mend.io - *Special Report: Software Supply Chain Malware*

Attacks Move Beyond Vulnerabilities

Malicious packages are a growing threat, and they may already have infiltrated your applications.

Key findings:

- **The fox is in the henhouse.** Mend.io's 360 degree malicious package protection reveals thousands of malicious packages in existing code bases.
- **Malicious package attacks skyrocketed.** The number of malicious packages published to npm and rubygems in 2022 **ballooned 315 percent** compared with 2021.
- **Malicious package versions grew 147 percent from 2021 to 2022.**
- **Malicious package exfiltration attacks are on the rise.** Risk vector analysis reveals a growing trend of exfiltration attacks that can combine several risk vectors — and often succeed. 85 percent of malicious packages discovered in existing applications were capable of exfiltration – causing an unauthorized transmission of information.

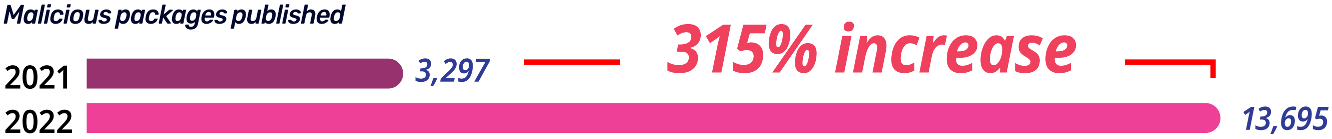
Malicious Package Trends

When it comes to applications and software, the key word is 'more.' Driven by the needs of a digital economy, businesses depend more and more on applications for everything from simplifying business operations to creating innovative new revenue opportunities. Cloud-native application development adds even more fuel to the fire, driving growth in the application development market that is forecast to reach $1.6 billion by 2030.

However, that word works both ways: Those applications are often more complex and use open-source code that contains more vulnerabilities than ever before. Then too, threat actors are creating and using more attack methods and techniques, often in combination.

Ultimately, we end up with a smorgasbord of attack opportunities for threat actors, who use an expanding quiver of risk and attack vectors to build more accurately targeted attacks. In other words, don't expect to see these numbers do anything but increase

Malicious packages published



Case in point: The multi-digit jump in monthly attack numbers from 2021 to 2022. The totals really started to ramp up in October of 2021, and we haven't looked back since. The 2021 monthly attack low was 13 in January. For 2022, it was 494.

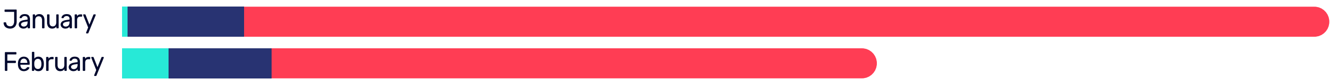
Malicious packages detected per month

Month	***2021 Total malicious packages detected per month***	***2022 Total malicious packages detected per month***
January	13	530
February	203	607
March	224	1,606
April	89	494
May	20	500
June	32	2,253
July	76	2,985
August	161	1,727
September	280	803
October	585	636
November	976	947
December	638	607
Total	***3,297***	***13,695 (315% increase)***

2023 starts with a bang

When we compare January-February malicious package activity over the past three years, it’s clear that one of these years is not like the others. As it turns out, the enormous activity of early 2023 came from spam attacks launched by several actors. One attack came from new npm user 'Zalastax', who our researchers observed uploading nearly 20,000 packages. According to the package's readme, it 'depends on every package in npm,' and the author even used a script to make that happen automatically. The team reached out to the author to understand his intention, and he replied that it was a "silly project with no purpose and done out of curiosity." In a separate attack, attackers jammed the npm repository with more than 15,000 spam packages that included phishing links.

Total malicious packages detected



	● **2021**	● **2022**	● **2023**
Jan.	13	530	**59,919**
Feb.	203	607	**27,133**

Source: Mend Supply Chain Defender

Version Control

On average, malicious packages had about four versions. We saw a trend toward first releasing a non-malicious version before releasing malicious versions. There are a couple of interesting things going on here. The initial clean release is likely due to a common belief that different, more stringent assessments are run on first releases compared with updates. We also see version releases reflect a learning curve. It’s hard to craft a package that has a high probability of exfiltration, so malicious actors often tinker and adjust their code between releases.

Malicious package versions published, 2021–2022

Month	***2021 Total malicious packages detected per month***	***2022 Total malicious packages detected per month***
January	28	1,895
February	461	2,216
March	373	4,118
April	194	6,132
May	39	2,033
June	115	4,975
July	203	6,407
August	930	6,961
September	1,323	2,590
October	4,013	1,960
November	4,593	2,409
December	2,843	1,203
Total	***15,115***	***37,379***

Source: Mend Supply Chain Defender

Malicious Package Attack Vectors

In order to deliver a malicious payload via an open source package, attackers must find a way to get the package downloaded. That means attackers need to find a way to fool someone – or something – into downloading it. Let's take a closer look at several of the ways that malicious packages can be downloaded. There are four basic attack vectors for malicious packages: brandjacking, typosquatting, dependency hijacking, and dependency confusion.

Brandjacking is an activity whereby an attacker acquires or otherwise assumes the online identity of another company or an owner of a package and then inserts a malicious code. It doesn't necessarily mean he actively steals something, but just takes advantage of an opportunity to take ownership related to the brand name.

In a **typosquatting** attack, an attacker publishes a malicious package with a similar name to a popular package, in the hope that a developer will misspell a package name and unintentionally fetch the malicious version.

With **dependency hijacking**, an attacker obtains control of a public repository in order to upload a new malicious version.

Dependency confusion happens when a malicious package in public repositories has the same name as an internal package name. The attacker then uses this so-called feature to trick dependency management tools into downloading the public malicious package rather than the private, non-malicious package.

Brandjacking and typosquatting were the original malicious package attacks, and they remain an integral part of the attack vectors used today. Dependency hijacking and dependency confusion are more recent additions.

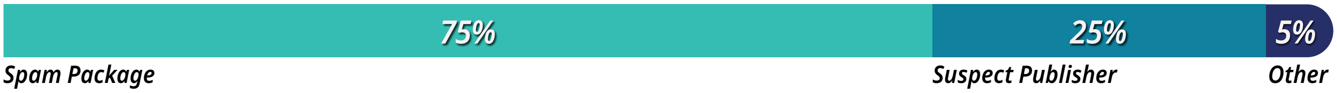
Attack Vectors and Risk Vectors: What's the Difference?

They may sound similar, but these terms describe separate things.
An attack vector is the method of attack used by a bad actor, while the term risk vector is used to indicate the risk of a given package.
Here's an example: Say we detect a malicious package with a name that's one letter off from that of a popular package; that malicious package contains a malware dropper and malicious obfuscated code.
The attack vector used is typosquatting — the attacker is hoping that a developer will misspell the package name and call the malicious version by accident. The malware dropper and obfuscated code are the risk vectors.

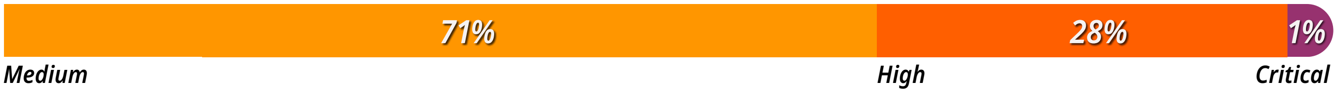
Risk vectors

Malicious package risk vectors have been steadily growing in both quantity and sophistication, and we wanted to get a clearer picture of usage and deployment. Using an algorithm-aided selection process, the Mend.io research team collected versions of 61,009 malicious packages from npm and RubyGems to analyze the risk types posed by malicious packages. By tagging the malicious packages according to risk types, then analyzing the frequency and distribution of these risk types, the team created a profile of risk vectors and risk severity across the data set. Spam packages emerged as the most prevalent threat source, followed by suspect publishers (publishers that had previously been known to release one or more malicious packages).

Top Risk Vectors



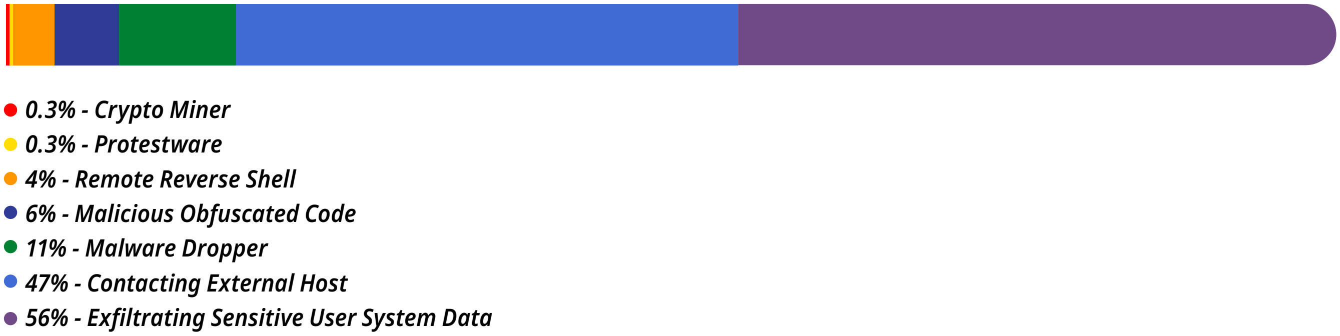
Severity breakdown



A closer look at 'other risk vectors'

Although small, the possibilities hidden in the '**other risk vector**' category intrigued us, so we dug into more granular detail to get a clearer sense of how threat actors were using and combining these more targeted risk vectors. Of this subset, attacks using exfiltration, often combined with contacting external hosts, emerged as a concerning trend.

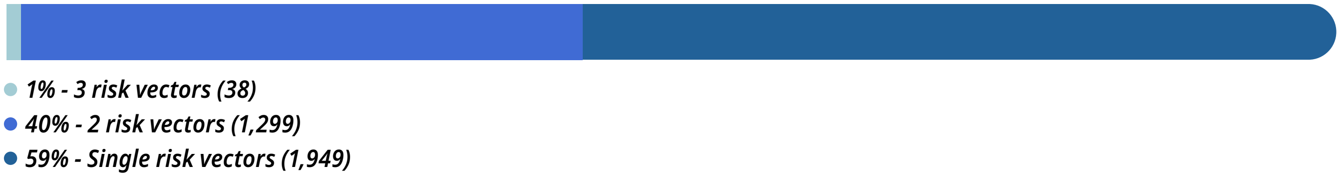
Other Risk Vectors



Note: Total is greater than 100 percent due to multiple vectors used in some attacks.

These attacks illustrated the increasingly sophisticated methods used by a growing number of attackers. Nearly 70 percent were of high or critical severity, and more than 40 percent contained more than one risk vector. For example, attackers combined exfiltration with malicious obfuscated code to avoid detection, and some added malware droppers to install malicious code and spread it across the code base to make it harder to remove.

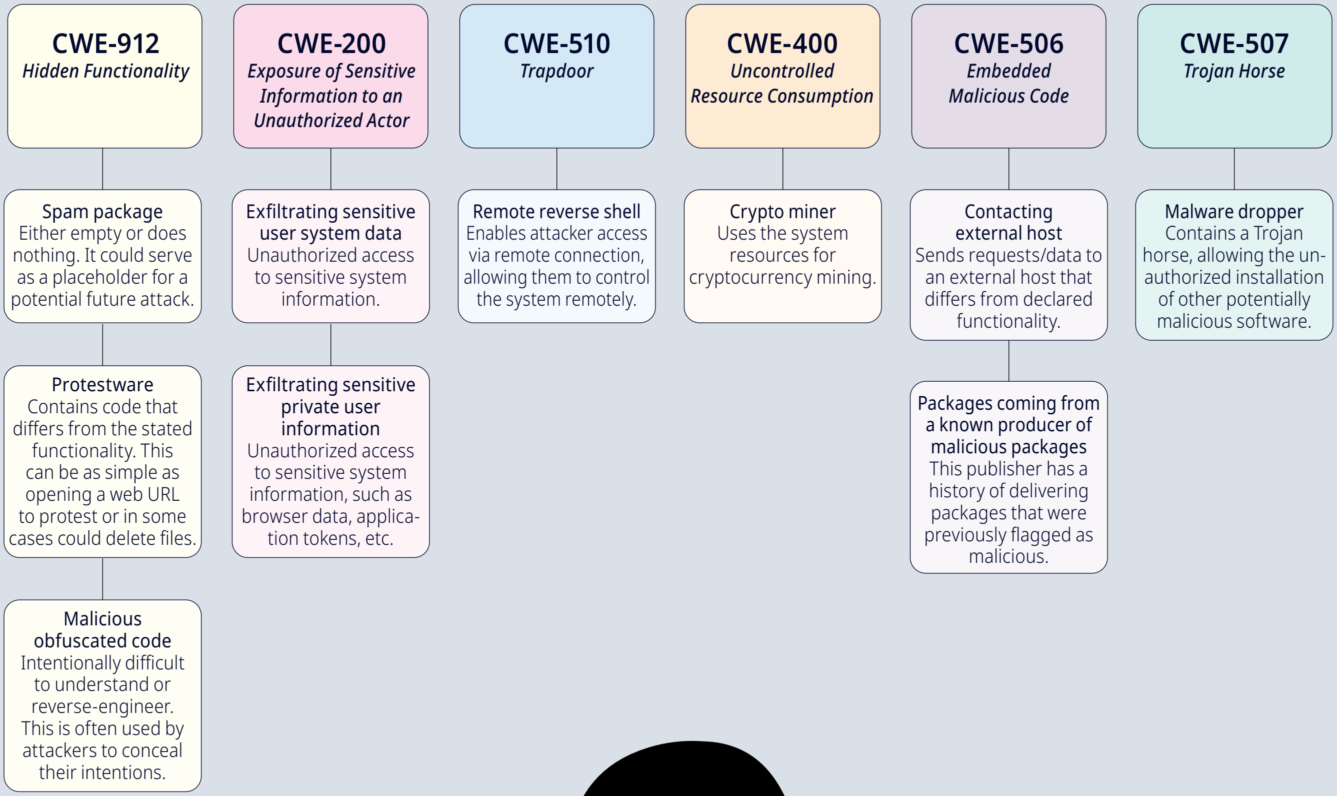
Multivector attacks



68% of attacks critical or high severity

A Family Tree With Poison Apples

Here's a look at how different attack risk factors relate to each other in the malicious package family tree.



CWE-912
Hidden Functionality

Spam package
Either empty or does nothing. It could serve as a placeholder for a potential future attack.

Protestware
Contains code that differs from the stated functionality. This can be as simple as opening a web URL to protest or in some cases could delete files.

Malicious obfuscated code
Intentionally difficult to understand or reverse-engineer. This is often used by attackers to conceal their intentions.

CWE-200
Exposure of Sensitive Information to an Unauthorized Actor

Exfiltrating sensitive user system data
Unauthorized access to sensitive system information.

Exfiltrating sensitive private user information
Unauthorized access to sensitive system information, such as browser data, application tokens, etc.

CWE-510
Trapdoor

Remote reverse shell
Enables attacker access via remote connection, allowing them to control the system remotely.

CWE-400
Uncontrolled Resource Consumption

Crypto miner
Uses the system resources for cryptocurrency mining.

CWE-506
Embedded Malicious Code

Contacting external host
Sends requests/data to an external host that differs from declared functionality.

Packages coming from a known producer of malicious packages
This publisher has a history of delivering packages that were previously flagged as malicious.

CWE-507
Trojan Horse

Malware dropper
Contains a Trojan horse, allowing the unauthorized installation of other potentially malicious software.

Malicious Packages In Existing Code Bases

If malicious packages are seen as invaders, one scenario is even more disastrous than inviting these invaders into your code during development: having invaders already in your code base, wreaking havoc while you remain unaware.

This scenario isn't just a hypothetical.

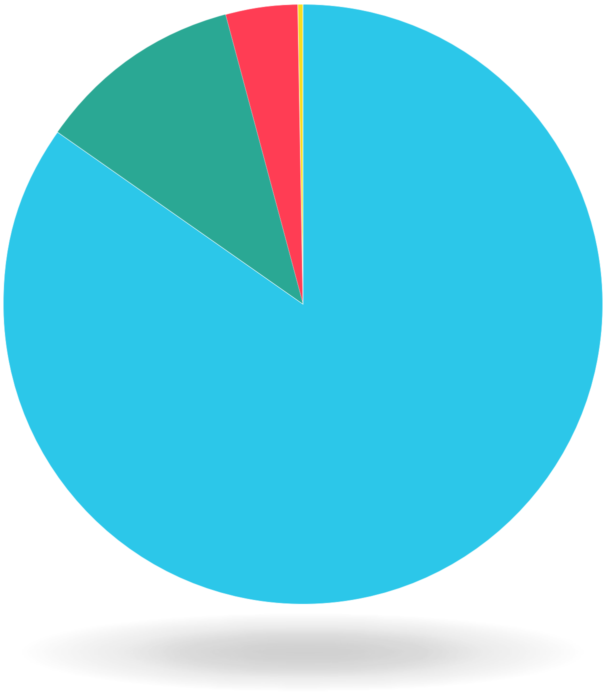
At Mend, our researchers recognized the need not only to detect malicious open source software and stop it from entering registries and repositories, but also to shine a light on existing code across applications that have already been built and released. When Mend launched a new feature to detect malicious packages that had been inserted into existing applications, we found thousands of instances of these packages in customers' code bases.

The identified packages had entered via repositories and registries that didn't use an existing Mend solution to block malicious open-source software.
When our researchers took a closer look at which packages were present, we found that they fit into four categories based on the motivation of the developer of the malicious package: exfiltration, developer sabotage, protestware, and spam.

Malicious package types seen in existing code bases, by motivation

- 🔵 Exfiltration 85%
- 🟢 Developer Sabotage 11%
- 🔴 Protestware 4%
- 🟡 Spam 0.1%

Source: Mend.io SCA



Nearly 85 percent of malicious packages discovered in existing applications were capable of **exfiltration** – causing an unauthorized transmission of information.

For organizations that rely on protected or confidential information, exfiltration represents an existential risk. Once a malicious package capable of exfiltration makes its way into a code base, it can potentially spread widely and create major breaches with both reputational and regulatory implications.

Perhaps counterintuitively, exfiltration's dangers exist in part because these packages don't wreak enough havoc on applications. Threat actors seeking protected information can maximize the amount of information they collect before a malicious package is detected and removed by ensuring their package works quietly, without attracting undue attention.

However, staying quiet isn't always what a threat actor hopes to achieve by distributing malicious open source software.

Some malicious package creators intend for their users to find out – like the developer responsible for **developer sabotage** in popular npm libraries "colors" and "faker." This developer intentionally committed an infinite loop that broke the libraries – and the millions of projects that depended on them.

This act of developer sabotage applied to some of npm's most popular libraries was responsible for over 11 percent of the malicious packages detected in Mend customer code bases.

The developer, Marak Squires, explained that his intention was to stop supporting large companies with his "free work." While so far Squires is a lone wolf (and the project has now been forked so that his work is no longer required), rogue open source developers could pose an ongoing threat to the proprietary applications that depend on their projects.

Protestware represents another emerging type of malicious package deployed by developers who want organizations to notice. The war between Russia and Ukraine gave rise to several malicious packages in this category.

Some of these protestware packages print messages about the war without altering files, but others are designed to identify IP addresses located in Russia and Belarus, then take specific actions like deleting and overwriting files.

For enterprises with a widespread global presence, protestware is an emerging area of risk that will likely evolve and change with each conflict. Its worst impacts may involve different geographic areas with each conflict, impeding organizations' ability to trust that the same piece of software will work the same way in every region.

Finally, spam packages are either empty or do nothing. They could serve as a placeholder for a potential future attack.

360 Degree Protection From Malicious Packages

Malicious packages are just as harmful as malware, but most companies have not built effective detection and blocking into their application security strategies. At Mend, we believe that such protection is an integral aspect of open source software security, which is why Mend's Software Composition Analysis (SCA) solution provides complete protection against malicious packages. Our technology:

- Proactively blocks malicious software before it's downloaded and detected
- Alerts on malicious software that may already be in the code base.

Our 360 degree protection helps developers secure against the growing threat of malicious packages without compromising speed or agility. Even better, it's based on the expertise of Mend Research. In the last three years, the Mend research team has successfully identified 100 percent of malicious RubyGem packages and 99.8 percent of npm packages. Mend researchers have also been the first to identify a number of new malicious packages, such as the dYdX crypto malicious package attack, so that organizations were able to take action against these security issues.

Organizational Impact: Malicious Packages Are More Dangerous than Vulnerabilities

When attackers gain access to your applications via a malicious package, they can impact your organization in multiple ways. Once a developer downloads a malicious package, how much damage it does will depend on several key factors:

1. **Intent** – When threat actors infiltrate using a malicious package, their intent substantially determines the impact. A threat actor trying to inform people about a war or protest action with annoying messages has a lower overall impact than one trying to steal information or turn developers' machines into cryptocurrency miners.
2. **Organization type** – Attacks designed to exfiltrate personal information will have a larger, potentially long-term impact on companies trusted with sensitive data, ransomware attacks that disable systems can have outsize impact in organizations like hospitals, where lives depend on uptime.
3. **Duration** – When malicious packages are discovered quickly and removed completely, the damage they cause can be limited. The greatest damage can be caused by packages that remain undetected for months or years while quietly delivering their payload.
4. **Spread** – Some of the most dangerous malicious packages are designed to provide initial access to a network, at which point the threat actor can move laterally through systems to steal passwords or protected information in order to gain even more access.

Unlike vulnerabilities — which can and do often exist for months or years in application code without being exploited — a malicious package represents an immediate threat to your organization.

Think of it like this: If your applications and organization are a house, then attackers are like burglars. A vulnerability is your proverbial unlocked window: It could let a burglar in some day, but that's only a possibility. On the other hand, a malicious package is like accepting a FedEx box that already has the burglar inside.

Conclusion

The increasing threat of malicious package attacks adds further urgency to the growing need for a new approach to application security programs. Certainly, the public sector is taking an increased interest in cybersecurity. Legislation such as the proposed Securing Open Source Software Act in the U.S., the Cyber Resilience Act from the European Union, and the Australian Cyber Security Strategy drives home the need for global organizations to rethink AppSec. The fact is, applications are the lifeblood of the global economy, and threat actors know it. And unlike open-source vulnerabilities — which can and do often exist for months or years in application code without being exploited -– a malicious package represents an immediate threat to your organization. Malicious packages don't enter your code base to do nothing. If they're in your application, your organization already has a problem. Organizations may have different thresholds for acceptable risk from vulnerabilities, but the only acceptable number of malicious packages to have in your application code is zero.
The question is, does your application security strategy have a plan in place to defend against this new threat?

How can mend.io help?

Ready to discover how to secure the software supply chain with modern application programs?

Learn More

About mend.io

mend.io, formerly known as WhiteSource, effortlessly secures what developers create. Mend uniquely removes the burden of application security, allowing development teams to deliver quality, secure code, faster. With a proven track record of successfully meeting complex and large-scale application security needs, the world's most demanding software developers rely on Mend. The company has more than 1,000 customers, including 25 percent of the Fortune 100, and manages Renovate, the open- source automated dependency update project.
For more information, visit www.mend.io, the Mend blog, and Mend on LinkedIn and Twitter.

FORRESTER
WAVE LEADER 2021
Software Composition Analysis
CYBER SECURITY EXCELLENCE AWARDS WINNER 2023
Application Security Solution - NA (100-499 employees)
CYBER SECURITY EXCELLENCE AWARDS WINNER 2023
Vulnerability Management - NA (100-499 employees)
CYBER SECURITY EXCELLENCE AWARDS WINNER 2023
Cybersecurity Startup - NA (100-499 employees)
Leader WINTER 2023
Leader PeerSpot
aws partner network
Advanced Technology Partner
DevOps Competency
RED HERRING 100 WINNER N. AMERICA
InfoWorld 2022 AWARDS
TECHNOLOGY OF THE YEAR