Audit Package Vulnerabilities

Ivan Shakhorski

Content

Overview

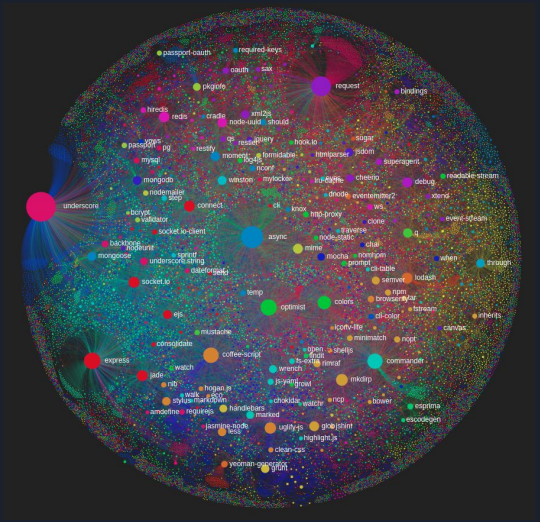
A security audit is an assessment of package dependencies for security vulnerabilities.

Security audits help you protect your package's users by enabling you to find and fix known vulnerabilities in dependencies that could cause data loss, service outages, unauthorized access to sensitive information, or other issues.

Understanding the problems

Some packages in their some versions may use methods, approaches or something, which allows them to be hacked, which to pull out any information, or broken the application. These can be either critical vulnerabilities or something very simple.

There are a lot of dependencies in npm, packages use other packages, a lot of deep dependencies, which makes it difficult to track, and it is difficult to update deep dependencies, so many outdated versions of packages can accumulate in which vulnerabilities could appear. Therefore, sometimes it is necessary to eliminate vulnerabilities.



Not space, but the scheme of npm package dependencies, every dot is package, every line is dependencies.

Tools for analytics and fixing

- Npm audit

- Snyk

Npm audit usage

01 npm audit - displays all vulnerabilities to the console based on its vulnerability database, also recently briefly describing the possibility of fixing and a link to the github issue related to this, if there is a fix.

02 npm audit fix - performs automatic correction of all vulnerabilities that he knows how to fix, and in which there are no breaking changes (although this is cheating, it can still break something).

03 npm audit fix --force - breaks everything to hell, making any changes that he thinks can fix vulnerabilities. Dangerous to use, but it exists, and can be used.

Npm audit usage

Moderate	Misinterpretation of malicious XML input
Package	xmldom
Patched in	>=0.7.0
Dependency of	cloudinary-video-player
Path	cloudinary-video-player > videojs-contrib-ads > video.js > @videojs/http-streaming > mpd-parser > xmldom
More info	https://github.com/advisories/GHSA-5fg8-2547-mr8q

Moderate	Misinterpretation of malicious XML input
Package	xmldom
Patched in	>=0.7.0
Dependency of	cloudinary-video-player
Path	cloudinary-video-player > video.js > @videojs/http-streaming > video.js > @videojs/http-streaming > mpd-parser > xmldom
More info	https://github.com/advisories/GHSA-5fg8-2547-mr8q

```
found 143 vulnerabilities (6 low, 84 moderate, 51 high, 2 critical) in 2976 scanned packages
  run `npm audit fix` to fix 114 of them.
  11 vulnerabilities require semver-major dependency updates.
  18 vulnerabilities require manual review. See the full report for details.
```

Npm audit benefits

Npm audit allows you to leverage their efforts to find and fix security problems in your code, instead of going the tedious route of manually perusing the dependencies in your project to identify security loopholes.

It identifies the security issues clearly and labels them in terms of the level of severity. This allows you to address them fast and easily.

If a fix has been published, it provides an out-of-the-box option for resolving the discovered anomalies.

Snyk usage

The disadvantage of using a third-party tool is that you will have to install it on your computer, and even worse, you will have to create a snyk account to use it.

What are the advantages of Snyk, it allows you to see in more detail what the vulnerability is, to see possible solutions, and so it allows you to handle each vulnerability individually, ideally in the case when you are trying to deal with it so as not to break any dependencies.

After these rituals, you have the opportunity to use tools to the fullest, but personally, in my experience, two main features came in handy:

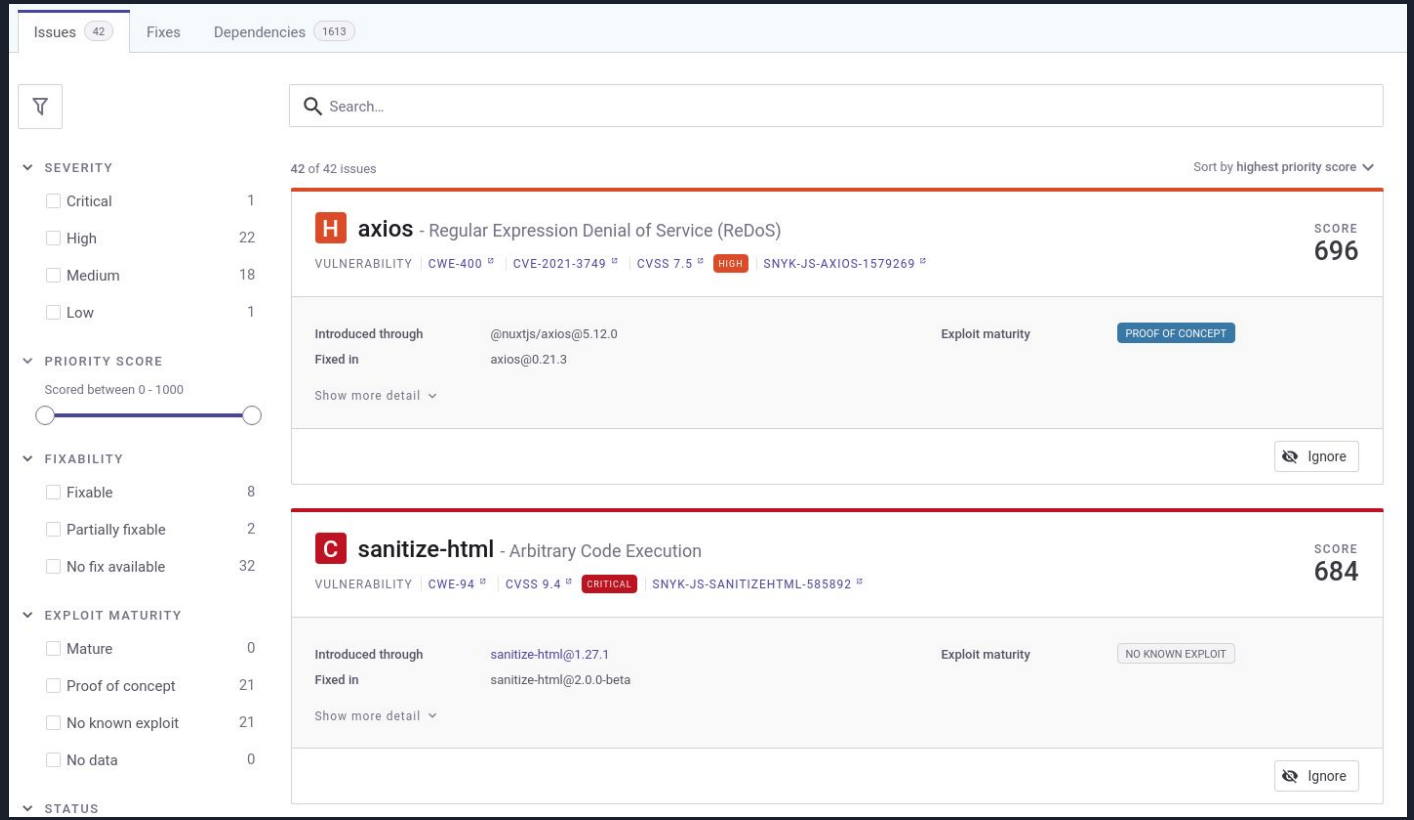
Snyk usage - snyk monitor

01

snyk monitor - allows you to generate a vulnerability report for your project based on package.json and package-lock.json files, Snyk analyzes what you use and which versions of packages you use, and this also concerns internal dependencies, when some packages use others and so on.

After based on own vulnerability database Snyk beautifully tells us which packages have vulnerabilities, and even more, it can describe what kind of vulnerability it is, how it can be fixed, if it possible. Well, a huge plus is that it does it in a beautiful, understandable form as a report that you can just open in the browser

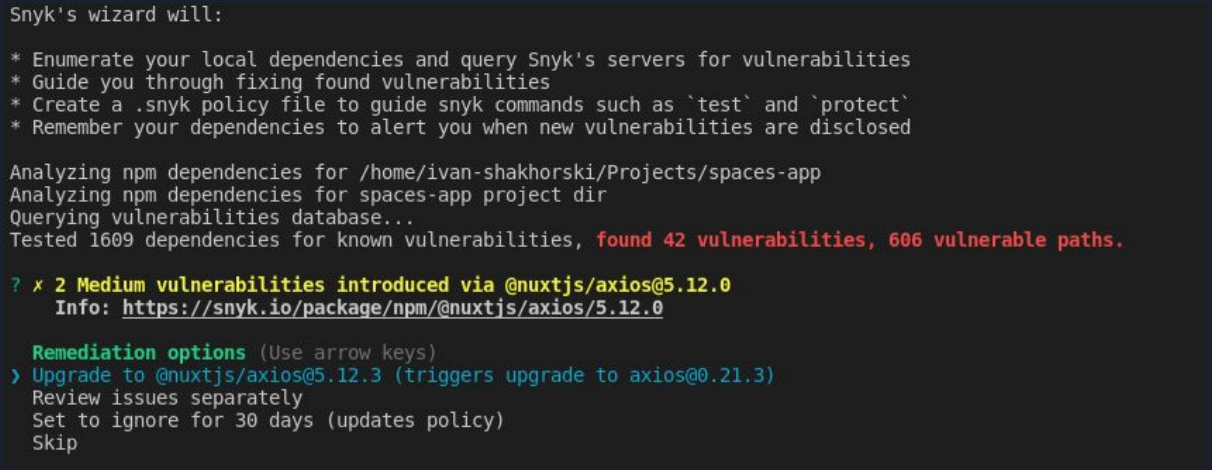
Snyk report


Issues 42
Fixes
Dependencies 1613
Search...
SEVERITY
Critical 1
High 22
Medium 18
Low 1
PRIORITY SCORE
Scored between 0 - 1000
FIXABILITY
Fixable 8
Partially fixable 2
No fix available 32
EXPLOIT MATURITY
Mature 0
Proof of concept 21
No known exploit 21
No data 0
STATUS
42 of 42 issues
Sort by highest priority score
H axios - Regular Expression Denial of Service (ReDoS)
VULNERABILITY | CWE-400 | CVE-2021-3749 | CVSS 7.5 | HIGH | SNYK-JS-AXIOS-1579269
SCORE 696
Introduced through @nuxtjs/axios@5.12.0
Fixed in axios@0.21.3
Exploit maturity PROOF OF CONCEPT
Show more detail
Ignore
C sanitize-html - Arbitrary Code Execution
VULNERABILITY | CWE-94 | CVSS 9.4 | CRITICAL | SNYK-JS-SANITIZEHTML-585892
SCORE 684
Introduced through sanitize-html@1.27.1
Fixed in sanitize-html@2.0.0-beta
Exploit maturity NO KNOWN EXPLOIT
Show more detail
Ignore

Snyk usage - snyk wizard

02

snyk wizard - this is already an automatic thing that immediately allows you to view each vulnerability individually through the console, and choose whether to take any action for it or not.



VS

- Internal npm utility, does not require any installation, authorization and the like
- Vulnerability database is better

- A good and convenient UI for working with a vulnerability report
- Allows you to selectively fix vulnerabilities automatically

I can't say which is better, I will say that both utilities are good, especially using them in a pair, snyk makes it great to start, since with it you can fix some of the dependencies more loyally, without consequences. After it, using npm, considering each vulnerability separately.

Life hack

- Try generate new package-lock.json file

  This can give a good start if you have a very outdated file, since when generating a new one, npm will install updated versions of internal dependencies where vulnerabilities could already be fixed, but here you also need to be careful, since this can also break something.

Conclusion

Based on my attempts and my mistakes, in order to have the minimum possible number of vulnerabilities in the project, it is necessary to maintain up-to-date, but at the same time stable versions of packages, update them in a timely manner.

Also, when eliminating a vulnerability, it will be important to consider each vulnerability separately, and after any actions to make sure that the solution is fully working, after which it is already possible to switch to the next one.

Unfortunately, the world is not perfect, and it will most likely not be possible to fix everything, but it will be a gesture of good code to maintain a minimum number of vulnerabilities

Thank you!

Sources

- Npm's docs
- Snyk's docs
- Npm Packages Dependencies Graph | Exploring Data