ANNUAL REPORT

REVERSINGLABS

THE 2025

Software Supply Chain Security Report

Attacks Grow in Sophistication – Targeting AI, Crypto, Open Source, and Commercial Software

Table of Contents

A Message from Our CEO

Software Supply Chain Risk Is Evolving. Be Prepared.

Mario Vuksan
CEO AND CO-FOUNDER,
REVERSINGLABS

This year's ReversingLabs Software Supply Chain Security Report has a clear message — "Software supply chain risk is evolving" — and a clear moral: "Your organization needs to be ready!"

ReversingLabs (RL's) 2025 report shows that the security of software supply chains lags, even as malicious actors score bigger and bigger wins targeting the commercial and open-source software running in homes, businesses, and powering critical infrastructure. RL's research over the past year identified compromises of open-source libraries and modules that are widely used in both the cryptocurrency and artificial intelligence sectors. We also uncovered widespread and exploitable flaws in commonly used open-source packages and third-party, commercial software binaries.

Unaddressed security risks such as these set the stage for bigger attacks in 2025. The question is: Are we ready to stare them down? The key to answering that question — and getting our response right — is changing the status quo for software security, which lacks incentives for software producers to secure their software and IT assets, and complicates efforts by end-user organizations to assess the risks lurking in the software and services their business relies on.

Incidents like recent campaigns by state actors targeting critical infrastructure and telecommunications networks underscore the urgency of efforts by both public- and private-sector entities to accurately assess the cyber risks facing them and take proper steps to secure their IT assets and data. To do so, however, both private- and public-sector organizations need to acknowledge a shift in threats and attacks, including growing instances of software supply chain compromises.

To help underscore this growing risk, RL researchers explored the attack vectors and methods that are favored by both cybercriminal and nation-state actors. Their work provides valuable insights into the evolving cyber-risk landscape and a useful preview of the kinds of threats and attacks that organizations will face in 2025.

Like any good report, our Software Supply Chain Security Report also provides recommendations on how to best address those risks: leveling up your software supply chain security tools and methods, while also pushing back on suppliers for more transparency about their secure development practices and any risks that may lurk in both the open source and proprietary code that powers their solutions.

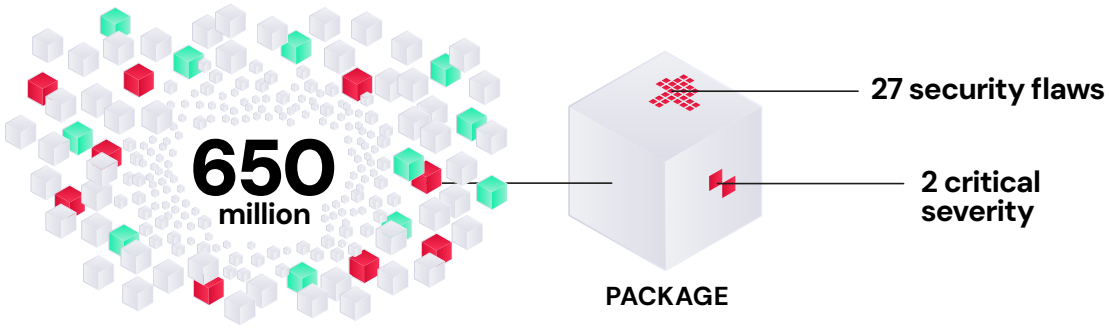
I hope you enjoy reading this report. We here at RL look forward to continuing our work helping to secure organizations from software supply chain threats in 2025.

Report Highlights

1. Software supply chain attacks got more sophisticated in 2024 as malicious actors launched attacks on the build pipelines of prominent open-source projects, singled out AI and machine-learning software supply chains, and took advantage of epidemic, exploitable flaws in black-box, commercial software binaries.

2. Open-source software risks shifted noticeably. Incidents of open-source malware dropped, while leaks of developer secrets and other sensitive information rose by 12%, fueling high-profile open-source supply chain compromises.

3. A survey of 30 npm, PyPI, and RubyGems packages found that large numbers of critical and exploitable flaws lurk in the latest versions of leading open-source packages that account for more than 650 million total downloads across the 30 packages and three package managers. The median number of security flaws discovered per package was 27 (avg. 68), with 2 critical-severity flaws per package (avg. 6).



4. Supply chain attacks on cryptocurrency applications and infrastructure were frequent in 2024, as attackers sought (and got) access to sensitive IT assets and diverted funds from cryptocurrency wallets. The shape and frequency of the crypto attacks are a warning sign that other industries should take note of.

5. The cybersecurity industry's heavy focus on Common Vulnerabilities and Exposures (CVEs) as the measure of cyber risk was called into question following a breakdown at NIST's National Vulnerability Database (NVD) and the CVE reporting system. New ways of assessing cyber risk are needed as organizations face growing risks from undisclosed software flaws, code tampering, and compromised or malicious software modules.

6. AI's explosive growth in the enterprise and the growing reliance of software development organizations on AI-generated code was accompanied by increased AI and ML supply chain cyberthreats, as malicious actors look to infiltrate widely used AI ecosystems.

Executive Summary

Research conducted by RL finds evidence of growing software supply chain risks affecting both software publishers and end-user organizations in the past year. Those include increasingly sophisticated "hands-on-keyboard" campaigns to compromise software supply chains and the targeting of applications and infrastructure linked to both AI and cryptocurrency. Noteworthy incidents in 2024 included the hack of the XZ Utils open-source package in March 2024, the compromise of the Justice AV Solutions (JAVS) commercial video software in May 2024, and the insertion of malicious code into the open-source libraries Solana, Ultralytics, and Rspack, which support millions of downloads and thousands of dependent applications.

Putting aside extreme incidents like software supply chain compromises, the past year saw an epidemic of vulnerable and insecure software continue. An analysis of 30 open-source packages accounting for hundreds of millions of annual downloads found the packages rife with critical-severity as well as patch-mandated CVEs that were being actively exploited by malicious actors. Beyond software vulnerabilities, the past year saw a 12% jump in instances of developer secrets and other sensitive data that were exposed via open-source repositories, according to data compiled by RL Spectra platform.

Add to those factors the vast and vulnerable population of closed-source, black-box commercial software. An analysis that RL conducted of more than three dozen common commercial binaries licensed to enterprises found clear evidence of insecure design, insufficient application hardening, and exposed data and development secrets. This mix of vulnerable open-source and commercial software makes enterprises and other end-user organizations a soft target for malicious actors. They are a big problem for organizations that hope to safely manage their software supply chain risks.

Making things worse: the proliferation of software supply chain attacks is occurring at the same time as a longstanding pillar of enterprise defenses, vulnerability management, is beginning to crumble. Specifically, the past year witnessed a startling drop in the number and quality of vulnerability disclosures from the NVD and a declaration in February 2024 by the National Institute of Standards and Technology (NIST) that it would cease enriching CVEs with critical information such as CVSS criticality scores, patching status, vulnerability descriptions, and the names of affected products — all critical information for security teams.

This proliferation of exploitable software flaws, leaked developer secrets, and faltering software-risk information helps explain why the parade of sophisticated supply chain and cyberattacks did not slow, despite what appears to be a steep decline in instances of malware on open-source package managers over the past year. Data compiled by RL's Spectra Assure™ platform shows what instances of malicious packages declined by more than 70% from 2023 to 2024 across three major open-source package managers: npm, Python Package Index (PyPI), and RubyGems. Discovered malware instances dropped by more than 85% on PyPI, alone. That's an impressive dip in a trend line that has been moving sharply upward in recent years and may reflect tighter developer security policies implemented by those platforms, as well as closer monitoring for threats across many of the major repositories.

Looking ahead, we're concerned. We are concerned about a favorable environment for malicious actors: one marked by an increase in supply chain risks, AI-powered threat actors, and waning public-sector support. The question is, what can companies do to address these growing risks? This report attempts to answer that question by exploring the many dimensions of the shifting software supply chain security landscape and highlighting strategies your organization can modernize to take on today's software threats.

Key Trends

Software Supply Chain Attacks Level Up

Despite widespread media attention in the last year to the problem of software supply chain security and concerted efforts by regulatory bodies and government agencies to promote supply chain security, attacks on both open source code and commercial-software vendors have grown. Among notable incidents in the last year were:

Crypto: A Canary in the Coalmine for Supply Chain Security

RL researchers observed a steady stream of malicious campaigns that targeted the infrastructure and assets for cryptocurrency in 2024. The motivations for these attacks are clear enough. The quote attributed to Willie Sutton that he robbed banks because 'that's where the money is,' also explains why cybercriminals and nation-state actors are targeting cryptocurrency exchanges and wallets.

Many of these supply chain attacks are straightforward. For example, a number of campaigns employed typosquatting to fool developers working on crypto-related apps to add compromised file dependencies that would enable the theft of cryptocurrency from users' wallets. We saw this happen in March 2024, when RL threat researcher Karlo Zanki exposed BIPClip, a malicious PyPI campaign targeting developers working on projects related to generating and securing cryptocurrency wallets.[1]

But crypto-focused attackers also employed more sophisticated and high-touch techniques to gain access to sensitive cryptocurrency applications and infrastructure. For instance, in November 2024, RL detected malicious code in a legitimate-looking Python package, *aiocpa*. That package had been originally engineered as a legitimate crypto-client in order to attract a user base, only to have a subsequent malicious-version update compromise cryptocurrency wallets using the client.[2]

The firm Checkmarx reported in November 2024, on the open-source package *@0xengine/xmlrpc*, an XML-RPC implementation that was first published in a benign, functional form in October 2023. As with the *aiocpa* package, the malicious actors behind *@0xengine/xmlrpc* subtly altered their package in the course of 16 updates, transforming it into a malicious tool capable of stealing secrets and sensitive data while also mining cryptocurrency on infected systems.

Shortly after that, unknown malicious actors compromised the npm package *@solana/web3.js*, a JavaScript API for use with the Solana blockchain platform, implanting malicious functions in two versions of the package, which ranks among the top 10,000 projects in the npm community, with more than 3,000 dependent projects generating 400,000 weekly downloads.[3]

Incidents such as these underscore the growing sophistication and diversity of software supply chain attacks and serve as a warning of what's to come for other sectors. While typosquatting campaigns and malware posted by sock puppet developer accounts are problems that won't disappear anytime soon, the past year's attacks on cryptocurrency infrastructure and applications show that software supply chain risks go well beyond malicious file dependencies to include sophisticated, long-horizon, hands-on-keyboard attacks, with malicious actors cultivating users and code dependencies before flipping their creation, or targeting high-profile projects and maintainers to blast malicious wares to thousands of dependent applications and millions of developers.

[1] Karlo Zanki, "BIPClip: Malicious PyPI packages target crypto wallet recovery passwords," ReversingLabs, March 12, 2024
[2] Karlo Zanki, "Malicious PyPI crypto pay package aiocpa implants infostealer code," ReversingLabs, November 28, 2024
[3] Paul Roberts, "Malware found in Solana npm library raises the bar for crypto security," ReversingLabs, December 5, 2024

Focus: Crypto in the Crosshairs

2024 saw a long list of software supply chain attacks targeting cryptocurrency applications and infrastructure via open-source repositories. Looking at the findings from both RL's threat research team and other security firms, we counted 23 crypto-related malicious campaigns involving malicious code being uploaded to open-source repositories. In the chart below, we categorize each of these discoveries by the platforms they lived on.

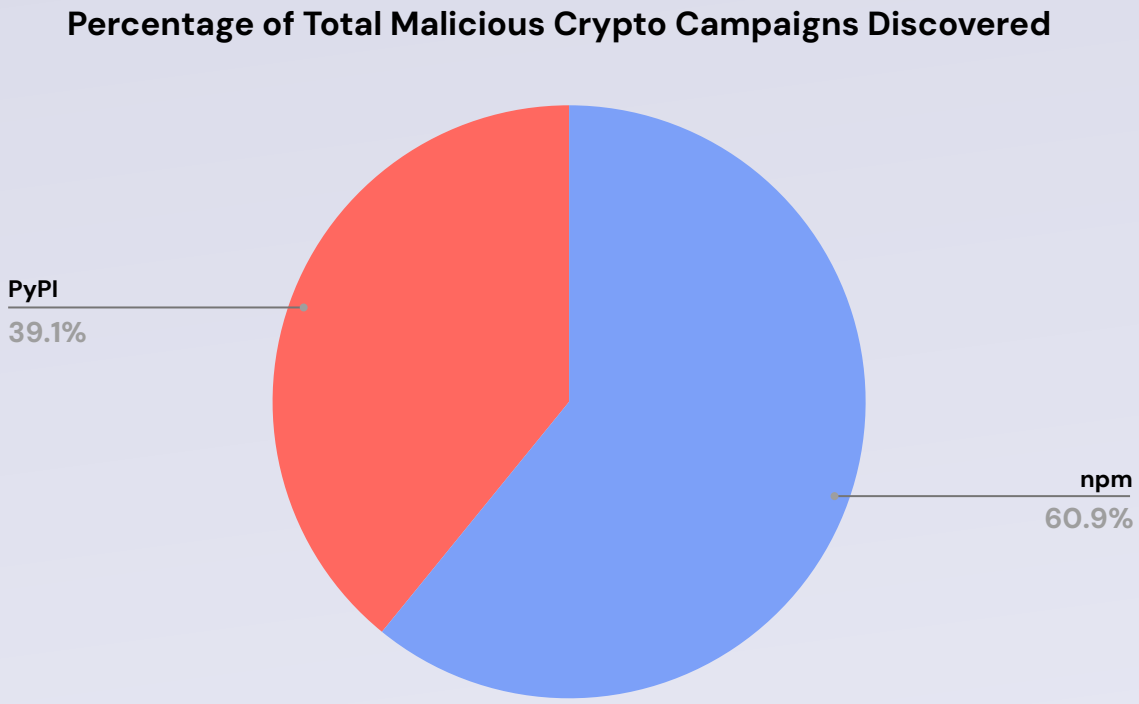


Figure 1: Source of Malicious Open-Source Campaigns Targeting Cryptocurrency.

Most of the malicious, crypto-related campaigns were on npm, which accounted for 14 of the 23 crypto-focused campaigns documented by security teams in 2024 (61%). Nine malicious campaigns were found on the PyPI platform (39%), with attackers posting packages designed to target sensitive cryptocurrency data and assets.

Two discoveries not included in the above chart are worth mentioning: the attack on the open-source Tornado Cash platform,[4] and the exploitation of Docker API.[5] Both campaigns exhibited sophisticated efforts by threat actors to either tamper with the platform's source code or exploit misconfigurations that gave bad actors access to carry out their malicious activities.

[4] Yehuda Gelb, "Tornado Cash Theft Uncovered: Malicious Code Drains Funds for Months," Checkmarx blog, February 26, 2024
[5] Ravie Lakshmanan, "Cybercriminals Exploiting Docker API Servers for SRBMiner Crypto Mining Attacks," The Hacker News, October 22, 2024

Check Out a Timeline of Crypto-Focused Supply Chain Attacks in 2024

Figure 2: Timeline of Crypto-Focused Supply Chain Attacks in 2024.

The XZ Utils Backdoor

First disclosed in April, the backdoor in the widely used XZ Utils open-source compression library (versions 5.6.0 and 5.6.1) was one of the most prominent supply chain compromises in recent memory. The malicious code allowed attackers with a private key to gain unauthorized access to affected Linux systems.

The hack hinged on a years-long, coordinated social-engineering campaign aimed at XZ Utils' longtime maintainer, Lasse Collin. That campaign featured several presumably phony developer accounts that carried out a pressure campaign aimed at getting Collin to allow code contributions from "Jia Tan" (JiaT75), a developer account that had become an active contributor to XZ Utils in the preceding months. The campaign eventually saw Collin handing control of the XZ Utils project to Tan, who then placed malware within the XZ Utils code.[6]

"The most amazing thing about this campaign is how the legit developer was slowly but continuously pushed to put maintenance rights into the hands of the threat actor behind this campaign," said Karlo Zanki, a threat researcher at RL.

State-Actor Attacks on Development Organizations

While the origin of the XZ Utils attack isn't known, it points in the direction of another notable development in the software supply chain threat space in 2024: efforts by state-backed threat actors to infiltrate development organizations in the guise of legitimate software developers.

In recent months, RL threat researchers, along with those from other firms, uncovered evidence of active campaigns by threat actors, likely linked to North Korea, to implant agents and code within development organizations. The campaigns include efforts to push malicious code to developers in the form of compiled Python (PYC) files that pose as developer skills tests given to candidates as part of phony job interviews. In these attacks, threat actors use fake user accounts on business platforms such as LinkedIn and pretend to be job recruiters from well-established companies. RL's analysis of the files in September 2024 showed a clear connection to the VMConnect campaign from 2023, also affiliated with North Korea.[7]

Serious Cyber Risks Lurk in Commercial Binaries

Much of the conversation about software supply chain security focuses on the risks lurking within open-source software packages — and the efforts by cybercriminals and nation-state actors to leverage open source code and platforms to their advantage. But open-source software does not represent the only risk in software supply chains — or even the most prominent. That risk lies in closed-source, commercial software.

"They say the world runs on open source," observes Saša Zdjlear, "but your business runs on commercial software."

Commercial Binaries: What You Don't Know Is Already Hurting You

When it comes to commercial software, what you don't know can — and will — hurt you. In fact, it already is hurting you.

[6] Paul Roberts, "XZ Trojan highlights software supply chain risk posed by 'sock puppets," ReversingLabs, April 11, 2024
[7] Karlo Zanki, "Fake recruiter coding tests target devs with malicious Python packages," ReversingLabs, September 10, 2024

Focus: Commercial Software's Seven Deadly Sins

Malware

Though less prevalent than the other deadly sins of commercial software, a number of cybercriminal campaigns in the last year implanted malware in commercial applications such as the Justice AV Solutions (JAVS) video surveillance software and iPany, a South Korean made VPN client.[8]

Tampering

Instances of unexplained or malicious tampering are relatively common in our scans of commercial software binaries, ranging from failed integrity validation checks that flag incomplete or corrupted file content to suspicious callouts to external command-and-control (C2) infrastructure.

Improperly implemented file-hardening features

This covers a wide range of security lapses often linked to hasty or insecure development practices such as poorly implemented address space layout randomization (ASLR) features, inadequate protections for exploits such as buffer overflows, and other security misconfigurations.

File rot

Old and outdated files were common within the modules we analyzed, including files containing known, exploitable vulnerabilities. Malicious actors prey on older software packages, since those are likely to have known or undiscovered security holes.

Exposed data

Commercial software binaries that RL scanned were found to harbor a large number of sensitive software secrets, including embedded and plaintext credentials, API tokens, private keys, and proprietary information that can be abused by threat actors to further malicious activity.

Known and exploitable or patch-mandated vulnerabilities

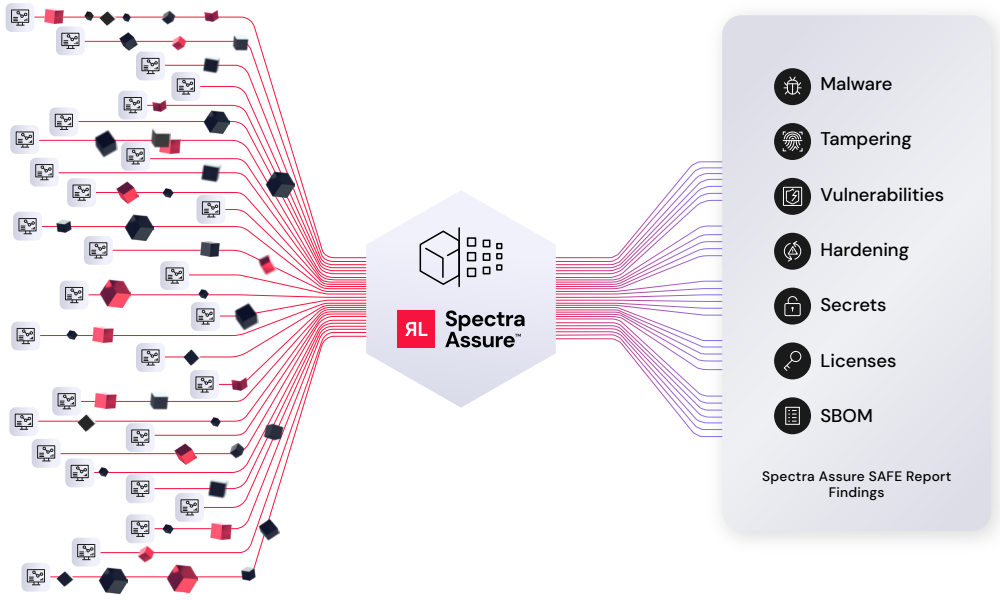
RL detected more than 100 known, exploitable vulnerabilities (KEV) or patch-mandated vulnerabilities spread across the packages we analyzed.

Licensing issues

Many of the commercial software binaries researchers scanned had licensing compliance issues. For example, code covered by "copyleft" licenses was common. These can expose organizations to legal liability for copyright infringement or breach of contract.

[8] Bill Toulas, "IPany VPN breached in supply-chain attack to push custom malware," BleepingComputer, January 22, 2025

To illustrate that risk, RL analyzed 30, widely used commercial software binaries using Spectra Assure. The applications we scanned included widely used commercial and open-source operating systems, web browsers, and virtual private network (VPN) software.



As part of their research, RL's experts scanned client executables as well as installer and setup files for the applications in question. And, to be sure that the analysis captured risks that are likely to be present in current enterprise environments, researchers limited themselves to files that were released within the last three years. Evidence of one or more of the "seven deadly sins" was common, with many of the scanned packages receiving a grade of "Fail" from Spectra Assure.

For example, our scans included 20 distinct versions of VPN clients from six, prominent vendors and found worrying trends. Among them:

- Seven of the 20 VPN packages contained one or more software vulnerabilities that are patch-mandated and/or that are being actively exploited by malware.
- Four of the 20 VPN packages we scanned contained exposed developer secrets - a popular target for malicious actors.
- A recently released Windows VPN client from a prominent vendor contained more than 50 distinct CVEs including:
  - Four CVEs that are rated "critical" and actively exploited by malware, and another 12 assigned a "high" criticality
  - Vulnerabilities dating back as far as 2009 - all with patches available, but not applied by the vendor in question

The prevalence of such severe software security issues — and the lack of attention they receive — shines a spotlight on serious risks that hide in commercial software packages distributed to and deployed within enterprises. Overlooked or disregarded by software producers and largely undetectable by their customers, these flaws provide cybercriminal and nation-state actors avenues for both initial access to sensitive networks and the lateral movement needed to carry out damaging offensive operations.

Hack of JAVS Highlights Risks Lurking in Commercial Apps

RL has opted to keep the names of the commercial vendors and applications we analyzed confidential. However, our analysis of software named in the publicly disclosed breach of Justice AV Solutions (JAVS) echoes and affirms many of the findings of our commercial software scans.

First detected in April 2024, the hack of the JAVS video-recording software involved malware placed within an installer application that allowed malicious actors to take over systems running the JAVS software[9], which is used in courtrooms, legal offices, correctional facilities, and government agencies around the world.

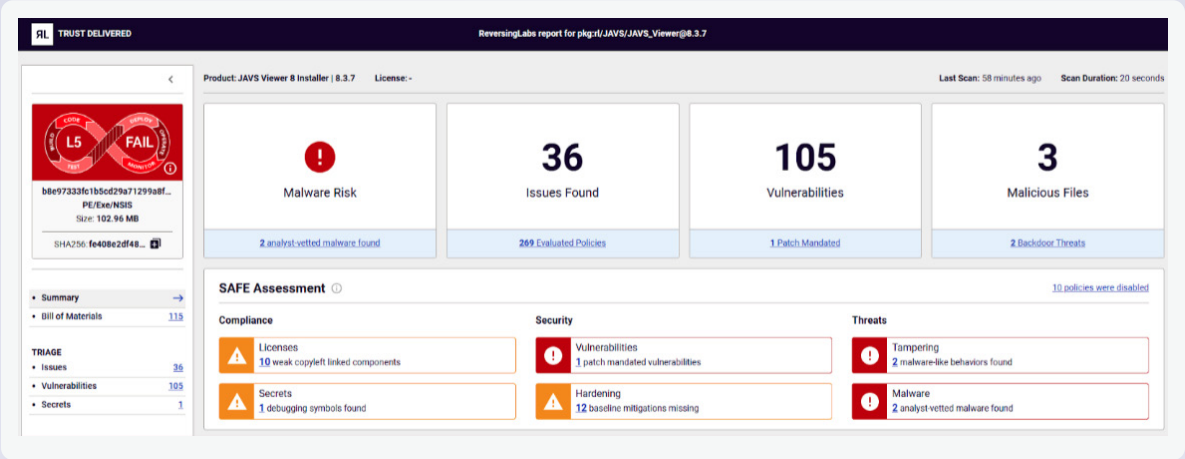


Figure 3: Spectra Report on JAVS Viewer & Installer Showing the Presence of Malware.

Implanted Backdoor Delivers RustDoor Malware

Analysis conducted by the firm Rapid7[10] determined that attackers compromised JAVS Viewer v8.3.7 with a backdoored installer that allowed attackers to gain full control of affected systems. The Trojan-style installer contained malicious functionality embedded in the *fffmpeg.exe* file, with execution triggered through the NSIS installer script. Once executed, the malware could bypass or disable security features, including Anti-Malware Scan Interface (AMSI) and Event Tracing for Windows (ETW), before downloading the intended payload, Rapid7 said in its report. Identified as the RustDoor/GateDoor malware, it collects credentials and disables security measures.

RL performed a differential analysis of the JAVS installer using our Spectra solution to determine whether any changes in the compromised installer were observable prior to release and deployment. The scan produced a failing grade for the installer, with 34 separate security-related issues discovered.

[9] Sergiu Gatlan, "JAVS courtroom recording software backdoored in supply chain attack," BleepingComputer, May 23, 2024
[10] Ipek Solak et al., "CVE-2024-4978: Backdoored Justice AV Solutions Viewer Software Used in Apparent Supply Chain Attack," Rapid7, May 30, 2024

JAVS Differential Analysis Exposes Signs of Tampering

Specifically, our analysis identified a large number of changes to the file, including nine new security issues in the updated installer executable. Most notable was the presence of files with "behaviors that match the backdoor malware profile" (TH15106) and the discovery of software components that "can change the system startup sequence" (TH16118). Malicious actors often attempt to register their code in the system startup sequence as a way to achieve persistence on the infected system (allowing the malware to relaunch each time the OS reboots.)

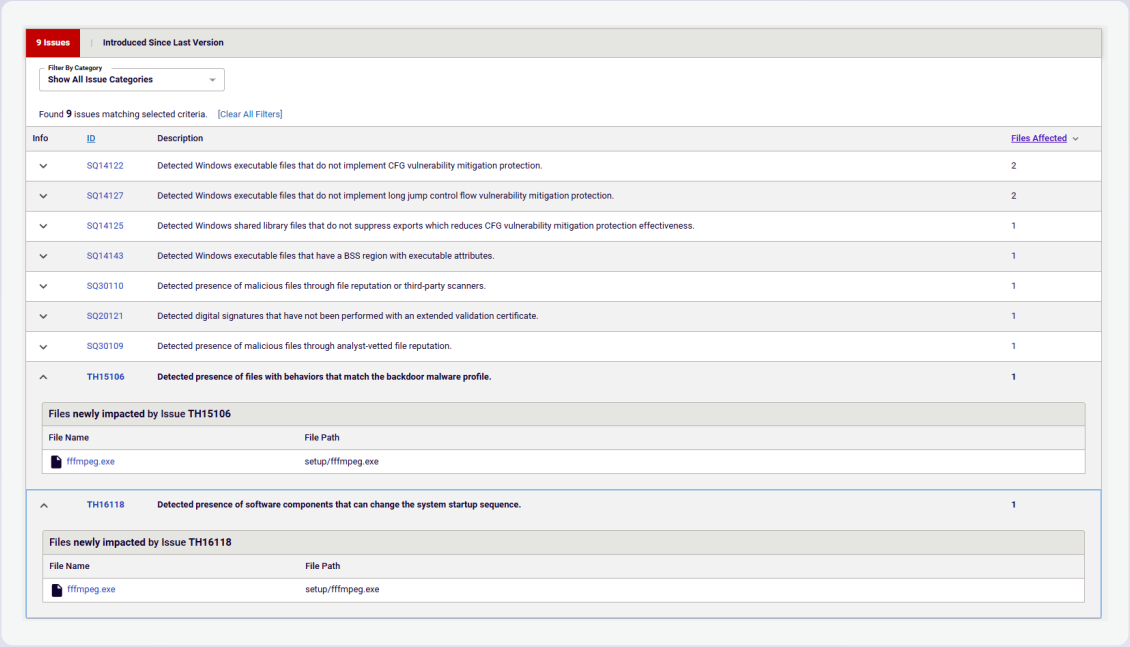


Figure 4: Spectra Report Showing Differential Analysis of the JAVS Installer File Identifying New Security Issues.

In the case of the JAVS installer, both issues were identified within a recently added executable file: *fffmpeg.exe*. That package name itself is a red flag for software producers and end-user organizations, being an obvious typosquat on *ffmpeg.exe*, a popular open-source package that is used by digital media firms to manage multimedia files.[11]

Our analysis of the *fffmpeg.exe* noted the presence of additional malicious files in the installer based on a hash-based file reputation lookup (SQ30109). The differential analysis also identified suspicious digital signatures for the installer package and fffmpeg executables that were not derived from an "extended validation" digital certificate. That affirms findings by Rapid7 that the *fffmpeg.exe* binary and the installer binary were signed by an Authenticode certificate issued to "Vanguard Tech Limited" rather than JAVS.

Our analysis concluded that the RustDoor malware distributed with the JAVS software was triggered by a command entry in the installation script that triggered the *fffmpeg* executable. Both that and the reliance on a certificate from an unaffiliated entity suggest that the malicious actors behind the campaign did not compromise the JAVS build environment. More likely, they compromised the web location that hosted the installer packages. Without access to the build environment, they were limited to repackaging the JAVS installer, wrapping in the malicious executable, and signing it with an unrelated certificate.

[11] ffmpeg.org

Links to Ransomware Groups

An attack against a vendor serving courts, prisons, and law enforcement related outfits might seem in line with nation-state groups' recent forays into the networks of government agencies to gain persistent access and exfiltrate sensitive information. But analysis of the JAVS compromise points towards cybercriminal- rather than nation-state actors.

Specifically, key elements of the attack like the RustDoor malware and the *fffmpeg.exe* file that helped deliver it, as well as the Authenticode certificate issued to Vanguard Tech Limited point to ransomware groups, including infrastructure associated with a ransomware-as-a-service (RaaS) group affiliate named ShadowSyndicate, according to research by Rapid7 and S2W. That suggests the motive behind the compromise was likely financial rather than strategic or ideological.[12]

Lesson Learned: Don't Trust, but Verify Commercial Binaries

The lesson from the JAVS incident is clear: The software your organization receives is not to be trusted, and should be verified.

Our differential analysis of the compromised JAVS installer exposed a long list of security issues, including the presence of malicious files. Had the publisher or its customers been aware of these findings prior to publishing or distributing the compromised JAVS update, that would have prevented widespread compromises of sensitive IT environments, including those attached to courts, police stations, prisons, and other organizations working within the justice system.

"When a software package matches behavior traits of malicious software and is flagged by security solutions, it is highly likely that the software package was tampered with by a malicious actor or a rogue insider," said Karlo Zanki, a threat researcher at RL.

The challenge is that widely deployed application security testing (AST) tools like software composition analysis (SCA), static- (SAST) and dynamic-application security testing (DAST) were designed to enforce secure coding practices, but not to identify supply chain threats like typosquatting, implanted malware, and unauthorized code changes, especially when those threats were imported via malicious or compromised file dependencies.

Absent tools that can spot changes in closed-source, commercial binaries, both development teams and end-user organizations have to trust that the software shipped to them is free of compromises or exploitable-software flaws. Incidents such as JAVS, 3CX, and SolarWinds remind us that blind trust in software safety is dangerous. Both software producers and their customers should never trust, and instead should verify the integrity of the black-box, commercial binaries that they produce or receive from suppliers.

Ignore the lesson of JAVS at your own risk!

[12] CVE-2024-4978: Backdoored Justice AV Solutions Viewer Software Used in Apparent Supply Chain Attack, Rapid7, May 23, 2024

The Great Unpatched: Open-Source Risks Persist

RL' s data from the past year delivers a clear message when it comes to open-source software risks: The landscape is shifting - and getting more dangerous.

Last year's The State of Software Supply Chain Security 2024 Report saw a huge spike in the number of malicious packages found on open-source package managers between 2020 and 2023. Instances of malicious code jumped more than 1,300% during that period, with everything from low-threat protestware and automated phishing campaigns to malware delivered directly from open-source packages, such as the r77 rootkit. RL's analysis of threats on open-source repositories in 2024 reveals a shift in the open-source threat landscape and underscores the need for software publishers and end-user organizations to stay vigilant.

Here are highlights from what RL researchers observed about open-source threats over the past year:

Vulnerable and Popular: Serious Risks Lurk in Open-Source Packages

When it comes to the cyber risks posed by open-source software packages, the mainstream (and social) media typically zero in on 'top 1 percent' type security failures like the social-engineered compromise of XZ Utils, or the glaring flaw in the Log4j Apache library that was both trivially exploitable and widespread.

Issues like those are no joke and deserve attention. But the focus on outlier incidents obscures an even bigger risk in the open-source ecosystem: the massive population of serious, exploitable-software flaws, configuration errors, and other problems lurking in widely used open-source modules.

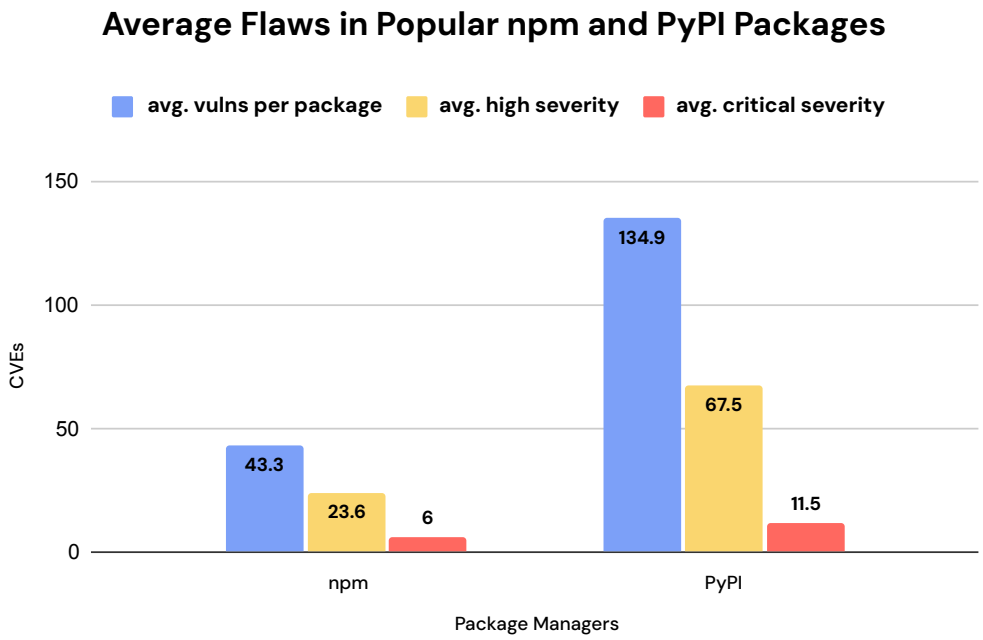


Figure 5: Average Vulnerabilities Detected in High-Traffic npm and PyPI Packages.[13]

[13] Surveyed packages are: npm: grunt-phpunit; sdp-translator; jest-electron; @weweb/renderer-puppeteer; turbo-linux-arm; mockserver-node; helm-bi-nary-linux; @expo/ngrok-bin-linux-x64; @expo/ngrok-bin-linux-arm; @daisy/ace PyPI: tensor2tensor; asciimatics; torchvision; igraph; cartopy; psd-tools; panda3d; memray; llama-index-core; ultralytics

To highlight the issues lurking in popular, open-source modules, RL surveyed top packages across three major open-source repositories: npm, PyPI, and RubyGems with the goal of identifying packages of recent vintage that contained high or critical flaws.

What we found was deeply concerning: open-source packages with millions - and even hundreds of millions- of downloads that contain critical and exploitable-security flaws. These receive little attention from maintainers over months and even years as they ignore available patches and push new versions of their packages containing the flaws to the public. Once deployed, these vulnerable packages provide fertile ground for both cybercriminal and state-sponsored hackers that wish to attack applications and other infrastructure that incorporates the flawed OSS packages.

Our scans across 30 top OSS packages from the npm and PyPI and RubyGems repositories found an average of 68 vulnerabilities across the 30 packages we scanned, with an average of six critical-severity and 33 high-severity vulnerabilities per package. For Python packages, the numbers were even more dire. Across the 10 high-traffic Python packages we scanned, there were an average of 134 flaws per package with 11 rated “critical” and 67 rated “high” severity. (Figure 5)

Critical and Patch-Mandated Flaws Taint Popular Packages

Our scan of the npm repository, for example, identified 10 packages that contained patch-mandated flaws. Together, the 10 npm packages account for more than 61,000 weekly downloads and 18 million total downloads since 2021.

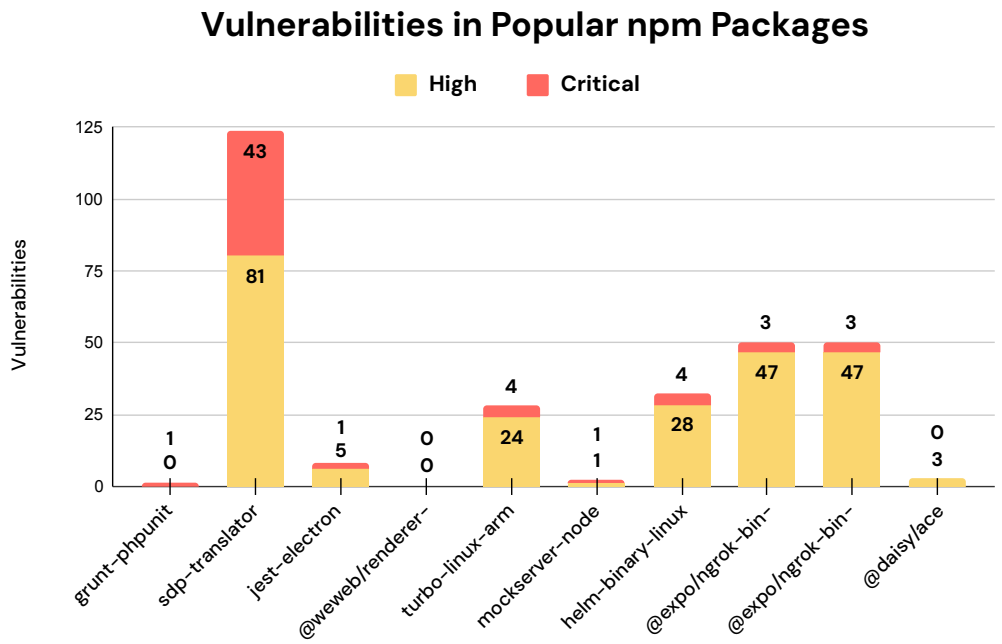


Figure 6: Critical- and High-Severity Flaws in Top npm Packages

Typical of this group of packages is *turbo-linux-arm*, a platform-specific binary for running Turborepo, a build system for JavaScript and TypeScript codebases on Linux distributions based on the Debian and ARM 64 architectures. It sports more than 10,000 weekly downloads and 9.5 million lifetime downloads.

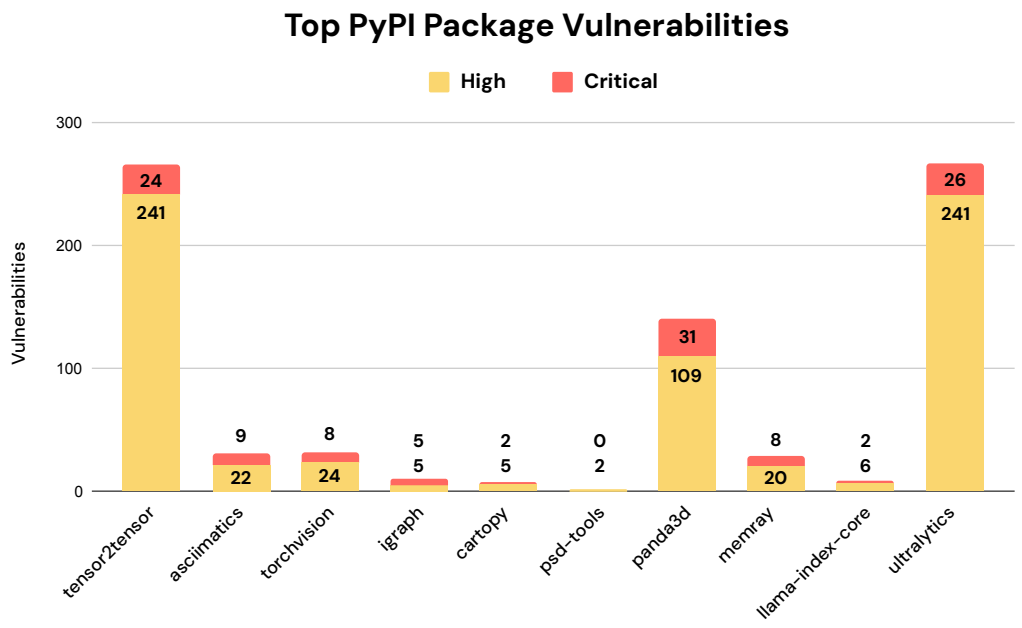


Figure 7: PyPI Critical- and High-Severity Flaws in Top Packages.

Our scan of the latest version of the *turbo-linux-arm* package (1.4.7), which was released more than two years ago detected 41 distinct vulnerabilities, with four rated "critical" and 24 rated "high" severity dating back to 2021. All the identified vulnerabilities had patches available.

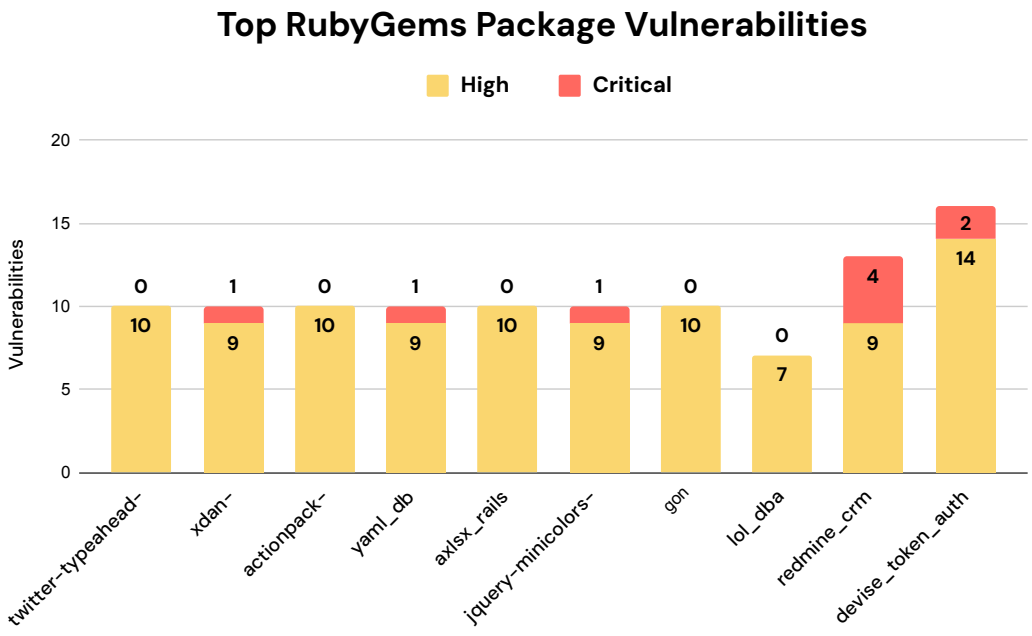


Figure 8: RubyGems Critical- and High-Severity Flaws in Top Packages.

Mind the Rot

One takeaway from our analysis of these popular open-source packages is the acute risks posed by "code rot:" developers' continued reliance on old and not-actively-managed open-source packages. Take the npm package *sdp-translator*, for example. First released eight years ago as a tool for achieving interoperability between Firefox and Chrome for the purposes of doing multi-party video conferences, *sdp-translator* acts as an adapter that feeds the right Session Description Protocol (SDP) to the browser in question.

Only 12 versions of the package have been released since it was first published in 2016, with the most recent version, 0.1.24 released more than eight years ago. Despite that, *sdp-translator* is still widely used with 16 dependent applications, more than 3,000 weekly downloads and 500,000 downloads over the life of the package.

That's a problem, as *sdp-translator* is rife with flaws. Our analysis identified 164 vulnerabilities in the latest version of the package, with some vulnerabilities dating back more than 10 years, to 2013. Even more alarming: Three quarters of the flaws identified in *sdp-translator* have severity ratings of "critical" (43) or "high" (81). Seven of them are known to be exploited by malware, while two are "patch mandatory," meaning they are listed on CISA's "Known Exploited Vulnerability" catalog.[14]

We saw a similar correlation between the age of the package and the number of vulnerabilities with the Python package *tensor2tensor,* a library of deep-learning models and datasets developed by engineers at Google and designed to facilitate machine learning (ML) research. Google has placed *tensor2tensor* in "maintenance mode" and encourages users to switch to its successor library, *Trax*. The latest version of *tensor2tensor*, 1.15.7, was published more than four years ago. Still, *tensor2tensor* has more than 3,000 weekly- and 7 million total downloads. Our latest scan of the package identified 445 distinct vulnerabilities dating back to 2019, the vast majority of which are in package dependencies. Of the 445 vulnerabilities, 24 of them are rated "critical" severity and 241 rated "high" severity. That includes an astounding 252 vulnerabilities with known exploits including one "patch-mandated" flaw (CVE-2023-4863) that can be actively exploited by malware. Almost all the detected flaws have patches available.

The problem of "code rot" is widespread within the software industry, as producers prioritize feature development and rapid-release schedules over security. The software industry long ago embraced the ethos of "if it ain't broke, don't fix it" — abstaining from applying software updates or phasing out old and unsupported software modules for fear of breaking existing functionality or slowing development cycles.

Who Pays for Vulnerable Code? Customers.

The consequences of those short-sighted decisions, however, often fall to customers: the end-user organizations that download and install applications containing vulnerable open-source components. These are routinely targeted by sophisticated actors that exploit critical flaws to gain a foothold, execute malicious code on privileged IT assets, or move laterally within a compromised IT environment.

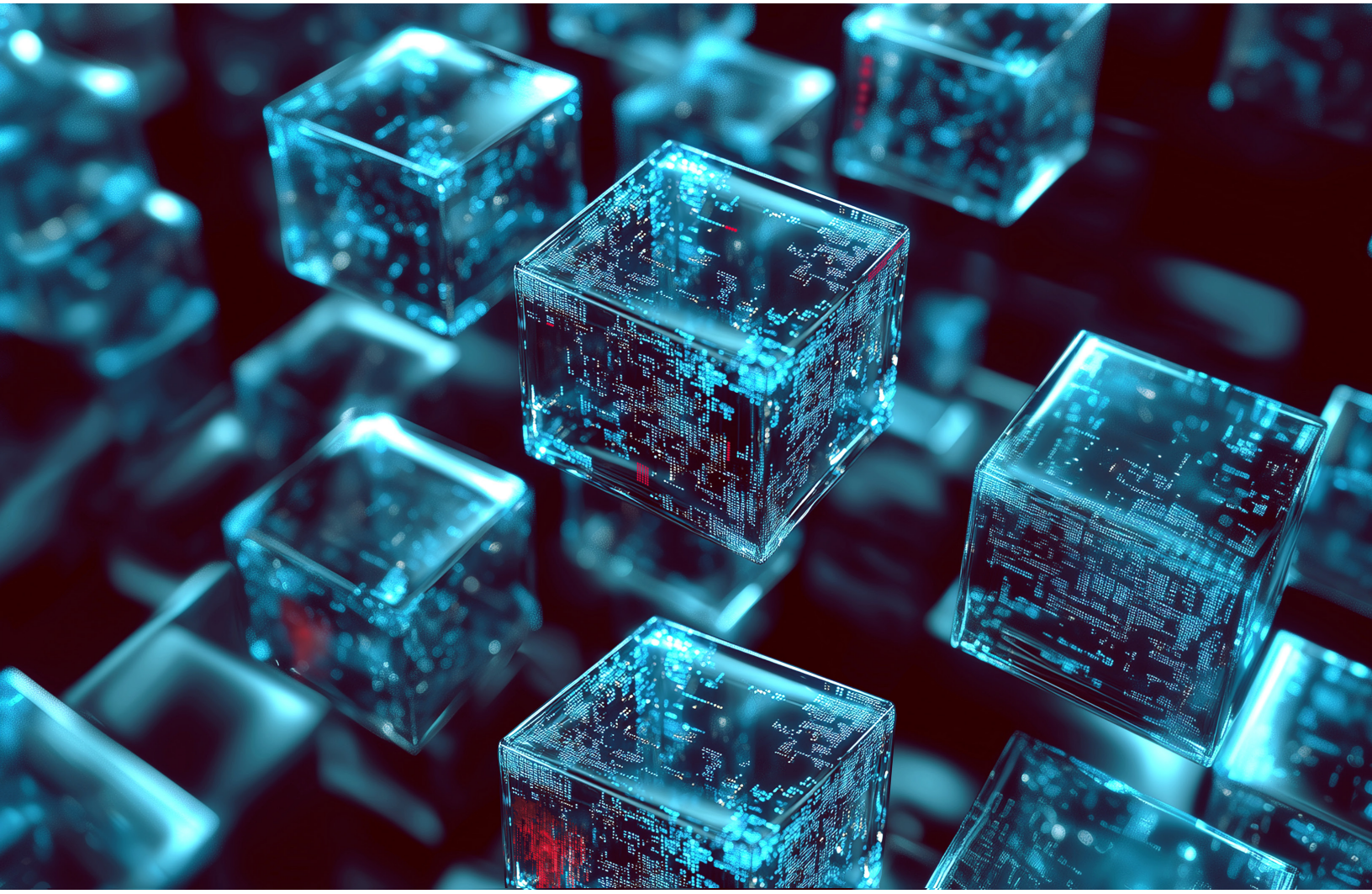
For example, companies using VPN technology from the vendor Ivanti have been targeted by wide-ranging, state-sponsored, and cybercriminal attacks since at least 2023. Those attacks exploited a wide range of previously undisclosed ("zero day") flaws in Ivanti's Connect Secure VPN and other products. A subsequent analysis of Ivanti's embedded software exposed that the attacks were driven by flaws and outdated code.

[14] https://www.cisa.gov/known-exploited-vulnerabilities-catalog

That included a base operating system, CentOS 6.4, that was more than a decade old and was declared end of life in 2020. Researchers also found a seven-year-old version of the OpenSSL software used in otherwise up-to-date Ivanti devices, and code with 396 “high” severity CVEs, 111 with active exploits, including two remote code exploits (RCEs).

While some of those issues were addressed by the company in the wake of the attacks, subsequent campaigns have found and exploited still more “zero day,” SQL injection, and RCE flaws in Ivanti software, according to an alert published by CISA in January 2025.[15]

Such reports are damaging to the reputation of software providers. But their customers bear the costs of insecure software applications and updates. In the case of the Ivanti hacks, victims included U.S. government agencies and defense contractors as well as technology and telecommunications companies, with the malicious actors behind the attacks maintaining persistence on compromised environments and using “live-off-the-land” techniques to move laterally and gain access to sensitive data and IT assets.



[15] CISA, “Threat Actors Chained Vulnerabilities in Ivanti Cloud Service Applications,” January 31, 2025

Focus: Serious Risks Lurk in Torchvision Python Package

RL's analysis of popular open-source packages makes one thing clear: Serious security issues are not limited to obscure, old, or inactive open-source modules. Our proof? Torchvision, a Python package of datasets, model architectures, and image transformations used for computer vision applications.

In the top 1% of all Python packages, Torchvision averages more than 3.4 million weekly downloads, with close to 360 million downloads over the lifetime of the package. A product of engineers at Meta, Torchvision is actively maintained, with 50 versions dating back eight years and 10 releases between January 2024 and January 2025 alone.

But frequent updates don't translate into better security. Torchvision gets a failing grade from RL's Spectra Assure analysis, with 45 identified vulnerabilities in the latest, scanned version of the package - four of them dating back more than six years. The vulnerabilities include eight with a "critical-" severity rating and 24 with a "high-" severity rating, as well as one that is considered "patch mandated" (listed on CISA's KEV catalog) and that is being actively exploited by malware.

Even more troubling: Those numbers have remained constant over time. The chart below tracks the number of total, critical- and high-rated Torchvision flaws over the last 10 versions of Torchvision, dating back to 2023.

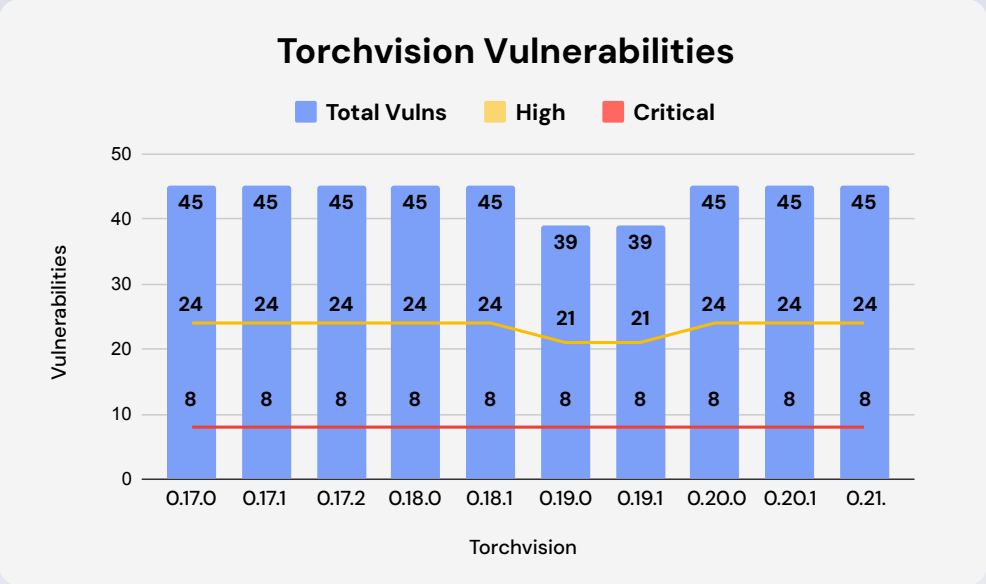


Figure 9: Total-, Critical-, and High-Vulnerability Counts in Recent Versions of Torchvision.

Among those vulnerabilities is CVE-2023-4683, a high- (8.8) severity flaw that describes a heap-buffer overflow in the widely used libwebp library that permits a remote attacker to use a specially crafted HTML page to perform an out-of-bounds memory write on vulnerable systems. A patch for the issue was released in September 2023 and the flaw is being actively exploited by malware and is considered a "patch-mandated" flaw.

As with many of the other packages RL analyzed, fixes are available for every single vulnerability Spectra identified in Torchvision. Despite that, the PyTorch team has continued to rely on and distribute unpatched, vulnerable software in the Torchvision package, potentially opening doors for malicious actors.

Leaked Secrets Persist on Open-Source Repositories

One of the key open-source risks that RL researchers monitor jumped in 2024: leaked developer secrets, which increased across almost every open-source package manager in the past year.

Development secrets are sensitive and confidential information that are used to verify users, grant system access, enable secure communications, or ensure the confidentiality and integrity of critical data. Such secrets range from passwords hard coded into software to encryption keys, API tokens used to secure information sent between applications, and SSH keys used for secure remote access and authentication.

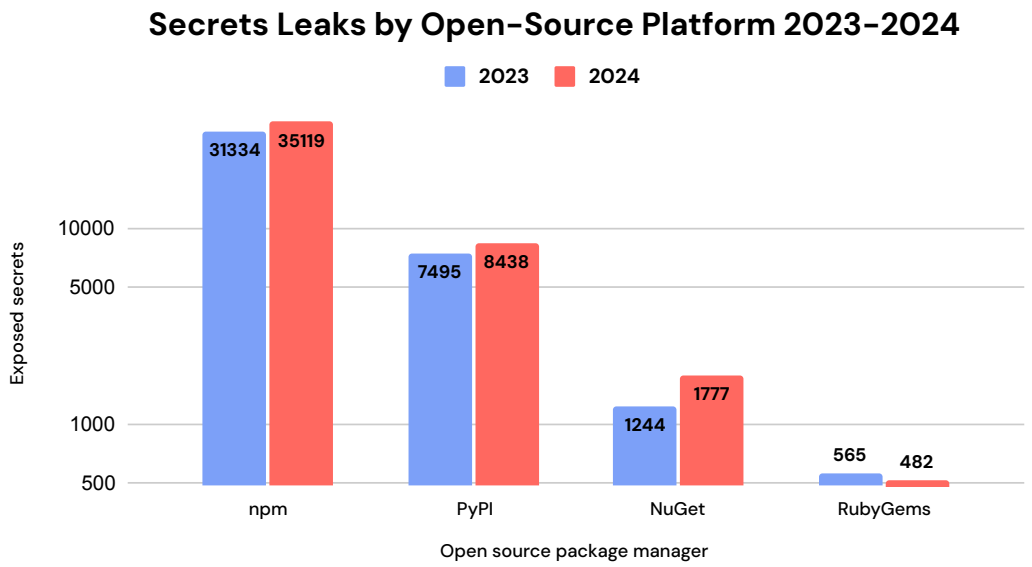


Figure 10: Secrets Leaks by Open-Source Platform 2023-2024.

Development Secrets Leaks Jump 12%

Secrets leaks across the four major open-source repositories RL tracks — npm, PyPI, NuGet, and RubyGems — continued to be a major issue in 2024, with the number of discovered secrets detected by RL standing at 45,816 through September 30, 2024 — a 12% increase from the same period a year earlier. Not surprisingly, the number of exposed development secrets tilted heavily toward the two most widely used platforms, npm and PyPI, which together account for around 95% of all the leaked secrets RL detected.

On npm, which is the largest open-source repository, RL detected close to 4,000 more secrets in 2024 compared with 2023: 35,119, up from 31,334. A similar trend was observed on the PyPI package manager, with incidents of leaked secrets increasing from 7,495 in 2023 to 8,438 in 2024 — a 12% increase. Incidents of leaked secrets spiked on the NuGet open-source package manager by more than 40%, up from 1,244 in 2023 to 1,777 in 2024.

Only the RubyGems platform experienced a decrease, RL's analysis found. Discovered secrets declined by about 14%, down from 565 in 2023 to 482 in 2024.

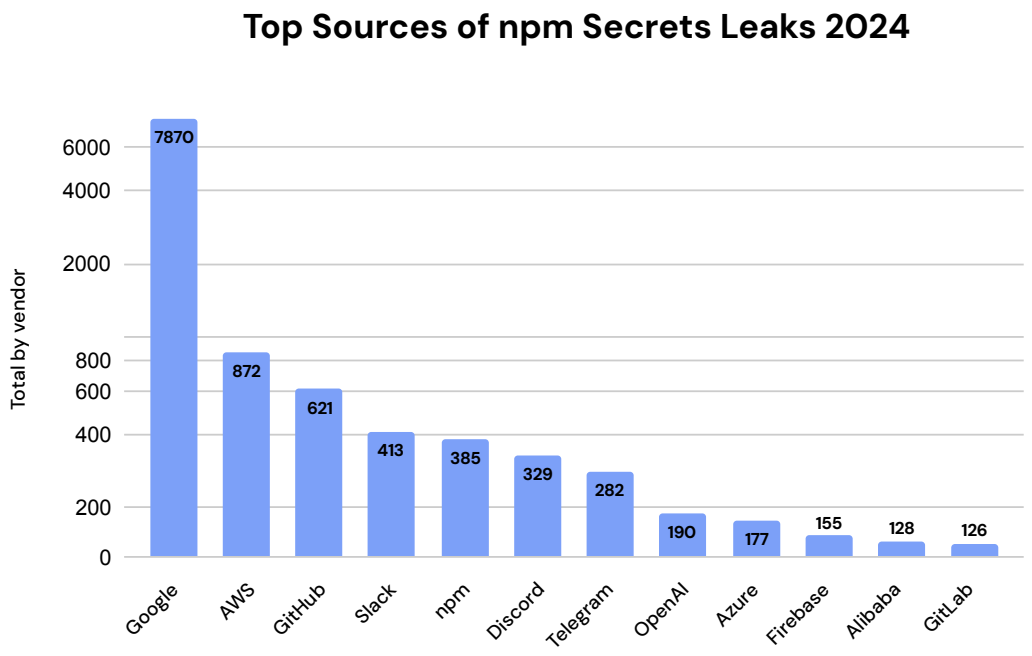


Figure 11: The Top 12 Vendors Associated with Secrets Leaks on the npm Platform.

Google, AWS (Still) Fertile Ground for Exposed Secrets

As we observed in previous reports, the sources of the development secrets leaks track closely with popular cloud-based platforms and applications, including Google, Amazon Web Services (AWS), Slack, and Telegram. Google's cloud was, by far, the largest source of leaked developer secrets for both the npm and PyPI platforms, accounting for close to 70% of all secrets discovered on the npm platform and 35% of the secrets discovered on PyPI. AWS was the second-biggest source of secrets on npm and the third on PyPI, while Slack, GitHub, and Telegram were also the source of substantial numbers of exposed secrets.

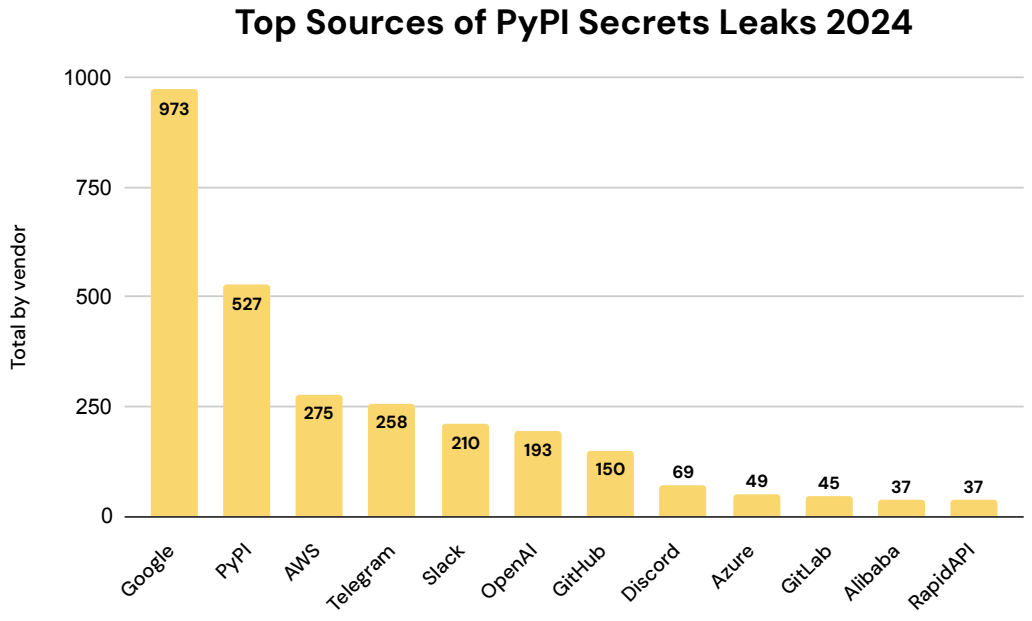


Figure 12: The Top 12 Vendors Associated with Secrets Leaks on the PyPI Platform.

When looking at the types of secrets that were most often discovered lurking in open-source repositories, the vast majority — close to 80% — fell into three categories: private keys, web-service API keys, or web-service access tokens. This analysis comes from data collected from both the npm and PyPI platforms, on which the vast majority of leaked secrets resided.

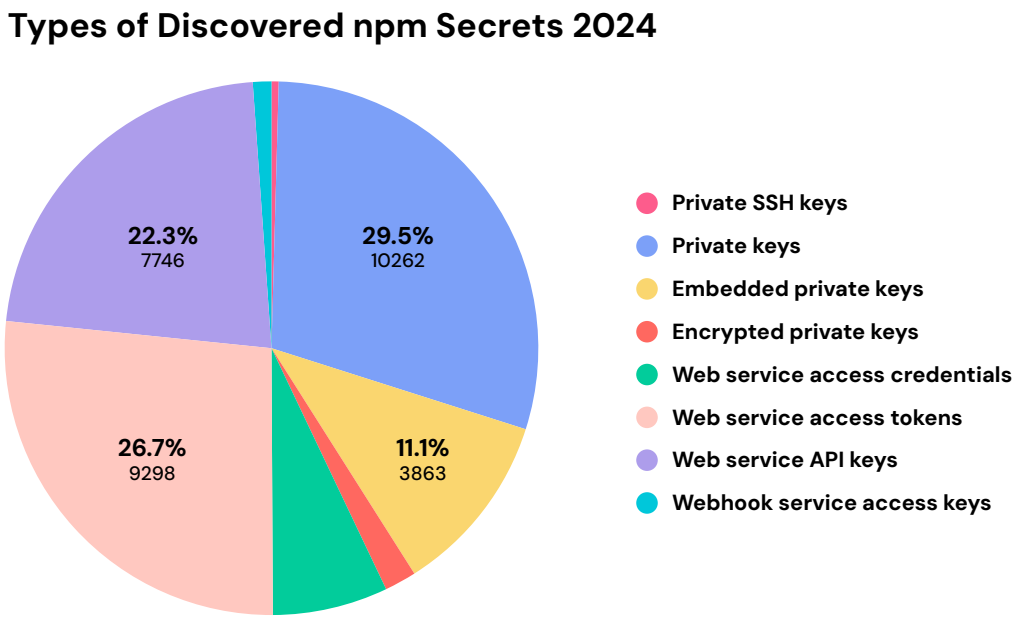


Figure 13: Secrets Discovered on npm in 2024.

These patterns accurately reflect the popularity and widespread use of those tokens within modern application development.

Secrets Leaks a Common Theme in Supply Chain Attacks

Supply chain security events in 2024 underscore the material risk that leaks via open-source repositories pose to individual firms, as well as the entire open-source development ecosystem. For example, in February 2024 RedHunt Labs found a leaked GitHub token belonging to a Mercedes-Benz Group employee that granted unrestricted and unmonitored access to the entirety of source code hosted on Mercedes-Benz Group's internal GitHub Enterprise Server.[16]

In July, a nearly catastrophic leak of a privileged access token put PyPI and the Python language at great risk. JFrog researchers found a compiled Python file in a Docker container.[17] If the token had fallen into the wrong hands, it would have granted access to any package stored on PyPI, as well as the raw code of the Python language itself — a breach that could have led to backdoors or other malicious code being distributed to tens of millions of machines worldwide.

In addition to that incident, the widely used AI library *ultralytics* was hit by an attack that saw malicious actors use a previously reported GitHub Actions script-injection flaw to compromise the Ultralytics build environment, steal API tokens and other secrets, and implant malicious code that was then distributed to downstream projects that use the Ultralytics library.[18] (As we noted above: *ultralytics* was among the most vulnerability-ridden PyPI packages we scanned in our security analysis of popular, open-source projects with 26 "critical"-severity flaws detected in the latest release.)

[16] Paul Roberts, "Lessons from the Mercedes-Benz GitHub source code leak," Feb 1, 2024
[17] Sead Fadilpašić, "A GitHub token leak could have put the entire Python language at risk," TechRadar, July 16, 2024
[18] Karlo Zanki, "Compromised ultralytics PyPI package delivers crypto coinminer," ReversingLabs, December 9, 2024

A compromised maintainer account with publishing privileges is believed to be behind the compromises of the Solana *web3.js* package, a widely used, open-source module with more than 3,000 dependent projects and 400,000 weekly downloads.[19]

To stop the flow of secrets and sensitive information to open-source platforms, development organizations are urged to clean up their code storage, access control, and secrets rotation policies — and to craft a unified secrets management strategy that includes regular code repository audits that look for secrets that may have accidentally found their way into publicly disclosed code.

GitGot? GitHub Features Exploited by Malicious Actors

Planting malicious code on open-source package managers is just one attack vector available to malicious actors — and not even the most effective. Another trend that RL observed in 2024 was the growing use of open-source infrastructure to further malicious campaigns.

In January 2024, for example, RL researcher Lucija Valentić wrote about the discovery of two malicious packages, *warbeast2000* and *kodiak2k*, on the npm open-source package manager that used GitHub to store stolen Base64-encrypted SSH keys lifted from developer systems, which were then used to install the malicious npm packages.

That aligns with an incident uncovered in April by researchers at McAfee, who discovered a flaw in a GitHub process for file uploads of comments on GitHub-hosted projects that was used to distribute malware containing URLs associated with authorized Microsoft repositories, in an effort to make the files appear trustworthy.

Leveraging open-source package managers to disguise malicious activities isn't a new phenomenon. RL threat researchers observed a similar campaign in 2023 in which GitHub Gist and git commit messages were being used to deliver malicious commands to infected machines.[20] The continued detection of these incidents raises the bar for developers and development organizations to be vigilant to ensure that packages, communications, and other interactions with open-source package managers and development platforms like GitHub are legitimate. Development teams and software producers should be on the lookout for any unexplained or unexpected behavior or network traffic leveraging GitHub infrastructure.

Malicious Campaigns Crop Up on NuGet, VS Code

The past year brought strong evidence that the risks posed by open-source supply chain attacks are spreading, as repositories such as NuGet and VS Code were leveraged by malicious actors and campaigns.

In July 2024, for example, RL researcher Karlo Zanki wrote about the continuation of a 2023 campaign that RL dubbed "IAmReboot" targeting the NuGet repository, a popular open-source package manager for .NET code.[22] Zanki's research captured the evolution of that campaign in which attackers shifted their code out of installation and post-installation PowerShell scripts to avoid detection. Then, in 2024, their obfuscation methods evolved with the introduction of a technique called "intermediate language weaving" to add malicious functionality to legitimate Portable Executable .NET binaries. In these attacks, the attacker takes a compiled .NET binary from a legitimate NuGet package and patches it in order to inject a module initializer into it.

[19] Paul Roberts, "Malware found in Solana npm library raises the bar for crypto security," ReversingLabs, December 5, 2024
[20] Karlo Zanki, "Malware leveraging public infrastructure like GitHub on the rise," ReversingLabs, December 19, 2023

Campaigns targeting the Visual Studio Code (VS Code) marketplace were also detected — though their true intent remains open to debate. For example, we wrote about the discovery of clipboard-helper-vscode, a package extension advertised as a VS Code add-on "designed to enhance the clipboard functionality within VS Code."[23]

A Drop in Open-Source Malware

Not all the news was bad. The past year saw a steep decline in malicious packages observed on common open-source platforms. Data compiled using RL's Spectra Assure platform found that incidents of malicious packages across the three main open-source repositories — npm, PyPI, and RubyGems — declined by 70% between 2023 and 2024, down from 12,130 total malicious packages in 2023 to just 3,580 in 2024. The vast majority of that decline was seen on PyPI, where incidents of malicious packages dropped by 87%. More than 6,500 fewer malicious packages were detected on PyPI in 2024 than in 2023.

There are many possible explanations for this steep drop in malicious open-source packages, but declining interest in software supply chains as a target for malicious actors isn't one of them. A more likely explanation is the greater attention paid to threats and supply-chain risks by the maintainers of open-source repositories.

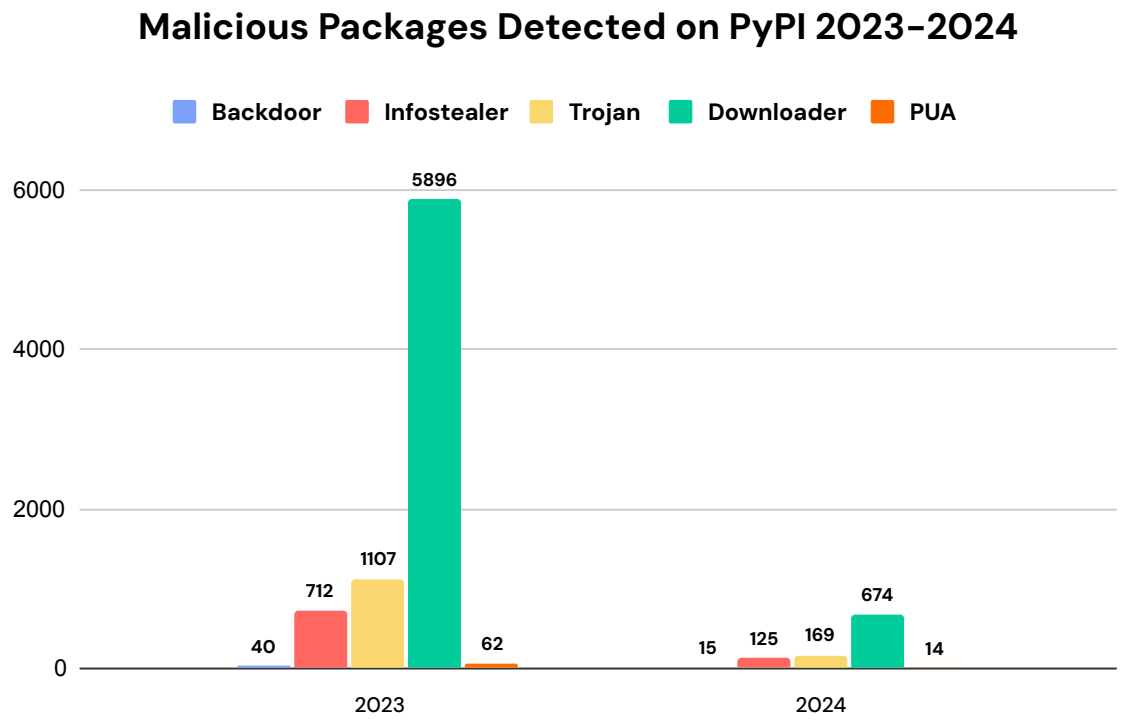


Figure 14: Instances of Malicious Packages Detected on the PyPI Platform.

[22] Karlo Zanki, "IAmReboot: Malicious NuGet packages exploit loophole in MSBuild integrations," ReversingLabs, October 31, 2023
[23] Lucija Valentić, "Malicious helpers: VS Code Extensions observed stealing sensitive information," ReversingLabs, April 3, 2024

For example, the Open Source Security Foundation (OpenSSF) launched its Malicious Packages Repository in October 2023 to coordinate and consolidate the scanning of open-source packages for threats across major repositories. Since then, the OpenSSF has issued credits to contributors for close to 29,000 submissions to the repository.

There has also been progress made in hardening developers' access to open-source repositories. PyPI's maintainers implemented mandatory multifactor authentication (MFA) for developer accounts in January 2024, making it much more difficult for malicious actors to take over developer accounts.[23] PyPI's move followed the lead of other platforms, including npm and NuGet, which made two-factor authentication (2FA) mandatory starting in March 2022.[24] RubyGems began requiring MFA for maintainers of popular packages in August 2022[25], and GitHub instituted mandatory 2FA in March 2023.[26]

The steep decline in malicious wares found on platforms that have implemented mandatory 2FA may be the most visible sign that stronger access controls for open-source developer accounts are paying dividends.

CVEs Lose Relevance

For the last 25 years, Common Vulnerabilities and Exposures identifiers (CVEs) have been the yardstick by which the security of software is measured. First introduced in September 1999, CVEs put a name (and number) to software vulnerabilities and have played a central role in industry awareness of software risk ever since. Major vendors such as Microsoft quickly incorporated CVE identifiers into their patch reports and security advisories and regularly cite the Common Vulnerability Scoring System (CVSS) to convey the risk associated with specific flaws.

In the intervening years, the CVEs associated with high-profile vulnerabilities became legendary and household names, including the Heartbleed flaw in OpenSSL (CVE-2014-0160); EternalBlue (CVE-2017-0144), the flaw in the Windows SMB Service that was exploited by the NotPetya wiper; and Log4Shell (CVE-2021-44228), the remote code execution flaw in Log4j, a widely used Java logging framework.

Cracks in the NVD Emerge

But last year revealed to us a CVE system that is faltering. The combination of a brittle, centralized reporting structure, scarce public funding, a lack of automation, and a growing pipeline of discovered software flaws led NIST to cease enrichment of CVE records on its NVD between February and May of 2024.

[23] Mike Fielder, "2FA Requirement for PyPI begins 2024-01-01," PyPI.org, December 13, 2023
[24] NuGet.org
[25] Ravie Lakshmanan, "RubyGems Makes Multi-Factor Authentication Mandatory for Top Package Maintainers," The HackerNews, August 17, 2022
[26] Securing your account with two-factor authentication (2FA), GitHub.com

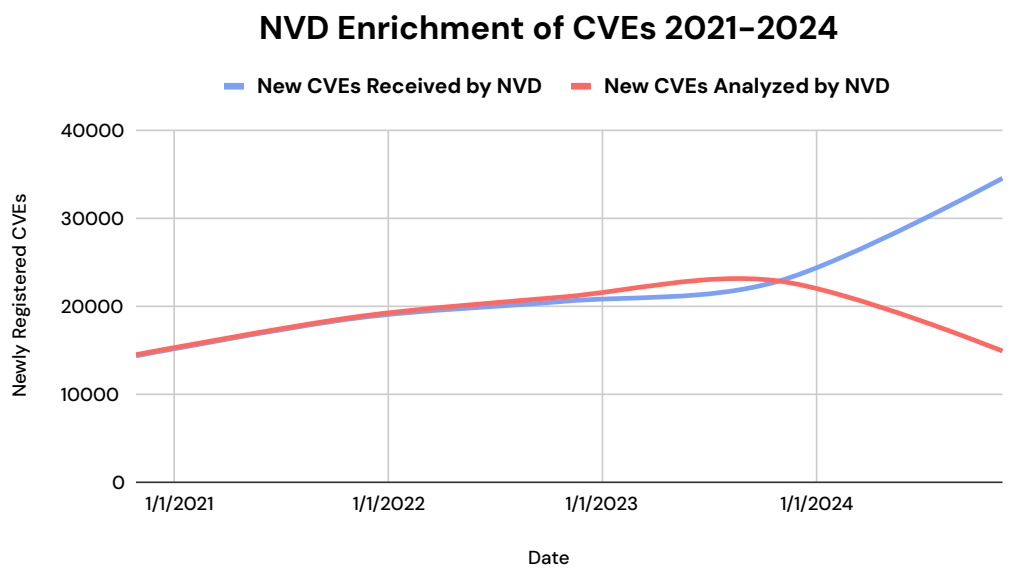


Figure 15: NVD Enrichment of CVEs 2021-2024.[27]

With NIST no longer "enriching" CVE records with detailed information about the linked software flaw, a backlog of thousands of CVE reports soon emerged, causing non-governmental CVE numbering authorities (CNAs) to scramble to take up the backlog. RL data reflects this disruption and the sudden shift in CVE-related activity from the public to the private sectors.

Looking at CVEs linked to open source code hosted on the npm package manager, there has been a big shift from NVD-assigned CVEs to CNA-assigned CVEs since 2023. The number of critical-, high-, and medium-severity CVEs added to the NVD declined by 74% during that time, while CNA-assigned and -enriched CVEs exploded from just a single reported CVE linked to an npm package in 2023 to 172 in 2024.

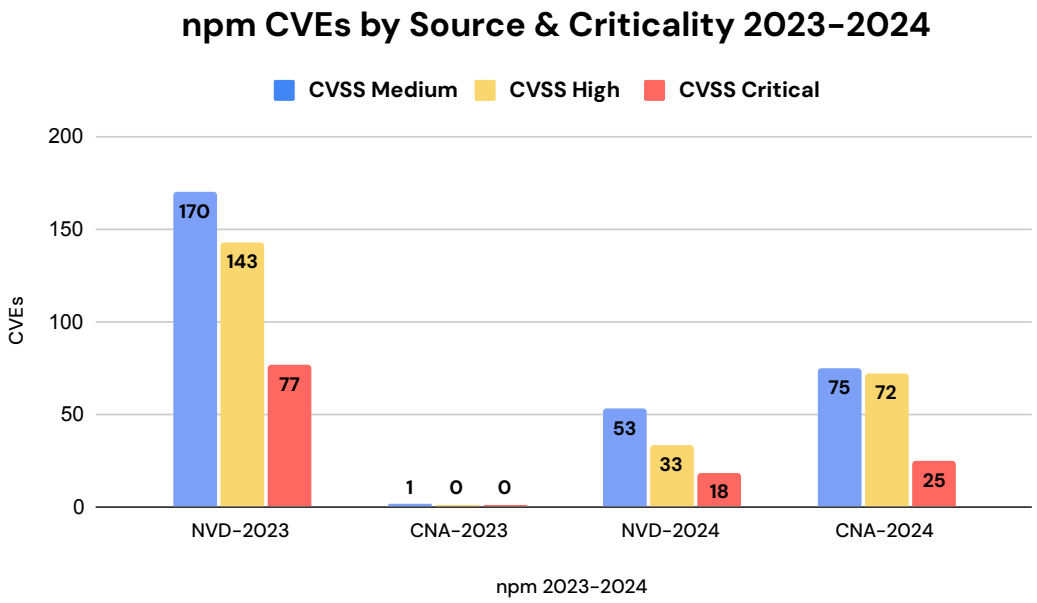


Figure 16: Medium-, High-, and Critical-Severity CVEs for npm Packages 2023-2024.

[29] Archive.org samples taken on October 29, 2020; November 11, 2021; October 26, 2022; October 25, 2023; and November 14, 2024

An almost identical pattern was observed with CVEs linked to vulnerabilities in open-source packages on the PyPI repository, with NVD-assigned and -enriched CVEs dropping by 74%, while CNA-assigned and -enriched CVEs with critical-, high-, or medium-severity ratings jumped from just three linked to PyPI packages in 2023 to 274 in 2024.

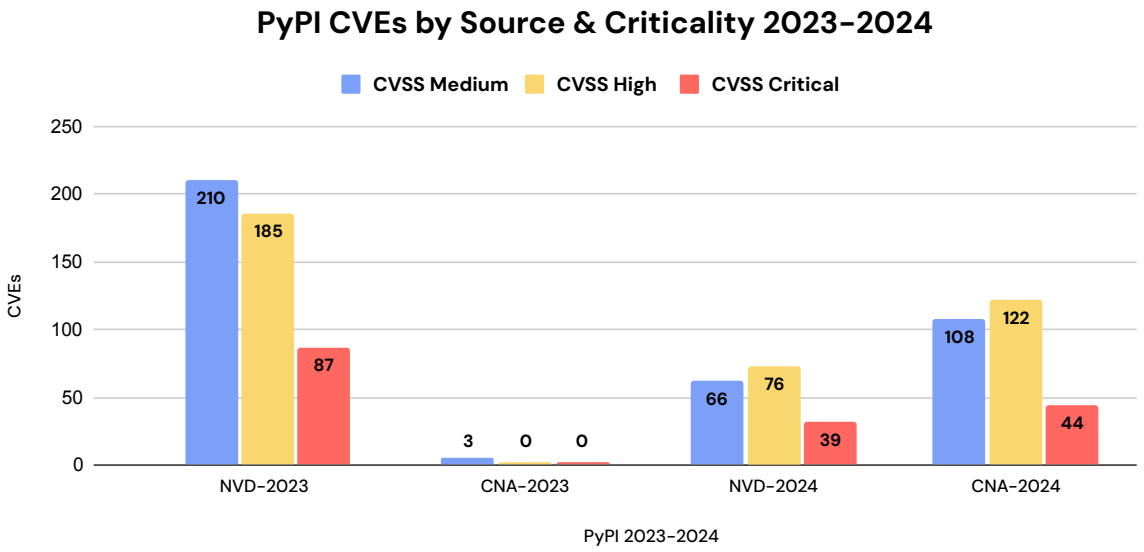


Figure 17: Medium-, High-, and Critical-Severity CVEs for PyPI Packages 2023-2024.

Research by the firm VulnCheck estimated that 93% of new vulnerabilities received during that time were not analyzed by the NVD — a gaping window for would-be threat actors.[28]

Soon after, in June 2024, NIST said that it engaged with a contractor to assist in tackling the growing backlog of CVEs and gave itself a deadline to clear the backlog of unenriched CVEs of September 2024. At the same time, MITRE worked to boost the number of private- and public-sector CNAs, hitting a CNA milestone of 400 in August 2024 by enlisting the likes of Wiz and Proton AG to help offload CVE enrichment work.[29] NIST's self-imposed September 2024 deadline ended up dropping. As of the end of November 2024, there was still a significant backlog of more than 20,000 CVEs awaiting analysis, NIST data showed.

The huge backlog of CVEs puts the core mission of the NVD and CVEs at risk. Enumeration and tracking of software vulnerabilities is useful — but only so long as it keeps pace with software risks. Once a gap opens between discovered flaws and enumerated and enriched vulnerability reports, organizations that rely on CVEs to plan and track their cyber-defense efforts are at risk from the growing numbers of yet-to-be-analyzed flaws. Our collective map of the software-risk landscape suddenly has large, obscured areas holding unknown quantities of flaws.

Vulnrichment to the Rescue?

To help address the backlog, CISA launched a new vulnerability-enrichment program dubbed "Vulnrichment" that aims to fill the gap caused by NIST's scale-down of involvement in the NVD last year.[30]

[28] Patrick Garrity, "The Real Danger Lurking in the NVD Backlog," VulnCheck, May 23, 2024
[29] Sarah Gooding, "MITRE Marks Major Milestone, Minting 400 CNAs as NVD Backlog Grows," Socket.dev, August 13, 2024
[30] Jaikumar Vijayan, "CISA's 'vulnrichment' aims to fix the NVD," ReversingLabs, May 15, 2024

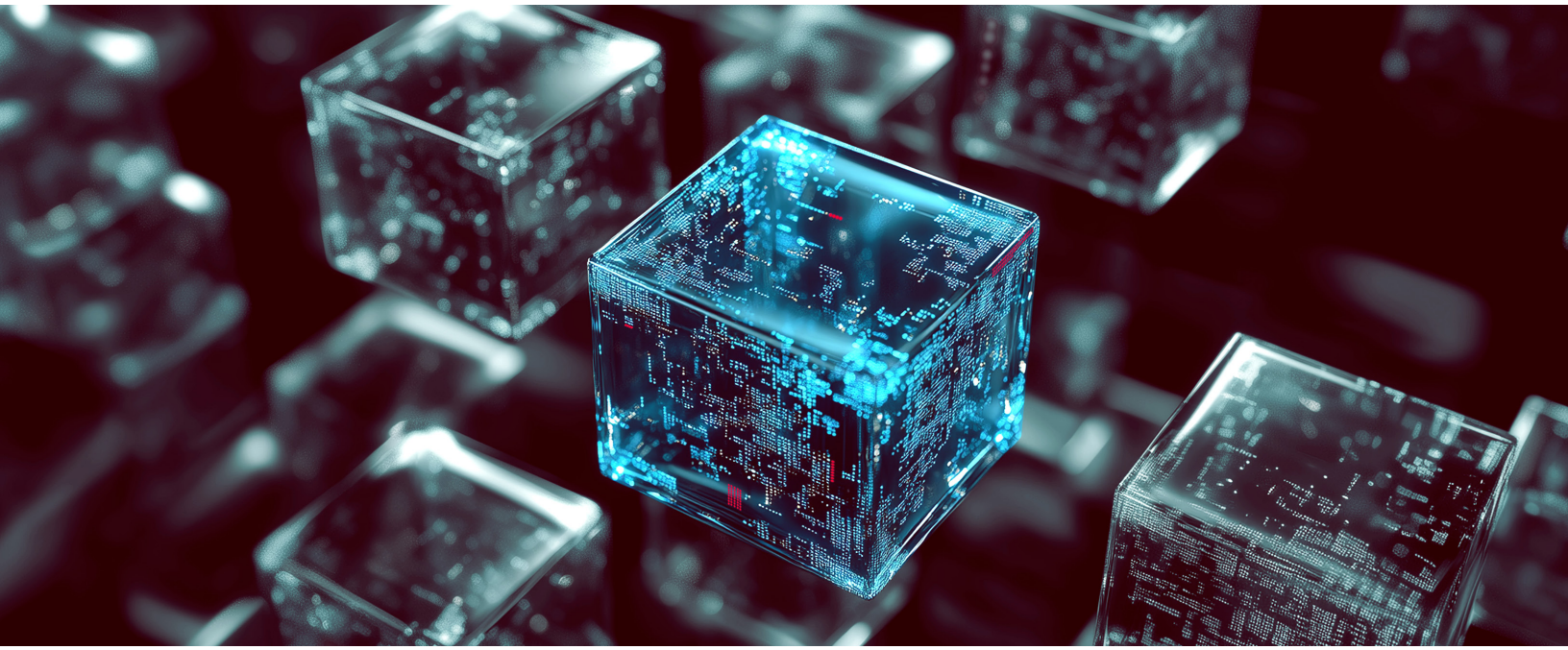
The program, which can be found on GitHub, is continually updated by CISA analysts with CVSS scores and other enrichment information to help the community mitigate vulnerabilities. CISA hopes that the program will encourage vendors to address broader vulnerability classes and facilitate a deeper understanding of software vulnerability trends.

Imagining a Post-CVE Future

Even without the turmoil around the NVD, the changing cyberthreat landscape and the growing problem of software supply chain attacks cast a shadow on security processes that are overly focused on chasing down CVEs. After all, the NVD and other vulnerability repositories present an incomplete picture of software risk.

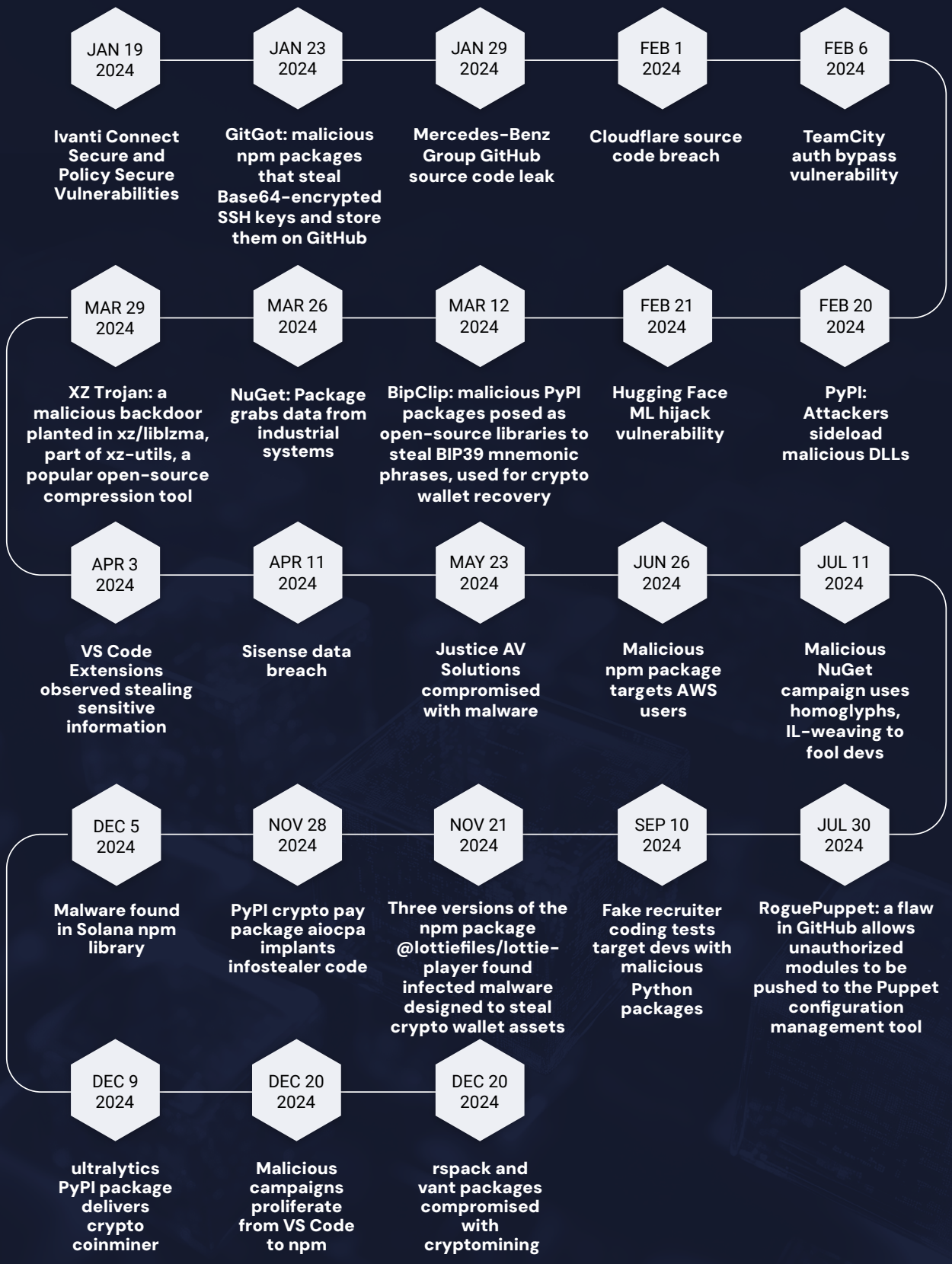
RL Chief Trust Officer Saša Zdjelar recently wrote that the CVE system's coverage is not exhaustive, missing threats in custom, proprietary, or less widely used software components that are often integral to supply chains. Even worse, the CVE system is inherently reactive, focusing on vulnerabilities after they have been discovered and directing scarce resources at reactive responses rather than proactive security measures and best practices that could prevent vulnerabilities from being exploited in the first place.[31]

Properly managing software risk in 2025 means focusing your organization's resources on the small number of flaws that are the most likely to be targeted — flaws such as those listed on CISA's Known Exploited Vulnerabilities (KEV) catalog. It also means broadening your mission to identify and address risks such as attacks targeting developers and development pipelines and threats such as malware or software tampering that have fueled attacks on SolarWinds, 3CX, and others. Such visibility should include the ability to detect the presence of malicious software, tampering, and other unsanctioned code modifications; leaks of development secrets; and the absence of proper software hardening.



[31] Saša Zdjelar, "Why SAFE. Why Now," ReversingLabs, August 1, 2024

Supply Chain Incidents 2024



What Comes Next

AI/ML Supply Chain Risks Get Real

With the rise of the use of generative AI in enterprises — and its wide adoption in software development — security concerns are mounting over how generative artificial intelligence (GenAI) and Large Language Model (LLM) machine learning can be exploited by attackers.

Of course, some of these AI-associated risks are just new variations on long-standing security problems plaguing organizations, such as spearphishing campaigns and malware. More concerning are the many new opportunities that AI and ML technology — and the technology ecosystem that supports them — are creating for malicious actors to infiltrate sensitive development or IT organizations where AI technology is being used. As RL's Dhaval Shah recently noted, the infrastructure attached to large language model (LLM) machine learning presents risks as well. Shah noted that Python's popular Pickle-format serialization files are "inherently unsafe" because they allow embedded Python code to run when the model is loaded.

While helpful for enabling certain AI features, this design also opens the door wide to malicious actors, who can abuse it to execute malicious commands, inject malware onto affected systems, or engage in inbound and outbound communications with malicious infrastructure, putting data at risk.

RL threat researcher Karlo Zanki provided proof of that in February 2025: reporting on the discovery of a malicious technique dubbed "nullif**AI**" in which malicious code was placed in Pickle serialization files, while evading protections built into the Hugging Face open-source platform. When run, the Pickle format serialization files loaded malicious code on systems that deployed them.[32]

Then there's the December 2024 campaign targeting the popular AI library Ultralytics, which saw malicious versions of the library published to PyPI containing code that downloaded the XMRig coinminer.[33] The compromise of the project's build environment was achieved by exploiting a known and previously reported GitHub Actions script injection. The campaign saw the malicious actors compromise the build environment related to the mentioned project and inject malicious code *after the code review part of the process was finished*. That led to a malicious update of a library getting pushed into an open-source project with close to 60 million downloads.

The security community has been working around the clock to address this growing attack vector, with the Open Worldwide Application Security Project (OWASP) Foundation leading the charge. In November 2024, it released the OWASP Top 10 Risks for LLM Applications in 2025,[34] with risks including more than just prompt injection, such as unbounded security, vector, and embedding vulnerabilities; system prompt leakage; and excessive agency.

OWASP's release of CycloneDX v1.6 also signaled a major win for AI and ML security in 2024 with the introduction of a machine-readable format for SBOMs that can be applied to ML models. OWASP also released the LLM AI Security and Governance Checklist, raising the bar for firms that are developing and promoting AI and ML to secure their AI and ML efforts.

RL experts agree: Serialized ML models need to be vetted with the same rigor as any other software package for supply chain risks. This is why it's essential for developer, application security, and third-party risk management teams to use a modern software supply chain security solution that can spot threats such as unsafe function calls and suspicious and malicious behaviors in ML files, particularly in risky formats such as Pickle, before they can impact an organization's infrastructure.

[32] Karlo Zanki, "Malicious ML models discovered on Hugging Face platform," ReversingLabs, February 6, 2025
[33] Karlo Zanki, "Compromised ultralytics PyPI package delivers crypto coinminer," ReversingLabs, December 9, 2024
[34] OWASP Top 10 for LLM Applications 2025, OWASP, November 17, 2024

Organizations Level Up Their Software Supply Chain Security

RL's examination of open-source and commercial software risks over the past year clearly shows that software supply chain risks are growing in both frequency and complexity, posing serious problems for software producers and end-user organizations.

Incidents such as the hijacking of the XZ Utils open-source project, as well as the campaigns to publish, promote, and then flip to malicious the *aiocpa* and *@0xengine/xmlrpc* open-source packages highlight the ability of malicious actors to conduct long-term, sophisticated, hands-on-keyboard software supply chain campaigns. These are designed to give them privileged access to hundreds or even thousands of development organizations.

At the same time, our analysis of widely used open-source packages such as *ultralytics* and *@solana/web3.js* remind us that popularity and a good reputation don't guarantee package security. The fact that widely used packages such as these were found to contain high-numbers of high-severity flaws and, in some cases, were infiltrated with malicious content shows that threats may lurk both in obscure and well-known open-source packages. Developers should not equate package downloads and reputation with package security.

Today's malicious software supply chain campaigns are fueled by endemic security flaws and other risks that exist not just in open-source binaries, but also closed-source commercial software binaries. The hack of the JAVS video-recording software was a prominent example of this, exposing courthouses, prisons, and other institutions to malicious intrusions via trusted and privileged software updates.

All this puts the onus on software producers and end-user organizations (their customers) to level up their software supply chain security by detecting unexpected and unexplained changes in the constitution and behavior of applications and updates, organizations of all kinds can thwart attempted supply chain compromises and keep their sensitive IT assets and data safe.

Nth Party Risk: Thinking Beyond the SBOM

One of the biggest transformations in the software industry in recent years has been the push by both regulators and customers for greater transparency around software risk. The drive for more software transparency is fueled by revelations about sophisticated supply chain attacks such as those carried out on SolarWinds and 3CX, which highlight the brittle nature of the software supply chain and the limitations of the current focus on known vulnerabilities and exploits, rather than software quality, availability, and resilience.

One response has been the call for software publishers to issue software bills of materials (SBOMs) that provide a list of ingredients — including open-source and third-party software — for software applications, updates, and patches. The push for adoption of SBOMs got a big boost when the White House issued its Cybersecurity Executive Order 14028 in May 2021.[35] The EO called on software-development organizations that do business with the U.S. federal government to provide the purchaser of their software an SBOM for each product. It was followed by additional guidance, including Presidential Memorandum M-22-18[36] (September 2022), which laid out the requirements for a software attestation form, and M-23-16[37] (June 2023), which addressed the kinds of software covered by the form.

[35] "Executive Order on Improving the Nation's Cybersecurity," Whitehouse.gov, May 12, 2021
[36] M-22-18 Memorandum for the Heads of Executive Departments and Agencies, Whitehouse.gov, September 14, 2022
[37] M-23-16 Memorandum for the Heads of Executive Departments and Agencies, Whitehouse.gov, June 9, 2023

In the European Union, there was a similar movement. The Cyber Resilience Act, approved in March 2024, requires manufacturers of products with digital elements sold within the EU (meaning both hardware and software) to provide an SBOM, detailing the components within their products as well as dependencies and potential security vulnerabilities.

While both software publishers and end-user organizations have heeded those calls and embraced the concept of SBOMs, there are lingering questions about their efficacy. As RL has reported, federal forms in which suppliers attest to the accuracy of their SBOMs are rife with vague language that weakens their impact.

SBOMs in their current form also strain to capture the full breadth and depth of software supply chain risks, which lurk not just in the code provided by third-party suppliers, but also in the countless companies that supply your suppliers. This notion of "Nth party risk" is a growing concern, especially given the growing complexity of software deployments, with the embrace of cloud computing, a heavy reliance on APIs, and the growth of no-code applications and AI- and ML-derived applications.

> **"We need a new era of software supply chain security management in which universal controls prioritize the mitigation of threats lurking deep in software supply chains."**
>
> Saša Zdjelar, Chief Trust Officer, RL

In short: Staying secure in 2025 is about more than chasing down "high" and "critical" CVEs in your environment. It's about recognizing the threats lurking in the open- and closed-source software and services that are the foundation of your business. The growing sophistication of malicious cyberactors, combined with the declining public-sector support for tracking software risk, demands a response. For software producers and development teams, that means improving your secure development practices, but also your ability to see, assess, and address the risks lurking in the closed- and open-source software you rely on. For end-user organizations, it means expanding your ability to detect risks and threats lurking in black-box commercial software binaries from trusted providers.

Our Methodology

RL's third annual Software Supply Chain Security Report brings together and analyzes public reports and data with non-public, anonymized data compiled by RL analysts and powered by Spectra Core, the world's fastest and most comprehensive software platform for automated static decomposition and analysis of binary files.

CVE and Vulnerability Data

Among the data that contributed to this year's report is vulnerability data, such as registered CVEs, gathered from public and private sources, including the OSV vulnerability library. Vulnerability data was broken down according to the corresponding open-source repository (npm, PyPI, RubyGems, and NuGet). Data was also sorted by the severity of the CVSS score, the year, and so on.

Security Policies

As part of its research on open-source platforms, RL downloaded and processed software packages from the repositories previously listed, analyzing them for violations of one of the scores of information security policies that RL monitors. That dataset included all versions of all the packages available in 2024, not just what was newly published in the last year. Developers can (and do) download older versions of most packages and, therefore, they are affected by the security flaws they might contain. In addition, RL's total package count includes any package versions that RL researchers have a record of from the repository in question, even if the packages are no longer available (deleted or removed from the repository).

OpenSSF Malicious Packages Repository Ratings

RL's binary scanning capabilities have enabled us to become a leading contributor to the Open Source Security Foundation (OpenSSF) Malicious Packages Repository, which is a public repository containing reports of malicious packages identified in open-source package repositories.

Launched in 2023, the OpenSSF Malicious Packages Repository is a collective effort to identify, flag, and remove malicious packages that are lurking on open-source package repositories and that have been linked to a number of sophisticated supply chain attacks traced to both cybercriminal and nation-state actors. Doing so prevents the packages from becoming dependencies of legitimate code or applications.

A wide range of companies contribute to the OpenSSF's Malicious Package Repository, including RL. As part of their work, contributing firms download, install, and execute packages from popular open-source package repositories such as npm and PyPI as they are published. Behaviors such as executed commands and network traffic are observed and compared to a set of known heuristics (patterns) exhibited by malicious packages. Those packages that have suspicious or malicious behavior detected are published to the new Malicious Packages Repository.

In 2024, RL received 7,599 OpenSSF credits for its contributions to the repository, making RL one of the top contributors to the repository, alongside firms such as Checkmarx.

Malicious Package Statistics

When tallying the number of malicious packages, RL package count numbers are deduplicated. A counted package correlates with a unique package name rather than each version of that package. In other words: A malicious package, as far as this report is concerned, is any open-source package that has at least one version that violates a security policy. For example, if an npm package lists 25 different versions going back five years and three of those package versions violate one or more security policies, RL counts the violation once — package X violated a security policy — not three times. Finally, malicious packages are grouped based on the creation timestamp of the malicious version, not when the malware or tampering was first detected, since the creation timestamp gives a more accurate picture when the actual problem or threat first appeared.

Outside of the data RL compiled from its own scans and research, this report also references the work and findings of other cybersecurity industry players to identify trends, with RL correlating, confirming, and (sometimes) overriding the findings of competing firms.

About RL

ReversingLabs is the trusted name in file and software security. We provide the modern cybersecurity platform to verify and deliver safe binaries. Trusted by the Fortune 500 and leading cybersecurity vendors, RL Spectra Core powers the software supply chain and file security insights, tracking over 422 billion searchable files daily with the ability to deconstruct full software binaries in seconds to minutes. Only ReversingLabs provides that final exam to determine whether a single file or full software binary presents a risk to your organization and your customers.

A Complete Approach to Software Supply Chain Security

Identify issues and exposures before release.

CLICK HERE >

Assess and Manage Third-Party Software Security Risk

Find hidden threats before deployment or updates.

CLICK HERE >

Understanding Complex Binary Analysis

Learn more about how Complex Binary Analysis helps identify malware and malicious code, without the need for source code.

CLICK HERE >

Leader's Guide to Software Supply Chain Security

New Gartner® report indicates that software supply chain attack costs will see a 200% increase. This report provides a three-pillar framework to ensure broad protection.

CLICK HERE >

Get started!

To learn more about ReversingLabs Software Supply Chain Security capabilities and solutions

REQUEST A FREE TRIAL

reversinglabs.com

RPT-Rev-03.12.25

REVERSINGLABS

Worldwide Sales +1.617.250.7518
sales@reversinglabs.com