Analysis of Node.js Vulnerability Detection from NPM: How Efficient these Tools.

Vidhya Prasad
Department of Computer Applications
Amal Jyothi College of Engineering Kanjirappally, India
vidhyaprasad2022b@mca.ajce.in

Grace Joseph
Department of Computer Applications
Amal Jyothi College of Engineering Kanjirappally, India
gracejoseph@amaljyothi.ac.in

***Abstract* — security of an application is not easiest thing to do. Node.js is one of the leading technologies that help developers for web application development. It is designed securely. However, while developing any application through node.js you will need to use various third party open-source-packages through NPM (Node Package Manager). Unfortunately as per the survey 14% of the NPM ecosystems are vulnerable and these indirectly account for 54% of the NPM packages. Ultimately, all these vulnerabilities are transferred to node.js applications. Security vulnerabilities are not new for open source backend framework similarly Node.js developers can understand the risk that hackers can pose for application and user data's. In this paper we are discussing about how we can detect the node.js security vulnerabilities and the efficiency of the tools which is used to solve the vulnerabilities.**

***Keywords*—Vulnerability Scanner, Web Security, Components with known vulnerabilities, audit, snyk.**

I. INTRODUCTION

The risks and vulnerabilities that come with third-party software, as well as the tools and techniques used to safeguard open source software, are all covered by open source security. When vulnerabilities are discovered, security tools can automate the discovery of open source libraries and dependencies in code, assess how those components are used in applications, and trigger alerts or repair procedures. Because of its collaborative and public character, open source software has grown in popularity in recent years, making it useful for both developers and malevolent actors. Once an adversary discovers that a program is vulnerable to a publicly known flaw, they can launch an attack against any application built with that open source code. To avoid this risk, it's critical to manage open source components and dependencies. Nonetheless, it is. Here in our study we are analyzing two tools which is used to scan and solve the vulnerabilities, they are NPM audit and Snyk. Npm, the Node.js default package manager, is one of the world's largest open source package ecosystems. This diverse ecosystem of open source packages has increased developer productivity and application performance, resulting in a win-win situation for development teams. Hundreds, if not thousands, of npm packages can be found in Node.js codebases. The packages' direct and indirect dependencies, as well as the security risks connected with them, may be unknown to developers. In 2018, Npm launched npm audit, a new command that does a moment-in-time security analysis of a project's dependency tree and generates a npm audit security report. The report contains information about the dependencies' security issues, as well as npm commands and recommendations for further investigation. The importance of using vulnerability scanners to unveil flaws in web applications before they are deployed has been realized by many organizations today. The system should be able to continuously monitor the software and generate informative reports about the vulnerable components, insecure versions, and security threats. Web applications have become the primary source of security vulnerabilities scanning tools for node dependencies. The tools under consideration are audit and Snyk.

II. LITERATURE REVIEW

Bodin Chinthanet, Serena Elisa Ponta, Henrik Plate [1] discuss the challenges associated with the construction of bill of materials of Node.js applications. The study highlights three lessons learned and the challenges that require attention by both researchers and practitioners dealing with Node.js applications and JavaScript in general.

Aman Anupam, Prathika Gonchigar, Shashank Sharma[2] reviews the tools for both feature-based comparisons (based on existing features) as well as result-based comparisons (based on scanning result). Feature-based comparisons focus on parameters like user-friendliness, vulnerability database, extent of scanning and features of command line interface of two tools.

Alexandre Decan, Tom Mens Eleni, Constantinou [3] analyse how and when these vulnerabilities are discovered and fixed, and to which extent they affect other packages in the packaging ecosystem in presence of dependency constraints. report findings and provide guidelines for package maintainers and tool developers to improve the process of dealing with security issues.

Mahmoud Alfadel,Diego Elias Costa, Mouafak Mokhallalati[4] the existence of the vulnerability that is the real problem. Furthermore, verify their findings at different stages of the application's lifetime and find that they are findings still hold. study argues that when it comes to software vulnerabilities, things may not be as bad as they seem and that consider vulnerability threat is key.

Chengwei Liu, Sen Chen, Lingling Fan [5]. unveils lots of useful findings, and further discuss the lessons learned and solutions for different stakeholders to mitigate the vulnerability impact in NPM based on our findings.

III. MOTIVATION

Open-source packages in general and npm in particular, are unquestionably fantastic. They increase developer productivity by providing each of us with a variety of pre-built functionality that is simply waiting to be used. We wouldn't be able to generate a fraction of what we do now if we had to write all of this stuff

DOI: 10.5281/zenodo.6907036
ISBN: 978-93-5607-317-3@2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

ourselves. As a result, a typical Node.js application nowadays uses dozens, if not hundreds, of npm packages. What we often ignore is that, in addition to their usefulness, each of these packages also includes its own set of Node.js security vulnerabilities. Many packages add new ports to the attack surface, hence extending the attack surface. Approximately 76 percent of Node shops utilize vulnerable packages, some of which are quite dangerous, and open source projects frequently become stale, ignoring security problems.

IV. METHODOLOGY

On GitHub, you may find the vulnerable node-based scanning project. The repository was cloned on the local machine, and the package.json file's dependencies were then installed. The command line scanners (both npm audit and Snyk) were run after installation. Both scanners produced a json-formatted report, which was subsequently translated to a readable format. For such conversions, the Snyk command line includes a tool called snyk-to-html. When you run npm audit, it checks all of the packages listed in package-lock.json for vulnerabilities. The audit is insufficiently sophisticated to determine what is included in the deployment. That is, the contents of the build folder as well as the packages that were used to create it. As a result, a npm audit is a preliminary security review that can help you find potential vulnerabilities in the packages you're using. However, it does not provide a complete picture of your application's vulnerability because:

- It's possible that code from packages with known vulnerabilities won't make it into your deployed code. Either because the code is solely used for development, or because the bundling algorithm determines that the code isn't needed and excludes it from the deployment bundle.
- It's possible that vulnerabilities have yet to be uncovered. As a result, the code you successfully inspected yesterday may fail an audit check today. Although no code has changed, a new vulnerability has been detected and reported to NPM https://www.npmjs.com/advisories.

A. How to Fix Security Vulnerability of using Audit

npm audit is a security feature built into npm that scans project for security flaws. It generates an evaluation report that includes information about the detected anomalies, proposed solutions, and more. It compares the current version of your project's installed packages to known vulnerabilities listed on the public npm registry. It will notify you if it identifies a security flaw. The severity of the identified vulnerability is detailed in the report. If no vulnerabilities were detected, the command will exit with a 0 exit code.

The effect and exploitability of the problem determine the severity level. The following are the severity levels and actions that should be taken:

Security Level	**Recommended Actions**
Critical	Resolve Straightway
High	Make a decision as soon as feasible.
Moderate	Resolve as time allows
Low	Resolve at your discretion

Under the hood, npm audit fix performs a full npm install. By default, it appears that an audit repair only performs semver-compatible updates. We use the command fix—force to upgrade the dependency.

$ npm audit fix –force

According to my understanding, "upgrading" entails not only "upgrading," but also "downgrading" in order to install the stable version that resolves the issue; sometimes, those issues arise in newer versions that may contain bugs.

B. How to Fix Security Vulnerability using Snyk

Despite the fact that development dependencies aren't supplied with the module when it's installed by users, it's still critical to evaluate the vulnerabilities that they include. If your build, lint, and testing packages, for example, are vulnerable, your entire CI/CD pipeline could become an attack surface from which a malicious party could attempt to escalate privileges (to gain access to your private CI/CD) or even modify your code, resulting in a persistent XSS hole or worse! Fortunately, it appears that all of the issues in this situation have an upgrade path or a fix available from Snyk. The JavaScript libraries are not scanned. Snyk also has its own vulnerability database, and because it's an open source program, newer vulnerabilities can be reported to the Snyk community if they're uncovered. The Snyk tool has a GitHub integration capability that allows a project hosted on GitHub to be automatically analyzed for vulnerabilities via a pull request. For a modern organization/company, GitHub support enables for a continuous open source management system. The Snyk CLI is a fantastic and effective tool for scanning security vulnerabilities in your applications, containers, and infrastructure as code. We'll look at the most powerful things our CLI has to offer in this cheat sheet. On your local system, you can use the CLI for scanning and monitoring, but you can also integrate it into your process. The Snyk CLI is the go-to tool for testing, monitoring, and remediating identified vulnerabilities in your apps, regardless of how you use it. You can see a description, the severity, and a link where you can learn more about the individual security issue for each vulnerability that the CLI shows in each of the three areas.

- The snyk test command checks for known vulnerabilities in a local project. It includes details on the vulnerabilities, their severity, types and descriptions, the number of vulnerable routes, and repair steps, among other things. Snyk CLI recognizes your manifest files automatically and tests the first one it finds.
- Assume your program has a vulnerability for which there is no patch or update available, or a vulnerability for which you do not feel it is currently exploitable. You might wish to advise Snyk to ignore the vulnerability for a while in that instance. To use the ignore command, run snyk test first and get the vulnerability ID from the CLI output.

But the snyk ignore will solve the problem for a certain period of time and it’s not reliable. Both these tools have some drawbacks, why means when I tested for the vulnerability that time the snyk showing 0 vulnerability and the when we run the npm audit it showing some vulnerabilities. So that we can say that there are some issues with these tools. These tools are not efficient. We need more reliable and stable vulnerability detection and avoidance tools to detect these vulnerabilities.

DOI: 10.5281/zenodo.6907036
ISBN: 978-93-5607-317-3@2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

V. BUILD MODEL

The model building is the main step in the fake vulnerability detection. While building the model user use some commands. The steps involved are:

1. Install npm

```
PS C:\Users\DA\Downloads\vulnerable-node-master> npm install
```

2. Make sure you have npm version 6 or higher installed.

```
PS C:\Users\DA\Downloads\vulnerable-node-master> npm -v
8.5.0
```

3. In the console, the audit report will be printed. Run the following command to get the report in JSON format:

```
PS C:\Users\DA\Downloads\vulnerable-node-master> npm audit
# npm audit report
```

```
body-parser  <=1.18.1
Depends on vulnerable versions of debug
Depends on vulnerable versions of qs
node_modules/body-parser
express  2.5.8 - 4.15.4 || 5.0.0-alpha.1 - 5.0.0-alpha.5
Depends on vulnerable versions of accepts
Depends on vulnerable versions of debug
Depends on vulnerable versions of finalhandler
Depends on vulnerable versions of fresh
Depends on vulnerable versions of qs
Depends on vulnerable versions of serve-static
node_modules/express

18 vulnerabilities (2 low, 3 moderate, 11 high, 2 critical)
```

4. Automatically apply the suggested fix. Run npm audit fix to have npm automatically correct the vulnerabilities. It's worth noting that some flaws can't be rectified automatically and will necessitate manual involvement or assessment. In the console, there will be more output.

```
PS C:\Users\DA\Downloads\vulnerable-node-master> npm audit fix

added 2 packages, removed 2 packages, changed 1 package, and audited 97
```

5. Fix for npm audit Under the hood, it performs a full-fledged npm install; all configs that apply to the installer will also apply to npm install.

```
PS C:\Users\DA\Downloads\vulnerable-node-master> npm audit fix --force
npm WARN using --force Recommended protections disabled.
npm WARN audit Updating log4js to 6.5.2,which is a SemVer major change.
npm WARN audit No fix available for ejs@<=3.1.6
npm WARN audit Updating morgan to 1.10.0,which is outside your stated d
npm WARN audit Updating debug to 4.18.1,which is outside your stated de
npm WARN audit No fix available for ejs-locals@*
npm WARN audit Updating serve-favicon to 2.5.0,which is outside your st
.
npm WARN audit Updating express to 4.18.1,which is outside your stated
npm WARN audit Updating body-parser to 1.20.0,which is outside your sta
```

```
fix available via `npm audit fix`
node_modules/pg
  pg-promise  4.7.6 - 5.9.1
  Depends on vulnerable versions of pg
  node_modules/pg-promise

4 vulnerabilities (2 high, 2 critical)
```

6. Using Snyk The steps involved are: install the snyk using npm.

```
PS C:\Users\DA\Downloads\vulnerable-node-master> npm install snyk@latest -g
```

7. Authenticating Snyk CLI.

```
PS C:\Users\DA\Downloads\vulnerable-node-master> snyk auth

Now redirecting you to our auth page, go ahead and log in,
and once the auth is complete, return to this prompt and you'll
be ready to start using snyk.
```

8. 8. Snyk test to test the data

```
PS C:\Users\DA\Desktop\ninjasworkout-master> snyk test

Testing C:\Users\DA\Desktop\ninjasworkout-master...


Issues with no direct upgrade or patch:
  X Directory Traversal [High Severity][https://snyk.io/vuln/npm:e
n express-blinker@0.0.6
    introduced by express-blinker@0.0.6
  No upgrade or patch available
```

9. The snyk ignore command alters the snyk policy file to ignore a stated issue for all occurrences based on its snyk ID or to ignore file system paths.

```
PS C:\Users\DA\Desktop\ninjasworkout-master> snyk ignore --id='npm:express-blinker:20180226
' --expiry='2022-1--10' --reason='Module not affected by this vulnerability'
```

VI. RESULT

Comparing to the NPM Audit, snyk is solving all the vulnerabilities. The npm audit updating all the dependencies but does not solving all the vulnerabilities.Using snyk ignore tool it ignoring the entire vulnerabilities.Npm audit decrees the number of vulnerabilities. Because of its user-friendliness and feature-rich command line scanner, Snyk scanner is more advantageous.Each susceptible component's dependency routes are included in Snyk's report.

```
PS C:\Users\DA\Desktop\ninjasworkout-master> snyk test

Testing C:\Users\DA\Desktop\ninjasworkout-master...

Organization:      vidhyaprasad9991
Package manager:   npm
Target file:       package-lock.json
Project name:      vulnerable-app
Open source:       no
Project path:      C:\Users\DA\Desktop\ninjasworkout-master
Local Snyk policy: found
Licenses:          enabled

✓ Tested 324 dependencies for known issues, no vulnerable paths found.
```

DOI: 10.5281/zenodo.6907036
ISBN: 978-93-5607-317-3@2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

VII. CONCLUSION

It has been recognized that automated node.js vulnerability scanning tools are required for any enterprise. Maintainers and auditors of software would benefit from a tool that would assist them in focusing their attention on functions that are likely to be the source of security flaws. After a thorough comparison of the two open source scanners, npm audit and Snyk, it is clear that Snyk scanner is superior due to its user-friendliness, feature-rich command line scanner, GitHub connection, and insightful report output. It has a lot of future scopes. Because the npm audit is updating the dependency, and it decreasing the count of vulnerabilities. So we need more reliable and flexible local CLI tool to solve the vulnerabilities.

REFERENCES

[1] Bodin Chinthanet and Bodin Chinthanet. Code-based Vulnerability Detection in Node.js Applications: How far are we? ASE 2020, 21 - 25 September, 2020, Melbourne, Australia.

[2] Aman Anupam1 and Prathika Gonchigar. Analysis of Open Source Node.js Vulnerability Scanners, International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 05 | May 2020

[3] Alexandre Decan and Tom Mens,Eleni Constantinou: On the impact of security vulnerabilities in the npm package dependency network, MSR '18, May 28–29, 2018, Gothenburg, Sweden.

[4] Mahmoud Alfadel and Diego Elias Costa, On the Threat of npm Vulnerable Dependencies in Node.js Applications: Article in IEEE Transactions on Software Engineering · November 2019.

[5] Chengwei Liu, Sen Chen: Demystifying the vulnerability propagation and its evolution via dependency trees in the NPM ecosystem.44th international software engineering(ICSE 2022) May 8-20 Virtual,May 22-27 in person,2022,Pittshurgh,PA,USA.

DOI: 10.5281/zenodo.6907036
ISBN: 978-93-5607-317-3@2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam