Remediating Vulnerabilities in npm Packages

How to quickly and painlessly reduce vulnerabilities in npm packages

One of the biggest pain points in managing application security — and open source security in particular — is quick remediation of open source vulnerabilities.

In order to address the challenge, DevSecOps tools and practices are increasingly being put in place to ensure that application security is addressed from the earliest stages of the software development life cycle (SDLC). These efforts must also cover open source components, which comprise up to 80% of our software products.

As AppSec practices shift left into development, the task of ensuring that open source libraries are up-to-date and vulnerability-free falls on developers' shoulders — and it is quite a task. Open source libraries are composed of a deep web of direct and transitive dependencies. Ensuring they are all updated to secure versions as soon as a vulnerability is detected — without breaking the build or slowing down development might often feel like a losing battle.

Our Research

In order to gain insight into how to speed up remediation without slowing down the development process, we analyzed Mend data on vulnerable npm packages — one of the most widely used package ecosystems in the open source community and commercial projects. We wanted to learn about the typical time frames of vulnerability detection and fixes, to help find the quickest and least painful way to a security fix.

The Challenge: Keeping Open Source Components Secure

In addition to the task of tracking all open source dependencies in the code base, staying on top of newly disclosed security vulnerabilities and fixing them quickly by updating to a secure version can also be a complicated task. Commercial software products have a distribution list to announce a new version release and push an update, but the process with open source updates is different, often presenting a challenge to already-busy security and dev teams. Developers need to track the publication of new security vulnerabilities, which may or may not include fix recommendations.

Package Updates and the Risk of a Broken Build

If a fix is available, developers often hesitate to update immediately due to concern that the fixed version won't be backward compatible with their code or other dependencies. In the fast pace of today's release cycles, fear of dealing with a broken build often trumps updating to a secure version. The result is outdated and insecure software that might eventually cost a company much more than a delayed release.

How Are Open Source Security Issues Detected?

Because of the decentralized and distributed nature of the open source community, open source vulnerabilities are managed differently than vulnerabilities in proprietary applications. The open source community is made up of a growing group of projects and contributors, that work independently, or are sponsored by commercial companies in the software development ecosystem.

Open source project maintainers often depend on their community to help uncover vulnerabilities and create the fixes as quickly as possible. When a vulnerability is detected in an open source library, the more common practice is to report it to the National Vulnerability Database (NVD). There are a few steps in this process:

1. Contributors, security researchers, CVE naming authorities (CNAs), and white hat hackers review the code using a combination of automated tools and manual methods to uncover vulnerabilities.

2. When a vulnerability is detected, it's reported to the open source project manager, who then notifies MITRE, or the GitHub Security Lab created to make the reporting process easier.

3. MITRE reserves a CVE ID number but does not publish any of its details for 60-90 days. During this grace period, details about the vulnerability are withheld, giving project maintainers time to develop a fix.

4. Once a fix is available or the grace period has ended, the CVE is published with detailed information about the vulnerability. At the same time, the CVE automatically appears in the NVD.

It's then up to developers to then make sure that they are using secure versions of open source components. Developers are tasked with tracking all of the open source libraries in their codebase, along with tracking the publication of new open source security vulnerabilities. Ideally, when a new vulnerability is published with a fix available, the vulnerable version in their code is updated to a fixed version as soon as possible.

Mend Insights: Fixing npm Vulnerabilities

In order to gain a better understanding of the process of tracking and managing open source version updates, our knowledge team went to the Mend open source vulnerabilities database, which covers over 270M open source components and 13 Billion files. We decided to take a deep dive into npm — as it is one of the most popular platforms in the open source dev community, and can give us a good understanding of open source vulnerability management.

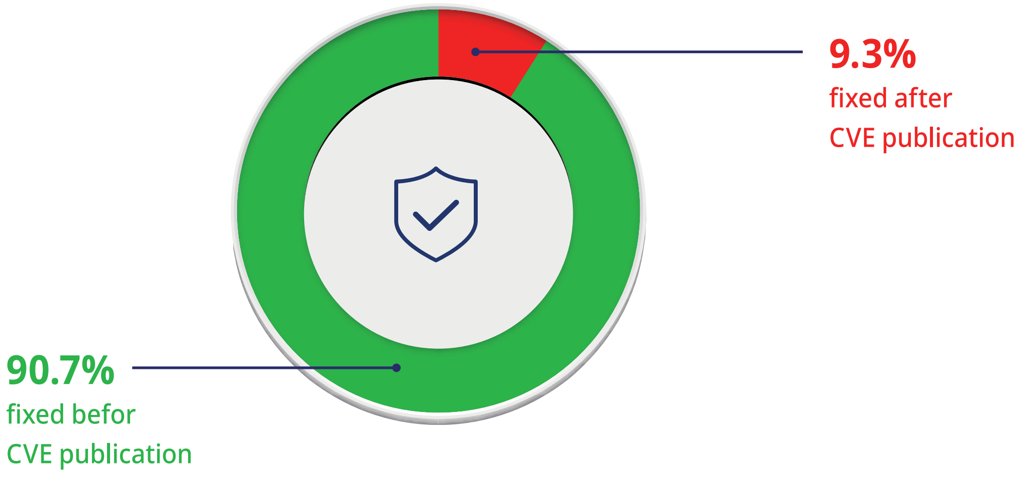
The research:

We analyzed the npm vulnerabilities published in 2021, checking the CVE publication date and comparing it to the release date of the vulnerabilities' fix, in cases where a fixed version is available. The research helped us understand what happens when an open source security vulnerability is detected and disclosed, so that we could help developers find a way to fix vulnerable versions as quickly and easily as possible.

The results:

Over **90%** of vulnerabilities in npm packages are fixed before the vulnerability is published on the NVD. That means that with the right tools and processes in place, developers can potentially install a fix before a security vulnerability is publicized, which ideally means before any black hat hackers are aware of it.

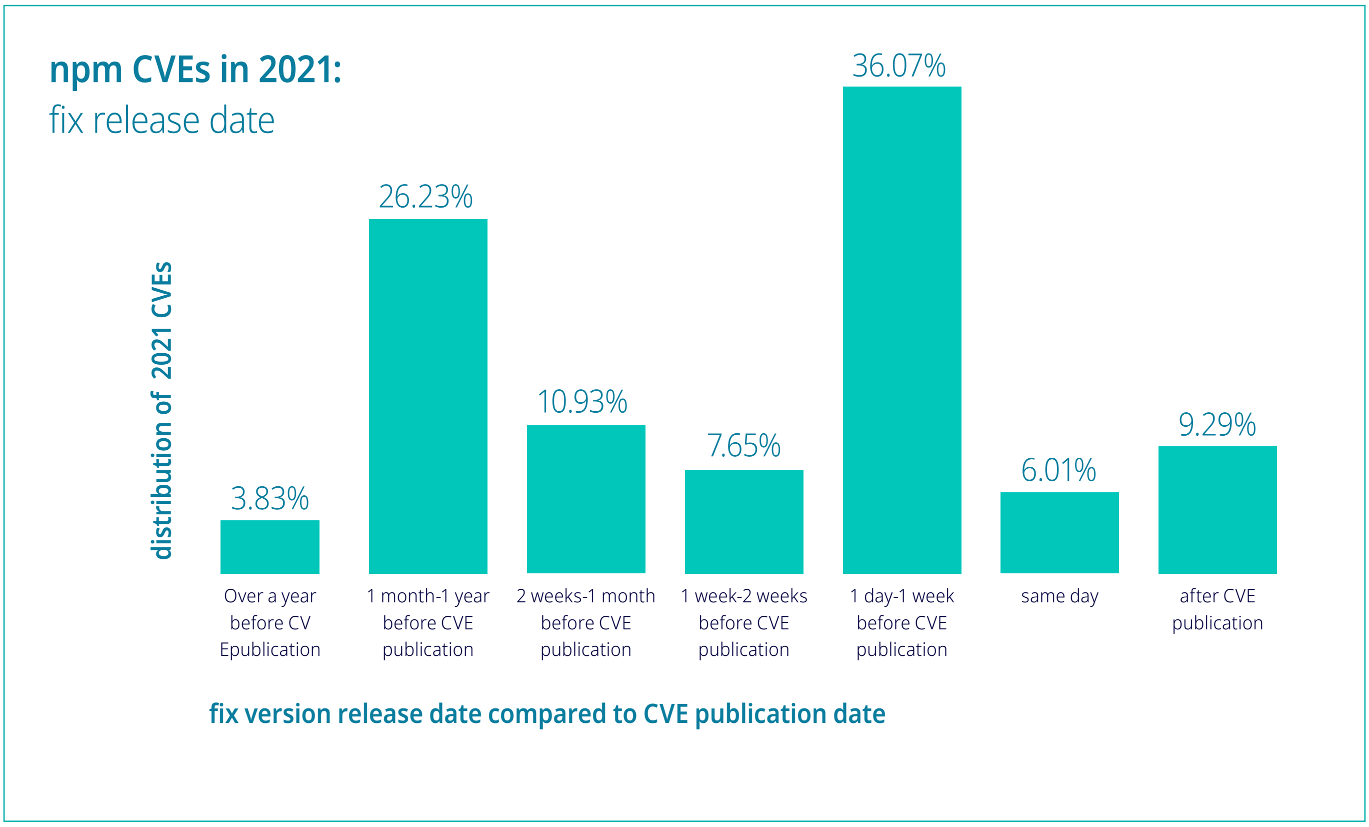
npm CVEs in 2021, per fix date



Reasons behind the gap between the release of a fix and the publication of the cve

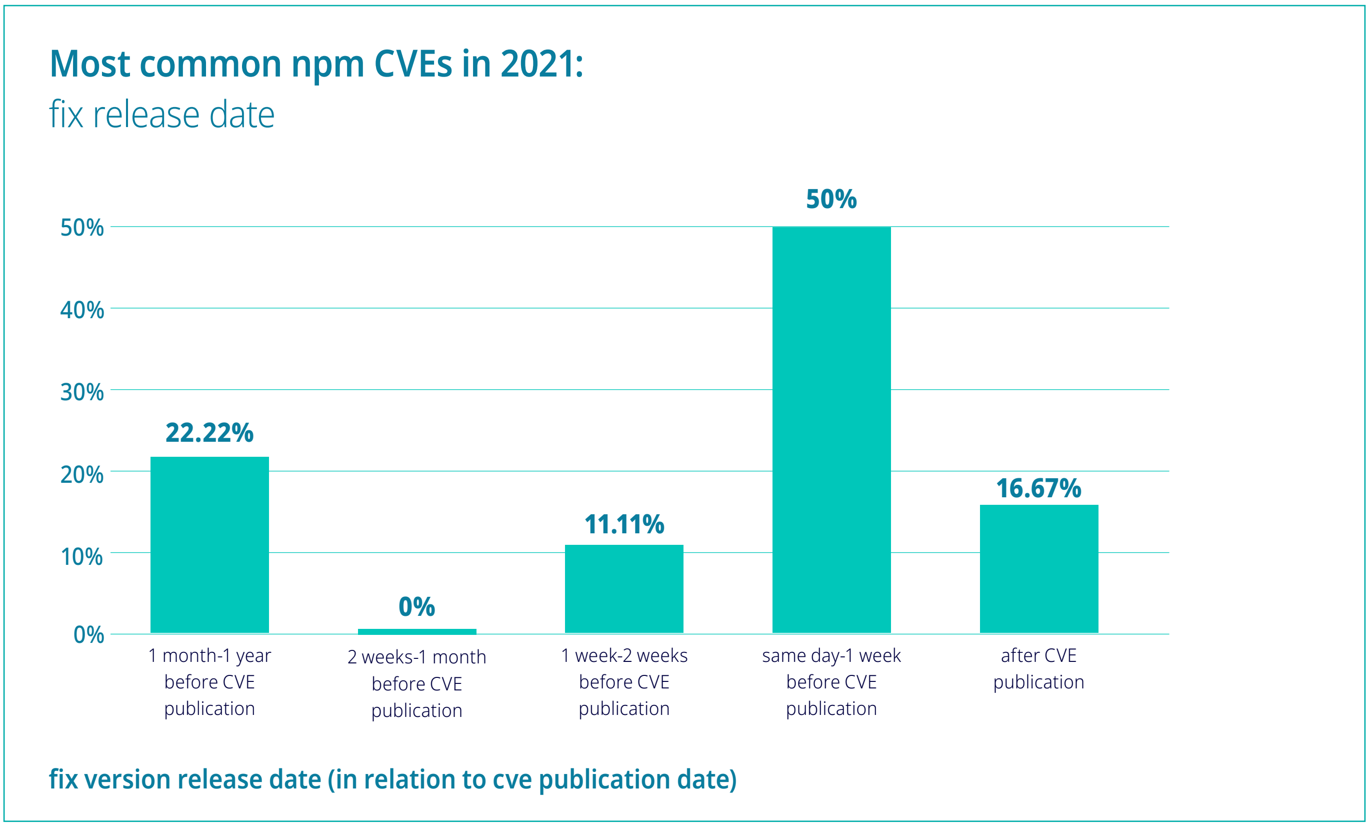
- Many of the biggest gaps between a fix and a vulnerability — a vulnerability published months or years after the fix — can probably be explained by the practice of security researchers analyzing old fixes and reporting them. In other words, when the vulnerability was fixed but not reported initially, but is later followed up by a security professional who reports it anyway.
- In other cases, the fix is published in parallel to or some time after the public disclosure. One way this happens is when the official publication of the vulnerability is forced due to a previous, “irresponsible” disclosure in public — in which case the package’s author has no choice but to publicize the vulnerability even if no fix exists yet.

Late fix releases might also occur if the open source project maintainer didn’t respond to a responsible disclosure within the agreed-upon time frame, and the researcher decided to publish it once the grace period ended.



Data shows that over half of the vulnerabilities have a fix available up to a month before the vulnerability was published on the NVD, and 85% of vulnerabilities are fixed a day or more before the issue is published. Some are even fixed a year or more before the issue is published.

When we looked more closely at the most common CVEs in our data, we found that most of them could have been addressed well before the issue was published on the NVD:



Our data shows that for a total of **83%** of the CVEs in the most widely used versions of vulnerable packages, could be fixed before the publication of the CVE, when information becomes available to all users — including malicious players

How Can Developers Update Vulnerable npm Packages with Confidence?

Data regarding fix dates vs. CVE publication dates shows that in most cases developers can fix security vulnerabilities even before they are published on the NVD.

However, anyone that's been in the trenches knows that is easier said than done. Manual tracking of CVE's or released fixes of vulnerable open source components is virtually impossible.

MEND**Renovate**

Mend Renovate helps developers integrate automated fix pull requests in the development environment, to ensure that fixes are implemented as soon as they are released.

Understandably, developers are often hesitant to push these security fixes, because of the risk that they will break the build. Mend Merge Confidence helps to overcome developer hesitance and enables fixes to be accepted and deployed faster.

Merge Confidence: Leveraging Crowd Data for Quicker Remediation

Mend Merge Confidence gathers dynamic crowd data about new releases and gauges the likelihood that a new release is "accidentally" breaking or backward-incompatible. Leveraging anonymized and aggregated data gathered from hundreds of thousands of repositories using Mend's GitHub app, developers can quickly assess the likelihood that a new release might cause them problems. For example, if a new release is a week old, passed 99% of everyone's tests, and already has 20% adoption, the chances that there's a newly introduced bug that nobody else has detected yet is extremely low, and the remaining users can feel confident to upgrade.

This confidence makes the difference, allowing developers to approve security fixes without hesitation and close the attack window where projects may be exposed to attackers.