems have been published at a relatively steady 15% average rate<sup>5</sup> in recent years, which was a significant reduction in pace from highs seen in 2019 and before.

FIGURE 1.2  
OPEN SOURCE NEW PROJECT GROWTH RATE OVER THE PAST 4 YEARS

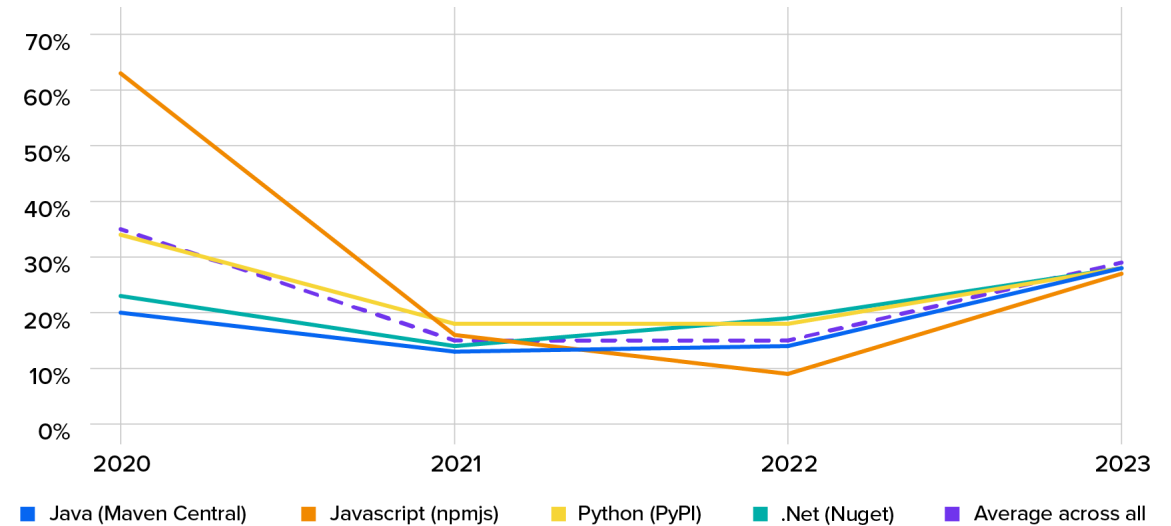
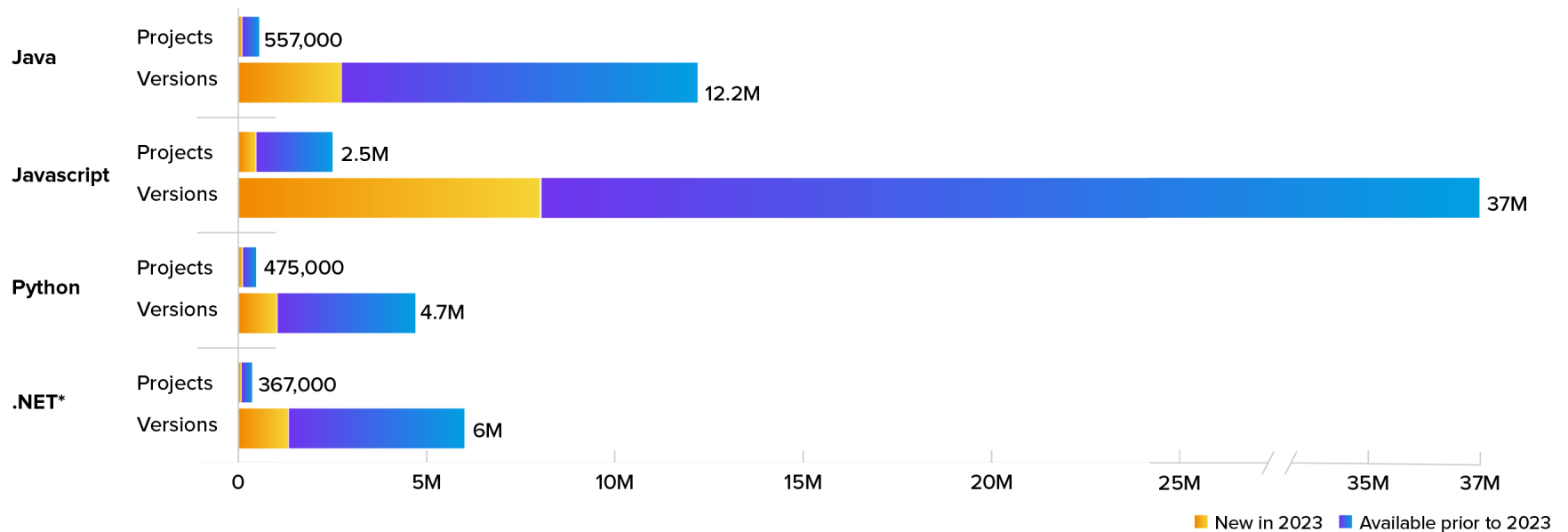


FIGURE 1.3  
OPEN SOURCE PROJECTS AND VERSIONS GROWTH



<sup>5</sup> In the 2021 report, we used a different method of estimating project counts for NuGet which resulted in a reduced number of projects being reported in 2022. We have corrected the NuGet historic numbers based on statistics published by the NuGet Gallery.

This two-year slump is most likely related to the COVID-19 pandemic period and associated slowdown. While some [studies](#) suggest productivity did increase during the 2020-2023 period in the U.S., a negative correlation emerges in open source production trends. This is further supported by another study that found productivity rates in information and communication technology did [decline](#) towards 2022. One other explanation could be that a lot of these projects are in fact coming from commercial activity and not people with spare time, which was in abundance during the pandemic.

To date, the data in 2023 shows the innovation slowdown is now over. Each monitored ecosystem showed a remarkably consistent project growth rate, varying just 2% across all four

monitored ecosystems to a total average growth rate of 29% year-over-year.

The rate of production growth is recovering across the board, and both Maven Central and NuGet are on track to exceed the rate of growth seen in 2020.

PyPi and npm, although growing, have not yet caught up to their original rate of growth but are on an upward trend. In a later section, we will see how breakthroughs and interest in AI and its related tooling are fueling the rate of growth in these ecosystems.

Between 2022 and 2023, the number of available open source projects grew an average of 29%. The average open source project in 2023 has released 15 versions available for

consumption, with specific ecosystem averages ranging from 10 to 22 across the different open source registries. That means 1-2 new versions every month and adds up to 60 million combined releases made available in the observed ecosystems.

## Open source consumption is decelerating

While we're seeing supply increase, consumption isn't keeping pace. The rate of download growth in open source consumption has slowed the past two years. In 2023, this trend continued with the average download growth rate sitting at 33%, which is exactly what it was last year. This is a stark comparison to the all-time high of 2021, which saw 73% year-over-year growth. There is a sign of slowdown in growth in the largest ecosystems, which is not surprising given the market saturation they already have.

Despite this, both of the largest ecosystems, Maven and npm, are each estimated to reach over a trillion requests in 2023, with npm reaching a staggering 2.6 trillion requests in total, continuing a modest growth that surpasses the total request rate of PyPI in 2022.

These two ecosystems account for 90% of the requests served with the remaining two growing at above average pace.

**FIGURE 1.4**  
CUMULATIVE ESTIMATED REQUESTS PER ECOSYSTEM OVER 6 YEARS

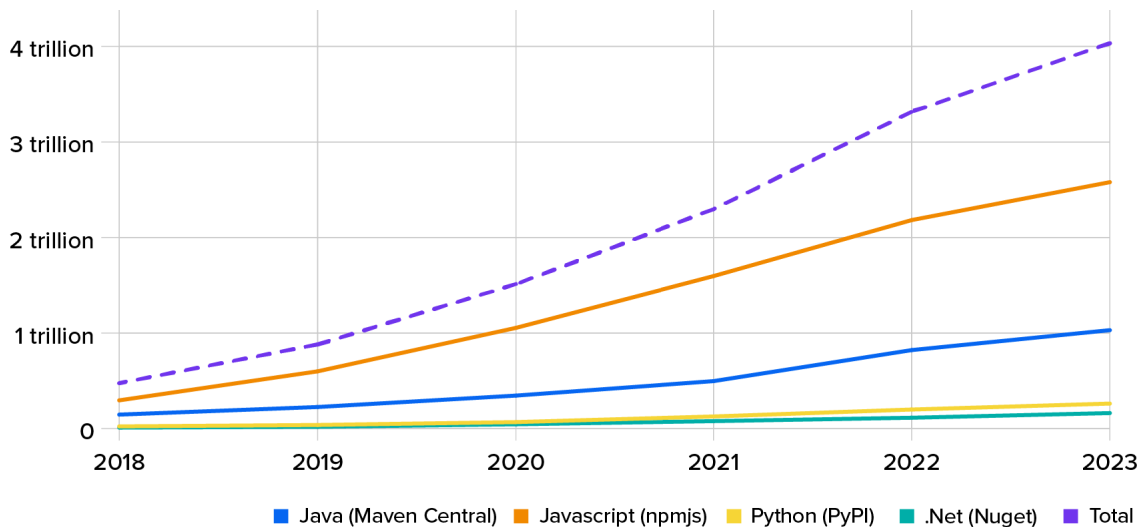




FIGURE 1.5

## GROWTH RATE OF THE MONITORED OPEN SOURCE ECOSYSTEMS OVER 5 YEARS

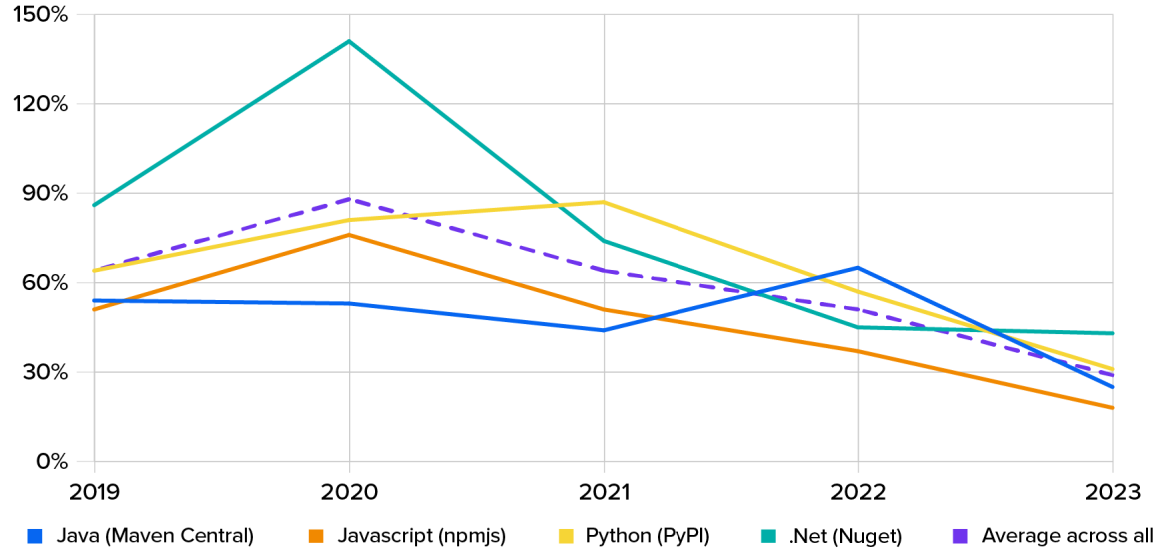
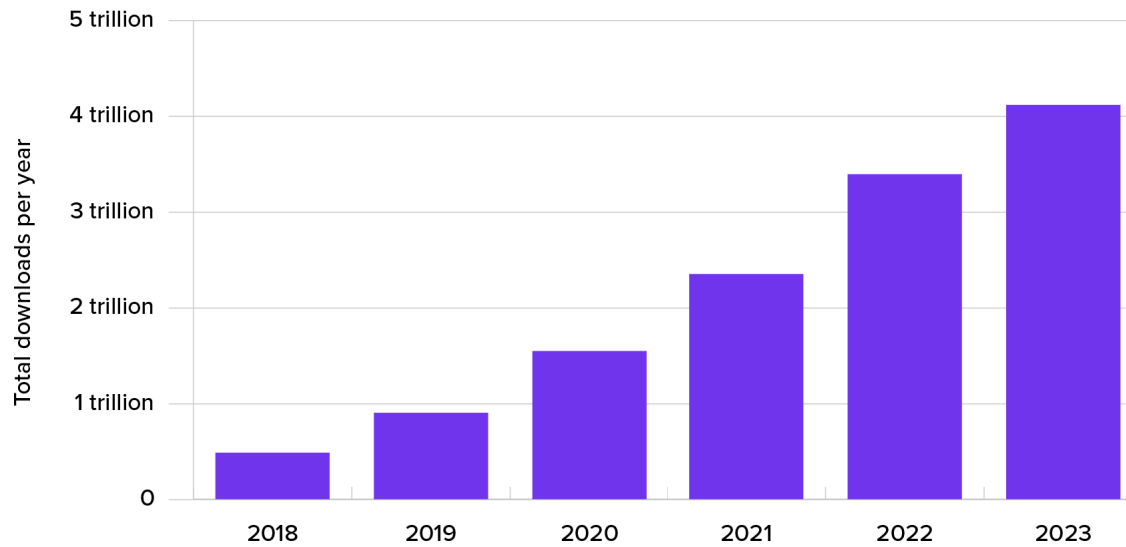


FIGURE 1.6

## TOTAL OPEN SOURCE REQUESTS OVER 6 YEARS



Requests are the fundamental measure of how popular an open source ecosystem is and how lively its usage is. Other factors within an ecosystem may vary, such as the larger size and complexity of Java packages compared to JavaScript packages.

Investigating the rate of growth for requests can reveal information about the state of open source adoption, as well as the growth of the software industry at large.

**Figure 1.5** charts these individual growth rates over time and displays an average across all 4 ecosystems.

Download requests for all open source ecosystems are still growing, but for a third year in a row there are signs that the pace of growth is slowing down.

We can see a clear delineation between the stabilization of large ecosystems like Maven and npm, and continued accelerated growth in PyPI and NuGet.

**Figure 1.6** charts the overall aggregate request growth across all ecosystems. It illustrates that although the pace of growth is slowing down, the absolute scale of growth continues to compound on previous years' rates. To put it simply, the pace of open source adoption still shows no signs of stopping.

## 2023 BY THE NUMBERS

JAVA

**1 trillion**  
projected request volume

**25%**  
YOY growth estimated

JAVASCRIPT

**2.1 trillion**  
projected request volume

**32%**  
YOY growth estimated

PYTHON

**261 billion**  
projected request volume

**31%**  
YOY growth estimated

.NET

**162 billion**  
projected request volume

**43%**  
YOY growth estimated

## Individual ecosystem analysis

### Java (Maven)

Through the first 7 months of 2023, 512 billion Java components were requested from the Maven Central Repository. This is a significant jump compared to the 821 billion requests in 2022.

Java continues to grow at a healthy pace, hitting an estimated 25% YoY request growth rate. If previous years are any indication, we may well see a spike towards the end of the year.

### JavaScript (npmjs)

npm is the juggernaut of open source registries, with an estimated download request count of over 2.6 trillion components (or to display it in full numbers: 2,579,310,885,518).

The growth of npm is the slowest of all the monitored ecosystems — estimated to be at 18% YoY. Nevertheless, owing to npm's substantial footprint, this translates to a staggering 400 billion requests, surpassing the combined total of requests served by PyPI and NuGet.

### Python (PyPI)

Python continues to expand at a high pace, fueled by the language's popularity and innovative uses, including AI. In 2023, PyPI served over 178 billion requests. This year, we estimate PyPI request volume will hit 261 billion packages. This represents 31% YoY growth.

### .NET (NuGet)

NuGet is the chosen ecosystem of the .NET family of languages and continues to serve engineers working with the growing set of Microsoft technologies. The rate of growth in NuGet is estimated to be the fastest amongst the cohort. Developers downloaded 113 billion NuGet packages in 2022, which was well above our estimate last year. In 2023, NuGet is estimated to serve 162 billion requests, representing 43% YoY growth.

## Open source software security concerns see no sign of slowing

In 2022, we reported a massive increase in the growth of malicious attacks on the software supply chain. Since our last report, this method of propagating security threats using trusted developer utilities and ecosystems has continued to evolve and flourish.

A troubling trend has emerged in the software supply chain over the past few years of tailor-made packages designed to run a malicious payload on download — without any developer interaction. This form of intrusion relies on developers not recognizing that the build breakage resulting from the fake package might be an indication that something nefarious has already happened on their system. We did a deep dive into types of malicious attacks in [last year's report](#).

In our YoY monitoring, at the time of writing in September 2023, we have logged **245,032 malicious packages** — meaning in the last year, we’ve seen the number of malicious packages tripled. Looking at it a different way, it also indicates that in one year alone, we’ve seen twice as many supply chain attacks as the cumulative numbers in previous years

This pace of growth is astonishing. It signals the role of the supply chain as one of the fastest growing vectors for adversaries to [execute malicious code](#). Furthermore, we have seen an increase in nation-state actors leveraging these vectors (see our deep dive section below).

This is alarming news. Even though many open source ecosystems have implemented new security policies, such as [mandatory MFA](#), they usually only address the issue of protecting existing open source publishers from attack. Often, packages containing malicious code are treated very similarly to packages with new security vulnerabilities — and they are taken down entirely based on a volunteer effort following a vulnerability removal process, which is not appropriate when the code is designed to be malicious from the start. This approach can lead to the malicious packages being up longer than necessary, leaving developers at risk.

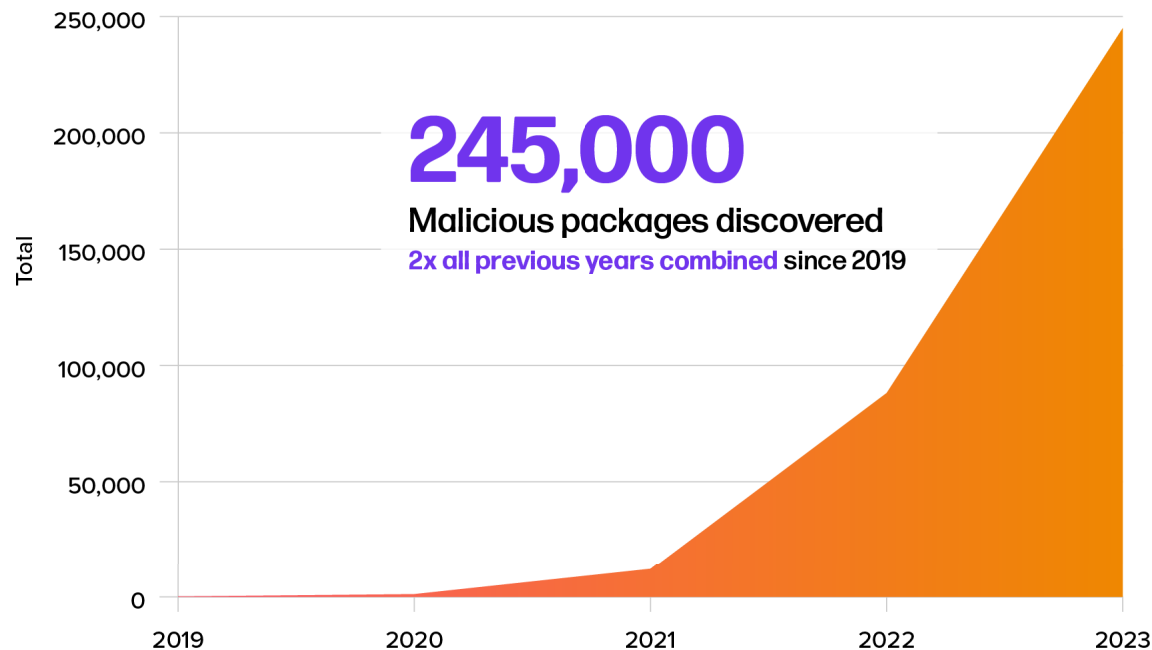
## Notable malicious packages and vulnerabilities

As we continue to document an overall rise in malicious attacks on open source ecosystems, the monitored 2022-2023 period has also seen more professional criminal campaigns emerge. The software supply chain lends itself well to the cybercriminal ecosystem — either as an initial access vector to [Initial Access brokers](#) or even as a means of distributing initial access malware for Advanced Persistent Threat groups. Here are several examples we’ve seen this year:

### Lazarus created PyPI package ‘VMConnect’ imitates VMware vSphere connector

In August 2023, Sonatype discovered a malicious Python package, ‘VMConnect,’ on PyPI, which mimics a legitimate VMware module. This is part of a wider cyber campaign called “Paper-Pin,” and is widely thought to originate from the Lazarus Group, a North Korean state-affiliated organization. The packages aim to download further malicious payloads from attacker-controlled URLs. The focus on VMware, a widely-used virtualization platform, is particularly concerning, as a successful compromise could have far-reaching implications for enterprise networks and is widely attractive to state-affiliated actors. [Read our full deep dive](#)

FIGURE 17  
NEXT GENERATION SOFTWARE SUPPLY CHAIN ATTACKS (2019-2023)



## ChatGPT histories were uncovered due to a vulnerability in Redis component used by OpenAI

In March of 2023, ChatGPT users experienced a data leak where chat histories displayed other people’s queries. OpenAI identified the issue as a race condition vulnerability in an open source component called Redis, which they use for caching user data. This flaw made sensitive data of about 1.2% of ChatGPT Plus subscribers accessible to others. The vulnerability was exacerbated by a recent server change that increased the probability of the race condition occurring. The issue underscores the importance of even rarely occurring vulnerabilities, especially in widely-used components like Redis, given their potential to cause widespread disruption and data exposure. [Read our full analysis.](#)

## PyTorch namespace confusion attack targets utilities aimed at AI developers

In the past couple of holiday seasons, we’ve seen some big supply chain attacks, including one on PyTorch, a popular machine learning framework. The attackers used a tactic known as namespace confusion to specifically go after the experimental “nightly” build of PyTorch. They managed to steal sensitive data, signaling that hackers are increasingly setting their sights on AI and machine learning tools. These tools are becoming more critical in various sectors, making them attractive targets. While only the experimental build was hit, the incident serves as a wake-up call for better security in the booming AI field. [Read our full analysis.](#)

## A timeline of attacks

We have continued to curate a timeline of known malicious packages and malware campaigns. This [interactive timeline](#) summarizes notable supply chain incidents, next-gen attacks and other incidents propagated using the software supply chain.

## Differentiating software vulnerabilities and malware

Up until now, we’ve been talking about malware and malicious attacks on the software supply chain - or maybe better stated as malware propagated using the open source supply chain. In this next section, we’re going to discuss software vulnerabilities. While the two concepts are related, they are very distinct, so we’d like to quickly define the difference between a vulnerability and a malware.

### Software vulnerability: A flaw in the code

A software vulnerability is akin to a flaw in code, much like a faulty lock on a door. However, unlike malware, vulnerabilities are not intentional. Instead, they represent weaknesses in software components or projects.

Similar to how a faulty lock compromises the security of a building by allowing unauthorized access, a software vulnerability creates a gap in the software’s security perimeter. This gap

becomes an entry point for intruders to exploit, gaining unapproved access to the system, application, or component.

### Malware: Malicious intent in open source

Malware, short for “malicious software,” poses a significant threat to open source software ecosystems. It encompasses a wide range of malicious programs, such as viruses, worms, trojans, ransomware, spyware, and adware, all designed to gain unauthorized access to information or systems.

With its various forms, malware’s primary purpose is to steal data, install harmful software, gain control of a network, or compromise software or hardware. Threat actors employ diverse distribution methods, such as infected email attachments, malicious websites, or compromised software downloads.

## Consumption behavior contributing to security concerns

Our report last year revealed a startling statistic — nearly 96% of component downloads with known vulnerabilities could be avoided as a better, fixed version is already available. This illustrates a clear need for organizations to pay closer attention to what versions they are adopting.



FIGURE 1.8

## SOFTWARE SUPPLY CHAIN ATTACKS, DEC 2021–AUGUST 2023



# 249,556,989

total Log4j downloads since Dec 15, 2021 | **29% vulnerable**

# 23%

Vulnerable downloads (Aug 20-27, 2023) | **3,490,799 total downloads**

There is widening evidence that whatever the standard practice for avoiding vulnerable components today, the controls are not having the effect needed to reduce the attack surface. For example, as of September 2023, downloads vulnerable to the infamous Log4Shell vulnerability still account for nearly a quarter of all net new [downloads](#) of [Log4j](#). It should be highlighted, that almost two years after the initial finding of this vulnerability, we're seeing this pace continue every week — **that a quarter of all downloads are of the vulnerable version of Log4j**. This is only part of the story. The reality is, nearly 1/3 of all Log4j downloads, ever, are of the vulnerable version.

As we discussed last year, the numbers for other critical vulnerabilities that have not received

as much widespread media attention are even more depressing.

This warrants concern and calls for behavioral adaptation at organizations because critical vulnerabilities are widely exploited by bad actors, even at the state level. For example, Log4Shell has [topped CISA/NSA charts](#) for active state-sponsored exploitation for well over a year now. This is also echoed in the OpenSSF's recently released [Consumption Manifesto](#), which calls for organizations to take responsibility for the open source they use, how it is consumed, and how they manage the risk associated with that consumption.

According to a joint consortium of national operators including CISA, NSA, NCSC-UK and others, attackers are exploiting older well-known vulnerabilities much more frequently than new [zero-day vulnerabilities](#). This is extremely important to understand. While we should of course worry about zero-days, we also know that 96% of vulnerable open source downloads have a non-vulnerable fix available. Those 96% need to be addressed.

For this year's report, we've taken a closer look at how vulnerabilities are consumed from Maven Central, with a special focus on what sort of geographic variance might exist.

According to a joint consortium of national operators including CISA, NSA, NCSC-UK and others, attackers are exploiting older well-known vulnerabilities much more frequently than new zero-day vulnerabilities.

## Vulnerable components consumed

Let's start off by looking at the top level. In 2022 we saw 12% of downloads served by Maven Central<sup>6</sup>, contained at least one known security vulnerability.

This number is important when considering that the easiest way to reduce risk of a supply chain incident caused by a vulnerability is to simply choose a better, non-vulnerable version of a component.<sup>7</sup> There is some improvement in these however, the number of vulnerable downloads in 2021 was 14% — and the number to date in 2023 sits at around 10%.

<sup>6</sup> Countries and regions with over 100,000 annual requests

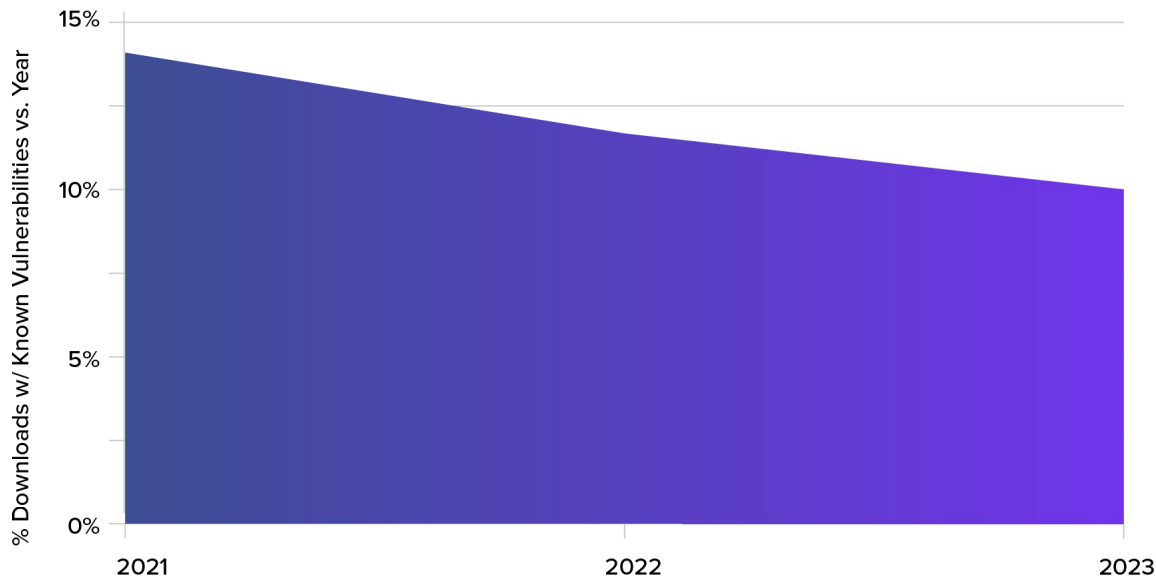
<sup>7</sup> A limitation in this report is that the numbers reported are of security vulnerabilities known as of August 2023, even when inspecting retrospectively. The amount of vulnerabilities known at the time of download might have been lower, which might partially explain the improvement over time.

However, when investigating the downloads that contained a vulnerability in 2022, it emerges well over a third of the components consumed that had known vulnerabilities were Critical<sup>8</sup> in severity — and a further 30.5% had a High Severity rating.

This trend holds true nearly universally across all regions — suggesting that component consumption is largely an unmanaged decision today. This is in contrast to the number of known critical vulnerabilities in the [National Vulnerability Database](#) — with over double the amount of criticals consumed over the spread of known vulnerabilities.

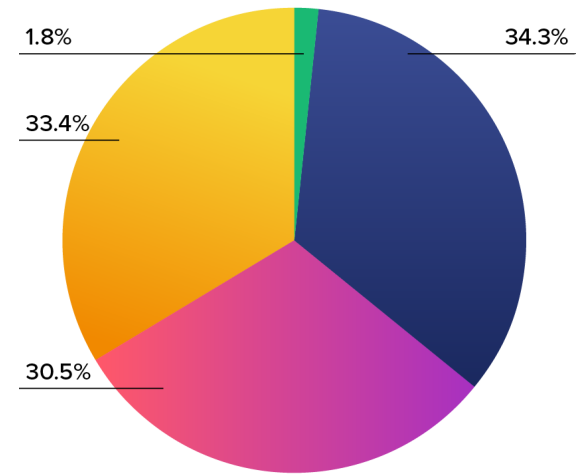
The increase of critically vulnerable components being consumed could be due to the fact that these vulnerabilities are found and reported primarily in more popular and widely-adopted open source software. Popularity begets more attention from good and bad actors, resulting in increased likelihood of a critical issue being present. It’s also worth noting that these more popular components have an official disclosure process to communicate through. Meaning, on average, these critical vulnerabilities should be the ones that are most noticed. But, as we’ve seen with the vulnerable version of Log4j, “knowing” is only half the battle. Organizations have to care, and they have to have an automated way to address this issue.

**FIGURE 1.9**  
PERCENTAGE OF COMPONENTS WITH KNOWN VULNERABILITIES SERVED FROM MAVEN CENTRAL



<sup>8</sup> As defined in CVSS 3.1 <https://www.first.org/cvss/specification-document>

**FIGURE 1.10**  
2022 VULNERABLE DOWNLOADS BY SEVERITY



**FIGURE 1.11**  
NVD – KNOWN VULNERABILITY SEVERITY

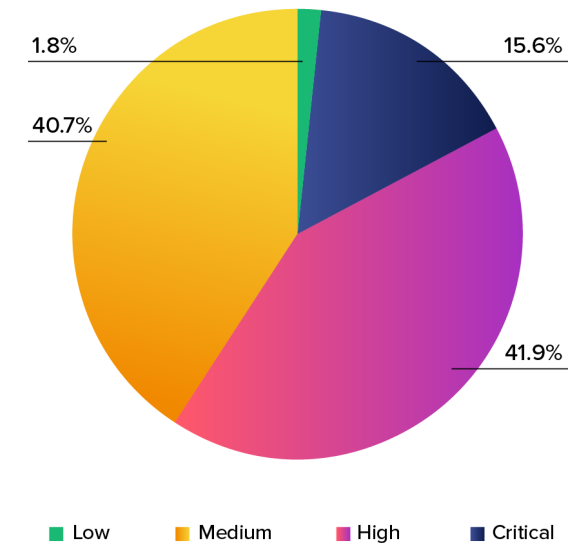
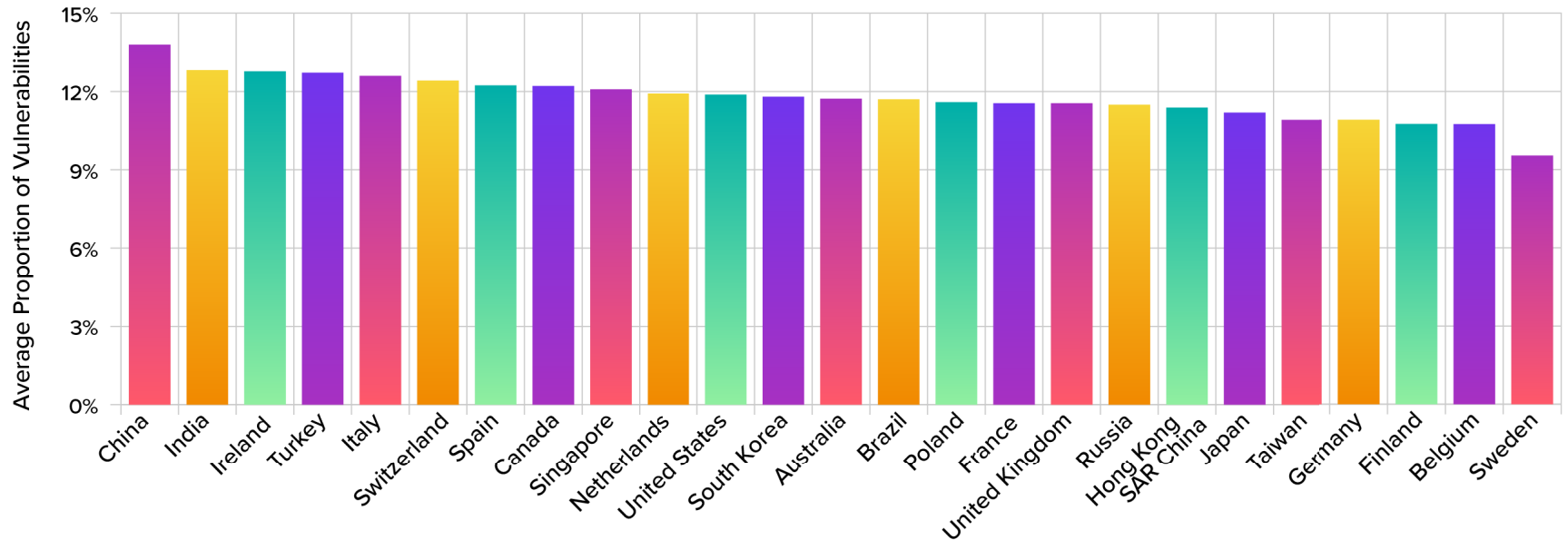


FIGURE 1.12

## AVERAGE VULNERABILITIES BY COUNTRY WITH OVER 1 BILLION DOWNLOAD VOLUME



## A global view of vulnerable open source downloads

Software development has evolved into one of the most globally influential industries, shaping various sectors and regions in unique ways. However, not all regions share the same level of emphasis on software development. To gain insight into how the trends we've explored thus far manifest on a global scale, we conducted an analysis that looks at open source vulnerability consumption, by country.

Our study focused on countries that collectively downloaded over 100 million open source components from Maven Central in the past year. By scrutinizing the percentage of vulnerabilities

associated with the software downloaded in each region, we start to gain insights into how different parts of the world manage their software supply chains.

In **Figure 1.12**, we delineate those that have stronger management programs, from those who don't, by plotting the percentage of vulnerabilities, against the average number of vulnerable downloads (approximately 12%) — and apply a ranking based on how countries compare to that average. But it's important to consider the context and one of the most important figures to come out of Sonatype's research: 96% of known vulnerabilities downloaded from Maven Central have a non-vulnerable version available.

The countries isolated above are 25 of the largest consumers of open source software in the

world. Even at the low end of our criteria, around a 100 million downloads, 9.5% of those downloads are vulnerable components. When you consider juggernauts of open source consumption like the United States, the EU (collectively), and China, tens of billions of vulnerabilities have entered into the supply chains that produce the software we all use and our governments run on.

While we're only scratching the surface with this regional view of vulnerable downloads, you can see a deeper dive into open source consumption patterns within specific economic regions in Chapter 3 of this report where we further unravel the intricacies of dependency management on a global scale. We also summarize the role regulations are having on the industry in Chapter 5.



CHAPTER 2

# Open Source Security Practices



In 2020 the OpenSSF project started collecting information on the software security practices employed by open source projects. This substantially increased transparency of open source development practices by centrally tracking the usage of commonly accepted best practices. In a previous version of this report, we analyzed the extent to which higher scores on these checks are associated with better security

outcomes and found them to be highly predictive of vulnerability status. In this year's report, we delve into the state of adoption of Scorecard best practices overall and also by ecosystem (focusing on Java and JavaScript). We look at checks today as well as over time, looking back at the prior year to see both how community practices have evolved and how the checks themselves have evolved over time.

## The state of scorecard check scores

In last year's report, we conducted an analysis of the connection between scorecard checks and known vulnerabilities. One outcome of this analysis was information on which checks are most closely associated with better vulnerability status. **Figure 2.2** summarizes this information.

### THE CHECKS

Our dataset consists of scorecard checks from 2022-06-27 and 2023-07-03 retrieved from the Scorecard BigQuery database. Each check is scored from 0 to 10. If the check fails or there is lack of evidence for a check, the Scorecard project assigns it a score of -1. We mapped these -1 values to 0 for our analysis.

#### BINARY ARTIFACTS

Checks whether binary artifacts are checked into the repository, reducing the score if binary artifacts are found.

#### BRANCH PROTECTION

Checks whether the project uses branch protection on its default and release branches to prevent maintainers from circumventing workflows like CI tests or code review when pushing changes.

#### CII-BEST PRACTICES

Checks whether a project has obtained an [OpenSSF \(formerly CII\) Best Practices Badge](#).



#### CODE REVIEW

Checks whether recent code changes have been peer reviewed before being merged.

#### DANGEROUS WORKFLOW

Checks whether the project avoids dangerous coding patterns in GitHub Action workflows.

#### DEPENDENCY UPDATE TOOL

Checks whether the project is using tools to help update its dependencies.

#### FUZZING

Checks whether the project is using fuzzing tools.

#### LICENSE

Checks that the project includes a license file.

#### MAINTAINED

Scores the project based on the amount of commit and issue tracking activity it has had over the last 90 days.

#### PACKAGING

Checks whether the project builds and publishes official packages from the CI/CD pipeline.

#### PINNED DEPENDENCIES

Checks whether project dependencies (including dependencies for GitHub Actions and Docker files) are pinned to specific versions.

#### SECURITY POLICY

Checks that the project includes a security policy.

#### SIGNED RELEASES

Checks that the project cryptographically signs releases.

#### TOKEN PERMISSION

Evaluates whether the project's automated workflow tokens follow the principle of least privilege.

#### VULNERABILITIES

Checks that the project and its dependencies have no open, unfixed vulnerabilities. Projects score 10 for no vulnerabilities and 1 point is deducted for each open vulnerability, down to a minimum score of 0.

#### VULNERABLE

A binary version of Vulnerabilities that we added to the dataset. A project is vulnerable if its Vulnerabilities score is less than 10.

**Code Review** (e.g. requiring review of pull requests before merging) was the most important practice, followed by **Binaries** (not checking binary data into the repository), **Dependencies Pinned** (pinning dependencies to specific versions), and *Branch Protection* (preventing direct pushes to the main branch).

The figure below shows how widely the scorecard best practices have been adopted. The chart presents the average value for each check across our 2023 dataset. Note that **Code Review**, which we found last year to be the practice most highly associated with good security outcomes, is not widely-practiced in general, with an average score of less than 1. Roughly 19% of projects score greater than 0 on Code Review, and only 2% receive the full score

of 10 (indicating that all code changes have undergone review). **Branch Protection**, which supports code review by requiring changes to go through the review process, also scores low (just 0.24 on average). The **Vulnerabilities** check measures whether a project has open, unpatched vulnerabilities, either associated with the project itself or with its dependencies. The Vulnerabilities score starts at 10 and is decreased by 1 for each open vulnerability with a minimum score of 0 (e.g. a project with 10 open vulnerabilities and a project with 15 open vulnerabilities both score 0). This means that the average score of 8.9 indicates that projects on average have at least 1 open vulnerability. Of course not every project is vulnerable: just over 14% of projects have a Vulnerabilities score of less than 10 (indicating they have at least one

open vulnerability). If we look at just vulnerable projects, we find that on the date our 2023 data was collected they had on average at least 7.7 open vulnerabilities per project.

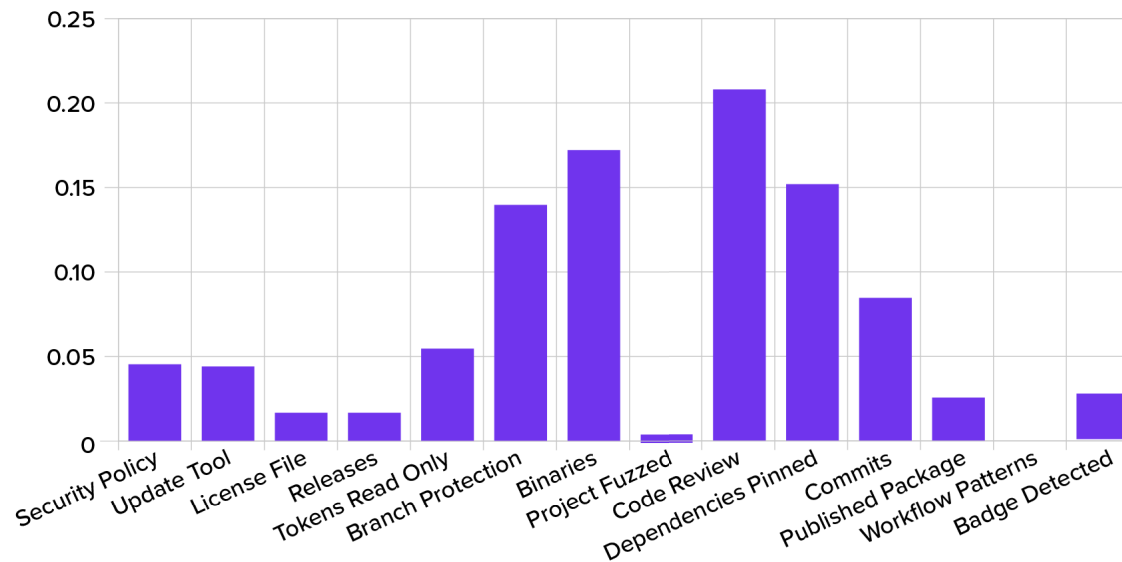
Another important low-scoring check is **Maintained**, which checks whether a project is being actively maintained, either via regular code commits or timely responses to issues. Maintenance is generally a prerequisite for security patches. On average, projects score about 0.66 on the Maintained check. Out of 1,176,407 total projects, just 118,028 of them (~11%) have a Maintained score greater than 0.

Looking at the data by ecosystem (shown in the figure below), we see that Java projects score higher on average than JavaScript projects for a number of security-relevant checks. Java projects score on average:

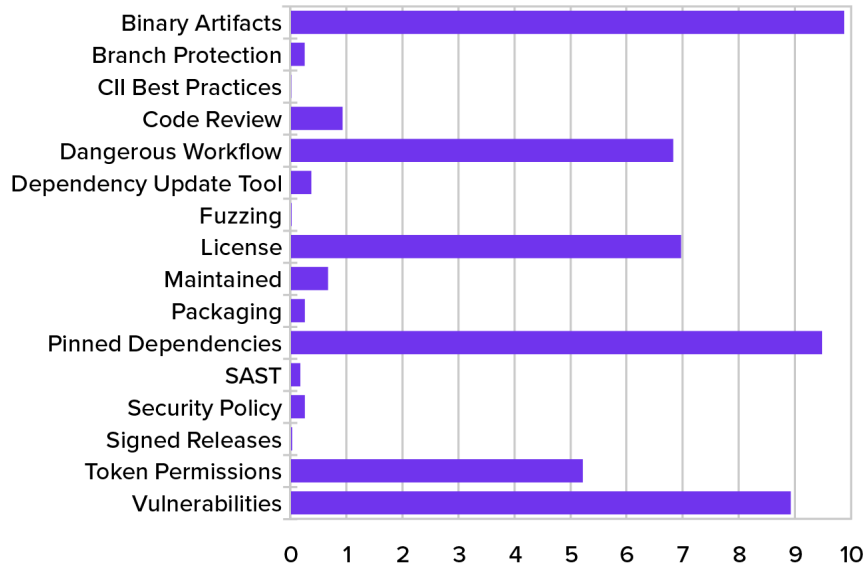
- ▶ 67x higher on Signed Releases
- ▶ 3.2x higher on SAST
- ▶ 3.1x higher on Maintained
- ▶ 2.0x higher on Branch Protection
- ▶ 1.7x higher on Dependency Update Tool
- ▶ 1.7x higher on Code Review

Java projects also score 10x higher on Fuzzing, though very few projects in either ecosystem take advantage of fuzzing technology. Rates of

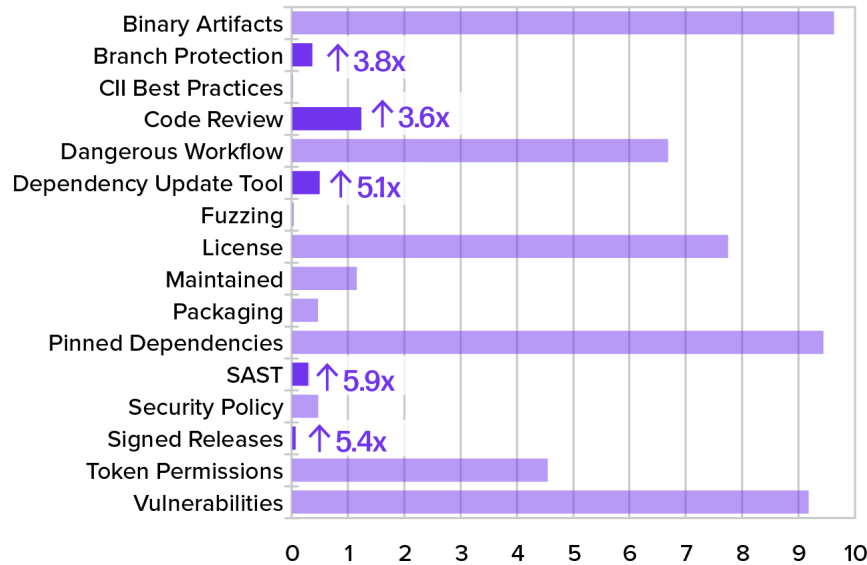
**FIGURE 2.1**  
**ELEMENTS MOST USEFUL FOR IDENTIFYING VULNERABLE PROJECTS**



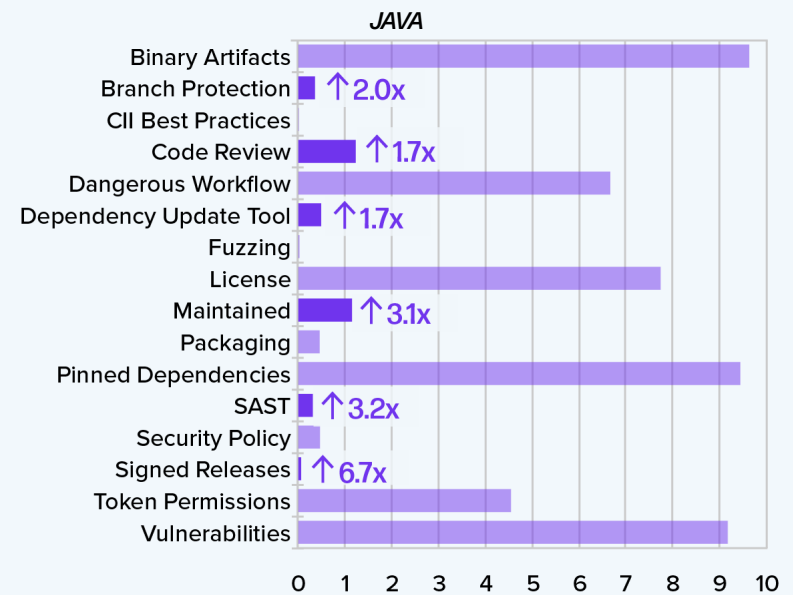
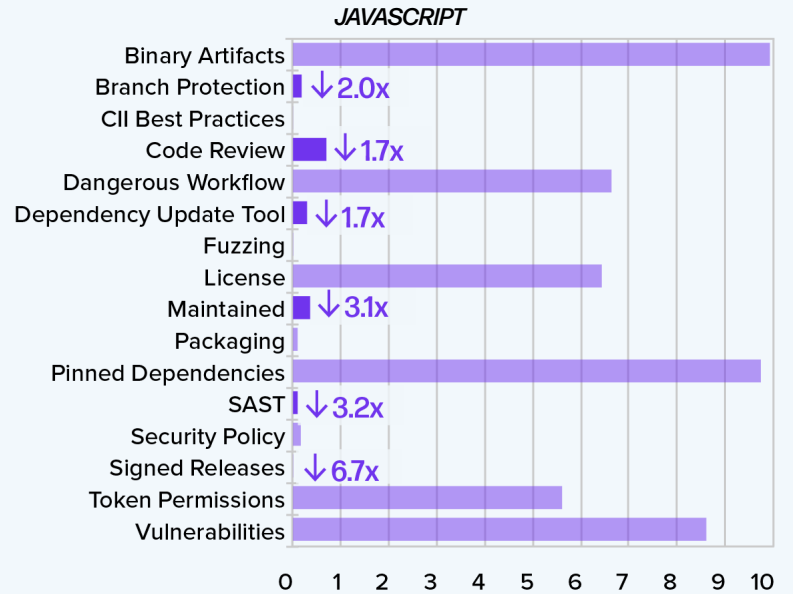
**FIGURE 2.2**  
SCORECARD CHECK AVERAGES, ALL PROJECTS



**FIGURE 2.4**  
SCORECARD CHECK AVERAGES, MAINTAINED



**FIGURE 2.3**  
SCORECARD CHECK AVERAGES, JAVA VS, JAVASCRIPT





vulnerability are comparable, with 15% of Java projects and 16% of JavaScript projects having unpatched vulnerabilities at the time of scan. Note that this is a measure of whether there are vulnerabilities that apply to the development branch of a project. This does not indicate whether the latest release of the project has a vulnerability, nor whether the recorded vulnerabilities will persist into the next release.

## The Importance of Maintenance

The figure above shows how the average scores change when we focus on maintained projects only (projects with a **Maintained** score greater than zero). There are several notable differences. Maintained projects are

- ▶ 5.9x higher on *SAST*
- ▶ 5.4x higher on *Signed Releases*
- ▶ 5.1x higher on *Dependency Update Tool*
- ▶ 3.6x higher on *Code Review*
- ▶ 3.8x higher on *Branch Protection*

Maintained projects also score better on *Vulnerabilities* and have slightly lower rates of vulnerability.

**The fact that 18.6% of projects stopped being maintained in the last year highlights the need to not only choose good dependencies, but monitor those dependencies for changes in their quality.**

## Year-Over-Year Differences

We also looked at the data from one year ago to see how Scorecard practices have changed. Surprisingly, we find that there has not been a steady improvement in scorecard scores. Instead, we see decreases in several key metrics. In particular, the number of maintained projects has decreased overall and also within each ecosystem. Overall, there are 24,104 projects that were maintained one year ago, but no longer qualify as maintained. This represents 18.6% of the maintained projects. Put another way, if you picked a random maintained project in 2022, there would be a 18.6% chance that this project is no longer maintained one year later.

At the same time these ~24k projects slipped out of the “maintained” category, there were 12,806 newly “maintained” projects (projects that are maintained now but did not qualify as maintained in 2022). This resulted in an overall decrease of 11,298 in the number of maintained projects. Of the projects that have become unmaintained over the last year, 1,285 were in the Maven Central (Java) ecosystem and 9,626 were in NPM

(JavaScript). This decrease is not necessarily a bad thing – it could simply reflect consolidation, with maintainers focusing on a smaller set of projects that have “stood the test of time”.

Overall, these changes highlight the importance of tracking the health of dependencies over time. Just as new projects are constantly being created, projects are also constantly reaching end of life status. Identifying these changes is an important part of maintaining a healthy set of application dependencies.

We have also seen an overall decrease in the amount of code review. The number of projects that had any Code Review decreased by over 15%. The decline was slightly less for Java (~11%) than for JavaScript (~17%). Even when just looking at maintained projects, there was a decrease of 8.6% in rates of code review. Comparing trends in scorecard checks over time is complicated by the fact that the OpenSSF Scorecard team does periodically change the checks in ways that can affect scoring. However, we reviewed the changes made to the code review check and, on aggregate, they appear to make the check easier to pass (e.g. by treating a PR

merged by someone other than the contributor as a review). If this data indeed reflects a decline in the amount of code review occurring in otherwise well-maintained projects then this is a worrying trend. We recommend ensuring that any open source libraries critical to business applications are practicing code review on every pull request to the main branch.

## Conclusion

Overall, these results provide guidance for open source users when choosing dependencies and important areas of focus for open source advocates and maintainers. The value of choosing

well-maintained projects stands out, as these projects tend to be more diligent about a number of important software security best practices. The fact that 18.6% of projects stopped being maintained in the last year highlights the need to not only choose good dependencies,

**Overall, there are 24,104 projects that were maintained one year ago, but no longer qualify as maintained. This represents 18.6% of the maintained projects.**

but monitor those dependencies for changes in their quality. The overall low rates of code review, even when considering just maintained projects, provide a clear area for improvement in open source development practices – especially given the importance of code review in predicting security status (established in last year’s report).

Based on our findings, enterprises looking to minimize their open source vulnerability risk should choose well-maintained projects that perform code review and monitor them to ensure they have not reached end-of-life.

CHAPTER 3

# Modernizing Open Source Dependency Management



In the last three State of the Software Supply Chain reports, we've delved into the intricate aspects of managing open source dependencies. We firmly believe that understanding the intricacies of both macro and micro decisions when selecting open source components fosters robust, streamlined, and secure software supply chains.

Each year, our cumulative analyses bring us closer to uncovering the key to optimal dependency management and ensuring safer, more efficient software supply chains. However, the incremental improvements we've observed in open source dependency management over the past three years, while noteworthy, fall short of our desired progress.

## Open source consumers are not paying attention

In last year's report, we posed this critical question: Who is primarily responsible for creating the greater share of risk in open source ecosystems,

open source maintainers, or consumers? The answer was overwhelmingly open source consumers. We wanted to see if this had changed, so re-ran some of the numbers for 2023. We saw no change in the consumption of vulnerable downloads within the Maven Central ecosystem.

This means, we once again saw that 96% of all known-vulnerable downloads were avoidable.

### The size of the "consumer" problem is concerning

This year, we again analyzed how the world consumes open source from Maven Central across 400+ billion downloads over the year. We compared consumers downloading vulnerable dependencies without a fixed version to vulnerable dependencies where a fixed version was available but not chosen.

### How Common Are Vulnerable Downloads?

From the average 37.8 billion monthly downloads from Maven Central, 3.97 billion vulnerable components were consumed.

**96%**

of all known-vulnerable downloads were avoidable

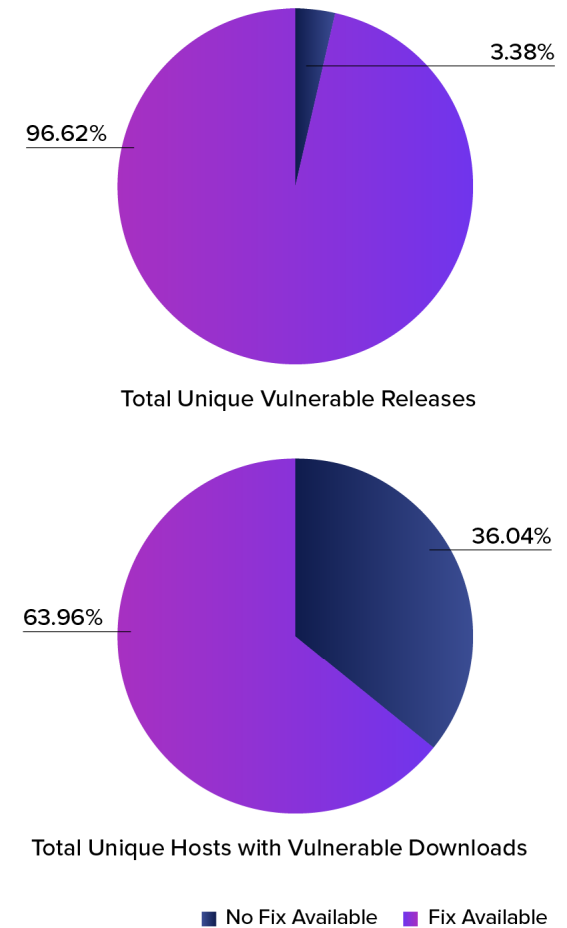
**3.97 billion**

vulnerable components were consumed, from the average 37.8 billion monthly downloads from Maven Central

**2.1 billion**

vulnerable Java Components are consumed each month

**FIGURE 3.1**  
CONSUMER VS MAINTAINER OF VULNERABLE DOWNLOADS



## How Common Are Fixed Vulnerabilities?

From there, we found that 1.8 billion or 3.8% of the vulnerable dependencies were due to a vulnerable version with no available fix – meaning 96.2% of known-vulnerable downloads had a non-vulnerable option available. That means 2.1 billion avoidable vulnerable dependencies are being consumed each month.

Consumers opted for these no-fix-available projects due to lack of alternatives. Interestingly, even though there are only a few vulnerable versions without alternative fixes, a significant portion of vulnerable versions with available solutions are still being downloaded.

This sentiment is echoed in the [Consumption Manifesto](#) recently published by the Open Source Security Foundation (OpenSSF). The manifesto emphasizes the need for organizations to shoulder the responsibility for the open-source software they employ, their methods of consumption, and their strategies for handling the associated risks.

## Why is making good choices so hard?

Developers face a multitude of challenges and responsibilities in their work, often leading to overwhelming and inefficient experiences when dealing with dependency management. This struggle has been humorously dubbed “[Dependency Hell](#)” in the development community. This bears repeating.

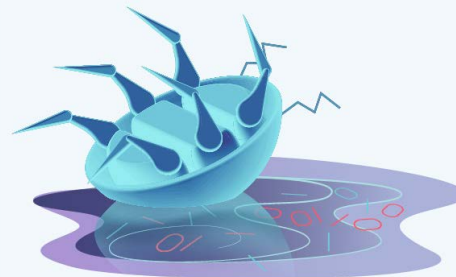
Organizations expect developers to make informed choices regarding open source components for their software projects. However, a significant portion (approximately 85%) of projects hosted on Maven Central are considered inactive, with fewer than 500 monthly downloads. This proliferation of inactive projects further complicates the already challenging task of selecting the most suitable components for a given project.

Consider for a moment the average Java application now boasts a whopping 148 dependencies, with around 10 releases occurring annually. Developers not only contend with the initial selection and management of around 150 dependencies but are also tasked with tracking an average of 1,500 dependency changes

per year per application, possessing security and legal expertise to choose the safest versions, understand ecosystem nuances, and sift through thousands of projects to pick the best one. Now, imagine the scale of these decisions for enterprises with tens of thousands of developers and thousands of applications.

The additional workload described above is merely a fraction of what software developers already face in their day-to-day responsibilities. The immense pressure to meet industry demands for efficiency and speed often leads to inefficiencies and risks within enterprises. When developers are inundated with an overwhelming array of choices and limited resources, it not only hampers productivity but also jeopardizes the organization’s success.



### Developers not only contend with the initial selection and management of around 150 dependencies but are also tasked with:




- ▶ tracking an average of **1,500 dependency changes** per year per application,
- ▶ possessing security and legal expertise to choose the safest versions,
- ▶ maintaining ecosystem security insights about new risks affecting projects,
- ▶ sifting through **thousands of projects** to pick the best one.



With all of this in mind, we also recognize that developers are making their best efforts given the circumstances. As we delve deeper into this issue, we encounter several key challenges:

- ▶  **Popularity** — When deciding which dependencies to use in a development project, popularity is often used as a proxy for quality (i.e., “everyone else is using it, so it must be safe, secure, and reliable”). Theoretically, this makes sense as more popular projects should be getting fixed faster. But they aren’t. As revealed in our 2019 State of the Software Supply Chain report, the popularity of a dependency does not correlate with a faster median update time. Developers may feel safe in selecting more popular projects, but just because a dependency is popular, doesn’t necessarily mean it’s “better.”
- ▶  **Clarity** — Oftentimes, developers aren’t manually selecting individual versions when building software supply chains and those dependencies are already part of a project that’s being used or built upon. As cited in the 2020 State of the Software Supply Chain report, 80-90% of modern applications consist of open source software. If an SBOM and proper DevSecOps practices are not implemented, developers and software engineering teams may have no way of knowing that those vulnerable components are being used, pulled, or built upon.

- ▶  **Automation** — Though there are plenty of open source automation tools, very few have security capabilities built in. Similar to the Clarity issue above, this automation may mask potential vulnerable dependencies, enabling developers to unknowingly build upon projects with known vulnerabilities.

- ▶  **Inactive Releases** — There are almost 500,000 projects within Maven Central, but only ~74,000 of those projects are actively used. That means 85% of projects are sitting in this repository and taking up space, potentially overwhelming developers with available options.

## Modern Dependency Management Practices

While not surprising, the problem of dependency management didn’t magically go away in the past year. So, as we continue on the quest to understand how we could potentially fix it, we honed in on the aspects that reflect modern dependency management — or a potential answer to solving the 96% problem.

Our emphasis lies on understanding the nuances of behavior within this domain and how it impacts the way we work. This includes:

- ▶ Defining the optimal component. What actually makes a “good open source component”
- ▶ Dissecting the optimal time to upgrade an open source dependency

- ▶ Reflecting on current upgrade behavior
- ▶ Analyzing global patterns of download behavior

## What makes an optimal open source component?

In the realm of dependency management, understanding the inherent risks and benefits of individual software component versions is paramount. To address this challenge, we conducted a comprehensive assessment of software artifacts and have developed a robust scoring system that evaluates components across five key dimensions.

These dimensions are as follows:

- 1. Security:** Uses a sophisticated method to gauge the “total risk” associated with a particular component version, considering all its vulnerabilities and common weaknesses. This scoring technique allows for a thorough comparison of different versions of the same component or across components by considering vulnerability counts, severities, and types.
- 2. License:** Based on License Threat Groups: Categorizes licenses into severity groups, allowing for informed decisions that align with your application’s licensing requirements.

3. **Age:** Evaluates a component's age in relation to the latest version, emphasizing the benefits of staying current within the software ecosystem.
4. **Popularity:** Analyzes download counts across repositories and other sources, enabling an understanding of component usage trends and popularity.
5. **Release Stability:** Quantifies the stability of a version by assessing its version label, factoring in development stages, such as pre-release, beta, etc, as well as double-publishing occurrences.

Our scoring system is not arbitrary. Built on [years of thinking about dependency management](#), it seeks to create a holistic representation of a component's risk and value, facilitating data-driven decisions in software supply chain management. By examining these dimensions, we can assess dependency management, prioritized upgrades, and optimized software development practices.

## Understanding component upgrade urgency

In the fast-paced realm of software development, staying current with component versions is paramount to ensure security, performance, and reliability. However, merely having access to better versions of components does not always guarantee that an upgrade is immediately necessary. Compare this to the way we buy and “upgrade” cars. You could get the newest model and upgrade every year. You have access to the new models (versions), and each year, there are improvements made that make it better. But, would that be effective? Usually, no. It would be even more time intensive, some of those new features may have bugs that need to be worked out — and you may simply not need those features. However, for example, there are definitely advantages to upgrading when there are new safety features, you know bugs have been worked out, and of course, if there is a recall.

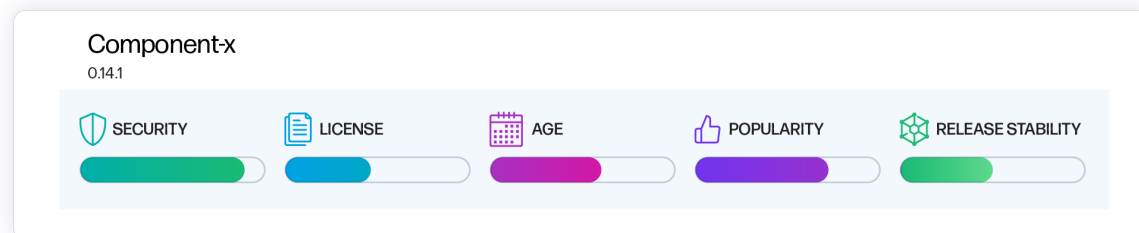
At first, the comparison seems unwarranted. Buying a car is considerably more time-intensive than upgrading a component. However, consider everything that needs to go into updating

**Merely having access to better versions of components does not always guarantee that an upgrade is immediately necessary.**

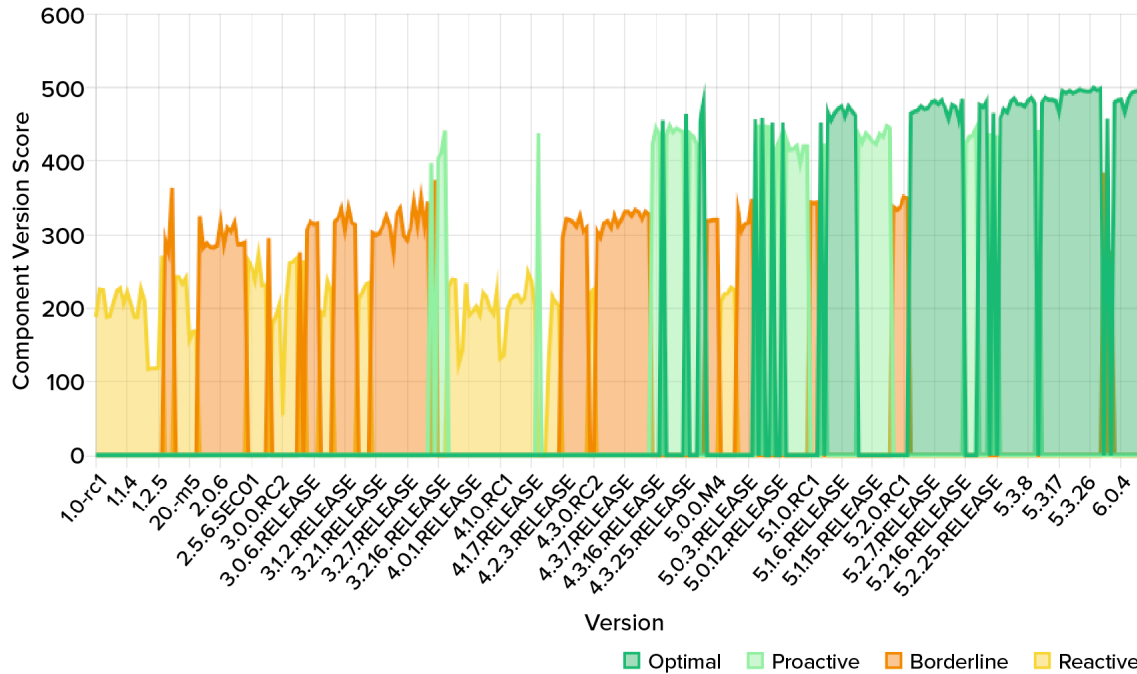
a dependency. For example, if your organization follows best practices, you will create a ticket for the work. Then, you may need discussion and approval to bring that work into the current iteration. Once the work is complete, a pull request and review from other members of the team — which could be immediate, but often is not. And this list continues through every step of your development process. In the best case, there are no issues, but it has still touched multiple team members such that a five-minute change is an hour or more. For enterprises at scale, a common component with frequent updates could introduce significant interaction and waste associated with developer time.

To better understand this cost, we looked into the concept of Upgrade Urgency to shed light on the need for a data-driven solution. Our approach is based on a component scoring algorithm, which not only scores a component but also categorizes its versions into distinct zones. These zones range from the best/optimal versions to reactive (worst) versions.

Proactively managing dependencies is of utmost importance because, as the saying goes,



**FIGURE 3.2**  
**UPGRADE URGENCY**



**FIGURE 3.3**  
**VERSION DOWNLOAD BY UPGRADE URGENCY FROM MAVEN CENTRAL**

Upgrade Urgency	Downloads by Urgency	Percentage Urgency
0- Optimal (best) Version	18,055,476,664	80.6%
1- Proactive	602,398,633	2.7%
2- Borderline	2,604,054,004	11.6%
3- Reactive	1,128,938,205	5%

software ages like milk, not wine. Remaining in a reactive state is not only suboptimal from a security perspective, but it also puts development teams at an immediate disadvantage and penalizes them when issues arise. Analyzing the observed patterns reveals that teams that proactively make upgrade decisions are in a significantly better position to make the most informed choices.

The upgrade urgency algorithm assesses how urgently a component should be upgraded by comparing its version score to the best available. It categorizes urgency as follows:

- ▶ **0 – Optimal version(s):** Component scores in the top 10% of the best version’s score
- ▶ **1 – Proactive:** Scores between the top 10% and 1 standard deviation below the max
- ▶ **2 – Borderline proactive:** Scores 1 standard deviation below the max but less than 2 standard deviations
- ▶ **3 – Reactive:** Scores 2 or more standard deviations below the best version for that component.

Unlike component scoring, upgrade urgency considers a version’s distance from the best available, whereas being on the best version eliminates the need for urgent upgrading.

We will apply these algorithms to explore download patterns across different urgency

zones, providing valuable insights into how software development industry professionals are making their component selections and upgrade decisions.

## Downloads by upgrade urgency

As you know, Maven Central is the de facto repository for Java-based open source libraries. It acts as a centralized hub, enabling developers to effortlessly discover, access, and integrate dependencies into their projects. Its widespread adoption and comprehensive collection of artifacts have made it an essential resource for Java developers worldwide.

In our analysis, we used Maven Central to examine the download patterns of component versions. We looked at a typical month of data and determined the upgrade urgency of each download. In a perfect world, developers would only be downloading optimal versions of their dependencies in order to minimize risk and future upgrade effort.

What we found is that a significant number of downloads (80%) are of the best available versions of the components being downloaded. At first glance this seems commendable, but it's worth noting that being on the best version doesn't necessarily mean it's a high-quality component; you might be on the best version of a really poor component. However, let's go back to our car analogy from above. If your

**During the time of download, each component, on average, had 10 superior versions available.**

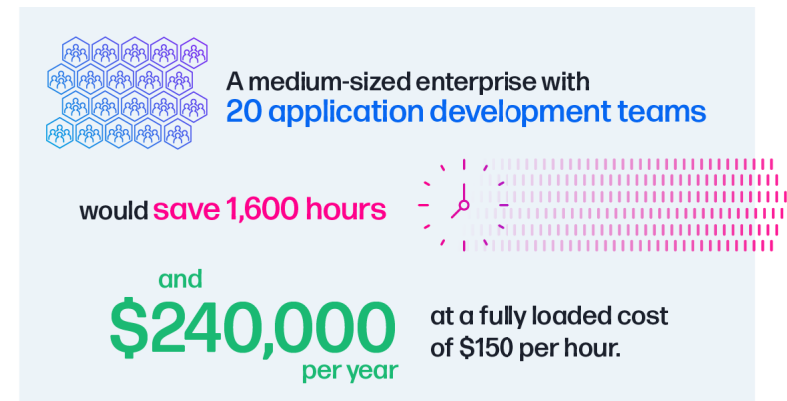
brakes only worked 80% of the time, that would be unacceptable. That would mean, a 1 in 5 opportunity your brakes stopped working while driving around town. To further underscore this point, think about it in terms of food preparation. Maintaining strict hygiene standards for 80% of food products in a restaurant also means the majority of ingredients are safe to consume, but the remaining 20% could lead to food poisoning or other health risks.

FIGURE 3.4  
TIME SAVED WITH OPTIMAL UPGRADE DECISIONS

By making optimal upgrade decisions, organizations could save



per development team per year.



With 2.6 billion downloads being classified as “borderline” and a further 1.1 billion falling into the worst, reactive category, there is some cause for concern. After all, we live in a world where it only takes just one vulnerability to “[set the internet on fire](#)”.

Moreover, a comprehensive analysis of the component scores for all downloaded versions reveals a striking pattern: during the time of download, each component, on average, had 10 superior versions available. This finding emphasizes that many developers not only miss out on the best versions but even when selecting versions that are considered “good enough,” relatively speaking, there are better alternatives they could have chosen. This reveals that the selection process is likely driven by heuristics such as just picking the latest version. However, we see that even that strategy has its pitfalls.

## A 25% reduction in false positives, over the course of a year, would give you a total resolution savings of 2X.

### A ton of wasted time

In prior years, we reported on how efficient timing for upgrades can yield significant cost savings. Opting for safer versions and prolonging their use can significantly reduce upgrade expenses. In a medium-sized enterprise with **20 dev teams, the potential gain equates to two extra development weeks per application each year.**

This year, we're continuing to dissect this cost savings coefficient further. In addition to the above, which takes into account optimal upgrade decisions, we looked at what the time savings would be when you consider the quality and accuracy of your security data, in making these decisions.

In answering this question, we found that when teams use better security data that reduce false positive findings by 25%, in combination with making optimal upgrade decisions, each team saved a total 1.5 months of time, per application, per year. This equates to a 2X boost in time saved over just the optimal upgrade process we described in our 8th annual report last year. In other words, if you improve the security data you use and reduce false positives, in tandem with upgrade efficiency, you double your gain.

In past years, we've looked at optimal as being a single component or a single version. There was a right and wrong version. This year, we've established that optimal is really a range. There is still a "right" or a "wrong", but on a spectrum. To further this, we examined the Maven Central repository to quantify wasted developer time. In this scenario, we define wasted time as the download of an optimal version of a component when another optimal version of that same component has already been downloaded by that user instance. To put this into perspective, let's again go back to our car analogy. Would you buy a car every time a new model year comes out? The optimal buying period might be when the model refreshes say every five years, rather than incremental changes each year. In our analysis, we consistently observed this phenomenon, which is all too frequent and raises concerns. The gains of such frequent upgrades are minimal and from a lead/management perspective prioritization can also be optimized/aligned based on things that are in the optimal range versus just getting the new component.

**We examined two significant players in the online information industry. One organization had 5.4% wasted time, whereas the other enterprise had 10.2% wasted time. That's an 89% increase in unnecessary downloads!**

However, we also noted great variability in this practice. To highlight this drawback we call attention to anonymized data from several companies in two different industries. We analyzed two organizations in the energy industry with a similar number of Maven Central downloads. Out of each company's total downloads, we found there was approximately 1.8 times more wasted downloads from one organization compared to the other.

We also examined two significant players in the online information industry. One organization had 5.4% wasted time, whereas the other enterprise had 10.2% wasted time. That's an 89% increase in unnecessary downloads! It is evident that mature enterprises can reduce this variability and wasted time. By having a centralized platform for storing, managing, and distributing quality software components, this is a very solvable issue.

### The value of an artifact repository manager

In the scenario above, we know much of this wasted time could be solved by simply using a good repository manager, like Sonatype Nexus Repository Manager. A centralized, binary



repository allows you to proxy, collect, and manage your dependencies so that you are not constantly juggling all of these various versions. Or, in this case, wasting time — downloading optimal versions, when another optimal version is already in your instance. It becomes the single source of truth for an organization’s software components and applications.

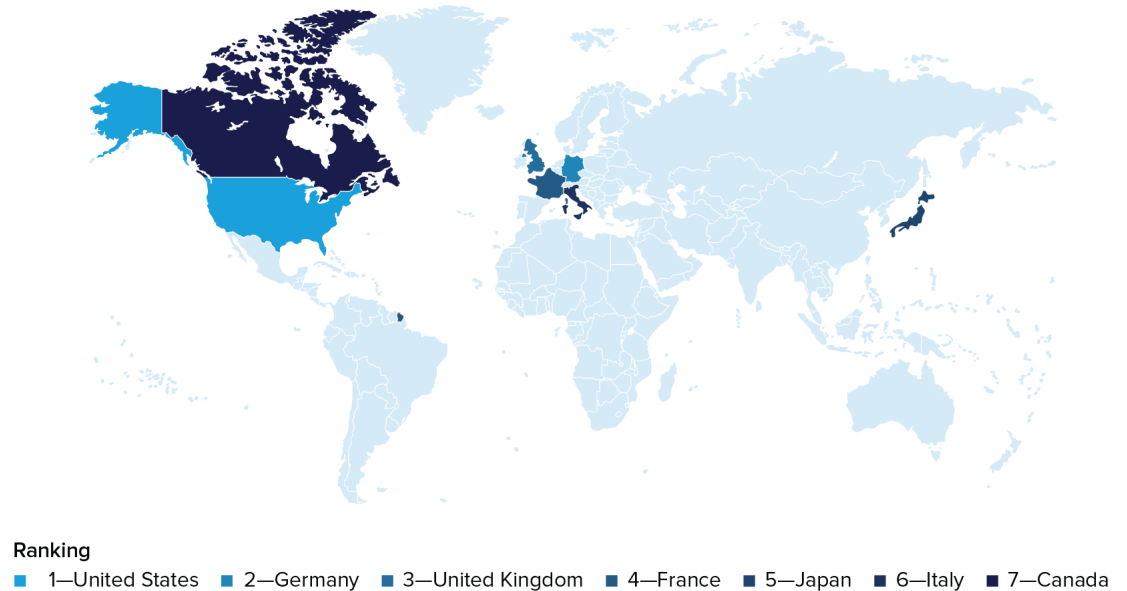
## Global patterns of open source consumption behavior

In today’s interconnected global landscape, various regional groupings and trade agreements play a pivotal role in shaping international relations, trade dynamics, and economic cooperation. We explored whether these groupings also inform software development in the form of component consumption patterns.

With the component score we can consider the quality of the component, with upgrade urgency we can assess how good the downloaded version is. Bringing it all together, we examined global download patterns. We gave equal weight to component scoring and upgrade urgency, and a little additional weight to the sheer quantity of downloads. Because picking the right components and staying on the best versions is surely harder at scale. Below are some observed global trends.

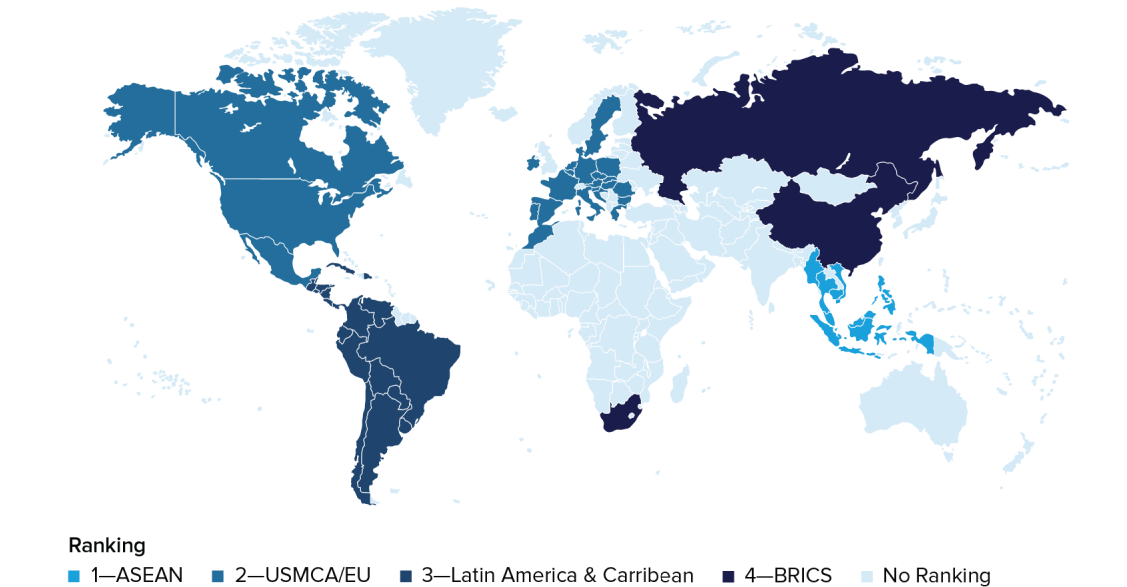
### GLOBAL PATTERNS

#### OPEN SOURCE CONSUMPTION RANKINGS AMONG G7 NATIONS



### GLOBAL PATTERNS

#### OPEN SOURCE CONSUMPTION RANKINGS BY REGION



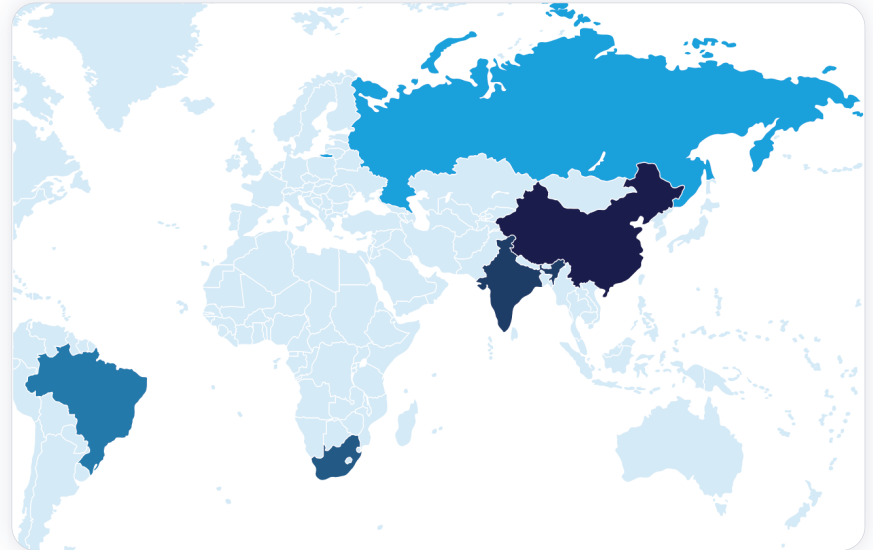
GLOBAL PATTERNS

OPEN SOURCE CONSUMPTION RANKINGS IN ASEAN NATIONS



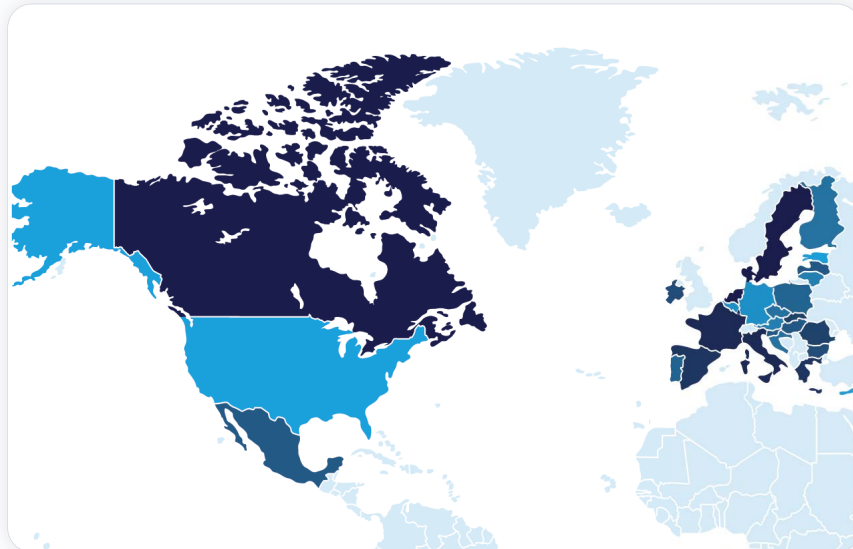
GLOBAL PATTERNS

OPEN SOURCE CONSUMPTION RANKINGS IN BRICS NATIONS



GLOBAL PATTERNS

OPEN SOURCE CONSUMPTION RANKINGS IN USMCA & THE EU



GLOBAL PATTERNS

OPEN SOURCE CONSUMPTION RANKINGS IN SOUTH AMERICAN & CARRIBBEAN NATIONS



Ranking Key

Highest cumulative score  Lowest cumulative score

Within the Group of Seven (G7) nations, renowned for hosting some of the globe's most advanced economies, the United States emerged as the frontrunner, boasting the highest cumulative score. In contrast, its northern counterpart, Canada, found itself positioned at the bottom of the list. Germany had the highest proportion of optimal versions downloaded. Italy had the highest proportion of reactive downloads among the G7 countries, at over 8%, which also made it one of the lowest ranked countries across the globe on that metric, only beating out China, Hong Kong, Macedonia, and San Marino. Some of this isn't surprising. While we talk later in this report about the sheer volume of regulations taking place around securing software supply chains, the United States has by far taken this issue the most seriously, at least in terms of focused regulation and national policy conversations.

## ASEAN

ASEAN (Association of Southeast Asian Nations) is a regional intergovernmental organization comprising ten member states in Southeast Asia. On average, the ASEAN grouping was the highest scoring group in regard to consumption behavior. The member countries of ASEAN are Brunei, Cambodia, Indonesia, Laos, Malaysia, Myanmar, the Philippines, Singapore, Thailand, and Vietnam. Among them are technological hubs such as Singapore and Malaysia as well as

many nations undergoing a digital transformation with heavy investment into IT infrastructure and digital innovation.

## USMCA & EU

Tied for a close second were the economic powerhouses of North America through the North American Free Trade Agreement (NAFTA), which has evolved into the United States-Mexico-Canada Agreement (USMCA), and the European Union (EU), the collaborative bloc of European nations. The USMCA trio had the US rank on top, followed by Mexico, and finally Canada. Estonia ranked among the very top of the EU. Unsurprisingly so, given its innovative approaches to e-government services, world recognition for a digital society and robust digital infrastructure. Germany is a prominent software developer hot spot in the EU, and closely followed Estonia in its component consumption score. The country's robust economy, strong engineering tradition, and technological infrastructure make it an attractive destination for software professionals. Cities like Berlin, Munich, and Hamburg host thriving tech ecosystems, offering a blend of innovative startups, established tech companies, and research institutions. In contrast, while the difference was not vast, it was unexpected to observe Sweden and the Netherlands placed near the lower end of EU member states, given their well-regarded technology sectors and developer practices.

## Latin America and the Caribbean

Next in the rankings were the Latin America and the Caribbean nations. Among the countries with the highest scoring downloads we saw Nicaragua and Bolivia. On the other hand, Panama, and Guatemala were ranked towards the end of the list. Despite the sweeping digitization trends happening across this economic grouping, some of the analysis was nevertheless skewed by the significantly lower download counts across many of the member countries.

## BRICS nations

Finally we examined the BRICS nations – Brazil, Russia, India, China, and South Africa. This examination unveiled that these regions scored comparatively lower in the quality of OSS consumption compared to the other geographical groupings. In conclusion, while each region possesses unique attributes that shape their component consumption habits, it's crucial to recognize that the rankings presented are a mere snapshot of their software landscape. They are not intended to comprehensively encapsulate the full scope of any individual country's software development maturity. As these regions navigate their own technological journeys, they are poised to further engage with open source initiatives, contributing to the global software community's growth and innovation.

CHAPTER 4

# Software Supply Chain Maturity





# Software Supply Chain Maturity – Peer Insights

In this year's report, we dive into insights gleaned from 621 enterprise engineering professionals, providing invaluable perspectives on software supply chain management and organizational development practices. Our analysis draws from a comprehensive survey that explored the multifaceted landscape of software supply chain management, including

the utilization of open source software (OSS) components, dependency management, governance, approval processes, and tooling. The survey also covered questions related to development practices, demographics, and job satisfaction levels, culminating in a more holistic view of software supply chain maturity at an enterprise and industry level.

## Expanding Horizons

Over the last decade, the software supply chain landscape has experienced rapid and

significant changes. In the past year alone, the emergence of generative AI capabilities and introduction of global regulatory initiatives spurred modernization efforts in software development across industries.

To further understand this ever-evolving landscape, we marry [last year's findings](#) with what changed today. We explore interesting year-over-year trends, examine the growing demand for Software Bill of Materials (SBOMs), and identify key practices aimed at not wasting developer time. Furthermore, we take a deeper look at companies affected by open source risk, particularly those who have experienced security breaches. Through this investigation, we aim to shed light on how these organizations may differ from their peers and provide insights into bolstering security measures within the software supply chain.

## Continuing Objectives and Methodologies

Consistent with our approach in previous editions, this chapter maintains two core objectives:

- ▶ Provide a benchmark and maturity model that enables organizations to assess their practices in comparison to their industry peers.
- ▶ Determine if reported software supply chain practices align with desirable outcomes.

To evaluate responses to the comprehensive survey, we will examine them within the

## EIGHT THEMES OF SOFTWARE SUPPLY CHAIN MANAGEMENT PRACTICES



### APPLICATION INVENTORY

Do you know all the applications within your organization's development/production pipeline, including their stakeholders/owners, architectural details, and SBOMs for the included OSS components?



### SUPPLIER HYGIENE

Can you identify your suppliers and ensure that your OSS components come from trusted, quality sources?



### BUILD & RELEASE

Do you understand how your software components and processes come together to build and release applications into production?



### PROJECT CONSUMPTION

What are your consumption policies governing the selection of OSS components within your projects?



### GIVING BACK

(Also referred to as "contribution") Do you actively and consistently contribute to the OSS community?



### POLICY CONTROL

Do you have risk management policies in place, and do they align with your risk tolerance? Do you employ automated policy enforcement?



### DIGITAL TRANSFORMATION

(Also known as an "execution plan") What strategies, resources, and training initiatives are in place to support adoption of new processes and tools as part of your digital transformation goals?



### REMEDiation

How do you implement fixes to address identified risks associated with OSS components?

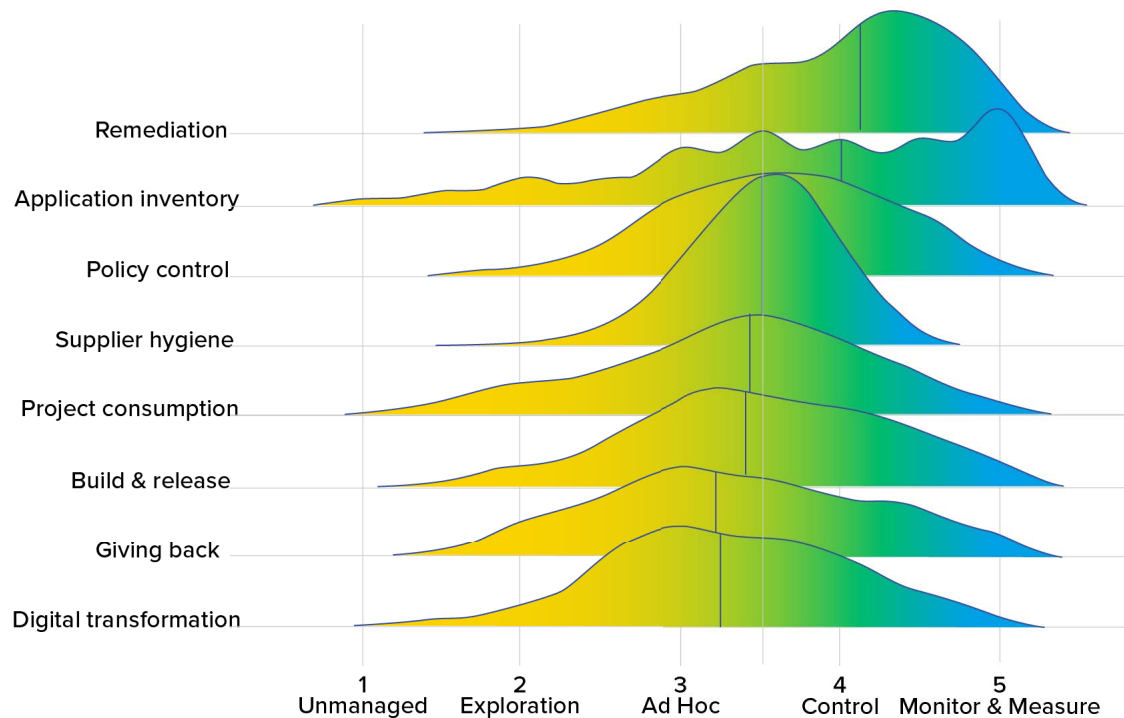


framework of eight key themes of mature software supply chain management practices.

## How mature are today's software supply chains?

We've distilled and presented collective insights from the 621 surveyed individuals in **Figure 4.2**. To develop this summary, we averaged each individual's responses on questions that align with the eight key themes. In each theme we

**FIGURE 4.1**  
SOFTWARE SUPPLY CHAIN MATURITY SCORE BY THEME

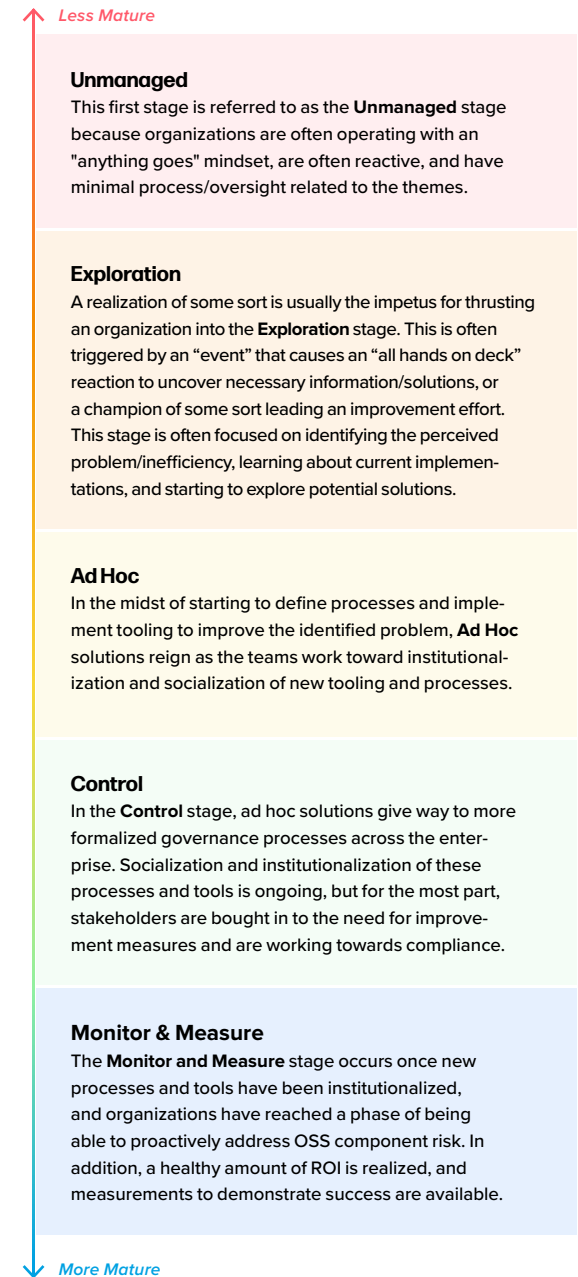


scored the responses from 1 to 5, corresponding to the five stages of software supply chain maturity. From Unmanaged (least mature) to Monitor & Measure (most mature), as noted in **Figure 4.1**

### Perception is disconnected from reality – again

In last year's survey, the [data showed](#) a notable pattern: respondents tended to self-report a higher level of software supply chain maturity than their actual stage of advancement, leaving many with an inflated sense of security. This trend persists into this year's findings

**FIGURE 4.2**  
FIVE STAGES OF SOFTWARE SUPPLY CHAIN MANAGEMENT MATURITY



Overall, this year's survey reveals that respondents generally demonstrated lower levels of maturity in the Digital Transformation theme and higher levels of maturity in Remediation. This consistent year-over-year trend underscores a fascinating disparity between how survey respondents self-assess their progress in actively addressing vulnerabilities (indicative of higher maturity) versus their capacity to garner essential buy-in, sponsorship, training, and operational processes for effective remediation (reflecting lower maturity).

From this, a striking statistic emerges: 67% of respondents feel confident that their applications do not rely on known vulnerable libraries. However, nearly 10% of respondents reported that their organizations had security breaches due to open source vulnerabilities in the last 12 months, while 20% were unsure if their organization had been breached. Pair this with the objective information we saw in Chapter 1 – that 1 in 8 open source downloads have a known vulnerability – and it's clear that no matter how we look at it, there continues to be a software supply chain management problem.

In addition, aside from remediation and application inventory, most respondents received ratings below the "4 – Control" level of maturity. The "Control" level represents a crucial milestone, where organizations move from a phase of uncertainty to a minimal level of maturity that enables the production of high-quality outcomes. Notably, the three maturity levels preceding "Control" (Unmanaged, Exploration, Ad Hoc) are considered suboptimal and

predominantly received the bulk of survey responses.

To get a sense of how your organization compares, take this [short quiz](#) on software supply chain maturity.

## How have software supply chain management trends changed?

The software supply chain plays a vital role in modern development processes. Understanding its dynamics is crucial for mitigating risk and ensuring operational efficiency. This section unveils key insights from our survey of engineering professionals, offering a comparative analysis of this year's responses to those of the previous year. By analyzing these year-over-year trends, we gain a clear view of significant changes and evolutions in the landscape of the software supply chain.

### Finding #1: Increased focus on open source risk

In a landscape marked by the growing volume of open source, efficiency in software development is the key to not wasting developer time. This year, our survey results highlighted the interplay between effective tools, developer satisfaction, and optimized work processes. Respondents reported an increase in the use

# 67%

of respondents feel confident that their applications are not using known vulnerable libraries

# 10%

of respondents reported that their organization had a security breach due to a vulnerability in the last 12 months

# 20%

of respondents were unsure if their organization had been breached in the last 12 months

### HOW MATURE IS YOUR SOFTWARE SUPPLY CHAIN?

[Take the quiz](#)

of integrated tooling (+9.8%), growing awareness of and focus on open source risk (+9.3%), improved dependency upgrade decisions (+9.6%), and greater comfortability in software supply chain management (+7.7%) – all of which point to a maturing market and an increasing industry-wide understanding of the downstream threats vulnerable open source poses

An overarching theme is clear: the combination of robust tools and happy developers plays a significant role in minimizing workloads.

### ADOPTION OF INTEGRATED TOOLING

One notable trend observed this year is a decline in the reliance on ad-hoc reports for obtaining risk information about open source libraries. A substantial decrease of 14.5% was recorded among respondents who received reports from external sources. However, a corresponding 9.8% increase was found in the integration of risk information into continuous integration (CI) and build processes. This shift aligns with the changing preferences, where more professionals now favor obtaining risk information within their existing workflows, rather than relying on separate reports.

### GROWING FOCUS ON OPEN SOURCE RISK

Another interesting finding: open source is very much a focus for engineering teams, and is a growing one at that. There was a significant increase in respondents disagreeing with the statement that “open source risk is not currently

a focus” for their organization (9.3%), and likewise, a decrease in those agreeing (-9.6%).

### IMPROVED UPGRADE DECISIONS

Respondents reported a decreasing number of times that dependency upgrades “always/often/frequently” broke the functionality of their application, and an increase in the number of times dependency upgrades rarely or never broke their application’s functionality. Not only does this point to respondents getting better at handling dependency upgrades and avoiding breaking changes, it also shows a deeper understanding of open source hygiene.

### GREATER COMFORT EQUALS GREATER EFFICIENCY

Overall, respondents feel more comfortable and familiar with topics surrounding software supply chain management this year. There was a 7.7% increase in those that kept up with software supply chain trends.

However, in line with the chapter’s earlier findings, a broader lack of overall maturity and efficiency in software supply chain management persists. While the focus on open source risk has intensified, this year’s findings revealed a decline in the use of automated open source processes, a lack of well-defined processes, and a lower number of respondents that follow their organization’s open source approval procedures. In other words, organizations continue to grapple with immature practices that waste time and resources – weighing teams down and slowing development.

## Changes in Supply Chain Management Trends

### ADOPTION OF INTEGRATED TOOLING

↓14.5%

decrease recorded among respondents who received reports from external sources.

↑9.8%

increase was found in the integration of risk information into continuous integration (CI) and build processes

### GROWING FOCUS ON OPEN SOURCE RISK

↑9.3%

respondents disagreeing with the statement that “open source risk is not currently a focus” for their organization

↓9.6%

respondents agreeing with the statement that “open source risk is not currently a focus” for their organization

### IMPROVED UPGRADE DECISIONS

↓9.6%

Respondents reported a decreasing number of times that dependency upgrades “always/often/frequently” broke the functionality of their application

↑4.6%

increase in the number of times dependency upgrades rarely or never broke their application’s functionality

### GREATER COMFORT EQUALS GREATER EFFICIENCY

↑7.7%

increase in those that kept up with software supply chain trends.

## Organizations continue to grapple with immature practices that waste time and resources - weighing teams down and slowing development.

A year-over-year decrease in agreement (9.5%) and increase in disagreement (3.4%) with the statement “We have executive sponsorship” underscores that achieving software supply chain maturity necessitates not only effective tools but also strong organizational support

In summary, although there is an increased focus on and awareness of open source risk, organizations have ample ground to cover in implementing best practices for mitigating these risks effectively.

75%

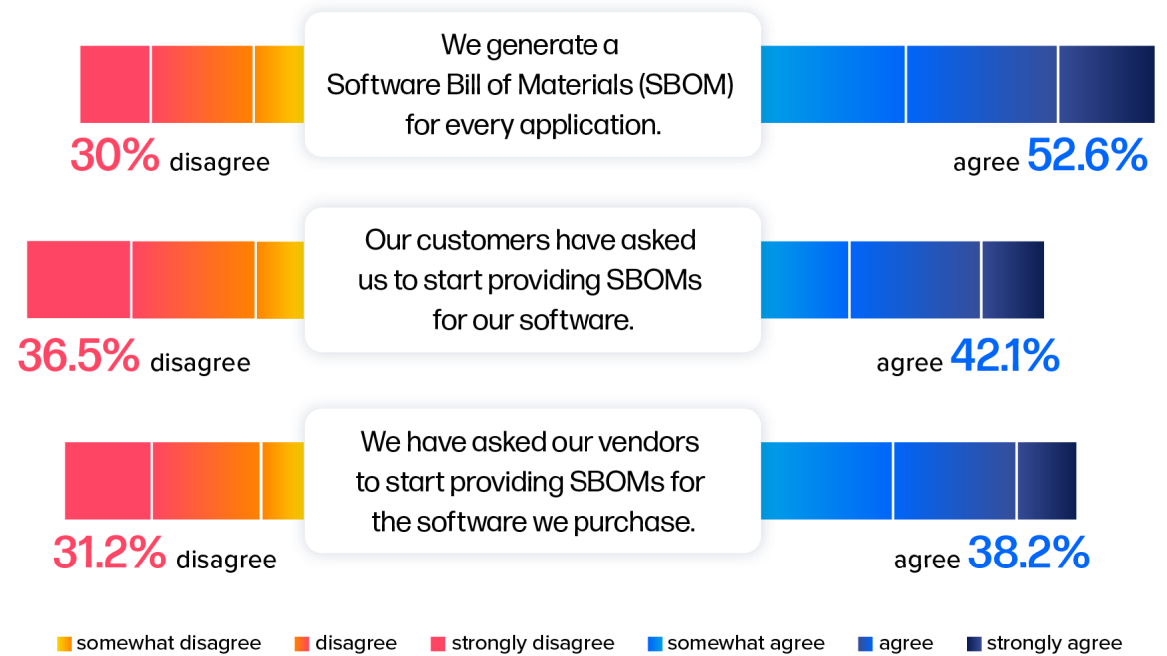
of leaders

25%

of engineering professionals

reported generating SBOMs for their applications

FIGURE 4.3  
SBOM SURVEY RESPONSES



### Finding #2: Rise in demand for Software Bills of Materials

Thanks to mandates outlined in President Biden’s [Executive Order on Improving the Nation’s Cybersecurity](#) and the growing global regulatory efforts, one open source best practice that is experiencing a growth in adoption is the use of an SBOM.

An SBOM serves as a comprehensive inventory, outlining the various components and dependencies of a software application. It provides a detailed list of the software’s building blocks, including open source and third-party libraries, along with their respective versions.

This document serves as a critical resource for understanding and managing the software supply chain. By enabling developers and organizations to accurately track and assess the security and licensing aspects of software components, SBOMs play a vital role in software development, cybersecurity, and regulatory compliance. SBOMs provide transparency and visibility into software composition, thereby facilitating efficient risk management, enhancing vulnerability response, and supporting effective software maintenance. Ultimately, while the creation of SBOMs themselves doesn’t equate to security, the adoption of SBOMs contributes to building secure, reliable, and trustworthy software

systems, benefitting both software producers and consumers.

This year, we took a deeper look at the usage and demand for SBOMs. Findings revealed that nearly 53% of the surveyed engineering professionals reported they are generating an SBOM for every application (**Figure 4.3**). Notably, respondents who adopted this practice were less likely to report security breaches.

However, when we contrast this with [survey results](#) from cybersecurity leaders at large enterprises (over £50 million/\$50 million annual revenue), we observed that 75% of these leaders reported generating SBOMs for their applications. Compared to the 53% reported by engineering professionals in our general survey, this discrepancy underscores the heightened demand for SBOMs at larger enterprises.

Although, this finding also might just track with [last year's data](#) that revealed how managers often report higher stages of maturity compared to other roles. Compared to respondents working in information security, IT managers were 1.8 times more likely to strongly agree with the statement “We know the Software Bill of Materials (SBOM) for every application.”

In addition, nearly half of the respondents have started asking vendors to supply SBOMs for purchased software, underscoring a higher demand for increased transparency in software supply chains

### Finding #3: The critical connection of hygiene and speed in risk mitigation

This year, we investigated how organizations have been impacted by open source risk,

**Our survey revealed that only 22% of organizations become aware of new open source vulnerabilities within a day of disclosure**

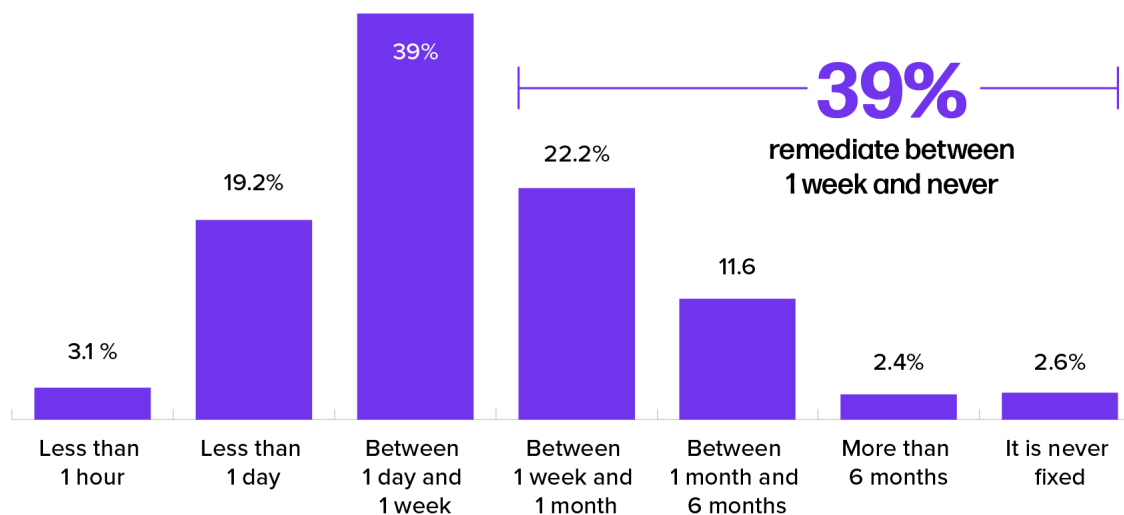
particularly those that have experienced security breaches.

Our analysis revealed a connection between security breaches and suboptimal management practices such as:

- ▶ deploying every change directly to production;
- ▶ breaking the functionality of their applications with dependency upgrades; and
- ▶ adhering to an entirely waterfall development practice

Additionally, organizations that experienced breaches exhibited delayed awareness and mitigation of vulnerability risks. Breached organizations were 2.9x more likely to take over six months to become aware of open source vulnerabilities after their discovery, and 1.7x more likely to report never fixing a vulnerability

**FIGURE 4.4**  
**TIME TO REMEDIATE KNOWN VULNERABILITIES AFTER DETECTION**





## The journey toward software supply chain maturity is not just a path to enhanced security – it's also a path to stop wasting developer time.

Overall, our survey revealed that only 28% of organizations become aware of new open source vulnerabilities within a day of disclosure. Thirty-nine percent (39%) discover them within one to seven days, and 29% take over a week to become aware

**The same survey found that 39% of respondents require over a week to mitigate vulnerabilities (Figure 4.4).** This means that the majority of bad actors have *days* to launch a malicious attack on enterprise targets.

In the context of organizations grappling with security breaches, these findings emphasize the critical connection between maintaining strong hygiene practices and swift risk mitigation. Organizations that have already embarked on the journey to software supply chain maturity are better equipped to effectively manage open source risks. With advanced software supply chain maturity, these organizations apply streamlined processes for identifying, responding to, and resolving vulnerabilities, significantly reducing the vulnerability window and contributing to their overall maturity

For organizations navigating the complexities of open source risk and striving for swift mitigation, the path to software supply chain maturity emerges as a pivotal strategy. This enables them to enhance their resilience, ultimately fortifying their security posture and safeguarding against potential breaches

These findings align with the eight key themes of software supply chain management outlined at the beginning of this section. These themes

encompass practical strategies aimed at not only enhancing security but also at optimizing developer productivity and reducing the waste of developer time.

The journey toward software supply chain maturity is not just a path to enhanced security — it's also a path to stop wasting developer time. By improving software supply chain maturity, organizations create an environment in which developers can focus on innovation, leading to more resilient and secure software.

CHAPTER 5

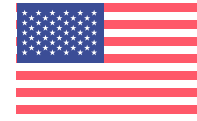
# Establishment and Expansion of Software Supply Chain Regulations and Standards



Worldwide, we continue to see a push for digital transformation not only in the private sector but increasingly through government guidance and regulation. In 2022, the [European Union Agency for Cybersecurity](#) (ENISA) identified the compromise of software supply chains through software dependencies as the foremost emerging threat. Recognizing the profound implications of cyber threats to national security and economic stability, the United States and the European Union (EU) have taken the lead in orchestrating robust regulatory frameworks and providing substantive guidance to fortify defenses against escalating cyber risks. Their comprehensive approach to cybersecurity includes stringent requirements for critical infrastructure sectors, enactment of rigorous data protection laws, and enhancement of international cooperation to combat cybercrime.

However, various regions across the globe are experiencing notable surges in cybersecurity endeavors outside the US and EU. Canada, Japan, Australia, Germany, and others are acknowledging the criticality of securing software supply chains by drafting legislation and building capabilities to thwart cyber threats. While the bulk of documented guidance and regulation is coming from the US and EU, it's clear the pressing need to safeguard the digital realm is a global imperative.

## United States



### National Cybersecurity Strategy (NCS)

The NCS has been hailed as the foundational document for advancing cybersecurity and is shaping legislation globally. Its comprehensive content underscores the urgency of enhancing cybersecurity both in the US and internationally. Addressing a wide range of areas, from infrastructure—addressing incidents like SolarWinds and NotPetya—to SBOMs alignment with EO 14028, the NCS emphasizes the growing importance of security in software design. Moreover, it underscores impending changes in accountability for software manufacturers. Specifically, those unable to prove that security is inherently integrated into their software design will face increased responsibilities and liabilities.

### Securing Open Source Software Act of 2023

In March 2023, Congress introduced legislation “To establish the duties of the Director of the Cybersecurity and Infrastructure Security Agency regarding open source software security, and for other purposes.”

### AI for National Security Act

In March 2023, the House Armed Services Committee introduced legislation “To make certain improvements to the enterprise-wide procurement of cyber data products and services by the Department of Defense and for other purposes.” In subsection (b), it inserts “including by

enhancing the security of the software supply chain of the Department” after “best interests of the Department.”

### FDA Cybersecurity in Medical Devices

On December 29, 2022, the Consolidated Appropriations Act, 2023 (“Omnibus”) became law. Within it, Section 3305 titled “Ensuring Cybersecurity of Medical Devices” updated the Federal Food, Drug, and Cosmetic Act (FD&C Act) by introducing section 524B on device cybersecurity. According to the Omnibus, cybersecurity mandates won’t apply to applications or submissions made to the FDA before March 29, 2023.

### SEC Regulation

Similar to the FDA, the Securities and Exchange Commission (SEC) has proposed a number of rules ([33-11028](#), [33-11038](#), [34-91742](#)) for public companies, the securities market, advisers, funds, and others within SEC’s regulatory scope. This move responds to various findings, including those by the [FBI and Congress](#), as well as

**The National Cybersecurity Strategy underscores the impending changes in accountability and liability for software manufacturers in the US.**

[CISA](#), that state “cybersecurity incidents are underreported.”

Among the new rules are requirements for organizations to demonstrate the development of robust processes and procedures for vulnerability management, detection, mitigation, and remediation. In addition to these requirements, the SEC has proposed specific rules concerning the disclosure of said processes and procedures.

**In July 2023**, the SEC adopted [new rules](#) compelling organizations to disclose significant cybersecurity incidents and provide annual cybersecurity risk management, strategy, and governance information with comparable requirements for foreign private issuers. Registrants must also describe their cybersecurity risk assessment processes and board oversight in an annual report.

SEC Chair Gary Gensler emphasized that any material event, including cybersecurity incidents, should be disclosed to benefit investors and the market. The rules mandate disclosing the specifics of a significant cybersecurity incident—namely, its nature, scope, timing, and impact—within four business days unless it’s deemed a national security risk by the U.S. Attorney General, in which case it can be delayed.

### Secure Software Self-Attestation Common Form

In April 2023, this document was released for comments. It represents a sequence of events

initiated by [EO 14028](#), which directed NIST to provide guidance on secure software development standards. EO 14028 also directed the Office of Management and Budget (OMB) to require federal agencies to collect information from software manufacturers that supply software products to the US government. The OMB responded with [Memo MM-22-18](#), setting the guidelines for requiring federal agencies to collect this information.

As part of these guidelines, software manufacturers must attest to using secure software development best practices highlighted in the memo and developed by NIST. While this is simply a first step, it aligns with the secure-by-design-and-default approach CISA has released. It incorporates secure software development best practices from the NIST Secure Software Development Framework (SSDF) and asks organizations providing software to the US government to self-attest that they have embedded this guidance into their software development process.

### National Cybersecurity Strategy Implementation Plan

**In July 2023**, the Biden-Harris Administration released the National Cybersecurity Strategy

Implementation Plan (NCSIP) to align roles, responsibilities, and resources in cyberspace.

It focuses on two shifts:

- ▶ Greater burden-sharing in cybersecurity by major entities
- ▶ Promotion of long-term cybersecurity investments

The plan outlines over 65 high-impact initiatives, from combating cybercrimes to building a skilled cyber workforce, which is aligned to the five pillars outlined in the National Cybersecurity Strategy:

#### ▶ Pillar One | Defending Critical Infrastructure:

This pillar focuses on updating the National Cyber Incident Response Plan to enhance coordination during cyber incidents and provide clear guidance to external partners, led by the Cybersecurity and Infrastructure Security Agency (CISA).

#### ▶ Pillar Two | Disrupting and Dismantling

**Threat Actors:** This pillar aims to combat ransomware and cybercrime with coordinated

**The OMB’s Self-attestation policy will require software manufacturers doing business with the US Government to attest to NIST guidelines based on CISAs secure-by-design and secure-by-default principles established in the National Cybersecurity Strategy.**



efforts, including disruption operations against ransomware ecosystems and offering resources to high-risk targets, co-chaired by CISA and the FBI.

► **Pillar Three | Shaping Market Forces and Driving Security and Resilience:** This pillar seeks to increase software transparency through a Software Bill of Materials (SBOM) to reduce supply chain risks, led by CISA, and explore a globally-accessible database for end-of-life software.

► **Pillar Four | Investing in a Resilient Future:** Focusing on cybersecurity standards and quantum-resistant cryptography, this pillar, led by the National Institute of Standards and Technology (NIST), aims to enhance U.S. leadership in international cybersecurity standardization and quantum-resistant cryptographic algorithms.

► **Pillar Five | Forging International Partnerships to Pursue Shared Goals:** This pillar involves development of international cyberspace and digital policy strategies, collaboration with partners and allies, and strengthening of knowledge and skills in cyberspace and digital policy, led by the Department of State.

Overall, the strategy emphasizes resilience, equity, and defense in cyberspace, led by 18 government agencies and coordinated by the Office of the National Cyber Director (ONCD). The NCSIP aligns with the private sector, civil society, international partners, and government

entities. It covers key areas like critical infrastructure defense, threat actor disruption, shaping market forces, investing in resilience, and forging international partnerships.

## Cyber Strategy of the Department of Defense (Declassified)

In September 2023, the Department of Defense (DOD) released a declassified summary of its classified 2023 Cyber Strategy, aligning it with national security priorities. It builds on lessons learned from cyber operations and the Russia-Ukraine conflict, emphasizing collaboration with allies and partners.

The strategy aims to maximize cyber capabilities for integrated deterrence, enhance cyber network defense, support non-DOD agencies, and safeguard the defense industrial base. Notably, it commits to bolstering collective cyber resilience among allies and integrating cyber capabilities into traditional warfighting efforts. This marks the fourth iteration of the DOD's cyber strategy, informed by significant experience in cyberspace operations.

## CISA Open Source Software Security Roadmap

In September 2023, at the [OpenSSF Secure Open Source Software Summit](#), CISA announced its [Open Source Software Security Roadmap](#), which outlines strategic goals and objectives for enhancing the security of open source software (OSS).

The roadmap focuses on four key goals to guide CISA in enhancing OSS security, aligning with broader cybersecurity strategies, and fostering collaboration with the OSS community and international partners:

- **Goal 1:** Establish CISA's role in supporting OSS security, partnering with OSS communities, and encouraging collective action.
- **Goal 2:** Drive visibility into OSS usage and risks, understanding the prevalence of OSS, developing a risk prioritization framework, and assessing threats.
- **Goal 3:** Reduce risks to the federal government by evaluating solutions for secure OSS usage, developing open source program office (OSPO) guidance, and prioritizing federal actions in OSS security.
- **Goal 4:** Harden the OSS ecosystem, advance software bills of materials (SBOMs), foster security education for OSS developers, publish OSS security best practices, and coordinate OSS vulnerability disclosure and response.

## NHTSA Cybersecurity Best Practices of Modern Vehicles

In September of 2022, the United States Department of Transportation National Highway Traffic Safety Administration (NHTSA) [published Cybersecurity Best Practices for the Safety of Modern Vehicles](#), a document aimed at addressing cybersecurity in modern vehicle manufacturing.



Among many recommendations, the NHTSA guides manufacturers to demonstrate processes and procedures to inventory software assets, track details related to open source software components, continuously monitor and assess risk, and disclose vulnerabilities. Most of these recommendations are also backed by the International Organization of Standardization (ISO) cybersecurity standards.

## European Union



### Cyber Resilience Act (CRA)

The CRA was proposed in September 2022, and it was met with mixed reviews. On the surface, it represents an improvement in holding manufacturers liable for the security of the software products they produce. However, it also represents a different direction for holding open source projects and creators liable for vulnerable software and attempts to potentially bring them in line with any traditional supplier. This approach presents interesting issues.

Although open source software is part of the software supply chain, how someone uses the software is often outside the understanding and control of the creator or open source project itself. As currently formulated, the CRA carries the potential to pose substantial barriers to many open source projects and even the distributors of open source software, potentially limiting their accessibility to European Union

markets. In some cases, it may discourage involvement in open source software in the EU altogether, putting EU-based companies far behind innovation and efficiency leaps in other geographical areas.

### Product Liability Directive (PLD)

The PLD delineates rules and guidance for the liability and responsibility of manufacturers of defective products as well as suppliers of defective parts for those products. A proposed update in September 2022 marked a notable expansion encompassing software and digital products, which had been excluded previously.

These updates put open source software and any related projects or activity within the potential sights of liability or responsibility associated with defects (i.e., vulnerabilities). For example, as written, there would be potential to find distributors of parts, such as Sonatype/Maven Central, liable for open source vulnerabilities resulting in loss or in some cases even distress to software product customers.

### Network and Information Security Directive (NIS2)

In December 2022, the European Parliament approved updates to the Network and Information Security Directive (NIS), referring to it going forward as NIS2. NIS2 is an evolution intended to modernize the approach of EU member nations towards cybersecurity. Among a host of updates, NIS2 includes call-outs for improved software supply chain security, greater attention to critical vulnerabilities, the increase of attacks via malicious threats, and the necessity of processes for disclosure and communication, such as Coordinated Vulnerability Disclosure (CVD).

NIS2 makes it clear that “Businesses identified by the Member States as operators of essential services in the above sectors will have to take appropriate security measures and notify relevant national authorities of serious incidents. Key digital service providers, such as search engines, cloud computing services and online marketplaces, will have to comply with the security and notification requirements under the Directive.” In addition, there are a number of

**As currently formulated, the CRA carries the potential to pose substantial barriers to many open source projects and even the distributors of open source software, potentially limiting their accessibility to European Union markets.**

requirements imposed based on different business categories: Highly Critical Sectors, Critical Sectors, and Essential Entities. In order to help countries address these requirements, ENISA has published guidance documentation including [Good Practices for Supply Chain Security](#). Failure to meet the requirements established within NIS2 can result in a number of sanctions.

## Canada



On [October 28, 2022](#), the Canadian Centre for Cyber Security (the Cyber Centre) released its [National Cyber Threat Assessment 2023-2024](#), warning that state-sponsored and criminal cyber threats are increasingly likely to affect Canadians. The report highlights five key areas of concern: ransomware, critical infrastructure, state-sponsored threats, influence operations, and disruptive technologies.

Among a number of criteria, the report also cites increased risk from remote work, more connected systems, and the proliferation of cybercrime tools. In assigning responsibility, the report points to threat actors in China, Russia, Iran, and North Korea that pose the greatest state-sponsored risk. In response, the Canadian government has invested in cybersecurity, including \$875 million in Budget 2022 to bolster defenses.

In February 2023, the Cyber Centre published [“Protecting your organization from software supply chain threats,”](#) which offers guidance

and best practices for securing software supply chains. The document outlines the importance of minimizing software supply chain risk, including the ability of malware to compromise updates, and the importance of remediating vulnerabilities in open source software components. Recommendations include vetting suppliers, monitoring them continuously, and incorporating supply chain risk management into security programs. Finally, organizations should have recovery plans to ensure business continuity when software supply chain attacks occur.

## Global partnerships

### Quad Cybersecurity Partnership: Joint Principles for Secure Software

#### UNITED STATES, JAPAN, INDIA, AND AUSTRALIA

Published in May 2023, the “Quad Cybersecurity Partnership: Joint Principles for Secure Software” document outlines the commitment of the Quad partners (the United States, Japan, India, and Australia) to enhance software security.

The main points are:

► **Recognition of security risks:** The Quad partners acknowledge the risks associated with the software supply chain being tampered with by adversarial and non-adversarial threats.

► **Promotion of secure software culture:** They aim to promote a culture where software security is a fundamental aspect of software development.

► **Minimum cybersecurity guidelines:** The Quad partners commit to establishing minimum cybersecurity guidelines for governments to guide software development, procurement, and usage, aligned with international obligations and domestic laws.

► **Engagement with the software industry:** They plan to engage with the software industry to ensure secure software practices are integrated throughout the software development life cycle, with the goal of reducing vulnerabilities.

► **Secure software development practices:** The document outlines high-level secure software development practices, including preparing the organization, protecting software and its development environment, producing well-secured software, and responding to vulnerabilities.

► **Government procurement guidelines:** Each Quad country intends to adopt guidelines for government procurement of software, including self-attestation by software producers regarding secure development practices and encouraging participation in national vulnerability disclosure programs.

► **Security measures for government software use:** The Quad partners commit to implementing controls and processes to protect government software and platforms from unauthorized access and usage.

The document's purpose is to strengthen software security across Quad countries, encourage the adoption of secure software practices, and establish guidelines for software procurement and usage in government contexts.

## Secure by Design and Default International Support

### AUSTRALIA, CANADA, UNITED KINGDOM, GERMANY, NETHERLANDS, NEW ZEALAND

In April 2023, the Australian Cyber Security Centre (ACSC) published “Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design and Default.” This is the same document that has become the guiding foundation for the US Cybersecurity Strategy. Though the [guidance has been led by CISA and the FBI](#) in the US, the document represents a collaboration across many international cybersecurity organizations:

- Australian Signals Directorate's Australian Cyber Security Centre (ASD's ACSC)
- Canadian Centre for Cyber Security (CCCS)
- United Kingdom's National Cyber Security Centre (NCSC-UK)
- Germany's Federal Office for Information Security (BSI)
- Netherlands' National Cyber Security Centre (NCSC-NL)
- New Zealand's National Cyber Security Centre (NCSC-NZ) and Computer Emergency Response Team New Zealand (CERT NZ)

The significance of this collaboration cannot be overstated as it represents a pivotal shift in cybersecurity responsibility towards software organizations. As previously discussed, the “Secure by Design and by Default” approach highlights the crucial integration of security into the design and default settings of technology products, effectively mitigating cyber threats. It emphasizes that the onus of security should not solely rest on end-users, compelling technology

**The significance of this collaboration cannot be overstated as it represents a pivotal shift in cybersecurity responsibility towards software organizations.**

manufacturers to prioritize security as a core business objective. To achieve success, a global shift towards embracing Secure-by-Design and Secure-by-Default practices is imperative, fostering transparency, accountability, and collaboration between manufacturers and customers. This necessitates manufacturers being held accountable for product security outcomes and prioritizing Secure-by-Design and Secure-by-Default principles when making technology procurement decisions. Overall, this guidance shapes future cybersecurity policies by advocating for proactive security measures and shared responsibility within the technology ecosystem.

## Are software supply chain regulations working?

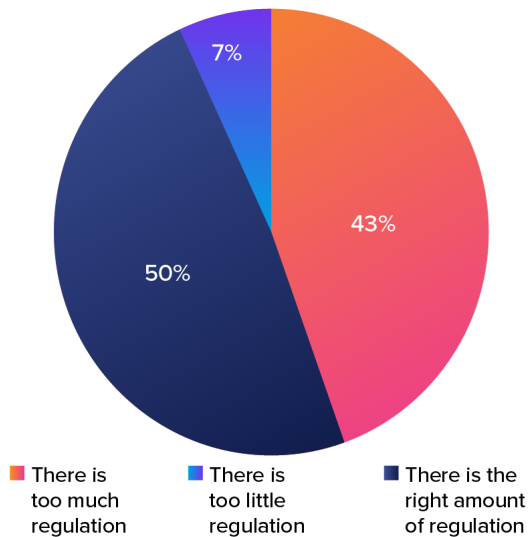
As we've seen from the analysis above, we're still in the very early stages of implementing true regulation that will hold people accountable for a more secure software supply chain.

But, especially in the United States, the conversation and incremental changes since President Biden's [Executive Order on Improving the Nation's Cybersecurity](#), is starting to become noticeable. As a direct requirement of the executive order, the Office of Management and Budget (OMB) was instructed to develop policy for federal agencies working with government contractors. Following that directive, the OMB released Memo [M-22-18](#), highlighting

the importance and potential requirement for SBOMs, especially as part of new [self-attestation standards](#). To understand this further, we surveyed 217 Cybersecurity Directors in organizations with over £50 million/\$50 million revenue in the UK and US respectively, and asked how much various policies were affecting their actions toward software supply chain management today.

A huge 76% of enterprises have adopted a Software Bill of Materials (SBOM) since the Order's introduction. Another 16% plan to implement SBOMs within the next year, showing increasing recognition of the correlation between open source hygiene and cybersecurity posture. Of the three-quarters of companies with SBOMs in

**FIGURE 5.1**  
HOW SURVEY RESPONDENTS FEEL ABOUT THE AMOUNT OF SECURITY GUIDANCE AND REGULATION IN THEIR COUNTRY



**76%**

of enterprises have adopted a Software Bill of Materials (SBOM) since the President Biden's Executive Order on Improving the Nation's Cybersecurity's introduction

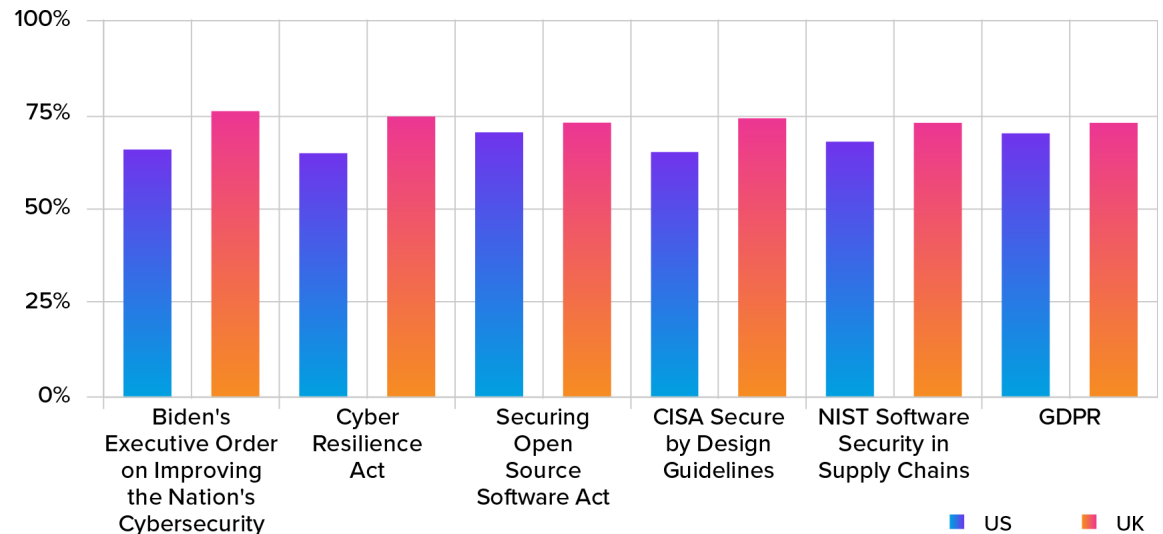
place, only 4% adopted them over three years ago, demonstrating how much practices have evolved since the Order.

Our research also confirmed the Order has influenced enterprises' software development practices in ways transcending SBOMs.

Respondents are increasingly investing in technologies to improve software supply chain management, including vulnerability scanning (30%), software composition analysis (24%), supply chain automation (23%), threat intelligence (22%), and bug bounty programs (20%). The regulation has also fuelled investment in skills and operations like employee training and awareness (26%), recruiting developer talent (21%), and processes to assess supply chain risks (24%).

We also examined attitudes to regulation in the UK and the US, uncovering that large enterprises generally see regulation as a good thing. In fact, 41% of security decision-makers see cyber regulation as the factor having the

**FIGURE 5.2**  
PERCENTAGE OF SECURITY LEADERS WHO BELIEVE REGULATIONS ARE EFFECTIVE FOR IMPROVING CYBERSECURITY



greatest positive impact on software security. Some, however, lament the volume of cybersecurity regulation, with 44% of business leaders believing there is too much government intervention on cybersecurity overall.

## Navigating the policy frontier: Global cybersecurity regulations

The global landscape of cybersecurity guidance and regulation in 2023 reflects a marked transformation compared to years past driven by the ever-increasing urgency to address evolving cyber threats.

Initiatives and regulatory actions taken by key players such as the United States, European Union, United Kingdom, Australia, Canada, Japan, and New Zealand demonstrate a shared commitment to improve digital defenses and safeguard critical infrastructure.

Three key themes have emerged:

- ▶ The heightened emphasis on security during software creation.
- ▶ Holding software producers accountable for their products.
- ▶ The need for robust processes to address cybersecurity incidents.

The global trend of cybersecurity regulations in 2023 demonstrates a growing collective endeavor to adapt to the ever-changing threat environment. Further international cooperation in this regard will be necessary to better prioritize secure software development practices. Regulations covered above as well as future related initiatives will play a pivotal role in shaping the future of cybersecurity policies and practices at scale worldwide.



CHAPTER 6

# AI in Software Development



In the ever-evolving digital landscape, artificial intelligence (AI) and machine learning (ML) stand out as transformative forces reshaping software development. As innovation takes center stage, an expanding toolkit of AI components and models play a pivotal role in driving this transformation.

Our exploration into the impact of AI and ML on software development draws from our survey of more than 800 developers (DevOps) and application security (SecOps) professionals and research. These insights reveal a broad adoption trend, with 97% currently incorporating generative AI in their workflows to some degree. While the impact and intricacies of these tools offer immense opportunities, they also present challenges, including concerns for security and the impact on jobs. Adding to these, and of particular interest to Open Source Software and the software supply chain, the consumption of AI and ML libraries has seen incredible rates of adoption. Unfortunately, many of the same concerns are present for teams including unplanned costs and increased liability.

This section covers two distinct but interrelated questions that have emerged on this topic — What role do AI and ML play in assisting developers, and what are the challenges that AI practitioners face in developing AI products? We explore both of those questions here

**Nearly half of the respondents—47% of DevOps and 57% of SecOps—reported that by using AI, they saved more than six hours a week.**

## Sonatype's risks & rewards of AI survey

For our recent AI-specific survey, we engaged DevOps and SecOps Leads responsible for software development, coding, developer relations, application security, threat intelligence, and analysis or security operations. Our primary objective was to understand how teams were using AI in their workstreams. Our questions ranged from frequency of use to what they found beneficial and challenging. We asked about the tools they were using, which industries are facing AI-related risks, and where they see the biggest opportunities. We'll give you a peek at some of the fascinating results below. For a more in-depth investigation, you can access the full report here.

## The intersection of AI and software development

### Code generation and testing

One of the most significant implications of AI in software development is its potential to

generate code. Platforms like [OpenAI's Codex](#), which powers tools like [GitHub's Copilot](#), can assist developers by suggesting entire lines or blocks of code. Nearly half of the respondents—47% of DevOps and 57% of SecOps—reported that by using AI, they saved more than six hours a week. Automated code suggestions mean faster development cycles.

Beyond speeding up the coding process, AI helps reduce the potential for human errors thereby expediting time-to-market and improving the quality and maintainability of software. Emerging AI-driven solutions can identify vulnerabilities, bugs, and inefficiencies in software code more swiftly than traditional methods. By understanding the context and intent of the code, these systems can predict potential failure

**The majority of SecOps and DevOps leads surveyed said they used AI for testing and analyzing.**

## OPPORTUNITY FOR DEVELOPERS AT ALL LEVELS

The benefits of AI for software developers extend across all levels of the experience spectrum.



### For senior developers: Enabling a motivated junior

- ▶ Senior Developers can leverage AI tools like GitHub Copilot to quickly complete tedious tasks.
- ▶ ChatGPT offers the promptness of AI technology while allowing juniors to be part of the development process and work closer with the Senior Devs versus just doing the “grunt work.”



### For juniors and seniors: Accelerating with an extensive reference

- ▶ Generative AI tools serve as a practical reference book for developers of all levels.
- ▶ ChatGPT helps look up common behaviors in programming languages and give clues on how to accomplish something without knowing basic syntax.
- ▶ Generative AI reduces time spent teaching standard techniques to juniors and helps senior developers with brainstorming.
- ▶ ChatGPT can also be a valuable tool for debugging.



### For juniors and seniors: Streamlining with private repositories

- ▶ GitHub Copilot Chat allows developers to query the repository directly.
- ▶ These tools benefit developers working with a legacy project with little documentation or local knowledge or when they need to learn a new code base.
- ▶ AI tools could reduce onboarding time for new hires from several months to a few weeks.



### For junior developers: Benefiting from an AI mentor

- ▶ Junior developers benefit from having an AI/ML tool to answer quick questions and provide insight into technical terms and jargon.
- ▶ ChatGPT saves time for senior developers by handling basic queries, allowing them to focus on more complex issues.
- ▶ Large language model (LLM) tools can help answer questions about why a particular tool or architecture might be chosen, helping juniors understand tradeoffs and potential decisions.

## The hottest new programming language is English.

points, enhancing the software’s resiliency. The majority of SecOps (82%) and DevOps (79%) leads surveyed said they used AI for testing and analyzing.

By and large, this is where most of the benefits are initially realized in software development teams today.

### AI tools

Among the 97% of DevOps and SecOps leaders who confirmed they currently employ AI to some degree in their workflows, most said they were using two or more tools daily. Topping the list at 86% was ChatGPT and GitHub Copilot at 70%. Notably, GitHub Copilot caters primarily to developer teams. The capability and widespread adoption of these tools underscores the notion that “the hottest new programming language is English.” Additionally, respondents also mentioned the utilization of other tools such as SCM Integrations, IDE Plugins, and Sourcegraph Cody.

## Navigating concern

While the prospects of Generative AI in software development are undoubtedly exciting, it doesn’t come without its challenges—even

if those challenges are just perceived and not realized. 61% of developers believe the technology is overhyped compared to the security leads at 37%. Although the majority of respondents are utilizing AI to varying degrees, it is not

**FIGURE 6.1**  
**NUMBER ONE CONCERN ABOUT USING GENERATIVE AI**



necessarily driven by personal preference. A striking 75% of both groups cited feeling pressured from leadership to adopt AI technologies, recognizing their potential to bolster productivity despite security concerns.

### Introducing security challenges

Three (3) out of four (4) DevOps leads have expressed concerns regarding the impact of generative AI on security vulnerabilities, especially in open source code. Additionally, more than 50% of these individuals believe that this technology will complicate threat detection. Interestingly, less than 20% of SecOps professionals shared these concerns.

Perhaps not surprisingly, 60% of large companies surveyed were more likely to be concerned about security risks compared to less than 50% for smaller organizations. The same went for the illegal use of unlicensed code, 49% of large organizations are worried about this, compared to 41% and 35% for mid-size and small organizations, respectively.

**This reinforces the concept that AI is a tool to augment human capability rather than replace it.**

## 3 of 4

DevOps Leads have expressed concerns regarding the impact of generative AI on security vulnerabilities

### Tools require guidance

AI tools, particularly Large Language Models (LLMs), offer significant assistance in various tasks, but they require considerable guidance to check their work and look for signs of bias. LLMs are not bound by fact, so they should not operate autonomously. For one, LLMs experience [hallucinations](#) or false information that require the person using the model to recognize when a mistake is made. AI is also not bound by rules or logical constraints. There is no shortage of YouTube videos where people engage in chess matches with ChatGPT. The tool's highly unconventional moves would certainly be deemed unacceptable in a legitimate game. This reinforces the concept that AI is a tool to augment human capability rather than replace it. It will require experience and knowledge to identify these mistakes when LLMs generate code. If not carefully monitored, the risk of developing technical debt, a concern of about 15% of each surveyed group, might become a reality.

### Job implications

One of the most palpable fears associated with the rise of AI and ML is job displacement. 1 in 5 of the SecOps leads in our survey noted this as



their top concern. The current shortage in cybersecurity talent, specifically, forces organizations to get creative to fill that gap. AI might be one way to do that. However, the reality points to a shift in roles rather than outright replacement. The technology is creating an environment where human creativity, intuition, and strategy are more valuable than ever. As shown above, in software engineering, junior, senior and every level in between can benefit from the use of AI in their workstreams and as a learning tool.

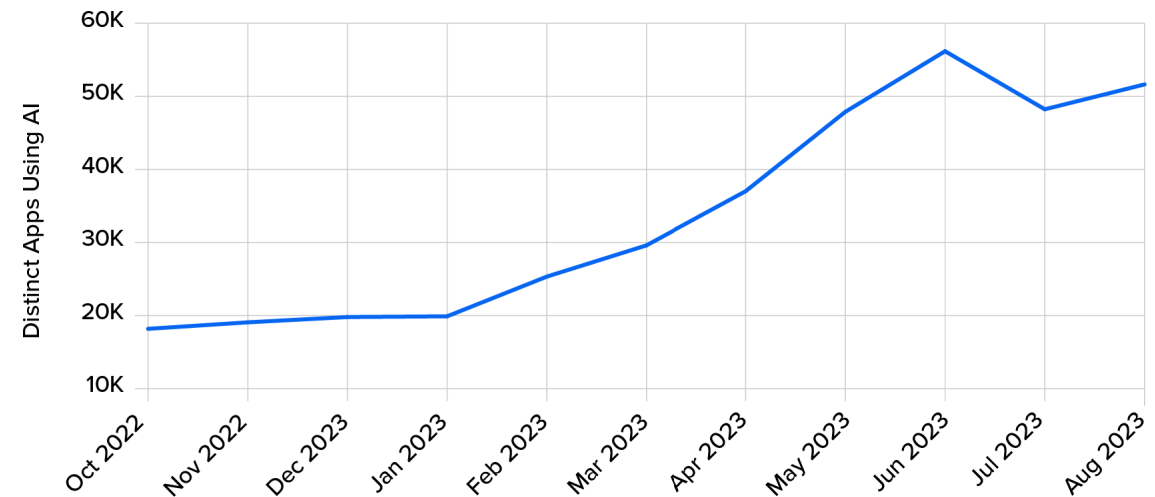
## AI's open source toolkit: Components and models in focus

### Doubling down on AI and ML: Enterprise adoption trends

The usage of AI and ML components has also experienced a remarkable surge in the enterprise. Over the past year, the adoption of these tools within corporate environments has more than doubled (**Figure 6.2**), reflecting a significant shift in how companies approach data science and machine learning. With the introduction of advanced AI models like ChatGPT, organizations have increasingly recognized the potential for enhancing decision-making, automating tasks, and extracting valuable insights from their data. This surge underscores a growing awareness of the transformative power of AI and ML, as businesses strive to leverage

FIGURE 6.2

### NUMBER OF ENTERPRISE APPS USING AI COMPONENTS



these technologies to stay competitive and drive innovation in their operations.

The word cloud in **Figure 6.3** showcases the versatile toolkit embraced by data scientists and AI practitioners. We observed a diverse spectrum of tools and frameworks gaining prominence. These range from classical machine learning stalwarts like scikit-learn to the deep learning powerhouse PyTorch, with HuggingFace transformers making a noteworthy ascent, highlighting the increasing influence of transformer-based models in contemporary AI endeavors.

In our comprehensive analysis of the software supply chain, we delved into the rapid expansion of components tailored to create or interact with Large Language Models (LLMs). Notably,

FIGURE 6.3

### TOP 25 MOST POPULAR DATA SCIENCE COMPONENTS

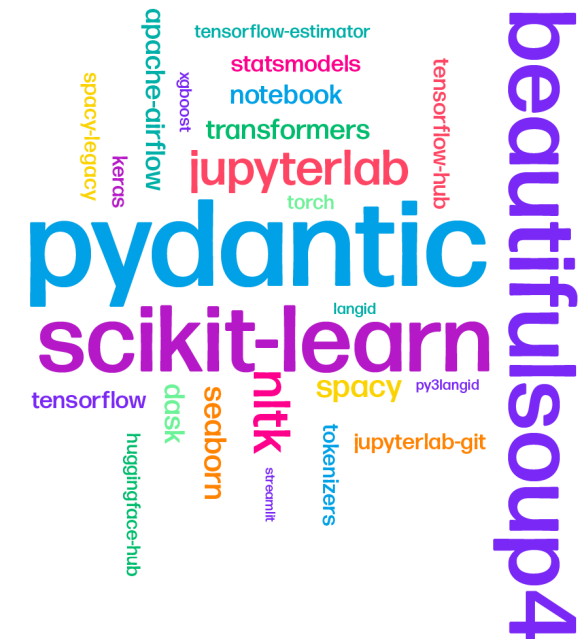
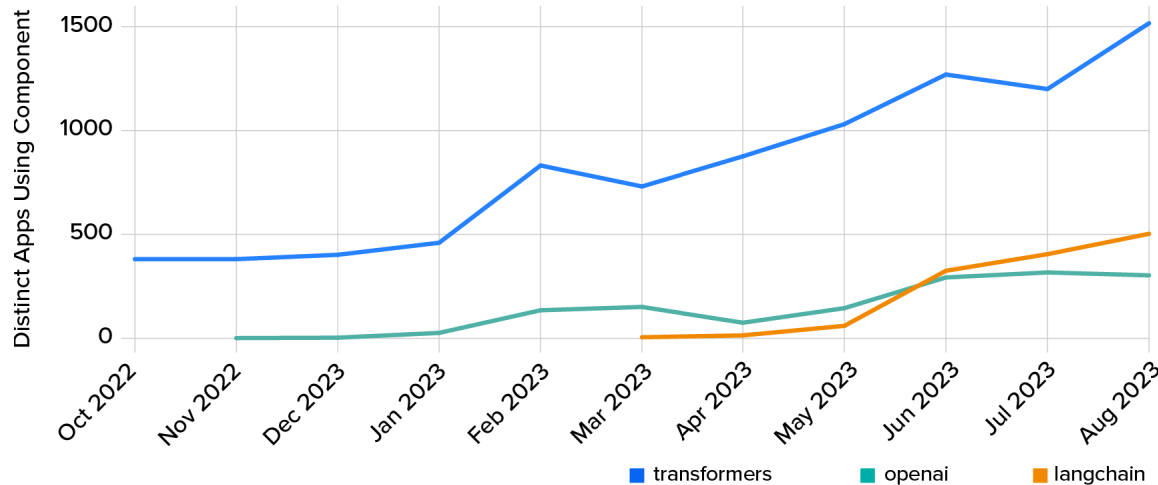




FIGURE 6.4

## LANGUAGE LEARNING MODEL GROWTH

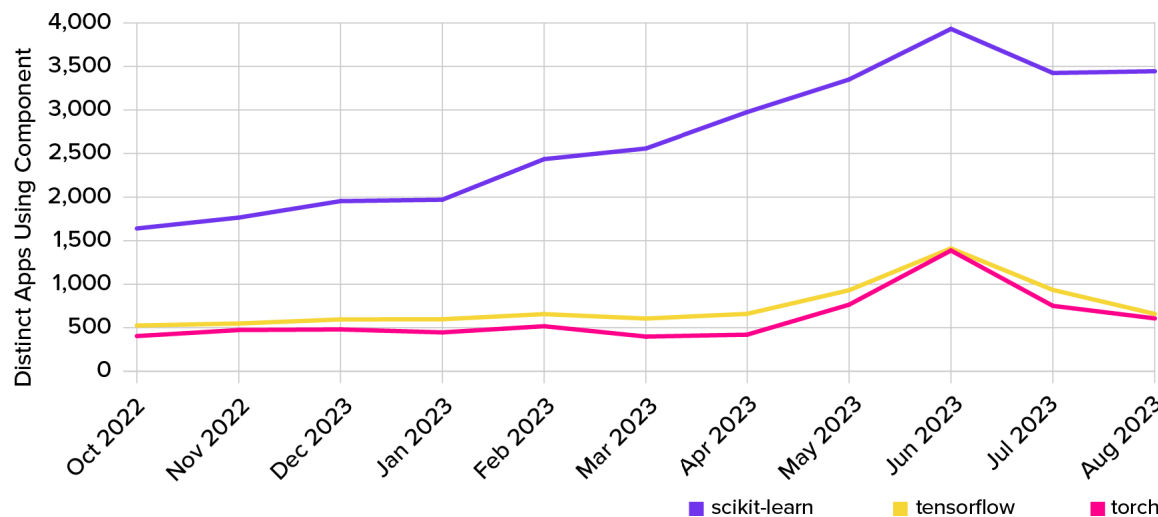


our research unveiled the meteoric ascent of libraries like Langchain, OpenAI, and Transformers, signifying a surging interest in harnessing the power of natural language understanding and generation (**Figure 6.4**).

Equally fascinating was the finding that this growth wasn't confined solely to cutting-edge generative AI components. Instead, it radiated across the spectrum, encompassing well-established frameworks such as scikit-learn, TensorFlow, and PyTorch (**Figure 6.5**). This diversification underscores a strategic shift among enterprises as they explore multiple avenues to extract actionable insights and value from their data. Hence, the evolving software supply chain embraces both novel LLM-focused tools and enduring traditional frameworks.

FIGURE 6.5

## TRADITIONAL MACHINE LEARNING GROWTH

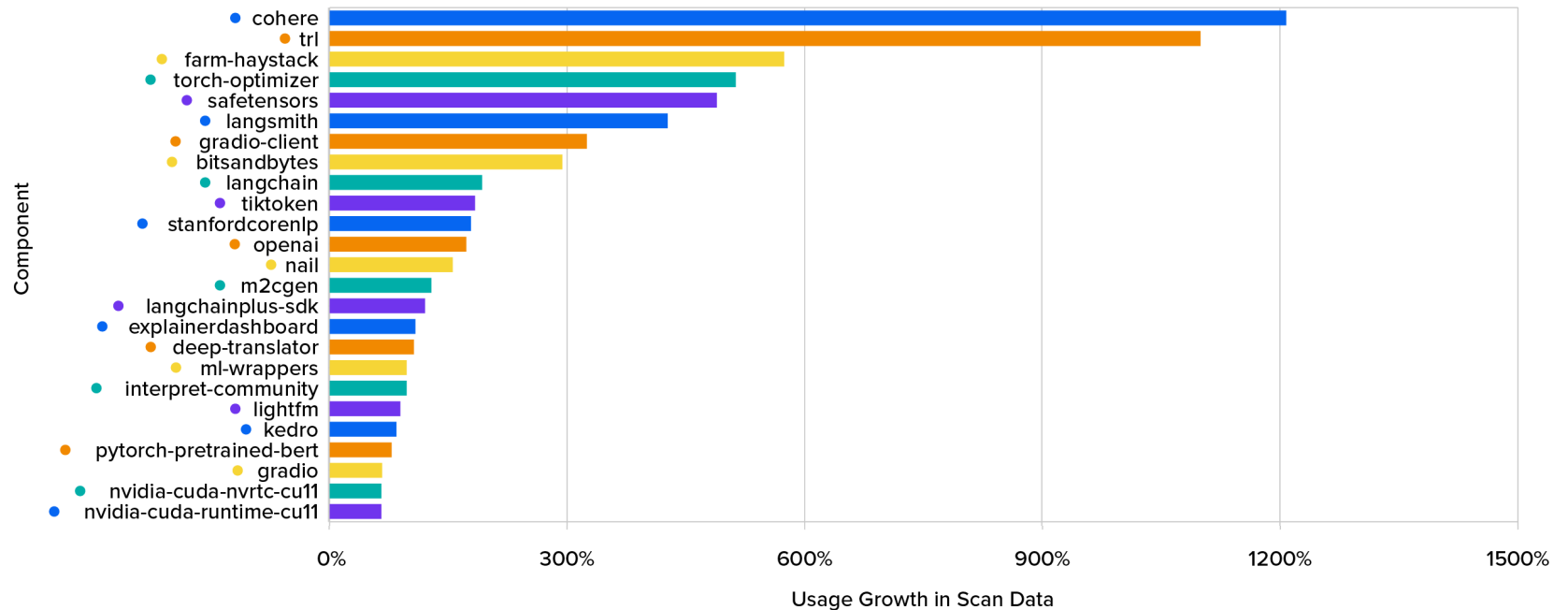


The growth of Large Language Models (LLMs) extends far beyond the offerings of tech giants like Google, Microsoft, and OpenAI, encompassing a thriving and diverse ecosystem within the enterprise sector (**Figure 6.6**). This ecosystem includes solutions such as Cohere, streamlining the integration of LLMs into enterprise products and services, as well as the rising adoption of Langchain and Langsmith for developing applications harnessing LLM capabilities. Notably, a significant portion of these applications are built upon open source LLM models.

This trend towards open source AI is further underscored by the widespread adoption of packages like Safetensors, enhancing the security of model weight serialization, and CUDA

FIGURE 6.6

## COMPONENTS WITH THE HIGHEST GROWTH RATE IN USAGE



GPU accelerators, facilitating faster model training and fine-tuning. Additionally, tools like Bitsandbytes provide quantization methods for these expansive open source models, optimizing inference speed and deployment on resource-efficient clusters.

While we have witnessed the remarkable output of these multi-billion parameter models, there remains a compelling case for interpretable machine learning. This demand is captured by the growth of components like Explainerdashboard, designed to elucidate the predictions made by machine learning models, ensuring

transparency and accountability in AI-driven decision-making.

### Pros and cons of LLM-as-a-service

Large Language Models (LLMs) as a service offer several distinct advantages to enterprises and companies. Firstly, they accelerate development, simplifying the integration of advanced language capabilities into applications through straightforward API calls. Additionally, LLMs as a service can deliver impressive performance benefits as the bulk of the processing is handled server-side, offloading computational demands from local devices.

However, there are notable drawbacks to consider. One significant concern is cost, as enterprises typically pay for each token sent and received, which can accumulate quickly with extensive usage. Data privacy and security are also paramount concerns, as enterprises may lack full transparency into how their data is being utilized by the service provider, raising potential privacy issues. Moreover, vendor lock-in poses a substantial risk, as reliance on a particular service leaves applications vulnerable to vendor outages, deprecated features, or unforeseen changes in model performance that may not align with the specific task at hand. Balancing these pros and cons is essential when evaluating

the integration of LLMs as a service into an enterprise's workflow. As more companies turn towards open source AI solutions, they encounter familiar challenges related to the consumption of these open source components, which, in this context, are the AI models themselves.

## Data scientist burden

Data scientists and engineers tasked with deploying open source Large Language Models (LLMs) shoulder a substantial burden, encompassing numerous critical decisions:

- ▶ **Model Selection:** With a vast repository of over 337,237 models available, they must carefully select the most suitable one for their specific application.
- ▶ **Parameter Size:** Choosing from a spectrum ranging from hundreds of millions to hundreds of billions of parameters, they must determine the ideal model size to balance computational requirements and performance.
- ▶ **Context Window:** Defining the appropriate context window, representing the number of tokens a model can take as input, is essential to optimize the model's responsiveness.
- ▶ **Version Selection:** Deciding which version of the model to use—be it chat-oriented, instruction-tuned, or code-tuned—requires careful consideration to align with the project's objectives.

▶ **Embeddings:** The selection of embeddings plays a crucial role in fine-tuning the model's performance for specific tasks.

▶ **Licensing and Security:** While navigating these choices, data scientists must remain vigilant about the licensing and security implications. They must verify the model's release license, ensuring it aligns with their intended usage, and be cautious of any potential licensing conflicts. Additionally, they must assess security risks, considering the possibility of uncensored or potentially inflammatory responses from the model and the implications of deploying versions that can be jailbroken.

## Licensing risk

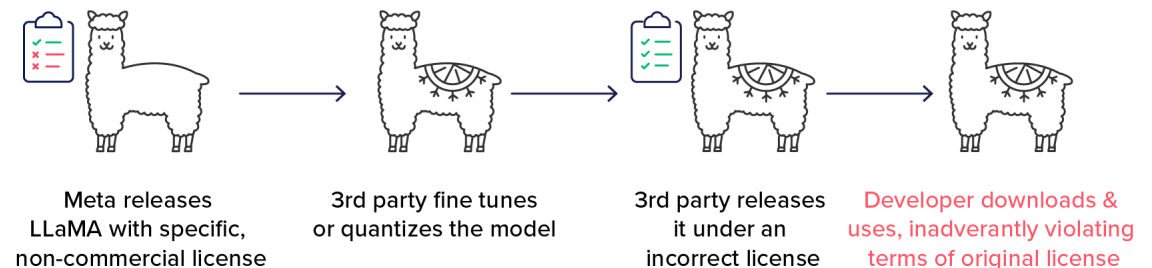
Deploying open source LLMs presents significant opportunities for natural language interaction with company products and knowledge bases. However, it is imperative to recognize the potential licensing risks associated with these models. In many cases, developers may fine-tune these models to suit specific applications, but the licensing terms of the foundational

model must be carefully considered. For instance, if a foundational model, such as Meta's LLaMA, is released under a non-permissive license that restricts commercial use or imposes other conditions, deploying a fine-tuned version with an incorrect license can inadvertently violate those terms. This situation can lead to legal liabilities and intellectual property disputes, even if the fine-tuned model itself is released with a different license. It underscores the importance of due diligence in understanding and adhering to the licensing terms of both the foundational and derived models to avoid unintended legal consequences. Furthermore, it necessitates model fingerprinting for lineage tracking to avoid these consequences.

As we have seen, LLMs are exceptionally proficient at generating human-like text and working code based on the input provided to them. AI models have gained these capabilities in part by scraping publicly available data off of the Internet without seeking express permission from copyright holders. Thus, this capability includes the potential to generate content that closely

FIGURE 6.7

### NAVIGATING LICENSING RISKS IN THE DEPLOYMENT OF OPEN SOURCE LANGUAGE MODELS



resembles copyrighted content. When LLMs generate text that mirrors copyrighted material without adequate attribution or authorization, it raises serious copyright infringement concerns.

It is clear that in this instance, technological innovation is ahead of legislation. As evidenced by the many court battles between generative [AI firms](#) and [content creators](#). Inevitably some of this stems from a misunderstanding of the technology. In contrast, there are also some egregious examples of the output of generative AI being a lot more indicative of memorization compared to generalization, such as when artists' ghostly signatures appear in AI [generated artworks](#).

The copyright issues around the training sets and outputs of generative AI aren't going away anytime soon. While the legal, and sometimes even philosophical, debates get resolved, some companies have taken proactive steps, offering legal protection for AI copyright infringement challenges to customers using products such as [Microsoft's Copilot](#). Overall, the devil is in the details, and the legal challenges are likely to help democratize the AI landscape as companies will have to become more transparent about the training datasets, model architectures, and the checks and balances in place designed to safeguard intellectual property.

In light of the potential risks associated with copyright infringement in generative AI, it is

## While concerns about job displacement persist, the prevailing sentiment is that AI aims to enhance human abilities rather than replace them.

imperative for enterprises to adopt a proactive approach. Over-reliance on a single LLM-as-a-service, or single foundational model, such as LLaMA, may prove detrimental in the face of regulatory challenges. Therefore, it is prudent to explore viable alternatives to mitigate potential consequences. In the event of a hypothetical scenario wherein copyright infringement is attributed to the training data of models like LLaMA, the critical question arises: Can you ensure that your applications remain compliant and unaffected by their utilization of LLaMA-derived models? Preparing for such contingencies is essential in a world where every company is becoming a data and AI company.

## Conclusion: A collaborative future

The rapid evolution and integration of AI, especially Large Language Models (LLMs) such as GitHub's Copilot and ChatGPT, as well as their open source alternatives, have brought about

a significant shift in software development. Our analysis and survey highlight this growth as a critical milestone in our technological era. These advancements offer transformative benefits, including increased productivity, advanced language understanding, and diversified AI components in engineering workstreams. However, along with these advantages, there are also challenges to be addressed.

Potential errors in AI-generated code, complexities in model selection, security, and licensing require meticulous oversight. While concerns about job displacement persist, the prevailing sentiment is that AI aims to enhance human abilities rather than replace them. Striking a balance becomes crucial. We must harness AI's unparalleled capabilities while remaining mindful of its challenges. Transparency, accountability, and due diligence should be emphasized.

As we journey forward, it is essential to ensure a balanced and responsible coexistence between AI tools and their human counterparts. A critical aspect of this task extends beyond tool assessment and involves crafting a governance strategy that assesses the impact of adopting both proprietary and open-source libraries and models on your software supply chain.. By navigating the potential pitfalls and maximizing the profound opportunities that AI presents, we can shape a future that optimizes the benefits of this technology.

# About the Analysis

Sonatype's 9th annual State of the Software Supply Chain report blends a broad set of public and proprietary data and analysis, including dependency update patterns for more than 400 billion [Maven Central downloads](#) and thousands of open source projects, survey results from 621 engineering professionals, and the assessment of hundreds of thousands key enterprise applications. This year's report also analyzed operational supply, demand and security trends associated with the Java (Maven Central), JavaScript (npmjs), Python (PyPI), and .Net (nuget) ecosystems. The authors have taken great care to present statistically significant sample sizes with regard to component versions, downloads, vulnerability counts, and other data surfaced in this year's report.



# Acknowledgements

Each year, the State of the Software Supply Chain report is a labor of love. It is produced to shed light on the patterns and practices associated with open source, development and the evolution of software supply chain management practices.

The report is made possible thanks to a tremendous effort put forth by many team members at Sonatype, including Aaron Linskens, Alli VanKanegan, Anna Hubbard, Ax Sharma, Brian Fox, Bruce Mayhew, Eddie Knight, Elissa Walters, Ember DeBoer, Ilkka Turunen, Jeff Wayman, Juan Morales, Leina Sanchez, Maury Cupitt, Mitchell Johnson, Nicole Lavella, Stephen Magill, PhD, Steve Poole, Tara Condon, Tiffany Jennings, Todd Baseden, Vlad Drobinin, PhD and Wayne Jackson.

We would also like to offer thanks for contributions, big and small, and for sharing perspectives with our many colleagues across the DevOps and open source development community.

Another very special thank you goes out to Alli VanKanegan and Leina Sanchez, who created the incredible design for this year's report.



Sonatype is the software supply chain management company. Recognized by globally renowned analysts as a leader in the industry, Sonatype enables organizations to innovate faster in a highly competitive market. We allow engineers to develop software fearlessly and focus on building products that power businesses. Sonatype researchers have analyzed more than 120 million open source components – 40x more than its competitors – and the Sonatype platform has automatically blocked over 245,000 malicious components from entering developers' code. Enabling high-quality, secure software helps organizations meet their business needs and those of their customers and partners. More than 2,000 organizations, including 70% of the Fortune 100 and 15 million software developers, rely on our tools and guidance to be ambitious, move fast and do it securely. To learn more about Sonatype, please visit [www.sonatype.com](http://www.sonatype.com).

**Headquarters**

**8161 Maple Lawn Blvd, Suite 250  
Fulton, MD 20759  
USA • 1.877.866.2836**

**European Office**

**168 Shoreditch High Street  
London E1 6HU  
United Kingdom**

**APAC Office**

**60 Martin Place Level 1  
Sydney, NSW 2000  
Australia**

**Sonatype Inc.**

**[www.sonatype.com](http://www.sonatype.com)  
Copyright 2023  
All Rights Reserved.**