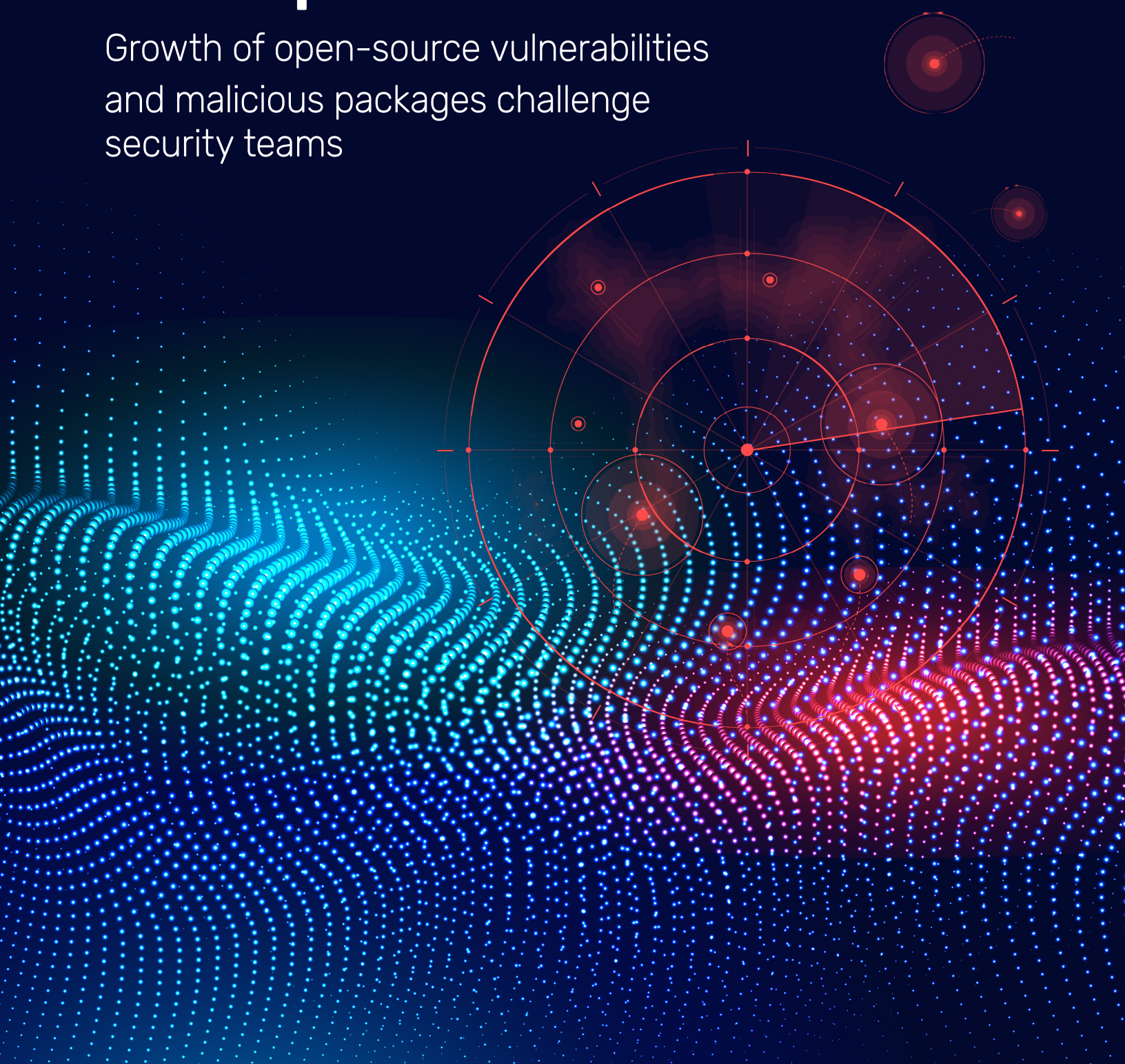
MEND

Mend Open Source Risk Report

Growth of open-source vulnerabilities and malicious packages challenge security teams



Executive summary:

The number of malicious packages that Mend identified and added to its vulnerability database in the first nine months of 2022 was 33 percent greater than the first nine months of 2021, reflecting both the growth in the number of published open source packages and the acceleration of vulnerabilities. At the same time, many companies struggle to close the remediation gap. While companies remediate thousands of vulnerabilities each month, it takes modern best practices to keep up with the wave of new vulnerabilities detected each month.

71 percent of IT and security leaders say their portfolios of applications have become more vulnerable to attack. Now, add in the fact that open-source code is used in 70 percent to 90 percent of applications today. Clearly, the ongoing rise in open source vulnerabilities poses significant risk for businesses, which rely heavily on applications for success.

More vulnerabilities, greater sophistication, less clarity

Unfortunately, this adds up to some attractive opportunities for threat actors, who are always quick to exploit new material. By using this increased cache of vulnerabilities to launch attacks that harness multiple vulnerabilities, attackers can exponentially boost their weapon arsenal — and target the flaws companies struggle to remediate. This also highlights the inadequacy of adhering solely to CVSS scores as a measure of how dangerous a vulnerability is. Effective prioritization requires not only severity details, but also context around how specific flaws can be exploited, both solo and in conjunction with others.

Growing challenge: Malicious packages

We have seen a steady quarterly increase in the number of malicious packages published in 2022, with a significant increase in Q3, which jumped 79 percent from Q2. At least ten malicious packages were published each day to npm and rubygems. Attackers are also deploying more sophisticated techniques. More packages contain telemetry, which enables data collection, and some are hidden more deeply, such as when valid content has a dependency containing malicious code.

Key findings:

- 33 percent growth in the number of open source software vulnerabilities that Mend added to its vulnerability database in the first nine months of 2022 compared with the same time period in 2021. This outstrips the estimated 25 percent growth in the amount of open source software available.
- According to a representative sampling of more than 900 companies from January-September 2022, only 13 percent of vulnerabilities seen were remediated, compared with 40 percent by those companies using repo integration.
- Data from Mend Supply Chain Defender shows a steady quarterly increase in the number of malicious packages published in 2022, with a significant jump in Q3, which increased 79 percent from Q2.

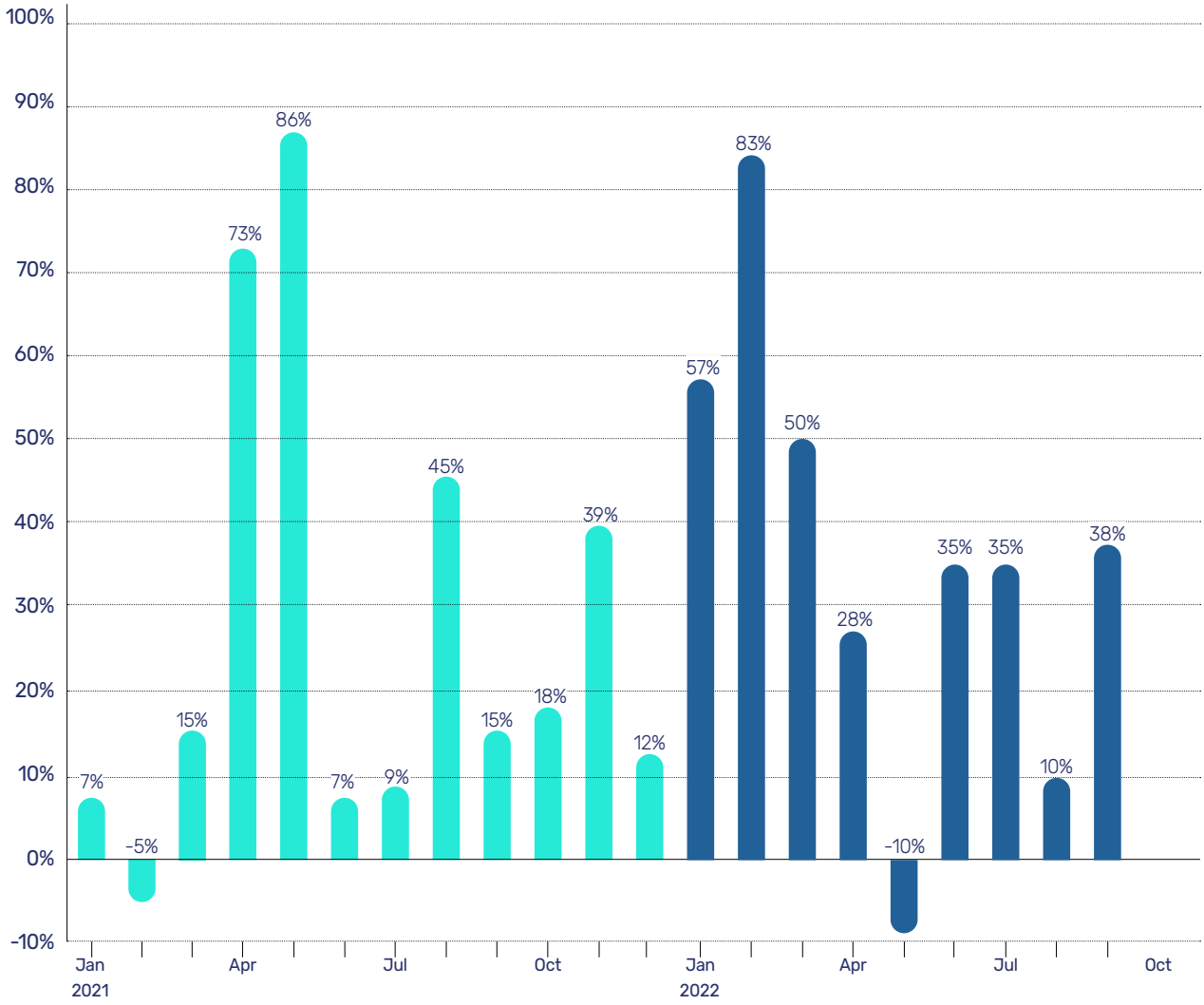
% OSS vulnerability growth

Open source vulnerability trends

Overall growth

In general, data from the Mend vulnerability database has shown open source vulnerabilities grow at a relatively conservative rate that maps roughly to the increasing prevalence of open source software. For instance in 2021, the number of new open source vulnerabilities that Mend added to its vulnerability database was 25 percent greater than the previous year, pretty much in line with the estimated ~25 percent growth in the amount of open source software available. However in 2022, we have seen a 33 percent increase in open source vulnerabilities through September.

Open source vulnerabilities by month, January 2021-September 2022

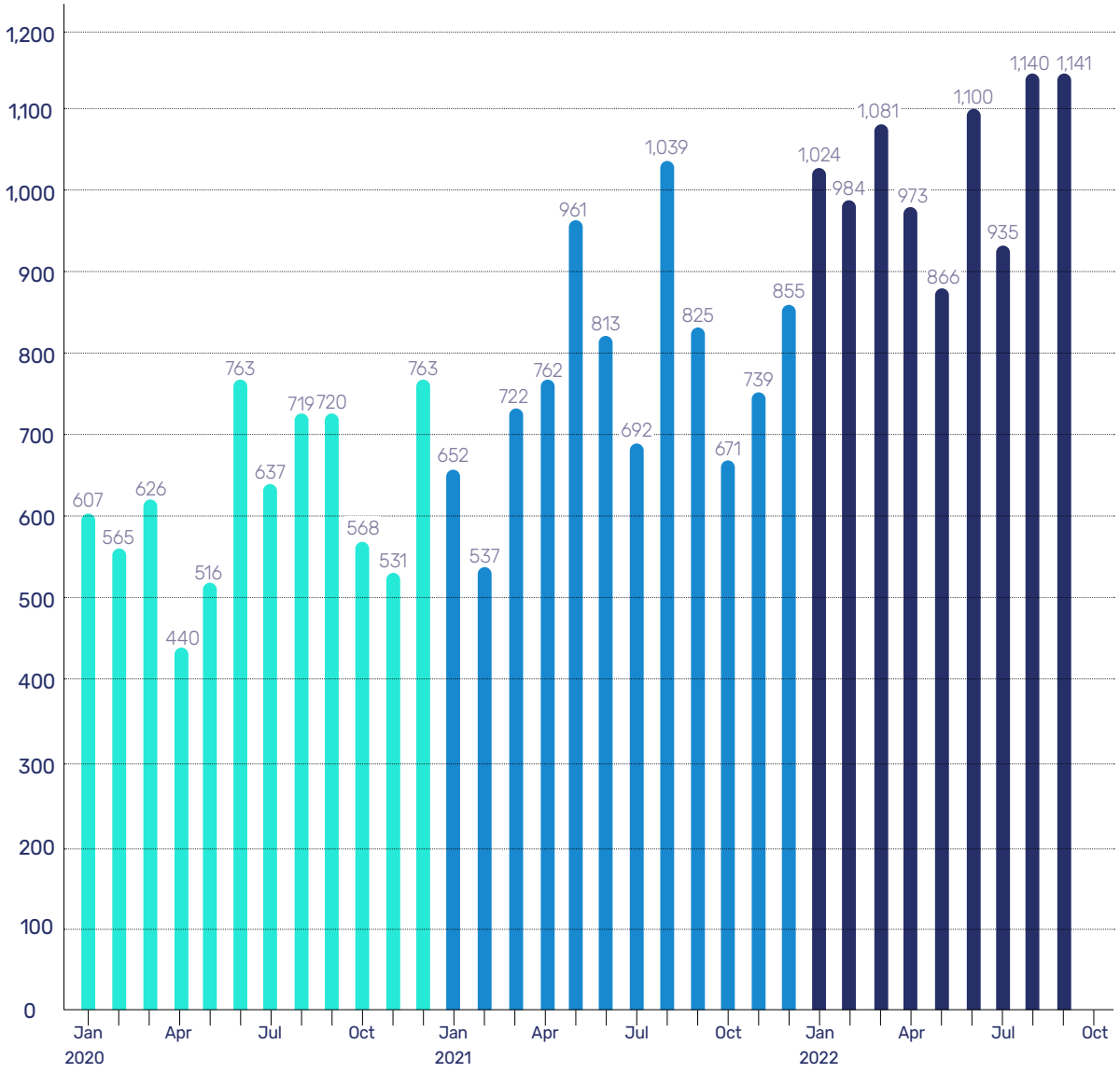


That growth in the total number of vulnerabilities represents a significant expansion of the raw material available to threat actors, who are well aware that applications and other software components constitute a juicy target. Even worse, the threat posed by this number of vulnerabilities is greater than the sum of their parts. Here's why: the actual risk is not represented by a total vulnerability count. Rather, it's the chance that those vulnerabilities can be used in a successful attack. And these days, malicious actors are launching increasingly sophisticated attacks that incorporate multiple exploits. That means that the chance of a successful attack that harnesses multiple vulnerabilities increases exponentially as the number of available vulnerabilities grows.

Moreover, it also highlights the inadequacy of adhering solely to CVSS scores as a measure of how dangerous a vulnerability is. For example, an FBI Flash Briefing on APT10, a Chinese actor targeting managed service providers and governments worldwide, included an extensive list of CVEs used by APT10 in their attack. More than half were low and medium security issues. What this means is that 'rank and react' based on severity doesn't always work. Attacker sophistication has obsoleted this method, as mid-and-low level vulnerabilities are now an active part of attacks. Now, defenders need context around how vulnerabilities can be leveraged in order to prioritize effectively.

Permanent growth mode

The combination of increased open source code creation and threat actors motivated to find new holes pretty much guarantees that open source vulnerabilities won't decline any time soon.



Source: Mend vulnerability database

Why are we seeing this jump? There are a few possible explanations for the increase in the number of known open source vulnerabilities thus far in 2022.

- Increased usage overall. Open source software usage continues to expand, as does our overall community of developers, not to mention an increase in the sheer amount of open source code available.
- At the same time, the field of OSS security research is also on the rise, with more companies in the field and more researchers at work. Additionally, automation also contributes to the high number of vulnerabilities discovered this year. Security researchers are using automated scanning and detection tools to find vulnerabilities, enabling them to find and fix security issues quickly, and at a greater volume. In some cases, researchers found multiple CVEs where a common problem affected many projects. Sometimes a single CVE was applied to many projects, but other times it was correct for them to each have their own CVE.
- The addition of CVE Numbering Authorities (CNAs) to the CVE program also contributes to the increase in published vulnerabilities. Comprising software vendors, open source projects, coordination centers, bug bounty service providers, hosted services, and research groups, CNAs are authorized by the CVE Program to assign CVE IDs to vulnerabilities and publish CVE Records within their own specific scopes of coverage. There are currently 251 partners from 35 countries participating.
- We also cannot discount the indefatigable efforts of the threat actor community, a very busy group who are financially motivated to stay one innovation ahead of the white hats. In short, as OSS code usage increases, so does the probability of growth in the vulnerabilities not known to us. This translates into more opportunities for attackers and a wider threat landscape.

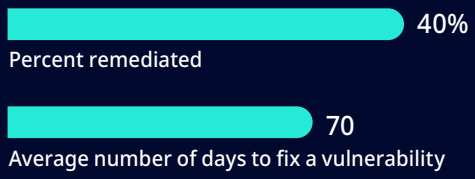
Open source security: The view from the user side

We wanted to take a look at open source security from the user perspective. By taking a representative sampling of approximately 1,000 North American companies across industries and company size from January through September 2022, we compiled data on critical and high severity vulnerabilities and remediation to present a snapshot into the state of open source security — and the difference Mend can make.

Baseline vulnerability remediation January-September 2022

Next, we took a representative sampling of companies that implemented Mend best practices through the repo integration.

Vulnerability remediation with repo integration January-September 2022



The results were telling. The increase in remediated vulnerabilities translates roughly into a three times reduction of risk, while the time to remediation was cut by 75 percent.

Remediation gap

While companies remediated thousands of vulnerabilities each month, many are left with a backlog of un-remediated vulnerabilities. Why is this happening? There are a number of reasons why companies face a remediation gap:

Lack of time and resources. It's no secret that application security teams are often overworked and understaffed, leading companies to make hard decisions on what applications to patch and keep current. Some focus on their flagship applications, for example, judging the business risk of not doing so as too high.

Lack of granular information. While the CVE severity index is a reasonable initial metric to use when deciding what to remediate first, it is insufficient for use by itself. It's crucial for application security teams to identify and prioritize vulnerabilities according to the risk they pose, both by themselves and when used in attacks that exploit multiple vulnerabilities. Many vulnerabilities, for example, do not pose a risk within an application and can be safely ignored if they can be identified. This also means that low and medium severity flaws cannot be neglected, as they can be used in multi-vulnerability attacks.

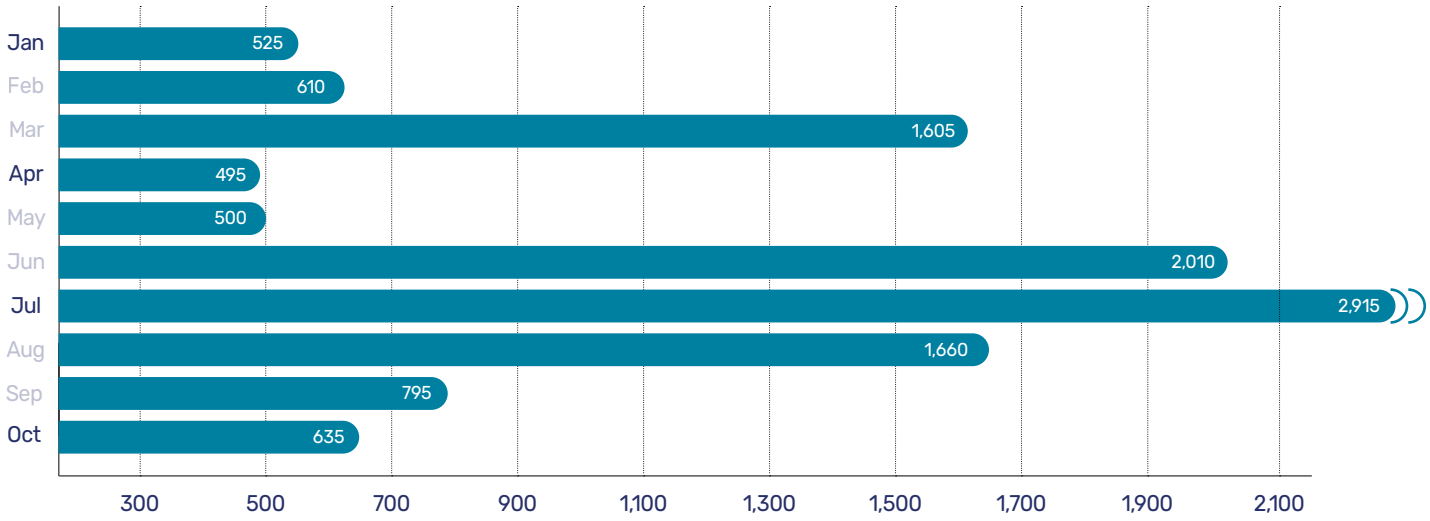
The need to balance security risk and functional risk. While application security is of foundational importance for business applications, so is the need to keep them functional. This can present knotty problems when deciding whether to patch or upgrade the many open source components and associated dependencies used to build an application. What is the risk that a regression error in a dependency update could cause production problems? Most teams cannot put in the massive amount of work needed to manually review each update. Instead, teams are starting to turn to tools that can automate dependency updates without compromising the functionality of an application.

Malicious packages

Mend's automated malware detection platform, Supply Chain Defender, detects and reports hundreds of malicious packages each month across npm and rubygems. From January through September 2022, we have seen a steady quarterly increase in the number of malicious packages published in 2022, with a Q2 to Q3 jump of 79 percent. At least ten malicious packages were published each day to npm and rubygems.

Attackers are also deploying more sophisticated techniques. More packages contain telemetry, which enables data collection, and some are now built more deeply within the software supply chain, such as when valid content has a dependency containing malicious code. Attackers also use legitimate hosting providers to ship malicious code, while others hide behind domain names, suggesting legitimate use cases. We saw the latter method in the attack on popular cryptocurrency exchange DyDx. In the case of DyDx, the malicious package versions contained a preinstall hook that made it appear as though a CircleCI script was about to be downloaded. This is brandjacking in its purest from — the domain looks as if it belongs to a legitimate CI/CD provider.

Malicious packages published per month, January-October 2022



Source: Mend Supply Chain Defender

Malicious package versions

On average, malicious packages had about three 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 versions published per month, January-October 2022

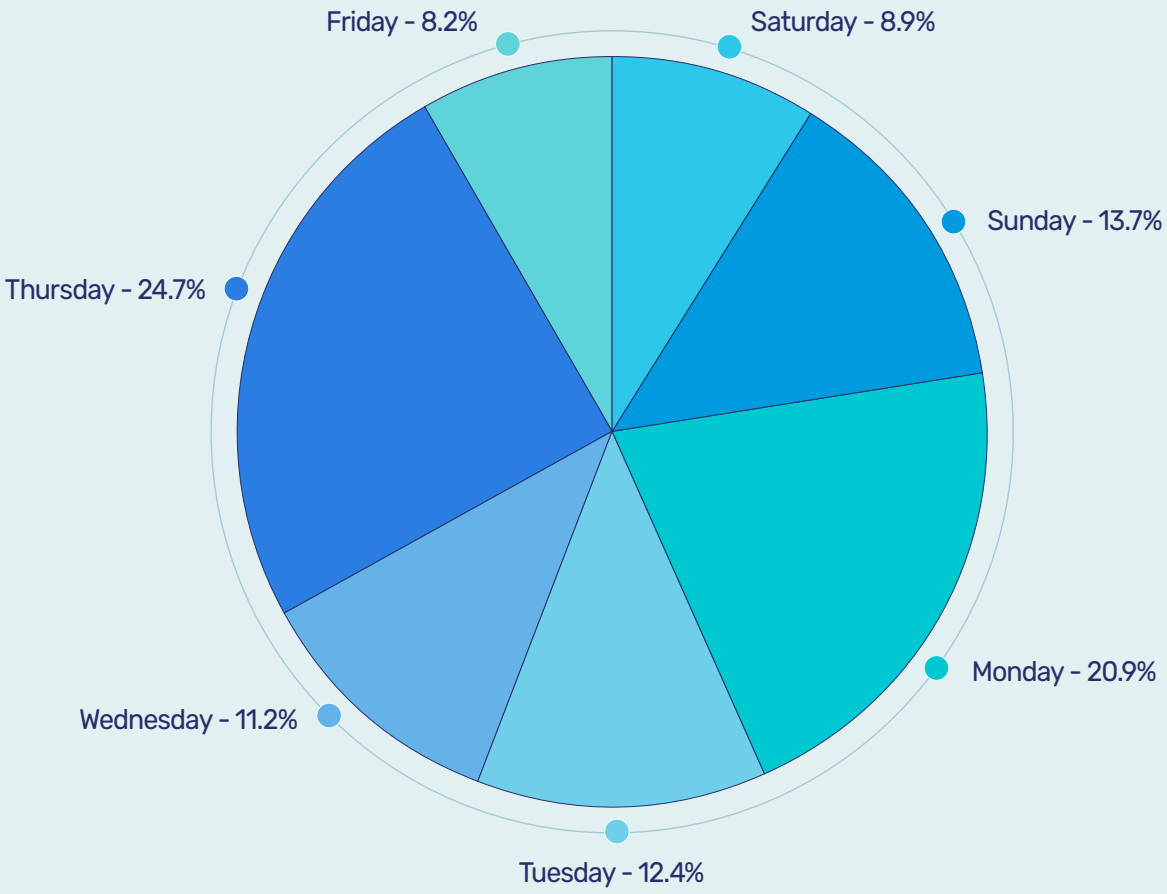
Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
Malicious versions published per month	1,835	2,210	4,087	6,097	1,914	4,651	6,198	6,729	2,535	1,925
Ratio of malicious versions per package	3.5	3.6	2.5	12.3	3.8	2.3	2.1	4.1	3.2	3.0

Attack timing: No accident

In our 2021 report, the most popular time for attackers to release malicious software was Friday, Saturday, and Sunday — also known as the weekend. We interpreted that as threat actors publishing packages during days with the highest probability of being missed by security researchers. The massive number of npm packages and the rate at which new ones are released makes it difficult to monitor, and an accumulation of data over two days could exacerbate the problem.

This year, almost 25 percent published on Thursday afternoon. Why the change? More and more attackers are aware that a preponderance of cybersecurity companies are in Israel, so they are banking on the fact that many companies there observe Friday and Saturday as the weekend. Translating CET to Israel's timezone, we see the attacks kicking in right about 4 p.m.

Malicious activity patterns



Source: Mend supply chain defender
(Data for this figure was taken in Central European Time (CET.)

Mining malware history for clues on malicious package innovation

Malware has come a long way since it first made the scene in the late 1990s. These days, attackers deploy a variety of innovative techniques to extract large amounts of money from businesses around the world.

A similar development is taking place with malware's upstart cousin — the emergence of malicious packages being uploaded to package registries. Like any younger sibling, malicious packages are just not as mature as generic malware. We see them lagging by about a decade. This makes sense, given that malicious packages represent a relatively new opportunity and attack surface, and cybercriminals are just starting to grasp their potential. Malware's history can also provide important clues about future malicious package trends. By looking at malware's evolution over the past 20 years in conjunction with current malicious package trends, we can predict a likely future direction for malicious packages. There are three primary areas: attack vectors, malicious techniques, and objectives.

Attack vectors

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.

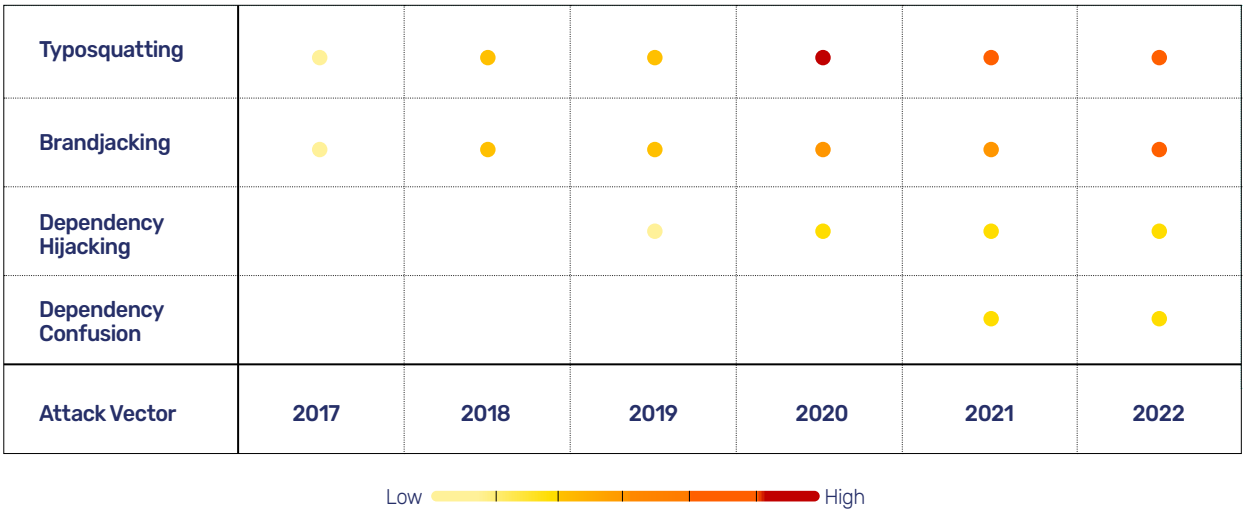
In a **typosquatting** attack, an attacker publishes a malicious package with a similar name to a popular package, in the hope that a developer will misspell a package name and unintentionally fetch the malicious version.

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.

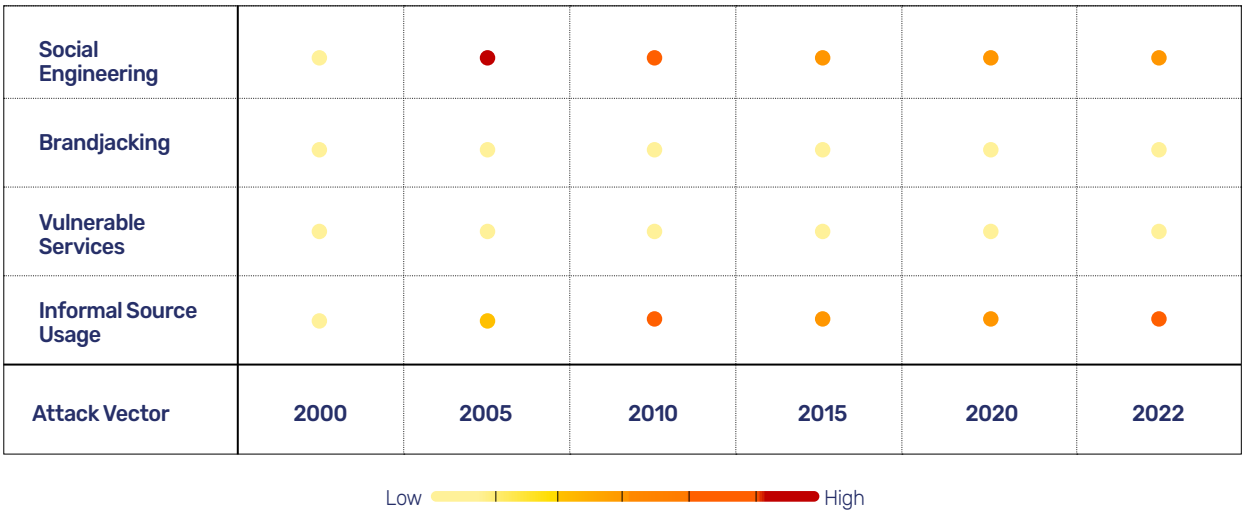
Brandjacking and typosquatting were the original malicious package attacks, and they remain an integral part of the attack vectors used today. Dependency hijacking and dependency confusion are more recent additions.

Malicious packages attack vector trends



There are also four common attack vectors used in malware: informal source usage, vulnerable services, brandjacking, and social engineering. The most commonly used attack vector is informal source usage, which refers to downloading or engaging with a site, company, or product that is clearly not well known nor has enough reputation to back up its legitimacy.

Malware attack vector trends



There are similarities between the two:

- **Dependency confusion** can be considered a vulnerability related to package registries managers, meaning it is considered as a vulnerable service attack vector. In the future, we'll see dependency management tools and package registries hurt even more by vulnerable services. Although it is a complex attack vector to exploit, there is a huge potential in it.
- **Brandjacking** appears on both lists, but is more commonly used with malicious packages than with generic malware. This is due to the obvious potential for attack that exists in package registries and in open source, where many individuals can own or contribute to the same project, and there are minimal verifications of authorization. Since dependency hijacking is very similar to brandjacking, we can add that into this area as well.
- **Typosquatting** can be considered as informal source usage, because with typosquatting, checking the owner of the package will nearly always reveal that it's not a reputable source. Typosquatting also shares similarities with the social engineering attack vector. It tries to target users who incorrectly type the package name that they actually want.

Prediction: While they don't share the same names, we see signs of every attack vector in generic malware being used for malicious packages. And since malicious package attacks are still relatively new, there is potential for increased use of both social engineering and vulnerable services. We expect to see an increase in attacks using these two vectors, both from malicious packages and in package registries themselves.

Malicious techniques

Attackers using malicious packages continue to rely on four common techniques: pre- and post-install scripts, basic evasion techniques, shell commands, and basic network communication techniques. While volume has grown, sophistication has not, although we are starting to see bad actors layer intermediate evasion techniques over basic evasions. A quick comparison shows significant maturation potential for techniques used with malicious packages:

In contrast, malware uses mature and sophisticated techniques to evade defenses, successfully infiltrate, and remain on infected machines, as well as achieving outgoing network traffic and code execution on the infected machine. Moreover, attackers are manipulating vulnerabilities found in other commercial or open source products to achieve better success rate or wider capabilities.

Evasion techniques. Although they exist in malicious packages, such techniques are extremely basic, such as the use of base64 encoding or hex encoding. We are starting to see more code obfuscation and even time delays that try to make it harder for dynamic analysis to detect the malicious activity, but this is still counted as basic or intermediate evasion techniques. Meanwhile, generic malware attackers can choose from a long list of advanced evasion techniques, such as anti-VM, anti-reverse engineering, filesystem and registry queries.

Persistence. Although attackers in malicious packages might be persistent in continually creating more and more malicious packages, only a few used persistence techniques on infected machines. Meanwhile, generic malware attackers can draw on extremely complex techniques to stay and keep running on infected machines; some examples are scheduled tasks, shortcut modifications, browser extensions, startup keys, and much more.

Vulnerability exploitation. We haven't yet seen a malicious package reach this level. On the other hand, generic malware exploits vulnerabilities even after infecting a machine to enhance their capabilities.

The last malicious technique we will analyze refers to the methods used to deploy, execute, and communicate once the attacker has infected the machine. **Malicious packages use basic methods to deploy, execute, and communicate on the machine, meaning that even if the package is successfully downloaded to the machine, it remains relatively easy to detect while deployed.** On the other hand, we continually see attackers use advanced techniques with generic malware.

Prediction: There are many opportunities for bad actors to refine their use of malicious packages. We expect to see more advanced evasion techniques sooner than anybody wants to. Malicious packages will start using persistence techniques. Vulnerability exploitation may lag, as it is not only difficult to develop, but generally useful only under special circumstances — for example, a new easy-to-use vulnerability emerges in a widely available product. Lastly, we expect to see more diverse and advanced approaches rapidly emerge in the general methods of deploy, execute, and communicate.

Objectives

Ransomware and adware are presently considered the most common malware types or malicious objectives for general malware. However, they are almost completely nonexistent in malicious packages.

There are reasons why it might be somewhat difficult to implement these types of malware in dependency management tools or package registries, but they aren't an absolute limitation. Attackers are starting to understand the potential of creating and deploying this type of malicious package. In the ongoing security cat and mouse game, we know that malicious actors are always motivated to overcome obstacles they might encounter.

When it comes to malicious packages related to cryptocurrency, we're seeing malicious packages with cryptominers. A few have tried cryptocurrency stealing, although there is nowhere near the amount of malicious cryptocurrency attacks happening compared to that in generic malware. We will see an increase in the amount of malicious packages focusing on cryptojacking and cryptominers in malicious packages. While bots have potential and still exist in generic malware, we see very limited numbers of malicious actors creating malicious packages for this intention.

Lastly, let's discuss stealing private information and reconnaissance in tandem, as there is considerable overlap on the malicious package side. Here again, we have a misleading edge case. At first glance, malicious packages are surpassing generic malware, but it's a measure of how common the methods are rather than their complexity. The most common objectives for bad actors using malicious packages are stealing private information and reconnaissance, while actors of generic malware have moved beyond these objectives. With that in mind, we might see a decrease in the popularity of reconnaissance, as the incentive for that is much lower than other objectives.

Conclusion

We have long held that preparation, planning, and consistent adherences to application security best practicess will help organizations construct a strong cybersecurity foundation. But as threat landscape activity continues to increase in volume and innovation, enterprises need to move beyond today's status quo in order to survive. Applications are the lifeblood of the global economy and threat actors know it. Attacks such as Log4j and the Solarwinds breach grab the headlines, but they represent a tiny fraction of the relentless attacks launched daily against applications.

Fortunately, we are seeing increased global commitment on cybersecurity from the public sector. Many governments, including the United States, the United Kingdom, and Japan are increasing regulations and standards to improve security across the software supply chain. That is but one step, however. As security debt continues to rise for most, it's important to find a way to prioritize the vulnerabilities that pose the highest risk. **A new approach is needed.** Organizations need to leverage prioritization and remediation tools that target the vulnerabilities that will most impact their systems and business if they want to manage their security debt wisely.

About Mend

Mend, 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.