DARKReading

WHITE PAPER

Malicious Packages: A Growing Threat to the Software Supply Chain

Sponsored by

mend.io

Table of Contents

The global economy runs on software applications, and their function and security is critical to every company's success. Many applications have exploitable vulnerabilities that modern defenders struggle to effectively detect and remediate. In addition to the growing number of vulnerabilities, today's security teams face the emerging challenge of malicious packages.

Software developers build approximately 80% of software applications using open-source code, which opens up a world of opportunity for today's threat actors.[1] Code package repositories such as npm and RubyGems allow anyone to store or publish packages, and unfortunately that can include packages containing malware. These are known as malicious packages. Today, threat actors create and use malicious packages to launch attacks with the goal of slipping past defenses, activating a harmful malware payload on an initial victim and helping threat actors gain access to wider supply chains.

Malicious packages employ tricky evasion techniques to breach supply chains and escape notice. Once downloaded, the malware in malicious packages can use a number of risk vectors to compromise systems while spreading down the supply chain to initiate further information theft, cryptomining, cryptojacking or creating botnets within a new victim's network. As a relatively recent phenomenon, malicious packages benefit from a lack of developer awareness and existing trust in open-source repositories and established supply chains. Worse yet, most current application security solutions and tools aren't equipped to identify and remediate these malicious packages.

Time is critical for organizations looking to secure their software supply chains, as malicious package attacks are increasing at an alarming rate. From 2021 to 2022, there was an identified 315% increase in the number of malicious packages published to the npm and RubyGem registries.[2] Companies must prioritize this rapidly growing problem. This starts by learning about malicious packages, exploring why they are dangerous, what contributes to their rise, why attempted solutions are failing and where your organization can look for real solutions to this software supply chain threat.

---

[1]Mend.io Open Source Risk Report. https://www.mend.io/wp-content/media/2022/12/Mend-Open-Source-Risk-Report.pdf
[2]Mend.io Malicious Packages Special Report. https://www.mend.io/malicious-package-protection-lp/?utm_sf_camp=blog

The Growing Threat of Malicious Packages

Today's cybercriminals and rogue security researchers never rest when it comes to searching for easier, faster, and more rewarding methods of attacking and breaching cyber defenses. Whether looking to corrupt target systems, breach networks for lateral movement, steal information or perform reconnaissance for future attacks, modern threat actors continue to raise the bar in their effort to access an organization's sensitive data. Malicious packages can deliver maximum bang for the bad guy's buck. It can be as simple as hiding a malware payload in open-source code and tricking a careless developer into using it, or elevating bugs in package manager systems and then benefitting from the opportunities afforded by the scale of a corrupted software supply chain.

According to Mend.io team sources, 2022 saw 13,695 individual malicious package supply chain attacks based on npm and RubyGem registries. Q3 in 2022 saw a 79% increase over Q2 in malicious packages published.[3] The recent explosion in malicious packages is easy to understand, as modern software supply chains provide excellent targets. These ecosystems offer numerous attack surfaces, including software dependencies, version control systems, testing and deployment tools, cloud-hosting providers, and numerous applications created with open-source code. The popularity of open-source use with developers also presents an opportunity as threat actors flood repositories like npm and RubyGems with 10 malicious packages per day or more according to the Mend.io Open Source Risk Report.[4]

Spamming is popular for delivering malicious packages because smart threat actors understand the value of scaling attacks with non-targeted distribution techniques, hoping for the one hit that opens the floodgates of downstream access. Most malicious packages work a reconnaissance angle, usually exfiltrating information for immediate financial gain or to aid in future targeted attacks. Jeffrey Martin, vice president of product at Mend.io, explains what security teams see today. "Spam packages are by far the most common type of malicious package, since that is the nature of spam. Packages created by known publishers of malicious content are next, followed by malicious packages that exfiltrate sensitive information from target systems. The most difficult types of malicious packages are those that combine risk vectors such that they use avoidance techniques as well as malware droppers that automatically activate and disseminate the malicious payload."

Attackers also understand the potential rewards of a patient trial-and-error approach as it relates to continually improving their odds of success. They tirelessly use package tester releases to help streamline processes, eliminate errors, and improve code writing in the effort to inject more malicious packages into software supply chains.

[3]Mend.io Open Source Risk Report. https://www.mend.io/wp-content/media/2022/12/Mend-Open-Source-Risk-Report.pdf
[4]Mend.io Open Source Risk Report. https://www.mend.io/wp-content/media/2022/12/Mend-Open-Source-Risk-Report.pdf

ATTACK VECTORS

Threat actors use malicious packages to attack and infect software supply chains as far as possible upstream for maximum distribution and damage downstream. Attackers currently employ the following basic types of attack vectors for malicious packages:

Typosquatting

This attack vector involves an attacker taking or "squatting" on a known, trusted package's name and attempting impersonation with slight typo changes designed to trick developers. A real-life example would be the case of reac1 and reect1, where these two packages tried to mimic research test packages, add new users to an operating system, and send outgoing http requests.[5]

Dependency confusion

In this attack vector, the threat actor creates a public repository package with the identical name of an internal package within the intended target system. The intention is to trick the target's dependency management tools into downloading the malicious public package instead of the safe internal one. Dependency confusion is a more complex attack vector but has massive potential for damage if successful. A real-life example of a successful dependency confusion vector attack is the mrg-message-broker case.[6] In this example, threat actors designed a malicious package similar to the grubhubprod-cookbook package intending to steal environmental data.

Brandjacking

In brandjacking attacks, the attacker steals control of the online brand identity of a popular legitimate package and then inserts malicious code into the original known and trusted package. This type of attack vector is challenging for malicious actors to pull off, but can deliver a high infection rate due to the popularity of known packages. An example of brandjacking occurred when attackers took control of popular npm packages style-resources-loader and sass-loader.[7] When an unsuspecting victim installed and activated the new malicious package, binaries downloaded a third-party component to collect system information, allow a remote host connection, and then allow remote code execution. Another well-publicized example is the September 2022 case involving the crypto company dYdX, where an attacker used a stolen employee npm account to begin sending new versions of legitimate packages owned and published by dYdX.[8]

Dependency hijacking

Dependency hijacking is an attack vector where a threat actor obtains control of a public repository in order to upload a new malicious version of an existing package.

Fake security research

This attack vector takes advantage of a popular repository policy that allows companies to create packages solely for research purposes. Attackers use that policy to create fake security research packages designed only to collect information on the sly.

---

[5]Mend.io NPM Threat Report. https://www.mend.io/wp-content/media/2022/02/Mend-npm-Threat-Repot-1.pdf
[6] Mend.io NPM Threat Report. https://www.mend.io/wp-content/media/2022/02/Mend-npm-Threat-Repot-1.pdf
[7] Mend.io NPM Threat Report. https://www.mend.io/wp-content/media/2022/02/Mend-npm-Threat-Repot-1.pdf
[8] Mensfeld, M, (September, 2022). Popular Cryptocurrency Exchange dYdX Has Had Its NPM Account Hacked. Mend.io. Retrieved April, 2023 from Mend.io

THE MOTIVATION BEHIND MALICIOUS PACKAGES

Why do threat actors put forth such effort to slip malicious code into software products and attempt to infect vast supply chains? The short answer is to make money, whether the payoff is immediate or long term. Most malicious packages are designed to stealthily infiltrate software, spread rapidly down a supply chain, and immediately begin gathering environmental information about every infected victim. Attackers use keyloggers, screen scrapers, spyware, and adware to gather a wealth of sensitive environmental information which can be quickly sold on the dark web or used to leverage and support future attacks involving more lucrative types of financial fraud.

For example, the Russian tech giant Yandex suffered a data leak in which source code from other global giants such as Google, Amazon, and Uber reached the public domain.[9] Armed with such sensitive data, malicious actors waste no time analyzing leaked code for vulnerabilities they can exploit in future attacks up or down the supply chain.

Control is another underlying theme driving the motivation for malicious packages. Attackers see the installation of malicious code and its flow down a software supply chain as a means to gain control over multiple devices and systems for whatever future nefarious activity promises the most significant payback. Installing a connectback shell enables threat actors to receive remote commands for execution on a target device. This three-step process includes connecting to the attacker's server, receiving execution commands, and sending back execution results. Using malware-loaded packages to install bots on target devices allows "bot herders" to expand botnets and launch future bot attacks which help steal more data and support cryptomining.

Cryptomining can be another lucrative benefit and powerful motivator behind malicious packages. Mining cryptocurrency is a resource-intensive undertaking, so attackers will attempt to silently gain control of multiple devices with widespread, non-targeted umbrella attacks to steal technical resources to use for mining cryptocurrency.

[9] Ben Ari, T, (January, 2023). Yandex Data Leak Triggers Malicious Package Publication. Mend.io. Retrieved April, 2023 from Mend.io

INCREASING SOPHISTICATION

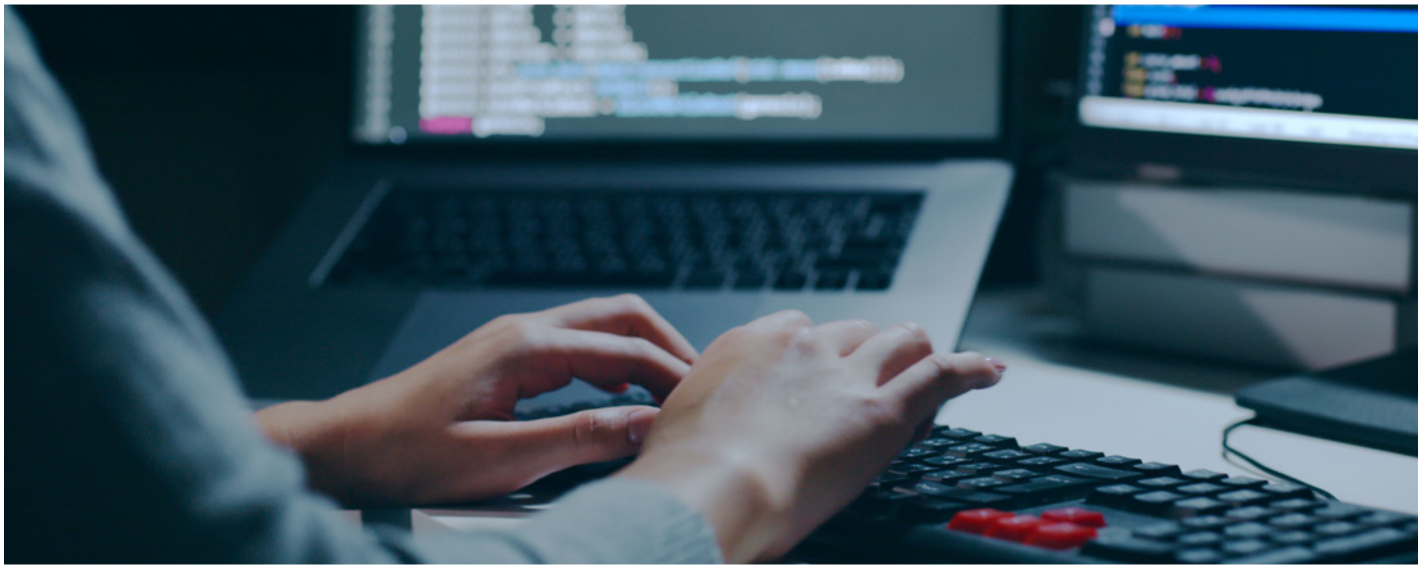
Despite a relatively short history, malicious packages are growing in terms of sophistication and potential negative impact across a supply chain. Early malicious packages resembled early malware as they were often indiscriminately targeted and relatively simple to remove. Not so anymore, as some malicious package attackers now apply an organized, calculated approach that evolves based on collected feedback from a three-release launch process. First tester versions may not contain harmful payloads, with their purpose often being just to navigate the usually more stringent initial defense assessments. Malicious cargo will typically load in the following package versions which include improved code that's been adjusted for better exfiltration, distribution, and downstream damage.

The use of evasion techniques by attackers is becoming more common and complex, complicating matters for security teams and developers. Most attackers rely on four standard evasion practices to disguise malicious packages: pre and post-install scripts, basic evasion techniques, shell commands and basic network communication techniques. Bad actors now layer intermediate evasions like code obfuscation and time delays over these more basic techniques to throw off dynamic threat analysis searching for malicious activity.

Today, more packages contain telemetry for data collection. Attackers now take full advantage of the noise and confusion created by "dependency hell," hiding harmful code in dependencies attached to valid content. They also design packages to bypass automatic detection tools, sending them from legitimate hosting providers or mimicking popular brand names to deceive their victims. Another level of sophistication utilizes packages' stateless and shifting nature to challenge defense teams. Safe packages downloaded today may become dangerous with malicious code tomorrow, as Maciej Mensfeld, principal product architect at Mend.io, explains: "Packages released today may not have many functionalities, and if you analyze the packages initially, they won't present any malicious behaviors. However, they may become malicious in a week or five weeks. This ability makes it much harder because you have to reevaluate packages daily."

Attackers employ increasingly sophisticated techniques to inject malicious packages into the software supply chain. Still, once inside, the objective is simple—silently infect as far and fast down the supply chain as possible. The greatest danger lies in the inherent developer trust in existing and established supply chains, allowing attackers a significant opportunity and advantage. Attackers seek to enter a supply chain upstream, leveraging this trust to allow the spread of their malicious packages downstream to impact a vast network of unsuspecting victims.

Unlike generic malware, malicious package attacks do not commonly use persistence techniques or vulnerability exploitation on infected devices, although some have been observed. Methods of deployment, execution, and communication still remain basic in many cases. However, looking forward we can expect the sophistication of these attacks to continue to grow.



Increased Risk

Modern organizations battling the growing threat of malicious package supply chain attacks face increased risk driven by three main factors: the widespread **adoption of open-source coding, lack of developer awareness and commitment to security,** and **application security program tools that fail to defend against malicious packages**.

Today's developers source between 70 and 90 % of the code used in software applications from open-source code repositories like npm and RubyGems.[10] According to Mend.io team experts, the npm repository alone offers around 3,190,220 code packages for developers to freely use in the creation of new applications. Using this open-source code makes life easier for developers, and many treat open-source registries like modern app stores, shopping for code with trust in its legitimacy and safety. This free use of open-source code creates a considerable security risk due to open-source registries' unrestricted and uncontrolled nature. Threat actors take full advantage of this opportunity as popular registries like npm and RubyGems continue to see a consistent flow of malicious packages daily.

A lack of developer buy-in regarding the importance of application security also increases the risk of malicious packages infecting a software supply chain. Balancing the need for security with demanding development deadlines can prove difficult for many teams and, unfortunately, security can suffer. Also, organizations traditionally incentivize developers to focus on software functionality and features, so they're not naturally geared toward prioritizing security. As a result, developers don't always check all open-source code for updates or vulnerabilities, and consistency checks aren't as routine as they should be—which all adds up to a greater opportunity for threat actors.

[10] Mend.io Open Source Risk Report. https://www.mend.io/wp-content/media/2022/12/Mend-Open-Source-Risk-Report.pdf

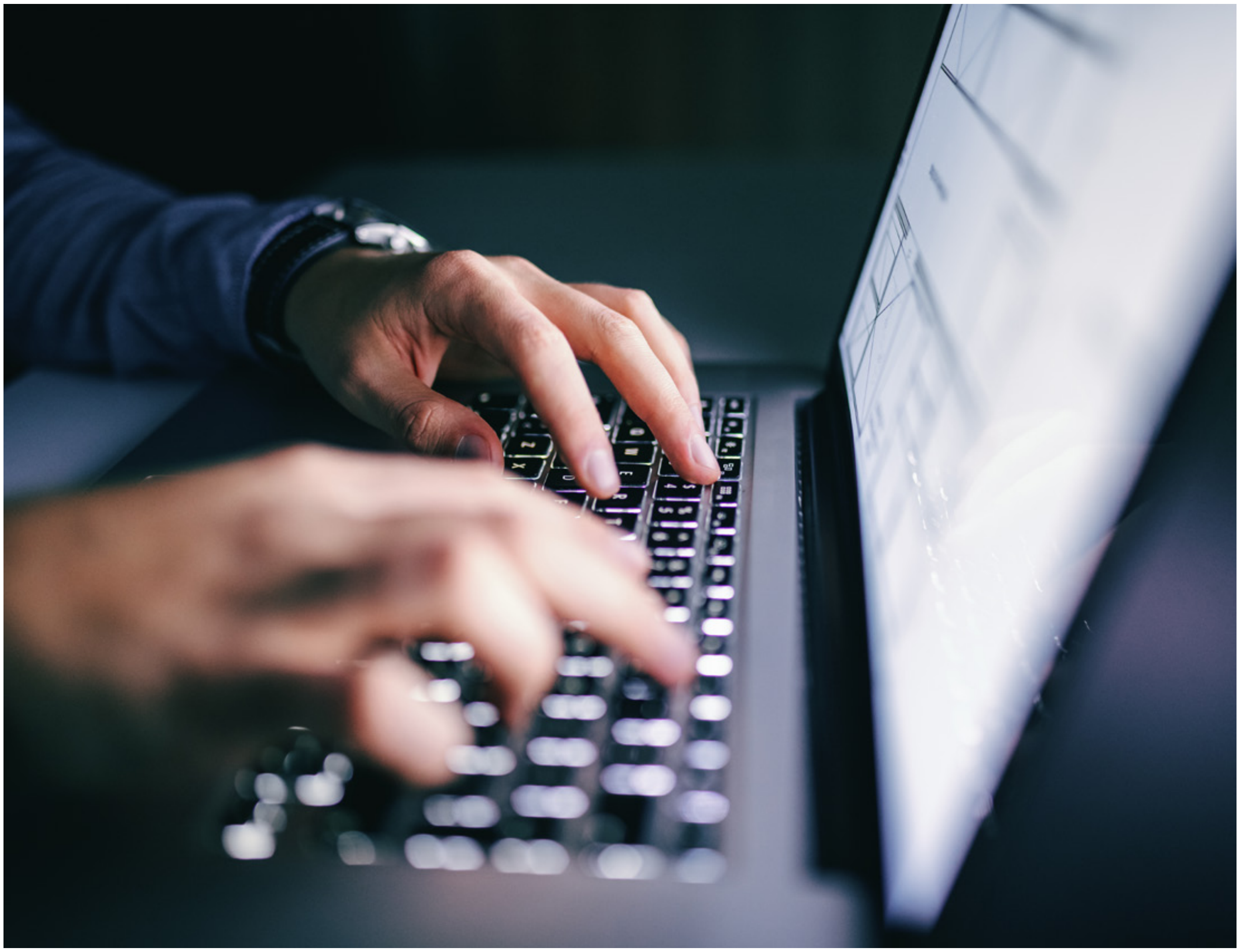
Failing Solutions

Many modern organizations struggle to combat malicious packages. "It's not that malicious packages are difficult to detect and remediate so much as that people lack the knowledge and technology necessary to do so in a timely manner," says Martin. "These are active attacks; being reactive is not enough, and any delay is too much."

Fast responses are critical when battling these attacks, and security teams often lack both the awareness and the tools needed to respond promptly. For example, in attack vectors such as dependency confusion, package management tools automatically download malicious code, leaving developers unaware of a malicious package's existence. The time lag this creates can be devastating to defenses.

Today's defenders find that current solutions, tools and approaches aren't enough to defend against the threat of malicious packages. "There's a general consensus that the tools out there and the solutions we're attempting right now aren't measuring up," Mensfeld says. "We don't yet fully understand all of the potential threats, all of the ways attackers look to exploit the supply chain, and for me, that's the primary reason why we don't yet have all the tools."

Many developers and security teams lack the technology for either blocking the introduction of malicious packages or detecting them in their existing code base. Many current tools attempt only one but not both, leaving security teams a difficult choice between blocking or detection functionality. Also, much of the traditional security technology available today lacks the capacity to effectively scan for malicious packages. The solutions claiming to provide scanning functions often employ rudimentary technology that misses the majority of malicious packages. Sadly, the de facto defense used by many organizations today is to rely on announcement updates from repositories that "find" and add the latest known malicious packages to their vulnerability database.



Combining Awareness With the Right Tools

The best defense against the growing threat of malicious packages is a knowledgeable and alert developer community in and around open-source registries like npm. The use of open-source code is here to stay, but there are measures organizations can take to make these code sources safer for the entire open-source community. Every organization should start by prioritizing the education of developers on security best practices while increasing their level of awareness and commitment to security throughout the entire software development lifecycle (SDLC).

Next, encourage your developers to follow some basic software development safety guidelines when using open-source code in their development processes, including:

- Never blindly assume ownership of open-source packages or trust the system of any registry
- Never install packages without running an assessment
- Encourage and incentivize developers to review all package details to help combat typosquatting
- Immediately report inconsistencies to package owners
- Maintain continuous environment awareness and update only when confident about the safety of content after deploying capable assessment tools

AUTOMATED DETECTION AND REMEDIATION TOOLS

As cybercriminals increasingly turn to automated attack methods to gain an advantage over defenses, it only makes sense for security teams to begin employing automated detection and response solutions. Automated detection and remediation tools powered by artificial intelligence and machine learning technology can gather data, perform analysis to identify threat patterns, continually run scans for suspicious activity, and quickly address threats by containing, remediating or alerting security staff.

When humans handle all detection and response duties, cyber incident response times bog down considerably. However, organizations that replace manual review with automated detection and response technology will enjoy improved accuracy rates. Legacy screening systems can deliver false positive rates as high as 95%, but automated detection technology can reduce false positive rates by up to 70%.[11] Today's industry-leading automated detection and response tools deliver perfect or near-perfect malicious content detection rate scores in popular registries like npm and RubyGems. By choosing automated detection and response tools that seamlessly integrate into native developer workflows, organizations can free up developers to focus on building better software instead of wasting valuable time investigating false positives.

[11] Abdel Hadi, D, (January, 2023). Reducing false positives using contextual AI. AI. Retrieved April, 2023 from aimagazine.com

Meet Mend.io's Malicious Package Protection Solutions

Malicious package supply chain attacks present a unique challenge for today's security professionals with few proven solutions. The malicious package blocking capabilities of Mend SCA and Mend Supply Chain Defender are two notable exceptions.

MEND SOFTWARE COMPOSITION ANALYSIS

The best time to stop malicious packages is before they enter your code base. With Mend SCA, organizations can block malicious packages from being downloaded or added to your artifact registry, ensuring that they can never enter your repositories or releases. Malicious packages can also be blocked in the repository, using automated scans at every code commit combined with a policy to block these packages.

Mend.io'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

Mend.io's 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.io Research. In the last three years, the Mend.io research team has successfully identified 100 % of malicious RubyGem packages and 99.8 % of npm packages. Mend.io 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.

MEND SUPPLY CHAIN DEFENDER

Mend Supply Chain Defender is the most comprehensive and effective detection and remediation tool available today for protecting npm and RubyGems registry open-source users from the supply chain damage resulting from a malicious package's toxic payload. Supply Chain Defender excels at scanning new open-source releases and existing packages with dozens of comprehensive tests to identify malicious content, and doesn't forget about previous packages potentially becoming malicious. "Our scans track current incidents as well as retroactively scanning historical packages to make sure things we weren't sure of didn't become malicious in later stages," Mensfeld says.

- **Supply Chain Defender operates in near real time, detecting and blocking malicious packages** in new releases or existing code bases. For example, in the dYdX incident in Sept 2022, Supply Chain Defender detected a malicious package within 30 minutes of its initial release. In another incident in October 2022, Supply Chain Defender alerted 17 minutes after the publishing of the first malicious "index.js" package according to Send.io team experts.

- **Supply Chain Defender is developer-friendly** as it's designed for exception-based alerting that doesn't interfere with developer workflows or slow down the development process. Supply Chain Defender also uses innovative classification rules that block suspicious packages before they reach developers, enabling them to work uninterrupted with code they can trust.

- **Supply Chain Defender helps manage security and compliance for your organization while making the open-source community safer.** Supply Chain Defender aids in the management of security and compliance processes across the SDLC by providing an analysis of open-source licensing and other metadata. This industry-leading automated detection and remediation tool sends malicious activity reports (sometimes hundreds daily) to respective package registry security teams as soon as possible to improve the overall security of the open-source community. As Mensfeld explains, "We don't only protect our customers, we also protect the community because we report all of our findings to the appropriate registries. If we consider an incident severe, we also contact the companies we believe are being targeted."

Next Steps

To talk to an expert or schedule a demo to learn more about how your organization can identify and remediate malicious packages with Mend.io's industry-leading malicious package security solutions, visit **https://www.mend.io/contact-us/**