Software Supply Chain State of the Union 2025

Expanding threat landscape jeopardizes software integrity

Table of Contents

Introduction

Managing and securing the entire software supply chain is foundational for delivering trusted software releases. However, this is often easier said than done. As a software security-focused company with a dedicated security research organization and 15+ years supporting development and security teams, JFrog understands the threats and challenges today's organizations face. In a post-AI world, these challenges are only accelerating, leaving most DevSecOps teams wondering: how do we keep up with all the change?

This report combines JFrog usage data from millions of users, CVE analysis by the JFrog Security Research team, and commissioned third-party polling data from 1,400 Security, Development, and Ops professionals to answer that all-important question. The resulting analysis provides context into the broad software supply chain and development landscape, reveals where persistent and new risks reside, and explores what it takes to secure your software supply chain in 2025.

We hope you find value in this report and welcome any feedback, which can be shared with us at data_report@jfrog.com.

Executive Brief

The software supply chain is evolving at an unprecedented rate, creating the potential to expose organizations to new threats at an untenable pace. When it comes to mitigating risk across the supply chain, "more" is not necessarily the best approach. The old adage of "work smarter, not harder" by simplifying toolchains and processes will best serve organizations who want to move fast, embrace new technologies, and dominate the competition.

Your Software Supply Chain — bigger, faster, more complicated

Open-source ecosystem growth shows little sign of slowing down and organizations that are eager to innovate are moving fast to take advantage of the latest technologies.

- Two-thirds of organizations (64%) report using 7 or more programming languages. 44% are using 10 or more. This is an increase YoY, up from 53% and 31%, respectively.
- Public repositories continue to grow. Of note, Docker Hub added 1.9M images in 2024 and Hugging Face added 1M.
- The typical organization brings in 458 new packages a year. That nets out to 38 new packages a month, on average, and varies according to the number of developers.

More risk, less clarity

While understanding the potential impact of a CVE continues to be a complex endeavor, it is just the tip of the risk iceberg.

- A massive backlog at NVD did not stop new vulnerabilities from being discovered. Over 33,000 new CVEs were reported in 2024, an increase of 27% YoY.
- The JFrog Security Research team detected 25,229 exposed secrets/tokens in public registries (up 64% YoY), of which 6,790 were active.
- In a deep analysis of 183 notable CVEs, the JFrog Security Research team found 63 to never be exploitable in JFrog Cloud customers' scanned applications.

In addressing security risk, don't skip the basics

Organizations have adopted varying levels of security frameworks and are using even more security tools, but some essential best practices are getting missed along the way.

- 71% of respondents indicate their organization allows developers to download packages directly from the internet.
- 73% of organizations use 7 or more security solutions. 49% are using 10 or more. This is up from 47% and 33% reported last year.
- Less than half of respondents (43%) indicate their organizations are scanning at the code and binary level.
- 40% of respondents lack full visibility into the provenance of software running in production.

AI adoption is shifting into fifth gear

There are more options than ever for teams to bring AI services to production, but this presents new concerns for organizations to address.

- Over a million new models and datasets were added to Hugging Face this year, but with that came a 6.5x increase in malicious models.
- Teams are turning to hosted models (64%), but nearly half of organizations are also self-hosting models in some capacity – both proprietary and open source.
- 37% of organizations currently rely on manual efforts to curate and maintain a list of approved models in an effort to govern model artifact usage.

What's in Your Software Supply Chain?

The modern software supply chain is global and expansive, integrating multiple technologies and sources, with millions of new packages and libraries added to the most popular tech ecosystems annually. Software development organizations are now leveraging an unprecedented number of languages and their corresponding package ecosystems. While legacy technologies remain widely-utilized, innovation across established and emerging open-source ecosystems presents opportunities and risks that will be explored further in this report.

Number of programming languages used in development organizations

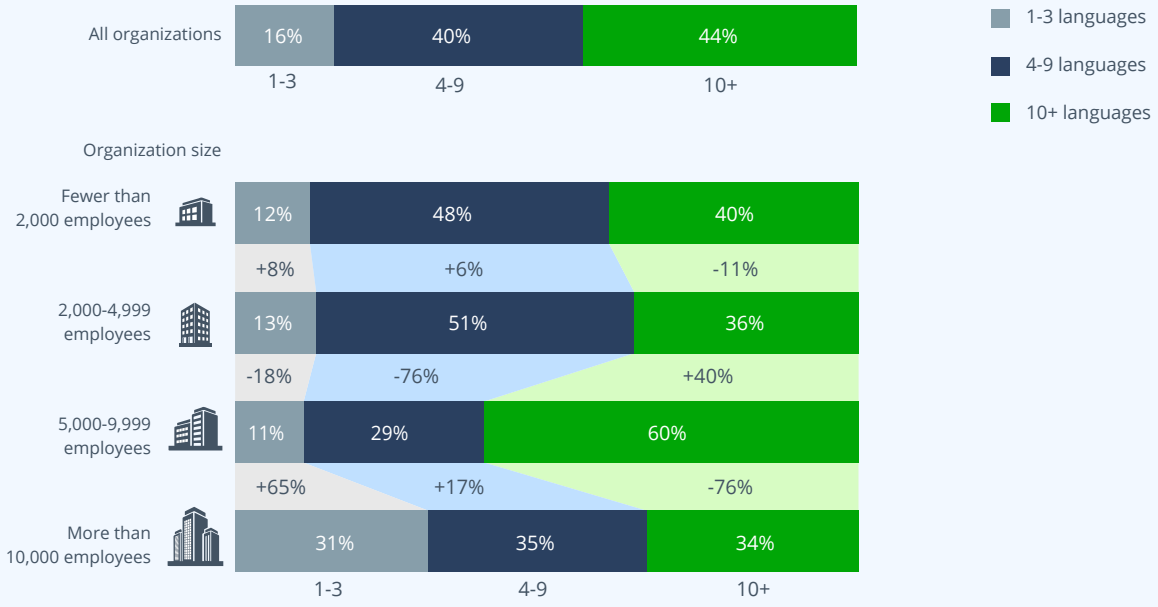


Figure 1.1. How many programming languages do you use in your software development organization? (Commissioned survey, 2024)

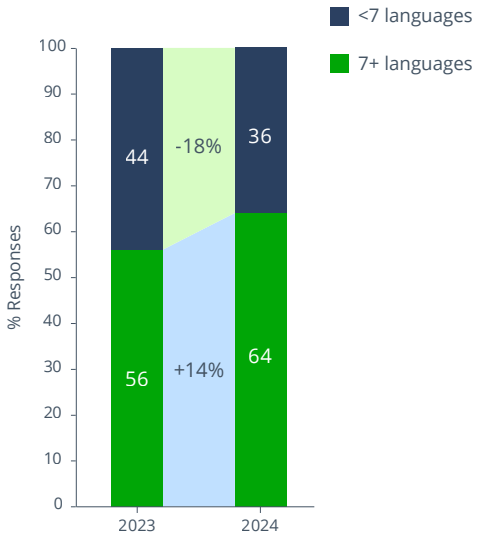


Figure 1.2

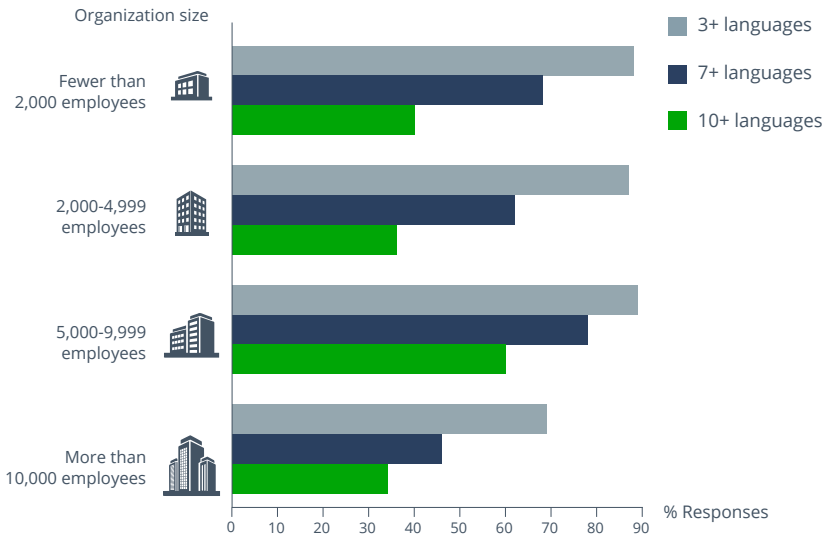


Figure 1.3

Nearly two-thirds of technology professionals (64%) report that their organizations are using 7 or more programming languages. Last year, just over half of respondents (56%) reported the same. This increase is reflective of the overall increase in complexity we are seeing across the software supply chain.

As organization size increases, we tend to see more languages in use – an expected trend. However, once the company size hits 10k or more employees, the reported number of languages in use actually decreases. This dip likely represents organizations past an inflection point where they realize they should start to take a more proactive approach in managing development and standardize using certain technologies to limit sprawl. It is also possible that larger organizations are maintaining established legacy applications with fewer new projects necessitating the usage of additional technology ecosystems.

New packages per year per package type

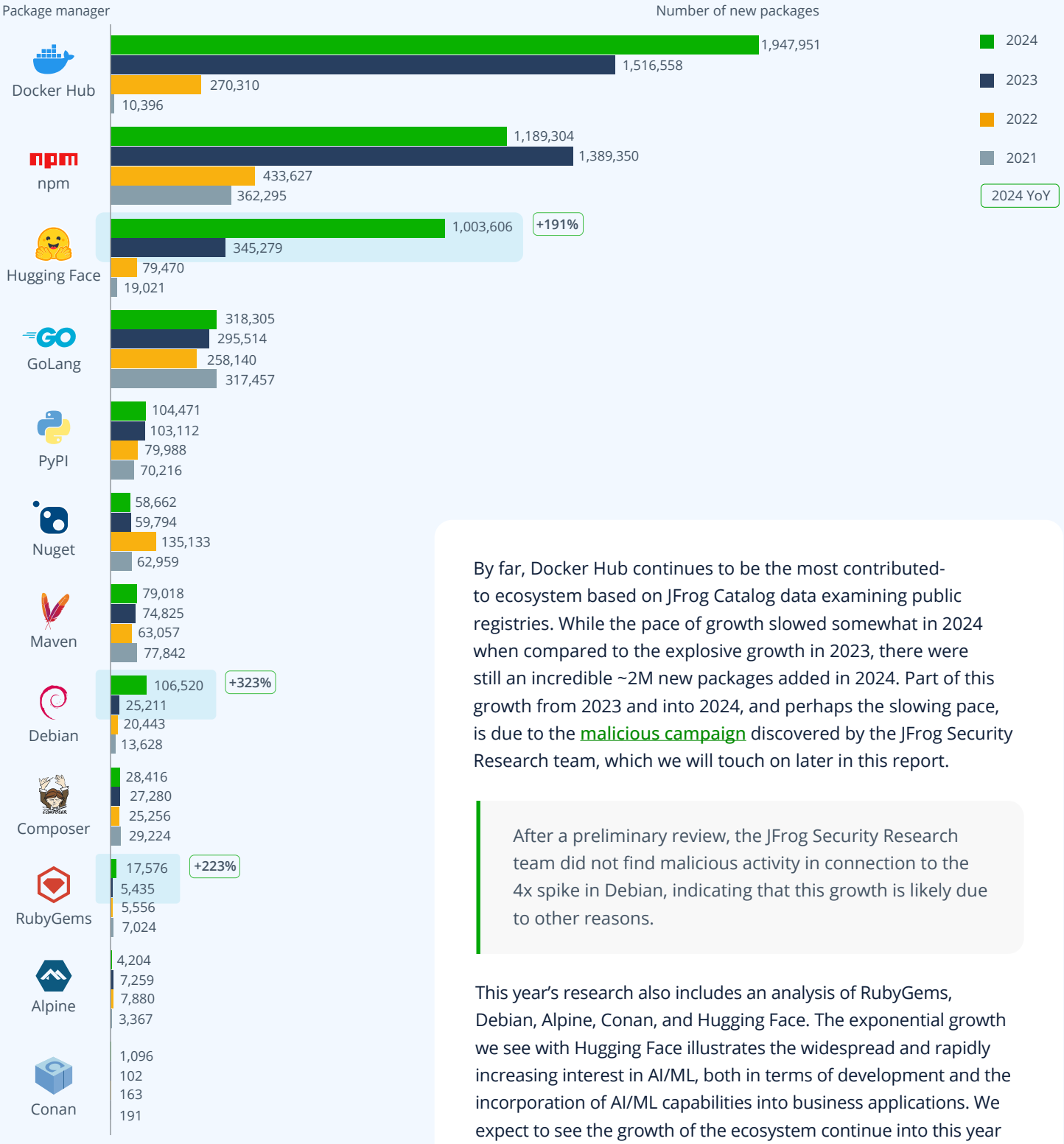


Figure 2. Number of new packages per year, displayed by package type (JFrog Catalog database, 2024)

By far, Docker Hub continues to be the most contributed-to ecosystem based on JFrog Catalog data examining public registries. While the pace of growth slowed somewhat in 2024 when compared to the explosive growth in 2023, there were still an incredible ~2M new packages added in 2024. Part of this growth from 2023 and into 2024, and perhaps the slowing pace, is due to the malicious campaign discovered by the JFrog Security Research team, which we will touch on later in this report.

> After a preliminary review, the JFrog Security Research team did not find malicious activity in connection to the 4x spike in Debian, indicating that this growth is likely due to other reasons.

This year's research also includes an analysis of RubyGems, Debian, Alpine, Conan, and Hugging Face. The exponential growth we see with Hugging Face illustrates the widespread and rapidly increasing interest in AI/ML, both in terms of development and the incorporation of AI/ML capabilities into business applications. We expect to see the growth of the ecosystem continue into this year and beyond.

Top package technologies in use by organizations

Package Type	Requests*	Number of Repositories	Artifacts
Maven	33.52%	104,955	2,567,881,564
npm	30.45%	48,549	674,010,130
Docker	15.45%	112,366	2,264,459,098
YUM	2.68%	14,669	20,785,724
PyPI	2.68%	22,352	66,838,230
Helm	1.61%	26,125	13,231,209
Nuget	1.45%	28,497	131,164,087
Debian	1.35%	8,184	8,066,185
Conan	1.33%	3,420	143,404,846
Gradle	0.99%	9,073	102,198,342
RubyGems	0.93%	3,736	46,728,889
Go	0.75%	9,034	16,511,299
OCI	0.47%	862	8,662,480
Cargo	0.13%	1,261	526,851
Sbt	0.12%	2,239	14,908,497
Helm OCI	0.07%	1,633	201,440
Ivy	0.06%	2,283	31,786,069
Composer	0.05%	2,413	614,957
Terraform	0.03%	3,566	675,684
Opkg	0.02%	529	33,812,836
Conda	0.02%	2,168	1,538,832
P2	0.02%	316	1,010,616
Pub	0.01%	363	166,878
Swift	0.01%	524	1,345,299
Alpine	0.01%	1,550	111,231
Cocoapods	<0.01%	1,400	2,973,045
Cran	<0.01%	2,403	816,170
VCS	<0.01%	273	1,692
Chef	<0.01%	1,530	150,462
Vagrant	<0.01%	680	7,326
Terraform Backend	<0.01%	2,307	395,004
Bower	<0.01%	985	44,161
Ansible	<0.01%	107	4,470
Puppet	<0.01%	1,530	17,758
Hugging Face	<0.01%	551	12,638

*% of total requests from 57 Billion requests in Q4

This year, we took an end-of-year (Q4) snapshot to get a more accurate look into the most popular technologies among the 35+ technology types that JFrog supports. While we continue to see the prevalence of well-established technology ecosystems, including npm, Docker, and Maven, there were some notable jumps in popularity for YUM and Cargo.

Over the past few years, we have watched the popularity of Cargo grow steadily, particularly as government entities push for more memory safe development. It remains to be seen whether the popularity of Rust will plateau or reach the widespread usage and adoption levels of more established languages like Java.

It is also worth noting the amount of OCI and Helm OCI usage. JFrog introduced dedicated repositories for OCI in early 2024 and many of our customers are already taking advantage of it. This indicates a growing preference for an open standard for containers and other technology ecosystems, and is why we expanded our Terraform repositories to natively support OpenTofu.

The use of common technologies differs by industry:

- **Automotive and IoT** companies leverage Maven (back-end apps), npm (front-end apps), Conan (embedded devices), Docker, PyPI (for AI/ML), and often bundle many of these together into generic packages (tar/zip images).
- **AI/ML and Robotics** companies leverage PyPI, ML models pulled from public repositories like Hugging Face and Tensorflow, and store these models in containers or generic packages (tar/zips). They may also adopt native repositories like Hugging Face or JFrog's Machine Learning Repository* for their models.
- **Insurance, Financial, and Retail** institutions leverage a combination of technologies like Maven, npm, and Docker, and with the increase in AI/ML, are starting to leverage PyPI and ML models to provide more enhanced offerings to remain competitive.

*JFrog's Machine Learning Repository was introduced in January 2025 and not included in the data of this report.

Figure 3. Technologies used, plus action counts, number of repos, and total size of artifacts stored for each (JFrog database, 2024)

Popular libraries

Rank	Docker	Maven	PyPI	npm
1	library/alpine	org.slf4j:slf4j-api	urllib3	@types/node
2	library/node	commons-io:cozmons-io	requests	semver
3	library/python	commons-codec:commons-codec	certifi	minimatch
4	library/nginx	org.ow2.asm:asm	charset-normalizer	glob
5	library/redis	com.fasterxml.jackson.core:jackson-core	setuptools	electron-to-chromium
6	library/busybox	com.google.guava:guava	idna	lru-cache
7	library/postgres	com.fasterxml.jackson.core:jackson-databind	packaging	caniuse-lite
8	library/ubuntu	com.fasterxml.jackson. core:jackson-annotations	typing-extensions	acorn
9	library/openjdk	org.apache.commons:commons-compress	wheel	debug
10	library/debian	org.apache.commons:commons-lang3	PyYAML	@babel/parser
11	grafana/grafana	org.codehaus.plexus:plexus-utils	python-dateutil	strip-ansi
12	library/golang	junit:junit	numpy	browserslist
13	library/hello-world	org.apache.httpcomponents:httpcore	click	@babel/types
14	library/maven	org.apache.httpcomponents:httpclient	MarkupSafe	tslib
15	library/docker	com.google.code.findbugs:jsr305	pytz	resolve
16	library/eclipse-temurin	com.google.errorprone:error_prone_annotations	cryptography	commander
17	curlimages/curl	commons-logging:commons-logging	cffi	qs
18	library/mongo	net.bytebuddy:byte-buddy	importlib-metadata	@babel/code-frame
19	library/centos	org.objenesis:objenesis	zipp	@babel/generator
20	library/amazoncorretto	org.apache.maven:maven-artifact	attrs	chalk

Figure 4. Top 20 downloaded packages for Docker, Maven, PyPI, npm into JFrog Cloud (SaaS) (JFrog database, 2024)

Many public registries offer download metrics for the packages contained within them, but these metrics can be misleading for various reasons, including the fact that they are impacted by things such as clients fetching the package whenever a build is run. Instead, our research deduces the libraries that are actually used based on what is being requested into environments in JFrog SaaS, which is used by thousands of customer accounts.

For Docker, it is no surprise that the top 20 images include the most popular operating systems and top development languages, presumably used as parent images. It is encouraging to see that all but one are either Docker Official Images or contributed by a Verified Publisher, indicating that care is taken to ensure these images are regularly maintained. Notably, the official Docker hello-world image is among this top group, possibly indicating a healthy collection of demos, proof-of-concepts, and developers learning to use Docker as this containerization becomes ubiquitous in modern software delivery.

In regard to Maven, PyPI and npm packages, there were no surprises in the top 20 used by organizations. That said, it is unclear whether these packages are being pulled in directly, chosen explicitly by software developers, or getting pulled in as dependencies or even transitive dependencies (aka dependencies of dependencies).

For example, Apache Commons Compress holds a seat at #9 in the popularity data. If you take a closer look at this library, you will find that it has direct dependencies on Apache Commons IO, Apache Commons Codec, ASM, and Apache Commons Lang – seats 2, 3, 4, and 10 respectively. This highlights the criticality of maintaining a current inventory of software artifacts included in an application, often in the form of an SBOM, in order to evaluate each individual ingredient and to have a better understanding of the blast radius should a specific component become vulnerable, compromised, or simply go missing in your software supply chain.

Pace at which new OSS packages are being injected into an organization

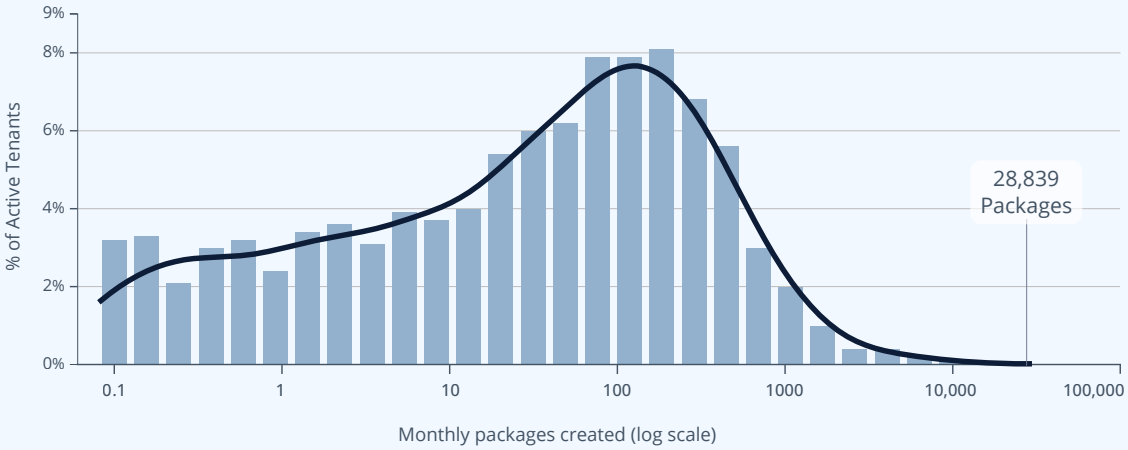


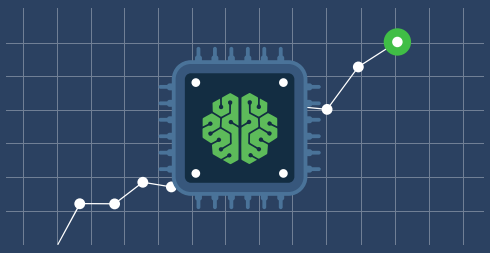
Figure 5. Distribution of new packages created monthly for active tenants in 2024 (JFrog database, 2024)

In 2024, organizations using JFrog Cloud, JFrog's cloud-native SaaS offering, brought a total of over seven million new packages into their software supply chain.

For the average organization, that is around 2,000 packages throughout the year – however, this number is buoyed by a few very heavy users. The largest single organization brought in 346,000 new packages over the course of the year, while the median organization brought in a much more manageable 231 packages.

If you exclude organizations that did not bring in any packages, the median number of packages jumps to 458, or 38 new packages per month. Based on the data, this number is likely the best representation for a typical organization. Even a pace of just over one new package per day can create significant challenges for organizations in regards to how they consider and manage the security, operational risk, and license compliance of what's being brought into their environments.

Key takeaways



AI explosion

We are witnessing an exponential growth in the availability of AI/ML components, with more community and corporate players getting involved by contributing to the ecosystem (e.g. Nvidia releasing NIM and NVLM). Organizations are moving quickly to begin adding AI services into their products, made evident by the 500+ Hugging Face repositories created by JFrog users today. It is important to have well-defined policies and strategies around how you consume and secure open-source models and datasets – a topic we will explore further on in this report.

Safeguarding applications and their development is top of mind

The U.S. government and other global political entities have been pushing for the use of safer development languages and frameworks. JFrog is now also starting to see the rise in use of Rust/Cargo in our own data, indicating that organizations may be rearchitecting applications or starting new projects from a more security-first footing. Additionally, the popularity of OCI can likely be explained, in part, by organizations' growing concerns about favored open-source technologies going private, business source, or now requiring a license.

Risk times a factor of 10

With two-thirds of organizations using 7 or more languages and nearly half using 10 or more, the risk for organizations exponentially increases since they now need to ensure that they have a consistent pipeline for multiple different languages, multiple different teams, and multiple different sources of threats. Each ecosystem has its own distinct vulnerabilities, malicious actors, and unique structures that must be taken into account during development to ensure the applications delivered to production are secure.

A package a day? Keep attackers at bay

Fast moving organizations are bringing in one or more new packages and versions a day, necessitating automated and improved processes for ensuring the security of these components brought into their software supply chain. As organizations continually look to improve velocity and empower developers and security teams to find novel solutions to business challenges, the pace of packages coming into organizations should only continue to rise.

The Accelerating Risk in Your Software Supply Chain

Organizations are in a race against bad actors, and must contend with a collection of key factors, which show no signs of slowing down:

- CVEs
- Malicious packages
- Open source licensing risk
- Operational risks (poorly managed packages, EoL, etc.)
- Secrets exposures
- Misconfigurations / human error

Overall, analysis shows that the tools Developers and Security professionals currently use are helping in some cases and hurting in others. For example, AI code assistants, if not used appropriately, can have a potentially negative impact, particularly in poorly or improperly configured functions.

Any YoY comparisons of Common Vulnerability Scoring System (CVSS) scores and Common Weakness Enumeration (CWE) information presented in this section will be skewed this year due to the months-long period when the National Vulnerability Database (NVD) was unable to analyze and assign properties to newly discovered Common Vulnerabilities and Exposures (CVEs), and the subsequent backlog that was created.

This NVD backlog is a tale of caution, highlighting an ever-persistent issue in our industry. As the number of libraries, and therefore CVEs, continues to grow, organizations need to consider the best way to manage this increased risk in a sustainable way. Further, with the current U.S. politics at play, the integrity and future viability of NVD and the National Institute of Standards and Technology (NIST) is not guaranteed.

Vulnerabilities found in a given technology or package type

In 2024, security researchers around the world disclosed nearly 33,000 new CVEs, a 27% increase from 2023, which continues the pattern of annual growth in CVEs discovered. While this is not surprising given the ever-increasing number of new open-source packages, the pace of CVE growth (27% YoY) is surpassing the growth rate of packages (24.5% YoY), which is an indicator that should be taken seriously.

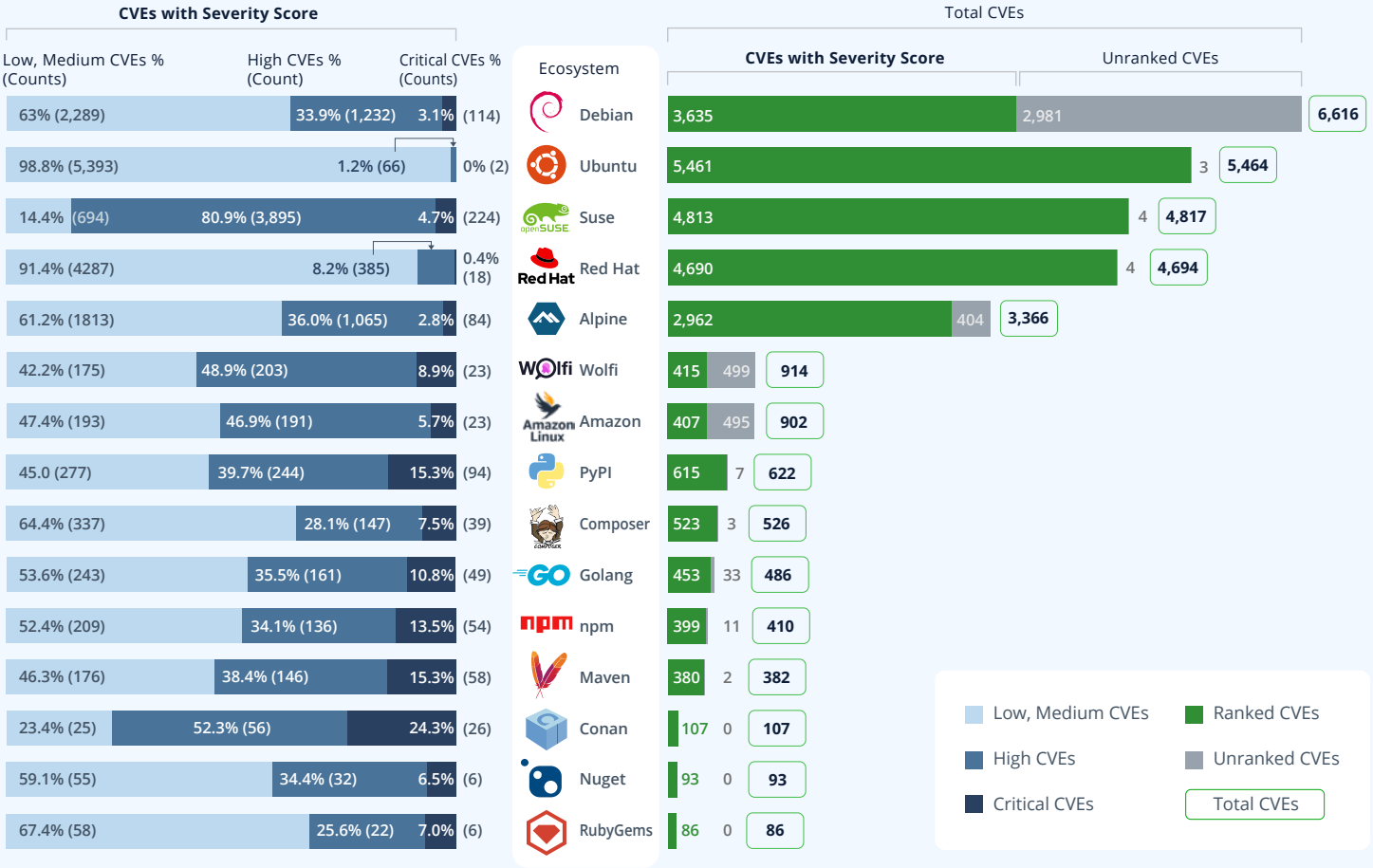


Figure 6.1. Number of discovered CVEs per package type in 2024

The first trend worth noting is the high growth of Debian CVEs YoY. An increase in Debian CVEs is not surprising given the 4x growth rate in packages contributed to the ecosystem in 2024. Fortunately, there are not a lot of critical and high CVE risks in the 2024 Debian CVEs.

Similar to 2023 data, Maven, npm, PyPI and Conan (a new addition this year) represent the highest percentage of critical CVEs, even as the total number of CVEs decreased for Maven and npm YoY. If we look at an end-of-year snapshot of the entire database, however, the persistent risk present highlights the significant level of risk in npm, Maven, and PyPI to a somewhat lesser extent.

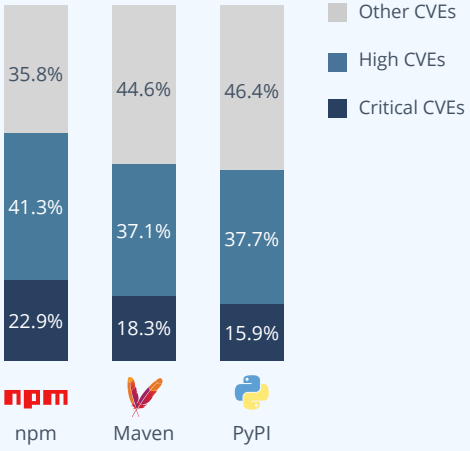


Figure 6.2. % Critical, High CVEs in popular ecosystems as of EoY 2024

Total removed and deprecated packages

* No data collected for the year

Figure 7. Total removed and deprecated packages (JFrog database, 2024)

Packages are not only added to technology ecosystems, but are sometimes deleted as well. The most stand-out piece of data from this set is the number of deleted Composer packages in 2024 vs. 2023. A manual review by the JFrog Security Research team found the following possible explanations:

- Most deleted packages have their GitHub source code repository as 'not available', meaning they were deleted by the author or made private.
- In one instance discovered, the deleted package was only renamed, so it could be that `packagist` marks the renaming event as 'deleted'.
- Some packages are deleted and marked as 'abandoned', although it is unclear what the criteria for an abandoned package is.
- It appears as though `packagist` likely ran a new automation in 2024 that deleted packages with invalid GitHub repositories.

Our data sources vary in their reporting of deleted packages; some provide this information while others do not. For certain ecosystems, we parse all available data and compare it over time, but this method does not account for packages deleted before our first data collection. In other cases, we receive periodic updates that include information since our last run, and only some ecosystems report deletions during these updates. Thus, we do not always have complete information on deleted packages.

Most common types of vulnerabilities

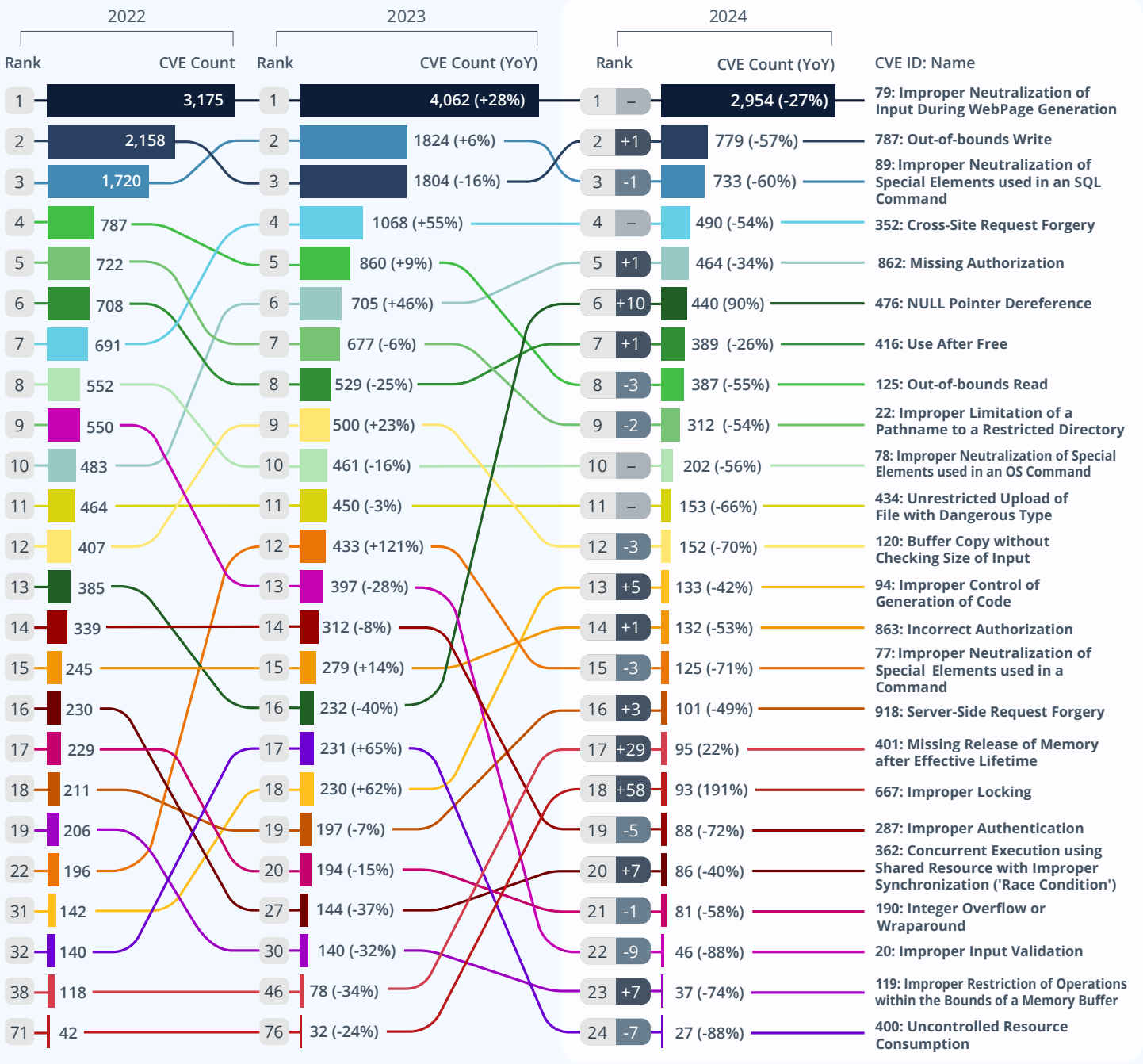


Figure 8. Popular vulnerabilities that were disclosed in 2024 in comparison to 2023, 2022, and 2021

243 unique CWEs IDs were assigned to CVEs in 2024, and the top three remain consistent YoY: Cross-site Scripting, Out-of-bounds Write, and SQL Injection. However, there were three new entrants into the top 20 most popular vulnerabilities, which each experienced unusually high jumps in growth:

401: Missing Release of Memory after Effective Lifetime

2024 #17 ↑ 2023 #46

362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')

2024 #20 ↑ 2023 #28

119: Improper Restriction of Operations within the Bounds of a Memory Buffer.

2024 #23 ↑ 2023 #30

To address the three most common CWEs (Cross-site Scripting, Out-of-bounds Write, and SQL Injection), which can be detected by SAST tools, we recommend teams scan their source code with automated SAST tools in order to prevent new vulnerabilities of these types. In addition, Out-of-bounds Write is an issue unique to low-level (i.e. memory-unsafe) programming languages such as C/C++. These can be prevented by moving to high-level languages as suggested by the U.S. government.

It is important to note that YoY trends for CWE types are prone to be affected by random factors and one-off events. For example, the backlog at NVD almost certainly affects the representation of prevalence of CWEs in 2024. Once all CVEs are properly cataloged, these numbers will likely change. Examining a broader timeframe of 20 years, for example, would reveal more meaningful trends as the popularity of low-level languages vs. high-level languages can affect the number of memory corruption vulnerabilities vs. high-level / web issues. The fluctuation in popularity of specific technologies, each prone to certain types of CWEs, can affect these trends as well.

Common vulnerability impacts for high profile CVEs 2024

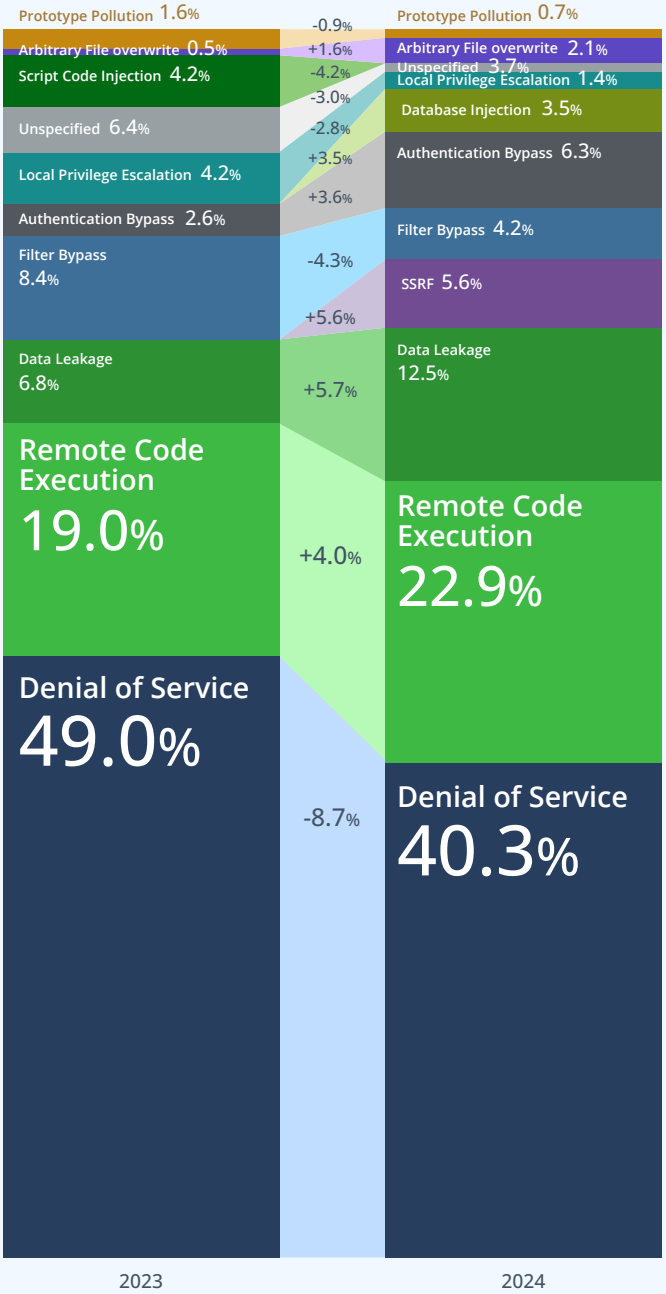


Figure 9. Common vulnerability impacts for high profile CVEs 2023 and 2024

This year, the JFrog Security Research team analyzed just over 140 High-Profile CVEs (HPCVE) based on their relevance and potential impact to JFrog customers. Denial of Service remained the top potential impact of exposure (58). Remote Code Execution (33) remained the second most common YoY, but increased from 18.9% to 22.9%. It is concerning to see Remote Code Execution increasing as a total percentage of HPCVEs due to the potentially devastating control it can give hackers.

Data Leakage (18) remained number three, but also jumped in total percentage. There were also increases in Authentication Bypass and SSRF, which is newly identified in this year's research. On the other hand, we saw decreases in Filter Bypass vulnerabilities.

The JFrog Security Research team considers several factors when prioritizing the CVEs for research. The team focuses on the relevant technologies for JFrog clients and prioritizes mostly the "High" and "Critical" severity issues (meaning CVSS score >= 7.5), but also by using machine-learning-based severity prediction when a CVSS score is not available. The team also prioritizes any vulnerabilities that are exploited in the wild or have high media profiles, even if they received a Medium or Low public severity rating.

Severity of the vulnerabilities being introduced into your software supply chain

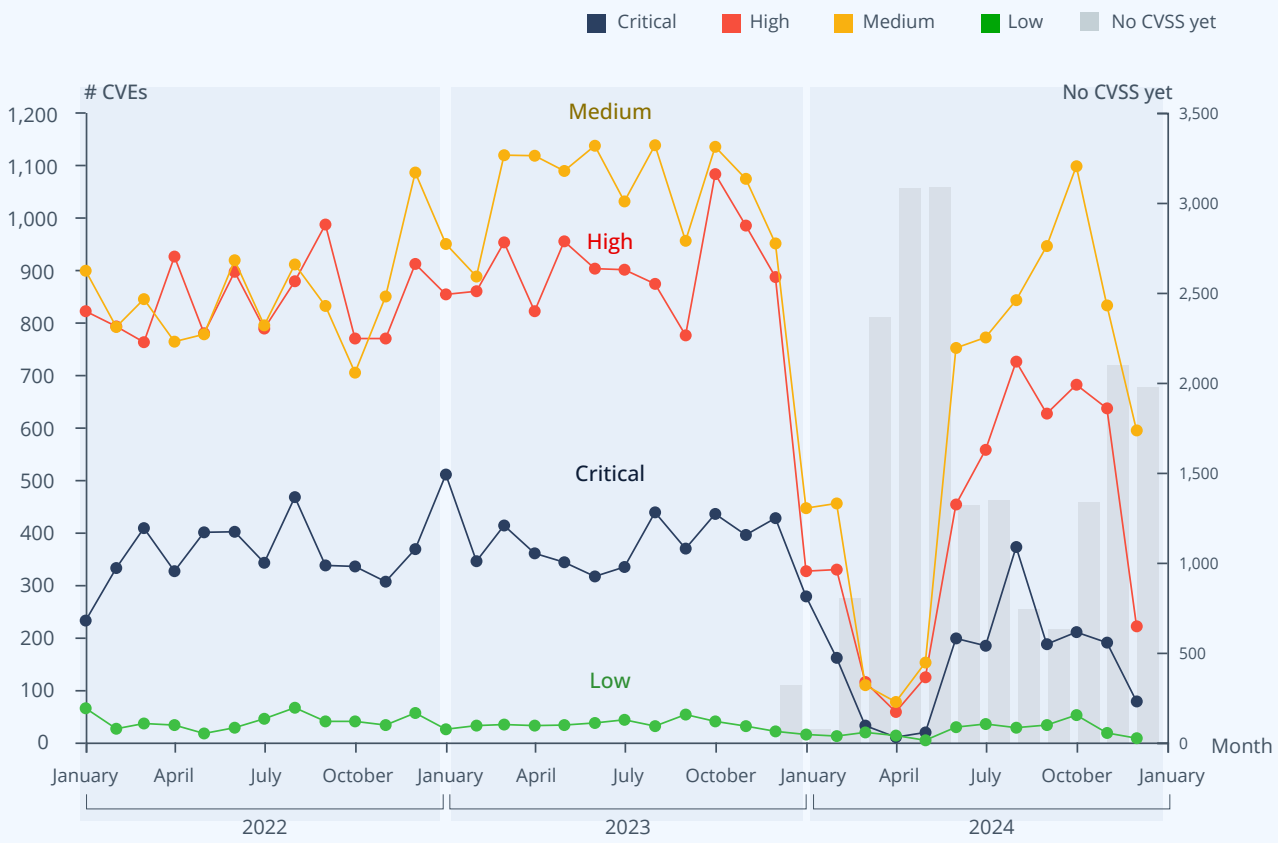


Figure 10.1. CVEs by month and severity within the last 3 years (National Vulnerability Database)

The data shows a significant drop and subsequent rebound of CVSS assignments in the middle of 2024. However, this is misleading. This perceived drop is due to the fact that in February 2024, NVD stopped researching CVEs while restructuring due to cutbacks. To solve this, NVD announced in June 2024 that CISA has been contracted to help their research. Had this disruption not occurred, we most certainly would have seen more predictable CVSS numbers.

To address resource shortages, NVD now mostly outsources CVSS scoring to external vendors known as Authorized Data Publishers (ADPs). Currently, CISA is the first and only data publisher. The JFrog Security Research team is continuously analyzing the scoring patterns of CVEs scored by CISA, and early analysis indicates that **CISA is giving even more exaggerated scores (weighting higher in severity) than NVD.** Looking to the future, there will also be concerns about potential inconsistencies in CVE scoring as additional Authorized Data Providers come online. This is a reality that organizations will have to contend with as they determine how to prioritize security efforts.

Based on the available NVD data, the trend continues to remain consistent with a high number of Medium and High severity CVEs, low amounts of Low severity CVEs, and Critical severity CVE amounts between the Low and High/Medium totals.

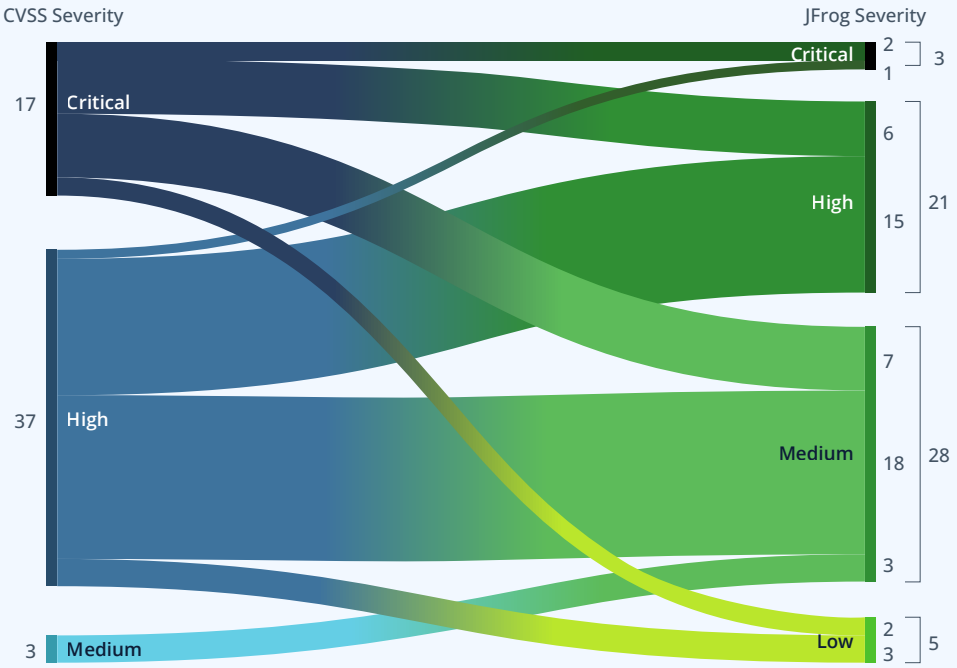


Figure 10.2. CVE severity scores
(Proprietary JFrog Security Research Severity, compared to NVD Severity)

However, not all CVE ratings are what they seem. The JFrog Security Research team regularly evaluates CVEs to determine their actual impact and assigns a JFrog Severity Rating. The JFrog Severity Rating, created by the DevSecOps experts at JFrog, takes into consideration the configuration requirements for vulnerabilities to be exploitable. The CVSS ratings look purely at the severity of a successful exploitation of the vulnerability as opposed to how exploitable the vulnerability is. Sometimes, the configuration or method of exploit is a non-standard setting for that package or dependency, making it very unlikely that the vulnerability will ever be exploitable. This over-weighting of CVE scoring continues to be a concern year-over-year.

The reason why this tendency to score CVEs higher is concerning is because no explanation of the scoring methodology has changed. Since scoring mechanisms are central to determining the initial perception of risk associated with a package, the over-weighting of CVEs increases the potential for false positives.

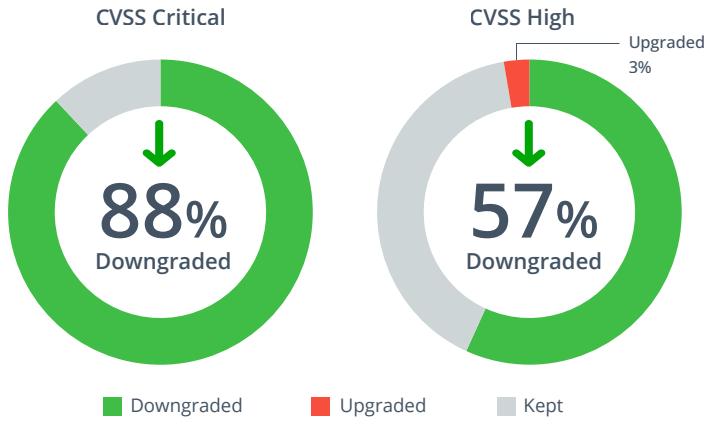


Figure 10.3

Based on the sample of 140 high profile CVEs, JFrog Security Research revealed that 88% of Critical and 57% of High CVE scores were not as severe as the CVSS scoring would have you believe.

Applicability ratings of high profile CVEs

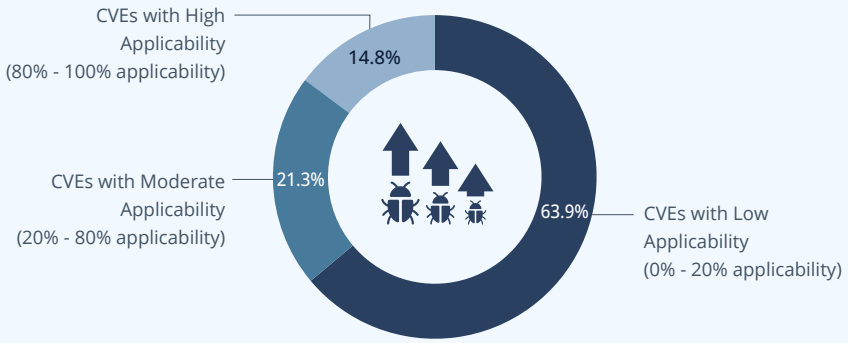


Figure 10.4. Applicability rating of 183 high profile CVEs
(Proprietary JFrog Security Research using CVE and JFrog databases)

Simply assigning severity scores to CVEs is insufficient for assessing the impact of a vulnerability on a specific software product. JFrog Security Research goes beyond just assigning JFrog Severity scores; the team also evaluates the conditions that affect the exploitability of these vulnerabilities. To this end, JFrog creates "applicability" scanners that determine whether the criteria for exploitability are met in a particular software product.

The JFrog Security Research team created applicability scanners for 183 CVEs that were made public in 2024 (CVE-2024-*), focusing on the High and Critical CVEs from the most popular components and technologies among our customers. This chart details how often the CVE was found to be applicable (i.e. able to potentially be exploited by a bad actor) among JFrog customers versus not applicable (i.e. not exploitable). Only 27 CVEs (15%) were found to be highly exploitable, with an applicability rate greater than 80% in artifacts scanned by JFrog Xray in 2024. By contrast, 117 CVEs (64%) were found with a low exploitability rate and an applicability rate of 0%-20%.

CVE-2024-24792 is one of the notable examples with a very high applicability rate (99.6%). This vulnerability can be triggered through a typical usage of the TIFF parsing package in the Go programming language, and frequently appears in applications that manage image uploads and processing. It is applicable in most scenarios because when the library is used to accept TIFF images that may be manipulated by a user, this CVE can be triggered, leading to a panic in the application.

On the other hand, CVE-2024-45490 (related to Expat, an XML parser written in C) is one of the least applicable CVEs, with less than 10% of cases deemed applicable. For an attacker to exploit this vulnerability, they would need to manipulate the "len" parameter passed to the library's API function XML_ParseBuffer(). However, this scenario is highly improbable, as developers typically provide the length of the XML document themselves, often using functions like `stat` or `XML_GetBuffer`.

Applicability types of High Profile CVEs

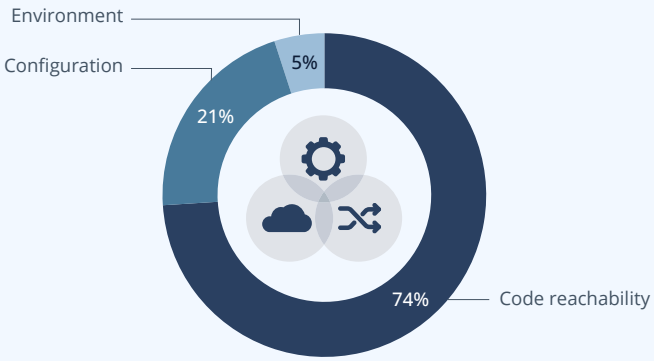


Figure 10.5. Applicability types of CVEs in 2024
(Proprietary JFrog Security Research using CVE and JFrog databases)

The JFrog Security Research team also looked into how these CVEs would be reachable and exploitable in an application. Determining whether a vulnerability is applicable or exploitable requires more than just evaluating the reachability of the vulnerable code through traditional call reachability analysis. It is essential to also examine the configuration settings of applications and libraries, as well as the environmental conditions of the underlying operating system. This holistic approach ensures a comprehensive assessment of potential risks, whereas merely identifying reachable code overlooks critical factors that can significantly influence vulnerability exploitation.

For example, in the famous “Sudoedit bypass”, CVE-2023-22809, the applicability of the vulnerability can only be determined by examining Sudo’s configuration file - “sudoers” - and looking for a specific non-default configuration. There is no way to determine whether the vulnerability is applicable by examining code reachability, as the vulnerable component “sudo” is a standalone utility and not a code library that can be invoked by 1st-party code.

Some malicious packages are worse than others

In our 2024 report, we highlighted the prevalence of malicious packages in the npm ecosystem. A 2024 end-of-year review of the popular package ecosystems confirms that npm maintains its status as the worst offender when it comes to the presence of malicious packages. It is worth mentioning, and perhaps not surprising given the rapid rise in its popularity, that there was a roughly 6.5x further increase in malicious models being uploaded to the Hugging Face ecosystem this year. Here are three noteworthy malicious attacks that merited attention from the JFrog Security Research team:

XZ Utils backdoor

On March 29th, a significant security breach was reported within XZ Utils, a widely used package in major Linux distributions, which contained malicious code allowing unauthorized remote SSH access. The sophisticated backdoor, found in versions 5.6.0 and 5.6.1, modified OpenSSH server routines to enable specific attackers to execute arbitrary payloads before authentication, effectively hijacking victim machines.

Source >

Docker Hub

Recent malware campaigns targeting Docker Hub have resulted in the creation of millions of “imageless” repositories containing malicious metadata instead of container images. Alarmingly, nearly 20% (about three million) of these public repositories hosted harmful content, ranging from spam promoting pirated material to malware and phishing sites, uploaded by automated accounts.

Source >

Hugging Face

Monitoring of AI models has revealed one family of models that execute code upon loading a Pickle file, granting attackers a connectback shell and full control over the compromised machine through a backdoor. This silent infiltration poses significant risks, potentially allowing access to critical systems, leading to large-scale data breaches or corporate espionage, while leaving victims unaware of the compromise.

Source >

Other sources of risk hiding in your code

CISOs and AppSec teams already know that it is important to scrutinize what you bring in from the open-source community. However, it is not the only area to be mindful of for holistic application security.

Misconfiguarations and mistakes — the impact of human error

2024 had its share of security incidents where data was exposed due to data leaks, exposures, and misconfigurations.

April 2024

Home Depot suffered a data breach after a third-party SaaS vendor leaked a subset of employee data exposing the personal information of 10k employees.

Source >

September 2024

Data belonging to ~2000 Fortinet customers stored on an Azure SharePoint site was accessed by a hacker who subsequently leaked it on the internet.

Source >

August 2024

Thousands of Oracle NetSuite customers were inadvertently leaking sensitive data to unauthenticated users through externally facing stores built with NetSuite SuiteCommerce or NetSuite Site Builder.

Source >

September 2024

Significant data exposure potentially affecting millions of users was discovered within Microsoft Power Pages, a low-code SaaS platform, due to misconfigured access controls.

Source >

September 2024

Over 1,000 misconfigured ServiceNow enterprise instances were found exposing Knowledge Base (KB) articles that contained sensitive corporate information to external users and potential threat actors.

Source >

December 2024

The data leak involving Volkswagen's automotive software company, Cariad, stands out as one of the most impactful SaaS misconfigurations of 2024. This incident exposed data collected from approximately 800,000 electric cars, including precise vehicle locations and information that could be linked to drivers' names.

Source >

State of leaked secrets in binary artifacts

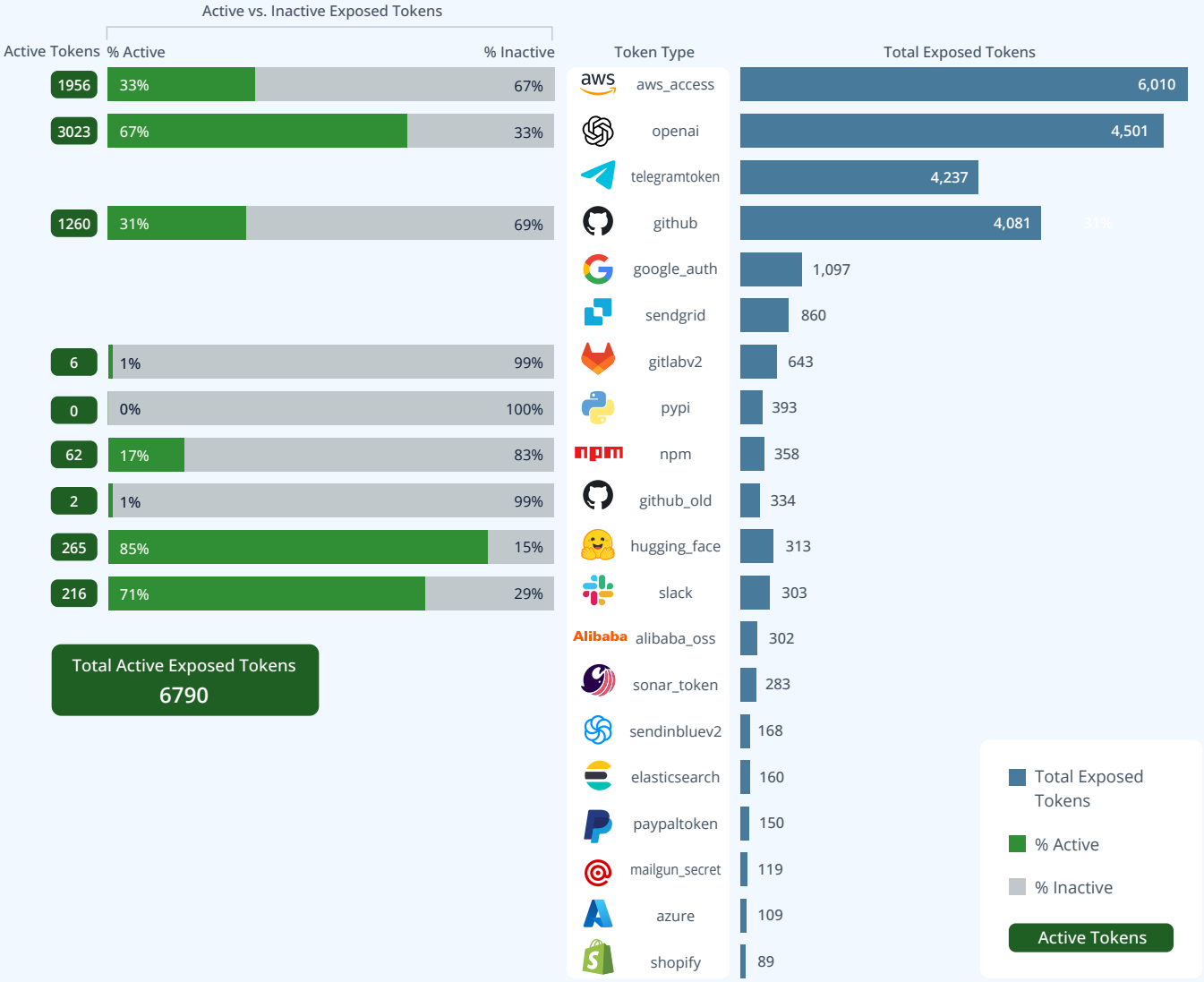


Figure 11.1. Top 20 most exposed token types in 2024

The JFrog Security Research team scanned millions of artifacts in the most common open-source software registries: DockerHub, npm, and PyPI. This year, they also noted where Active Tokens were found (i.e. tokens that could be used at the time of data collection).

From year to year, there were increases across nearly every type of token discovered; there was a 66% increase in total secrets exposed YoY. The most exposed tokens were the same as in our last report, and also saw significant increases YoY: AWS (increased 70%), OpenAI (increased 103%), Telegram (increased 62%), and GitHub (increased 82%). GCP tokens also saw a significant spike, up 86% YoY.

Hugging Face tokens are a new token type that was added to JFrog Security Research team's scanners this year, indicative of the growing popularity of open-source models and data sets. Hugging Face tokens represented the highest percentage of Active Tokens compared to others on the list (~85% active). Notably, the JFrog Security Research team identified 6,790 secrets active at the time of data collection, revealing a huge potential source of access to proprietary systems for bad actors.

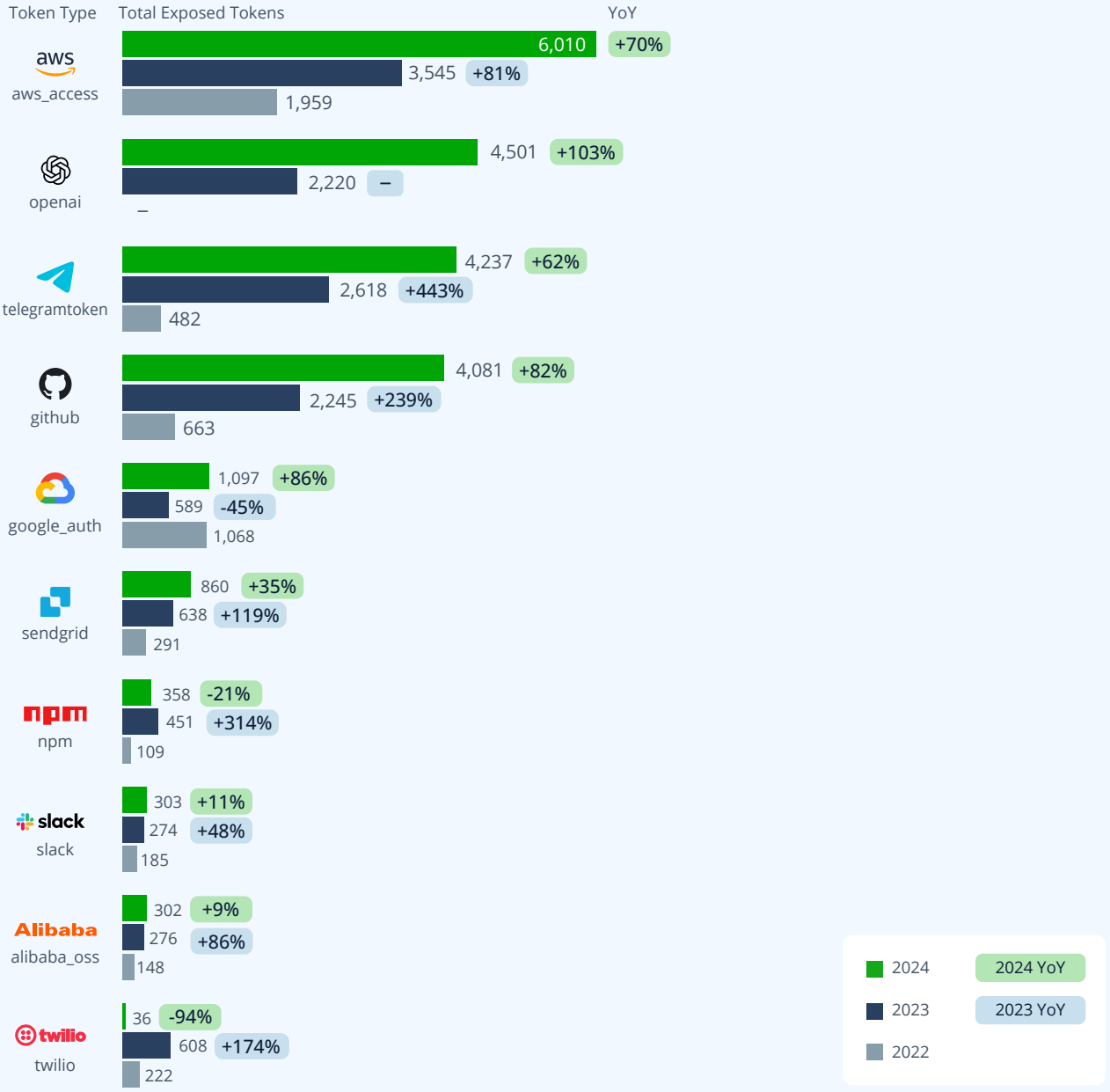
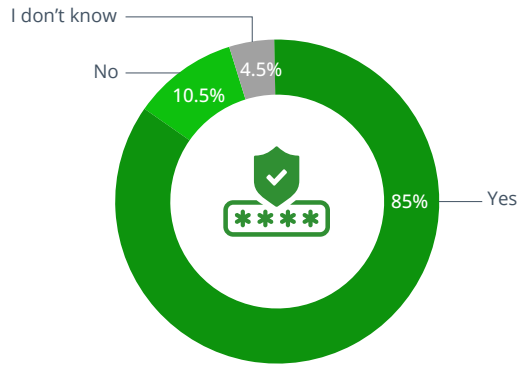


Figure 11.2. YoY comparison of most common exposed tokens

Does your organization have security measures in place to detect secrets left in code bases and/or leaked tokens? (Commissioned survey, 2024)

Organizations are investing specifically to keep secrets out of the hands of bad actors or the public. However, there is a significant 15% of organizations that either do not have measures in place or are not sure. Given the state of leaked secrets and the fact that we saw increases across nearly every type of token discovered, this is a concerningly high portion of the respondent population who are leaving themselves vulnerable.



How severe can a secret leak be?

In June 2024, the JFrog Security Research team discovered and reported a leaked access token with administrator access to Python's, PyPI's and Python Software Foundation's GitHub repositories, which was leaked in a public Docker container hosted on Docker Hub.

As a community service, the JFrog Security Research team continuously scans public repositories such as Docker Hub, npm, and PyPI to identify malicious packages and leaked secrets. The team reports any findings to the relevant maintainers before attackers can take advantage of them. Although JFrog encounters many secrets that are leaked in the same manner, this case was exceptional because it is difficult to overestimate the potential consequences if it had fallen into the wrong hands; one could have supposedly injected malicious code into all PyPI packages and even to the Python language itself.

The JFrog Security Research team identified the leaked secret and promptly reported it to PyPI's security team, who revoked the token within a mere 17 minutes. Although disaster was averted this time, the incident serves as a stark reminder of the catastrophic potential that a single leaked secret can unleash. Given that PyPI is one of the world's major repositories, the repercussions could have been far-reaching, affecting countless users and projects. If such a breach can occur in Python/PyPI, which is a highly maintained and widely used infrastructure, it underscores the vulnerability that exists across all platforms and languages, and highlights that this threat could strike anyone at any time.

Key takeaways

The need for data redundancies

Organizations and security tools that rely solely on NVD vulnerability data are at risk of missing critical CVE information due to the significant backlog delays experienced by the NVD over the past year. These delays mean that newly disclosed vulnerabilities and their potential impacts may not be promptly included in the database, leaving organizations unknowingly exposed to emerging threats. To mitigate this risk, it's essential to supplement NVD data with additional sources, such as vendor advisories and threat intelligence feeds, to ensure timely awareness of critical vulnerabilities. Organizations should evaluate any security scanning tool's data sources to ensure broad coverage and redundancy.

Applicability, impact, prioritization

With an ever-increasing volume of CVEs to contend with, security and development teams could become paralyzed in an effort to triage every single vulnerability. Understanding the applicability, attack vector, and potential impact of a CVE in an application is crucial to focus efforts on remediating vulnerabilities that actually matter. The JFrog Security Research team continues to find over-inflated risk assessments in CVSS scores, which appears to be an accelerating trend with CISA now contributing to enriching CVE records as the first Authorized Data Publishers.

Exposed secrets can happen to anyone

Organizations must maintain vigilance in safeguarding against exposed secrets and work to extend protections to developers working on personal and community projects. Even if a developer's system becomes compromised from a personal project, the impact can spread to corporate systems as well. The implications of someone finding an exposed secret or leaked token could be extremely severe. In the case of the JFrog-discovered Python/PyPI secret, the holder of such a token would have had administrator access to all of Python's, PyPI's and Python Software Foundation's repositories, potentially making it possible to carry out an extremely large-scale software supply chain attack. If it can happen to Python/PyPI, it can happen to anyone.

Malicious actor sophistication

Bad actors are becoming more creative and resourceful in their efforts to infiltrate the software supply chain. In the case of the XZ Utils backdoor, the attacker built up a credible reputation as an OSS developer over the span of multiple years and used highly obfuscated code in order to evade detection by code reviews. Other actors are exploiting AI tools by identifying instances where AI code assistants recommend "hallucinated" libraries, then quickly creating that library containing malicious code. Organizations must maintain vigilance, even with well regarded open-source projects, and set operational risk policies to prevent "overnight" libraries from accidentally being pulled into their supply chain.

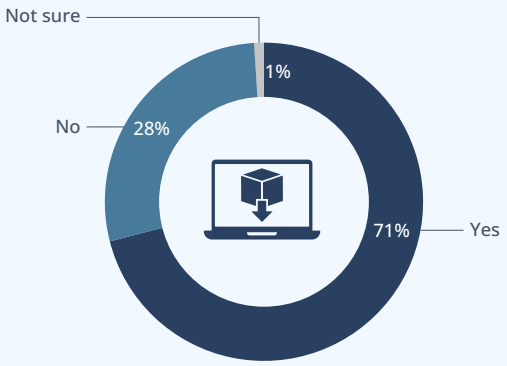
How Organizations are Applying Security Efforts Today

This year, we polled 1,402 Security, DevOps, and Engineering professionals expanded our survey questions, and incorporated findings from other JFrog-sponsored research reports to capture a more holistic view of how teams are managing application risk across the software development lifecycle (SDLC). While we found that most teams have security frameworks and tools in place, we were surprised to discover the prevalence of some risky activities such as downloading third-party packages or libraries directly from the internet.

Sourcing restrictions

One of the best things organizations can do to address risk in their software supply chain is to prevent risk from entering at all. This goes further than "shifting left", which typically means infusing security practices in the development phase, and requires going "left-of-left" by blocking risk before it even has a chance to enter the software supply chain in the first place.

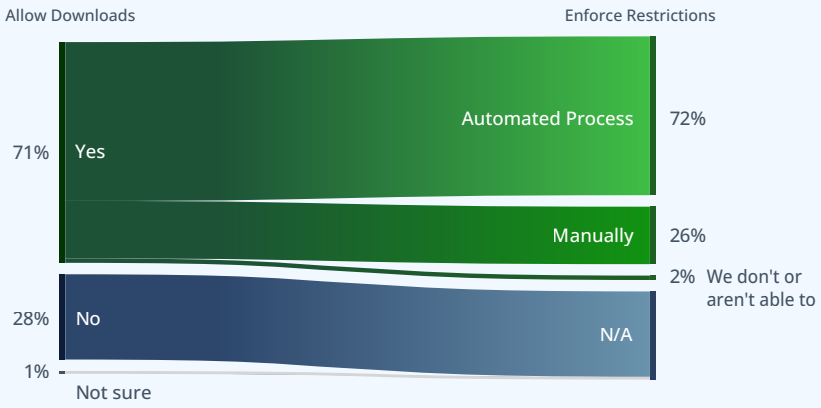
Does your organization allow developers to download packages, or other software components directly from public registries or other sources from the internet? (Commissioned survey, 2024)



An alarming 71% of organizations allow their developers to download software components from the internet. Best practice is to restrict developers from downloading packages or libraries directly from the internet because the risk is simply too great, with the potential to expose an entire organization to attacks through a single developer's machine. Traceability is also compromised, because there is no way to know what a developer is downloading if you allow them to download directly from the internet.

However, the fact that this many organizations are allowing it indicates a need. This is where an artifact management solution that can proxy upstream public registries can be particularly useful. An artifact repository with proxy capabilities acts as a central control point for every component that enters the software supply chain, enabling teams to track and secure them. With this type of solution, organizations can safely mitigate the risk associated with this type of activity, and prevent potential damage of incalculable scale.

How does your organization track and enforce sourcing restrictions for packages or other software components directly from public registries or other sources on the internet? (Commissioned survey, 2024)



More than 1 in 4 (26%) respondents claim their organization manually tracks and enforces sourcing restrictions for packages or other software components directly from public registries or other sources on the internet.

The 72% of respondents indicating they have automated processes is surprisingly high, but this may be influenced by which stage of the SDLC the respondent is referring to. For example, it is possible that developers are manually examining packages and dependencies prior to checking them into their code base, but automated processes may run during the CI/CD cycle or later on during audits of release builds. The challenge with this approach is that it creates rework for the developer and can allow for exposures early in the SDLC, as mentioned previously in this report.

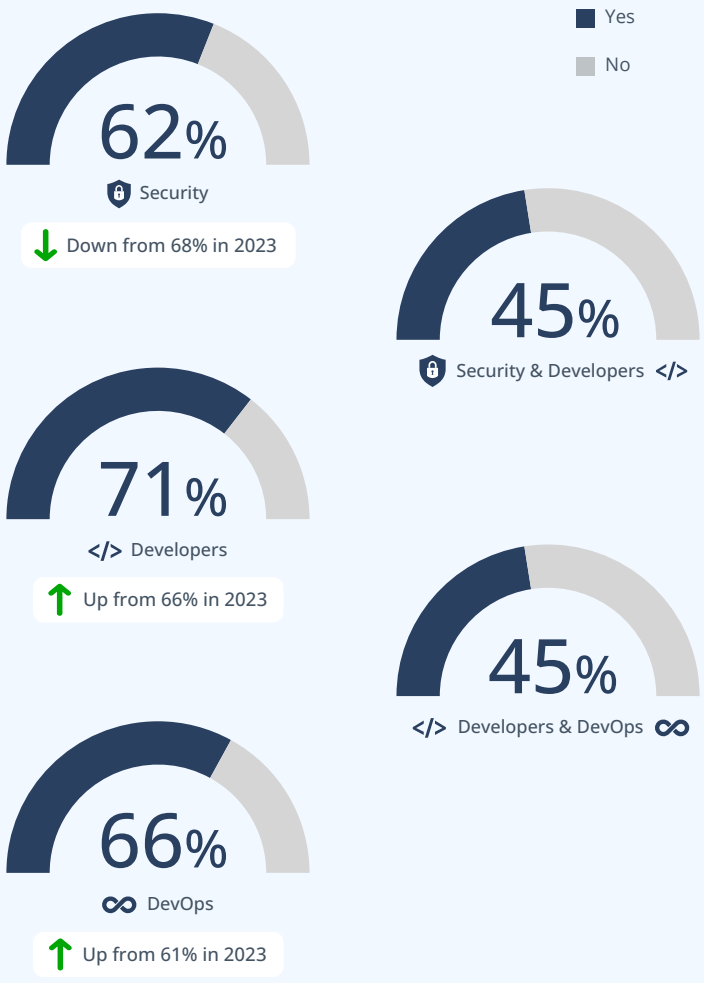
Of the total survey population, 19% indicate that they allow developers to download packages directly from the internet and also leverage **manual** approaches to enforce sourcing restrictions. This requires an extensive amount of difficult, manual work and is not an effective approach to blocking risk.

Those who report that they allow developers to download components directly from the internet and also leverage **automated** processes to track and enforce sourcing restrictions represent 52% of the total respondent population.

Q **Who manages the process of getting the latest version of software packages, libraries, and frameworks: security or developers? Select all that apply.**
(Commissioned survey, 2024)

When compared to previous years, it seems as though developers are now taking a more active role in managing the latest packages, but security professionals still hold responsibility, most likely when it comes to reviewing and approving packages for use. Organizations typically note a combination of teams responsible for the acquisition process with 'Developers + Security', and 'Developers + DevOps' equally leveraged.

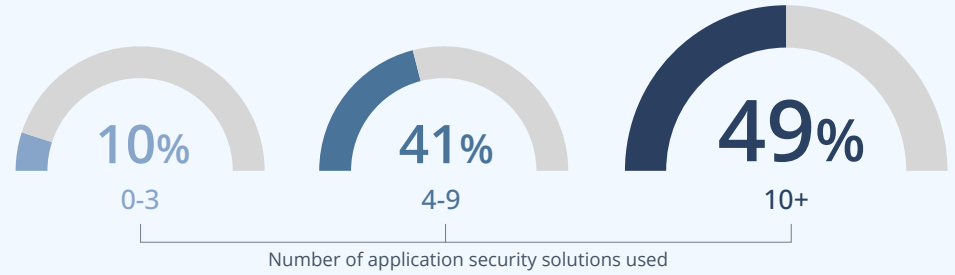
In order to increase velocity, organizations need to adopt approaches and solutions that empower developers to bring in the new and latest versions of libraries self-service, and automate the approval of those packages that fall in compliance with security policies. They must also support a waiver management program that can enable the team responsible, whether AppSec or Security, to integrate this waiver program into their overall risk management strategy.

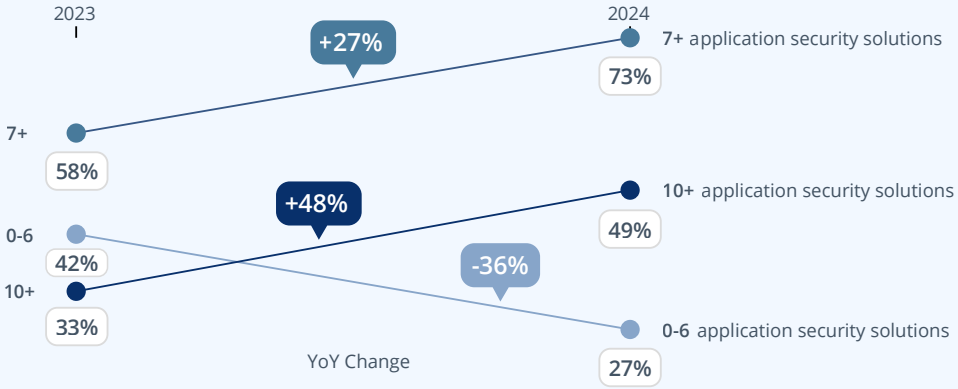


Scanning, scanning, scanning

Although organizations are using more security tools than ever, coverage gaps still exist. Lack of scanning across both code and binaries, and inconsistent scanning across the SDLC and production stand out as common blindspots.

How many application security solutions are you using?
(Commissioned Survey, 2023 & 2024)





Respondents report using more application security solutions in 2024 compared to 2023. By the end of 2024, 73% report using 7 or more application security solutions, juxtaposed to 58% in the prior year.

This is contradictory to what we would expect based on the focus in the market around tool consolidation and what we hear from JFrog customer leadership about their desire to streamline a secure software development process. The data suggests that ASPM (Application Security Posture Management) – a new category of tools that allow organizations to maintain multiple scanning solutions while filtering out duplicate results – are being used by organizations to maintain over-coverage to prevent the risk of missing something. However, ASPM is a "bandaid" for security tool sprawl and not a solution. We do not foresee this growth in total security tools used to continue next year as organizations refocus their consolidation efforts.

Does your organization apply security scans at the code or binary level (or both)? (Commissioned survey, 2024)

Code and binary scanning

43%

Down from 56% in 2023

Code scanning only

29%

Up from 27% in 2023

Binary scanning only

25%

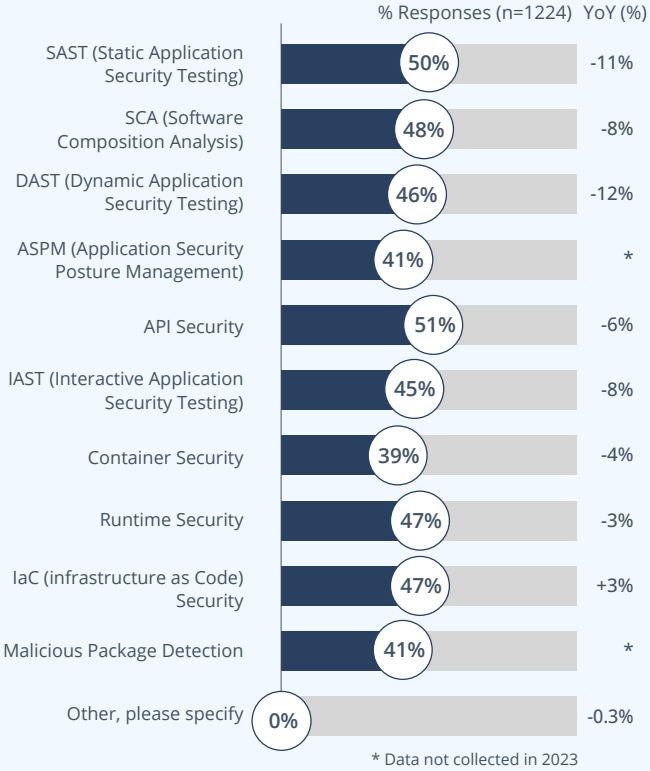
Up from 12% in 2023

This year, applying security scans at only the binary scanning level doubled in popularity: 25% of respondents say they apply security at this level versus just 12% in 2023.

43% of respondents say their organization applies security scans at both the code and binary levels, a slight dip from 56% in 2023. This is a somewhat alarming trend, as organizations should ideally be scanning at both the code and binary level to prevent and catch risk as early as possible. Reasons for taking this approach include the fact that there are certain types of vulnerabilities that only manifest at the binary level.

For example, secrets injected into binaries or memory-corruptions inserted by the compiler can create security issues not present in source code or accidentally left in the builds that end up in production.

What types of application security solutions are you using? (Commissioned survey, 2024)

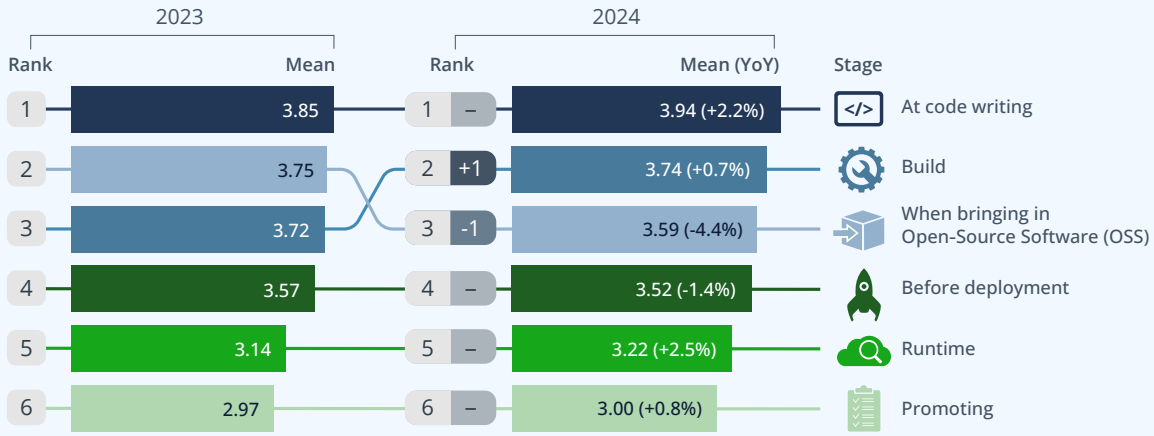


No single tool garners overwhelmingly unanimous usage. API security and SAST are the only two to reach or exceed 50%.

With continued emphasis on shift-left efforts, the high usage of SAST makes sense. It is also not surprising to see organizations investing in API security tools given the prevalence of modern microservice applications where APIs present a potential weak spot for exploitation by bad actors.

Relative to each other, the rates of tool type usage remain consistent YoY, but the overall percentage of respondents who indicate they use a specific tool type has decreased. This is particularly interesting given that we also see the total number of security tools increasing, as shown earlier in this report. This may indicate an overlap in the types of tools used, or that different teams each have their own preferred security tools that end up providing the same functionality as other teams' preferred tools. Organizations should consider a security tool audit to identify where they have overlaps or gaps in security tooling.

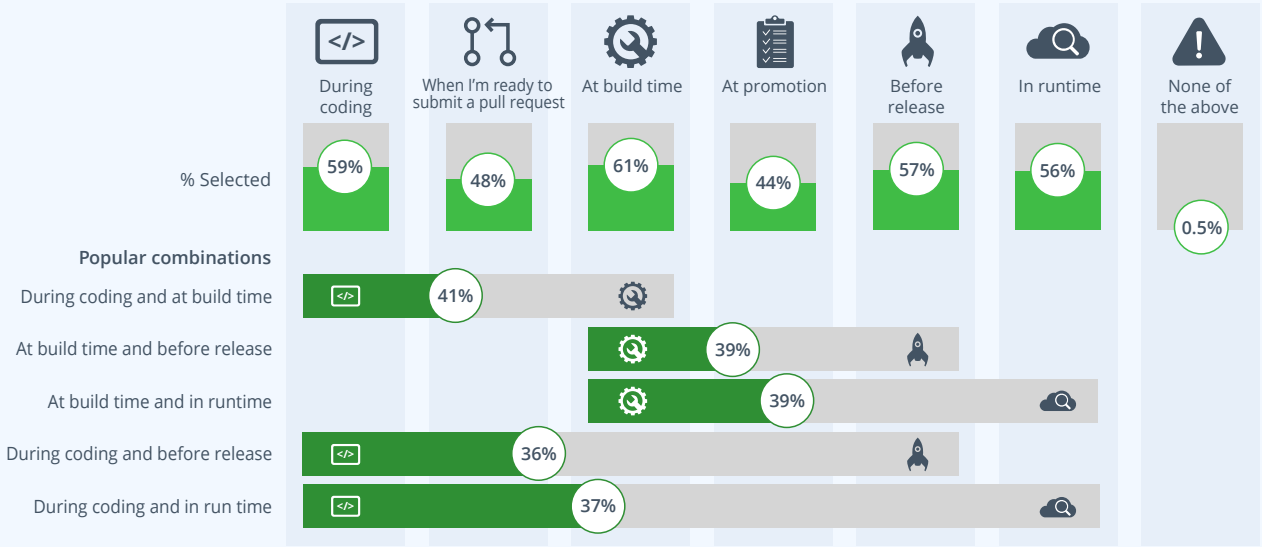
Where do you feel is the best place to apply security in your SDLC? (Commissioned survey, 2024)



When asked to rank where within the software development lifecycle is best to apply security, the top three reported points remain the same YoY:

At code writing	Build	When bringing in open-source software
M=3.94	M=3.74	M=3.59

At what point in development does your organization typically perform security scans? Select all that apply. (Commissioned survey, 2024)



"During coding" remained the most reported point within the SDLC where organizations typically perform security scans, with nearly 3 in 5 respondents claiming that their organization typically performs security scans at this stage.

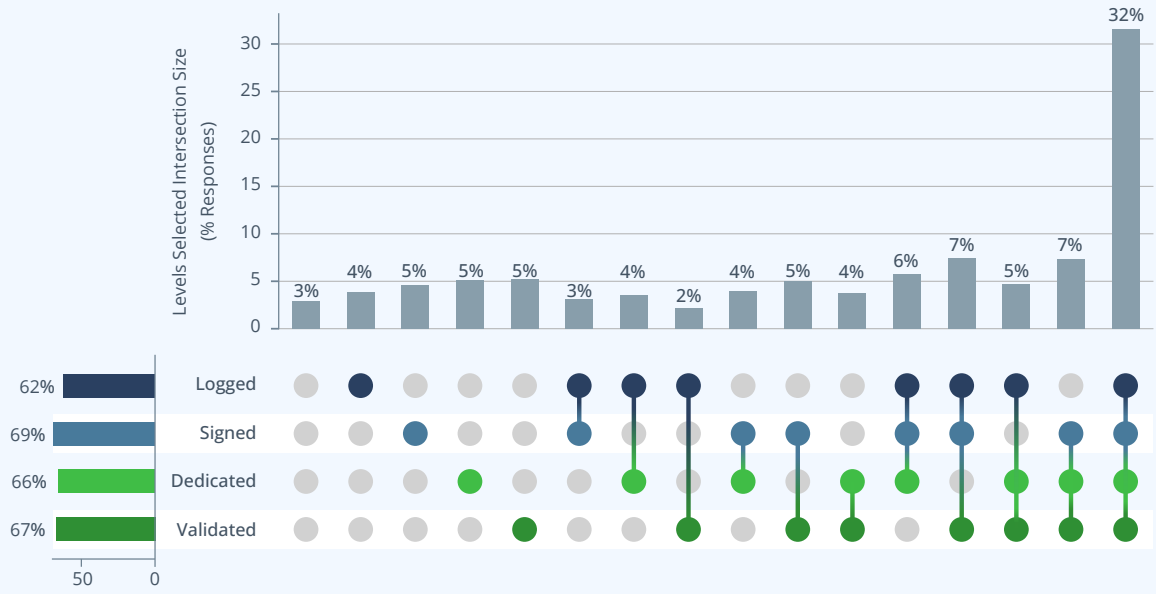
41%	During coding	+	At build time
39%	At build time	+	Before release
30%	At build time	+	In runtime
36%	During coding	+	Before release
37%	During coding	+	In runtime

Establishing visibility and control across application pipelines

It might be obvious that because businesses build applications, they would need to manage risk holistically at the application level. However, while organizations already define and track applications across the SDLC, the level of control and traceability varies significantly. It is essential to strengthen these two elements in order to confidently manage risk and ensure trust in the software released.

Control starts at the sourcing stage, before there is even an "application" in place. The components and libraries that are integrated into the development process fundamentally shape the security posture of the final product. By carefully evaluating and selecting third-party resources, organizations can mitigate risks well before they materialize in the application.

Which of the following security framework levels are implemented in your organization? (Commissioned survey, 2024)

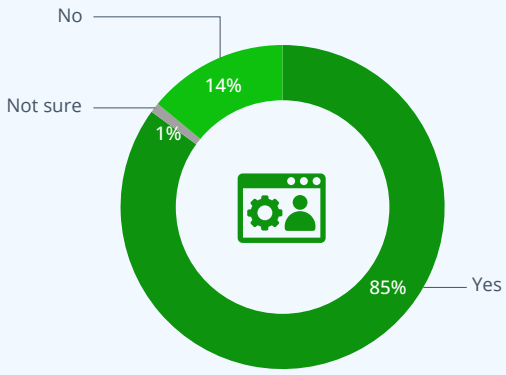


- **Logged** - Packages signatures are validated before publishing
- **Signed** - Packages are built on a dedicated host
- **Dedicated** - Package build data & metadata is signed
- **Validated** - Package build metadata is logged

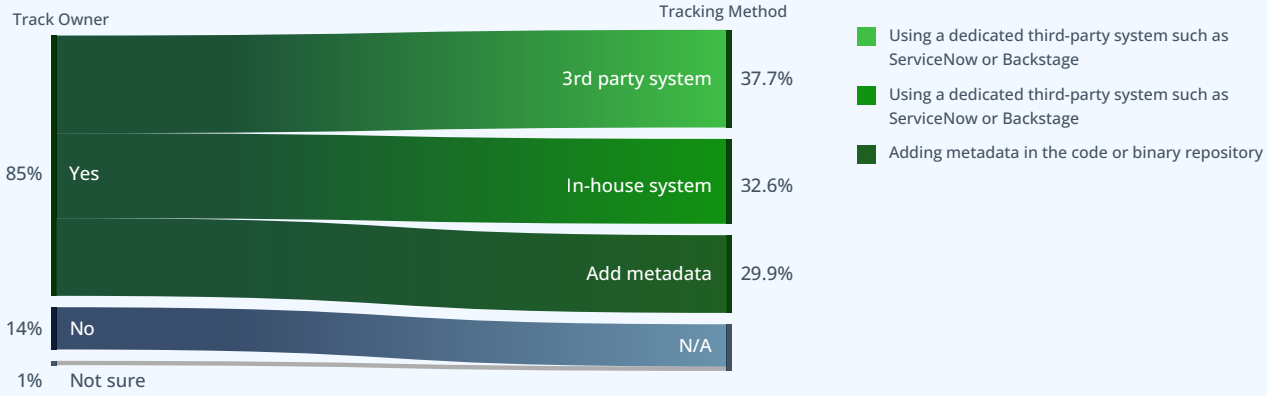
A majority of organizations surveyed have turned to frameworks such as Supply-chain Levels for Software Artifacts (SLSA), to help improve the security and integrity of software supply chains. The data shows significant adoption of at least one SLSA level, while just over a third of respondents are adopting all SLSA levels.

Tracking the owner of each application

For each application you build in your organization, do you keep track of the owner of the application (i.e., team / individuals) (Commissioned survey, 2024)



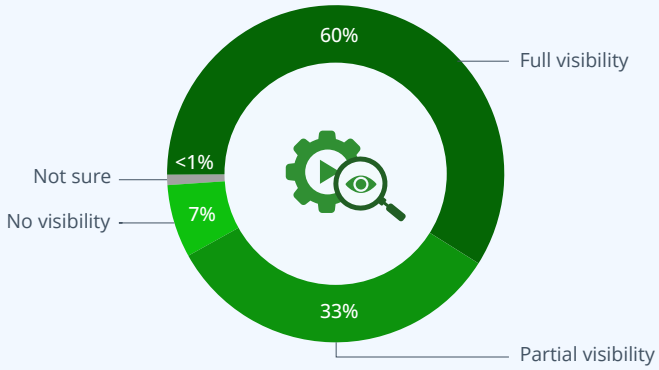
For each application you build in your organization, how do you keep track of the owner of the application? (Commissioned Survey, 2023 & 2024)



Tracking the owners of applications and the various microservices that comprise them is essential for many reasons, including quickly remediating issues, understanding interdependencies across applications, and establishing proper governance and business continuity plans. While most organizations track the owner of the applications they are building, the way they do so varies widely. Responses are nearly evenly split between dedicated third-party systems, dedicated in-house systems, and using metadata in the code or binary repository. Not a single respondent reports using any method other than these three.

Do you have visibility into the provenance of software running in production (i.e., who committed the code for a given service, what tests and validations it went through, where dependencies came from)? (Commissioned Survey, 2023 & 2024)

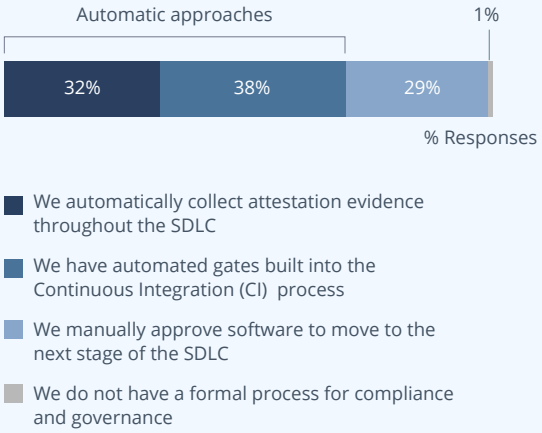
Only 60% of organizations say they have full visibility into the provenance of the software running in production. About a third (33%) have partial visibility, and fortunately, just under 8% have no visibility or aren't sure about their provenance.



Understanding the provenance of software is essential for assuring the quality and security of software released, and is quickly becoming a mandatory requirement of various government regulations. The roughly 8% who report having no visibility should, at a bare minimum, take an inventory of their codebases and any external packages, and ensure they have implemented automated CI/CD that tracks and assigns build versions. Although many take source control for granted, it is possible that a fraction of this ~8% have yet to implement a robust source control solution that tracks code changes during development.

How do you ensure standards for software testing / quality are adhered to during the software creation and release process for compliance and governance purposes? (Commissioned Survey, 2023 & 2024)

The methods used to ensure standards for testing and quality also vary across organizations. The majority (70%) leverage automated approaches, but nearly a third (29%) still require manual approvals to advance across stages of the SDLC.



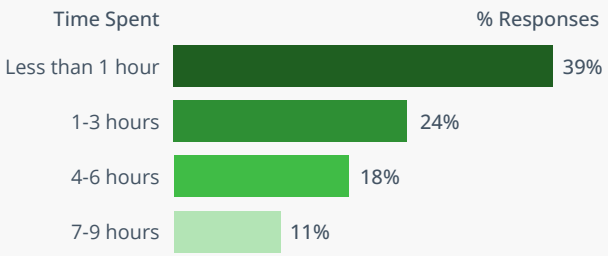
How much time security efforts are costing your organization

In research conducted by IDC and commissioned by JFrog, 60% of professionals say their developer and/or security team typically spends 4 days or more remediating application vulnerabilities in a given month. The cost of this averages out to about $28k per developer per year spent on security related tasks. Not only does this have financial implications, but a negative effect on developer experience (DevEx) as well.

IDC: The Hidden Cost of DevSecOps
Published: September 2024 | IDC #US52537524

Developer time spent outside working hours to address security issues

Developers are spending ~3.6 hours a week outside working hours to address security issues. This creates an environment ripe for burnout.



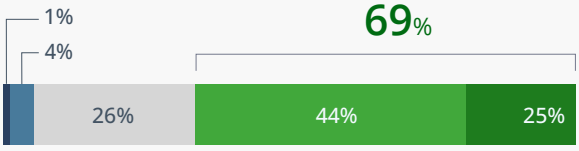
~3.6
hours a week spent per developer outside working hours to address security issues

Money spent on security related activities

The average organization spends $28k per developer per year on security related tasks. While DevSecOps is a business imperative and essential for developing secure applications, inefficient or poorly implemented tools and processes waste developer time and create business costs.

$28K
spent per developer per year on security related tasks

Too much context switching



- Strongly agree
  - I am consistently moving between tools or environments
- Agree
  - I often switch tools or environments
- Undecided
- Disagree
  - I sometimes switch tools or environments
- Strongly disagree
  - I rarely or never leave my tools or environments

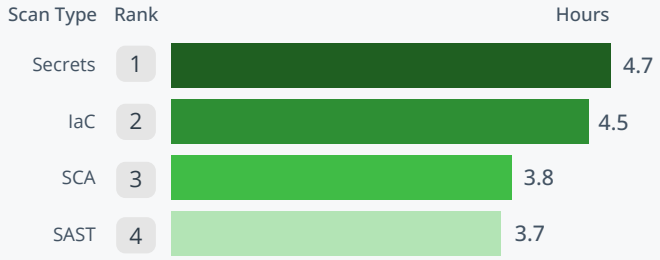
69% of developers agree their security related responsibilities require them to switch contexts frequently. As organizations seek to improve DevEx, they should consider that regularly switching between tools will hurt those efforts and make it less likely that developers will engage in security activities.

69%
of developers agree their security related responsibilities require them to switch contexts frequently

Time spent by types of scan

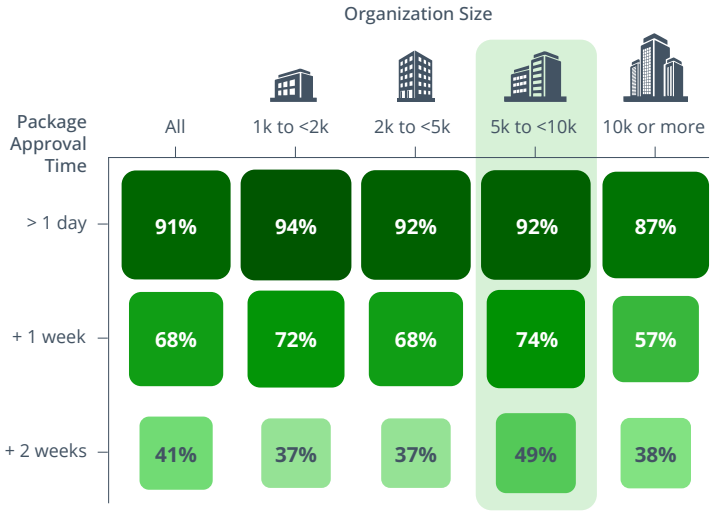
Developers are spending the most amount of time on secrets scanning, indicating either the need for more training on coding practices for handling tokens and secrets or the need for more efficient secret handling tools. It is also crucial to ensure that no secrets are left in code, which is the equivalent of dropping your house keys attached to a key chain that states your address. Oftentimes, the secrets left in code can give attackers unfettered access to critical systems and data.

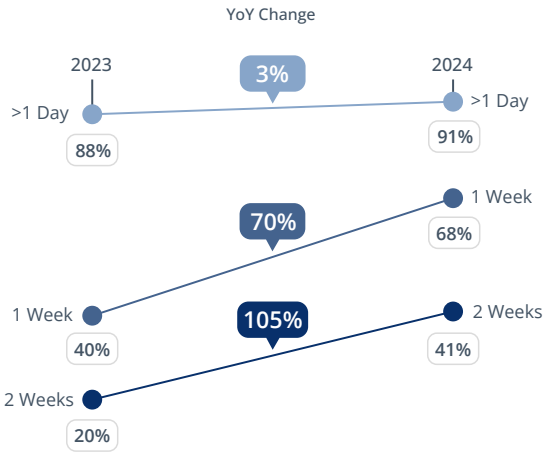
4.7
hours spent on secrets scanning by developers



The IDC study was conducted online in June 2024 and collected responses from 210 U.S. and European developers, development team leaders and managers, and product owners who use DevSecOps. The study sought information on the business impact of developer time on DevSecOps, DevSecOps tools and tasks consuming developer time, the value of developer time on DevSecOps, and the impact of security tasks on developer flow and satisfaction.

How long does it typically take to get approval to use a new package/library?
(Commissioned Survey, 2023 & 2024)





Developers are waiting longer than ever for new packages. Midsize organizations (5k to 10k employees) tend to wait the longest of any size, with 92% waiting more than one day, 74% waiting more than a week, and 49% waiting two weeks or longer. While it is encouraging to see developers more involved in the process, it is clearly still inefficient, likely due to reviews and other manual efforts. More is needed to make the process of bringing in new components truly self-service.

Key takeaways

71%

Missing foundational practices for software supply chain security

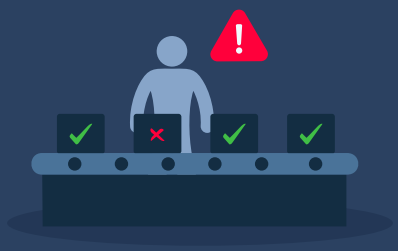
Organizations need to control, or at least have strong visibility into what is coming into their software supply chain via their developers and the dependencies referenced in their applications. Over 71% of organizations allowing developers to download directly from the internet is concerning, and a major violation of software supply chain security best practices. An artifact management solution to proxy public registries should be in place at every organization.

More scanners, more problems?

Organizations appear to be using more security tools than ever, but could this be positively or negatively impacting their security posture? Either way, there are still gaps in coverage and many organizations are not scanning at both the code and binary level, which is problematic.

DevSecOps without compromising DevEx

Security efforts are taking hours of developers' time every week. Organizations can and should be looking for ways to reduce the impact of security efforts on developers without compromising the security status of their applications. Smart prioritization, contextualized results, and automation are key in this domain.



Uplevelling application management

85% of organizations track the owners of applications they build internally, but the way they ensure application standards varies dramatically, with almost a third leveraging manual efforts for promoting their software from one stage to the next. Any manual intervention represents a potential for risk, either intentional or accidental, and reveals an obvious area for improvement by organizations.

The Next Frontier of Risk: AI and Machine Learning Development

Nearly every security tool and an increasing number of developer tools now tout AI-enhanced capabilities to accelerate development and improve the detection and remediation of vulnerabilities. In this section of the report, however, we focus on the **building** of AI tools, rather than their use.

The Artificial Intelligence/Machine Learning (AI/ML) software supply chain is the next frontier of risk for organizations, and it is much further to the left on the maturity curve than traditional software development. In fact, according to research JFrog commissioned with InformationWeek, 79% of firms say security concerns are slowing the use and/or integration of AI/ML features into software.

Trends in AI adoption and DevSecOps

The InformationWeek survey explored how well software developers and cybersecurity teams understand the importance of integrating application security into the software development lifecycle. It also looked at how teams are protecting their organizations against malicious code and avoiding improper use of AI technologies. Key insights from that survey include:

AI Adoption And DevSecOps: Staying Ahead While Staying Secure

Published: September 2024 | InformationWeek & JFrog

Lack of business-wide AI security confidence

79% of firms say security concerns are slowing the use and/or integration of AI/ML features into software

The top three AI security concerns within companies are data exposure through LLM usage, malicious code in AI models, and AI bias

64% of organizations are either not at all confident or only somewhat confident in their ability to comply with new and emerging regulation around AI usage in software

AI supply chain visibility is muddled

49% of firms have no reliable way to control usage of ML models in their apps

Less than a quarter of organizations have a single source of truth for all software components, including AI models

More than two-thirds of organizations have no reliable method for tracking open-source packages in their software containing transitive dependencies to ML models

AI policies are still lacking

58% of companies either have no policy in place or don't know if they have a policy that sets rules for how developers use open-source AI models or components

60% of companies don't have a policy for how developers source or license their training data

Enforcement is even spottier

68% of respondents report they have no way to enforce AI component usage or depend on manual review to do so

59% say they have no mechanism or rely on manual review to enforce policies about training data

The study JFrog commissioned with InformationWeek was conducted online in May 2024 and collected responses from 210 IT and cybersecurity professionals predominantly located in North America. Respondents hailed from companies of all sizes with job titles from executive level to staff. More than 21 vertical industries are represented including consulting, banking and financial services, education, government, technology, healthcare, and manufacturing.

While the InformationWeek study uncovered interesting trends, the remainder of this section dives a bit deeper into how organizations are actually bringing AI services into their applications and governing their usage.

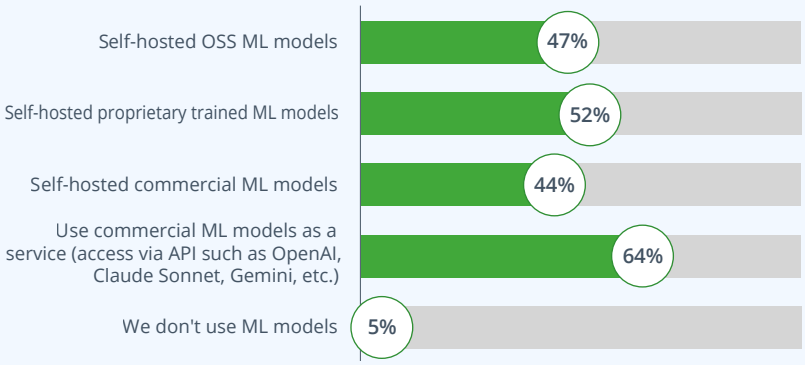
Usage, governance, and scanning of ML model artifacts

What is your primary method for consuming ML models as part of the applications you are developing? (Commissioned survey, 2024)

The ways in which organizations are bringing AI services and applications to life varies, and the research indicates that organizations are using multiple methods at once.

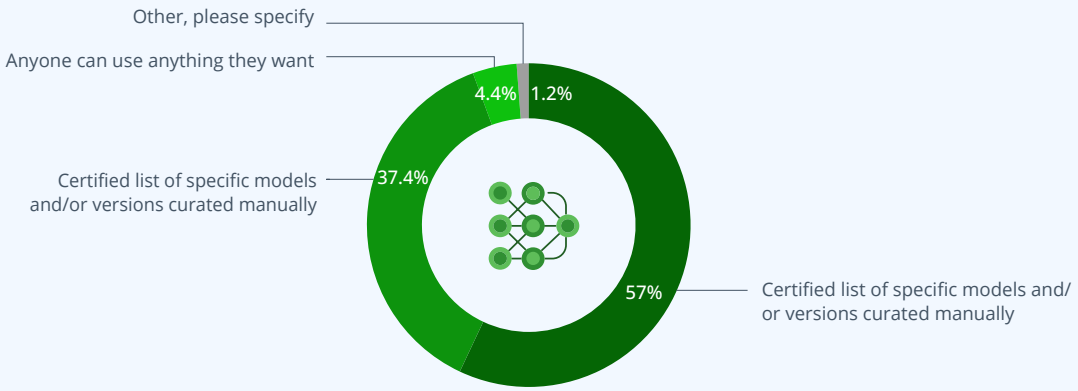
By far the most popular approach is using commercial models accessed via APIs (64%).

This enables organizations to gain access to powerful, general-purpose AI capabilities quickly and without the upfront development or infrastructure costs.



However, we also see organizations investing in self-hosting models, with over half self-hosting proprietary models built for their specific business needs. Nearly 1 in 2 respondents cite self-hosted OSS models as their primary method of consuming machine learning models.

How do you govern ML model artifact usage within your development organization? (Commissioned survey, 2024)



More than 1 in 3 (37%) professionals say they govern ML model artifact usage via a manually curated list of specific models and/or versions.

As revealed in the InformationWeek study, 49% of firms have no reliable way to control the usage of ML models in their applications. This may explain why 4% of survey respondents are willingly taking no action, manual or otherwise, to control what developers are using.

Of the total survey population,

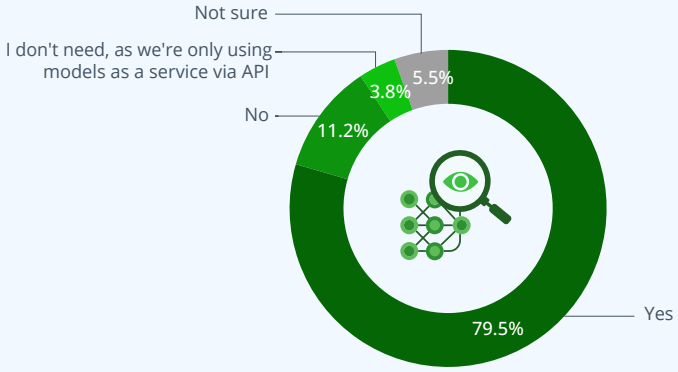
16%

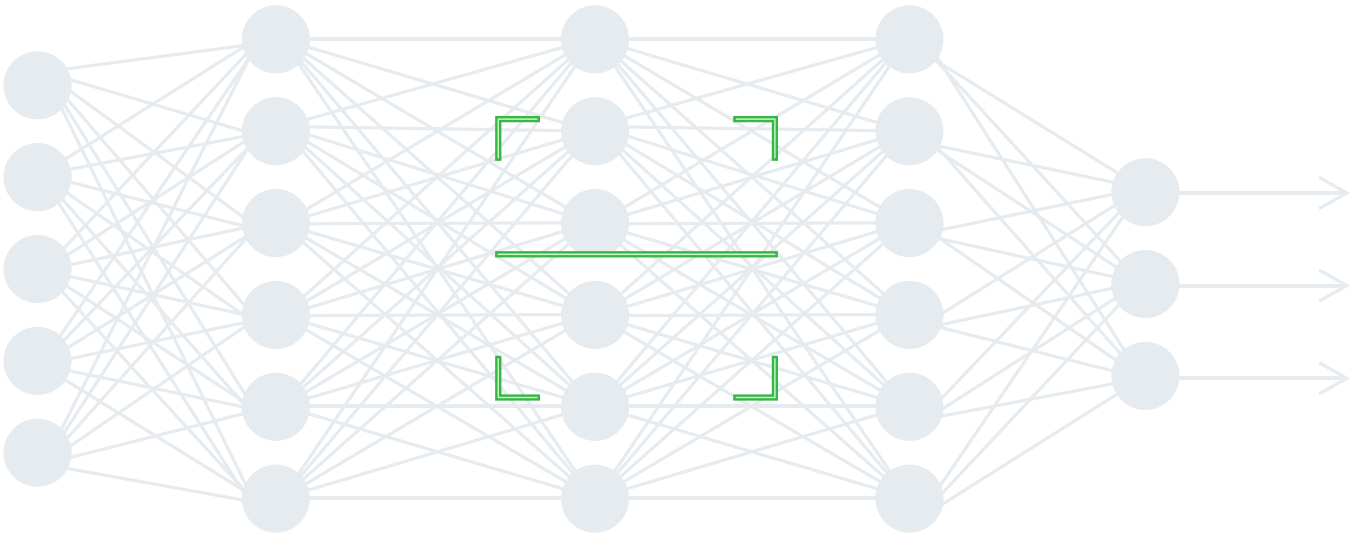
consume self-hosted OSS models and govern model artifact usage manually.

Does your organization have a mechanism in place for scanning ML model artifacts for security vulnerabilities or malicious models? (Commissioned survey, 2024)

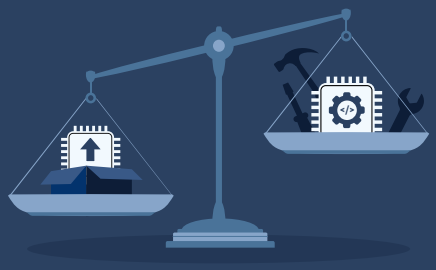
A significant majority (79%) of organizations say they have some mechanism in place for scanning ML model artifacts for security vulnerabilities or malicious code. 11% do not have a mechanism in place.

Fortunately, only 3% of the total survey population report consuming self-hosted OSS models and have no form of scanning mechanism in place to prevent vulnerabilities or malicious models. However, more still needs to be done both at the organization and security industry levels to ensure the right tools and policies are in place to secure AI/ML development.





Key takeaways

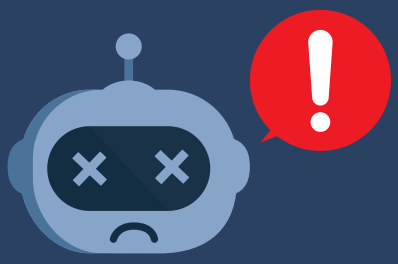


Accelerating AI through commercial models

Turning to commercial models seems to be a popular means to accelerate bringing AI services into business applications. Accessing commercial models via API also saves organizations time and money when it comes to acquiring the tooling, resources, and expertise to build and manage their own in-house models. Furthermore, it may behoove organizations that do not have much experience with AI/ML to entrust the security of their models to providers with more expertise in this domain.

AI supply chain visibility is unclear

Many organizations struggle to establish reliable methods for managing the use of machine learning models within their applications, and they often lack a single source of truth for all software components, including ML models. Additionally, a significant blind spot exists in their ability to effectively track open-source packages that contain transitive dependencies related to ML models. When these critical gaps are present in the ML software development process, it not only becomes more challenging for organizations to manage their AI/ML supply chains efficiently, but also increases the risk of security vulnerabilities.



Overconfidence in AI security

While 79% of organizations report some level of model scanning, the current solution landscape for AI/ML model security is still in its infancy with naive detection techniques. For example, current approaches led to a 96% false positive rate in malicious model identification on Hugging Face, while simultaneously missing threats due to simplistic scanning techniques. While many security tools are trying to capitalize on covering model artifacts, organizations must seriously evaluate the efficacy of security providers' model security solutions.

Methodology

This report incorporates a combination of insights derived from JFrog usage data, CVE analysis results from the JFrog Security Research team, and commissioned third-party survey data. Here is a more detailed look into each source:

JFrog Platform usage data

Technology usage trends highlighted in this report come from an end-of-year snapshot of anonymized usage data of the JFrog Platform for Cloud, representing thousands of customers, hundreds of thousands of repositories, and Petabytes of data.

Package popularity is represented by Action Count (upload/download), total number of artifacts, total number of repositories, and total artifact size for a given package type. Action Count provides a good representation of how often different package types are being called and generated by developers as an indication of actual use in software development.

It is possible that a handful of enterprises could skew these rankings. However, because we also look at artifact actions, we can safely conclude which package type is actively being used as part of the development process.

Analysis by the JFrog Security Research team

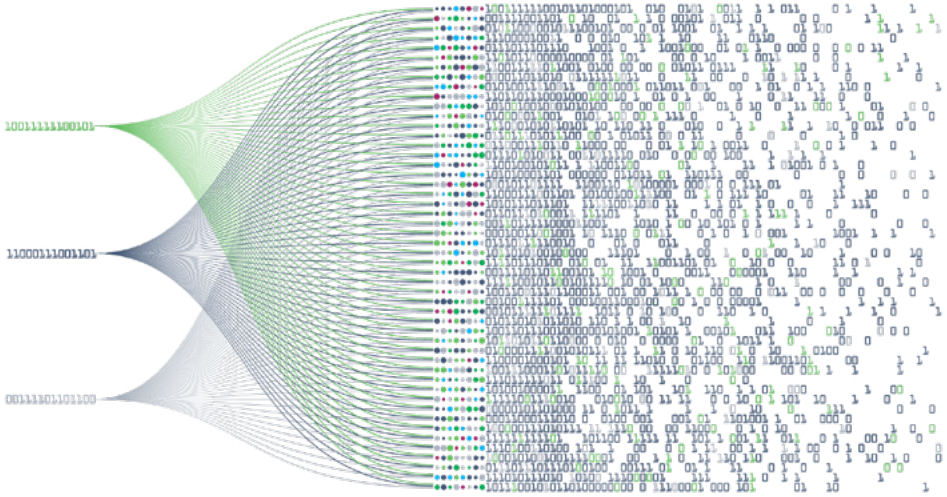
As a designated CNA, the JFrog Security Research team regularly monitors and investigates new vulnerabilities to understand their true severity and publishes this information for the benefit of the community and all JFrog customers.

This report includes data pulled from public sources via the JFrog Catalog service, CVE information pulled from the National Vulnerability Database, and proprietary analysis performed by the JFrog Security Research team on those data sources.

Commissioned survey results

JFrog commissioned Atomik Research to conduct an international online survey of 1,402 respondents working in select industries[1] throughout the United States (n=375), the United Kingdom (n=205) , India (n=206), Germany (n=205), France (n=205) and Israel (n=205). The sample consists of full-time employees who hold specific job functions[2] within their organization's information technology, information systems or technology departments. Moreover, all respondents indicate their employing organization consists of 1,000 or more total employees and confirm the presence of a software development team with at least 50 team members within their organization. All participants had the option to access English, French, German, Hebrew, and Hindi translations of the online questionnaire.

The margin of error for the overall sample is +/- 3 percentage points with a confidence level of 95 percent. Fieldwork took place between November 22 and December 09, 2024. Atomik Research is a creative market research agency.



1. To qualify for participation, all respondents must indicate they are employed at organization that serve the following industries: (a.) aerospace (b.) architecture and engineering (c.) automotive (d.) banking, financial services, insurance & fintech (e.) energy, oil, gas (f.) government or public sector (g.) healthcare and life sciences (h.) hospitality (i.) manufacturing (j.) retail (k.) technology (l.) transportation and logistics (m.) utilities, telecom & power

2. To qualify for participation, all respondents must indicate having job functions of, or similar to, the following: (a.) AI specialist or AI engineer (b.) application security engineer (d.) cybersecurity engineer (e.) data scientist (f.) developer (g.) DevOps architect (h.) DevOps engineer (i.) engineering manager (j.) machine learning specialist or ML engineer (k.) platform engineer (l.) security architect (m.) security researcher (n.) site reliability engineer (o.) software architect (p.) software developer (q.) software engineer (r.) solutions architect in addition to indicating employment in their organization's information technology, information systems, technology departments or IT product development department.

About the JFrog Platform

The JFrog Platform is a highly scalable and open cloud-native solution that integrates with the package technologies and tools in the software supply chain. It provides organizations with full control and traceability as software components flow from developers to all deployment environments, including ML models, edge devices, and production data centers.

This data report contains "forward-looking" statements, as that term is defined under the U.S. federal securities laws, including but not limited to statements regarding the JFrog usage data and the software supply chain.

These forward-looking statements are based on our current assumptions, expectations and beliefs and are subject to substantial risks, uncertainties, assumptions and changes in circumstances that may cause JFrog's actual results, performance or achievements to differ materially from those expressed or implied in any forward-looking statement.

There are a significant number of factors that could cause actual results, performance or achievements, to differ materially from statements made in this press release, including but not limited to risks detailed in our filings with the Securities and Exchange Commission, including in our annual report on Form 10-K for the year ended December 31, 2024, our quarterly reports on Form 10-Q, and other filings and reports that we may file from time to time with the Securities and Exchange Commission. Forward-looking statements represent our beliefs and assumptions only as of the date of this press release. We disclaim any obligation to update forward-looking statements.