

Analysis of Open Source Node.js Vulnerability Scanners

Aman Anupam¹, Prathika Gonchigar¹, Shashank Sharma², Prof. Prapulla SB³, Dr. Anala MR⁴

¹Student, Department of Computer Science and Engineering, R.V. College of Engineering, Bengaluru, Karnataka, India

²MTS 1, Enterprise Solutions, PayPal Bangalore, Karnataka, India

³Assistant Professor, Department of Computer Science and Engineering, R.V. College of Engineering, Bengaluru, Karnataka, India

⁴Associate Professor, Department of Computer Science and Engineering, R.V. College of Engineering, Bengaluru, Karnataka, India

Abstract - Today, web developers don't write the entire code from scratch, rather they rely on many open source components. The widespread use means developers don't have an in-depth knowledge of how the code exactly works. Often these components are not kept up to date with the latest release and patches. These unmaintained versions add to the security threats. One way of keeping a track of such dependencies is to continuously monitor them with scanning tools. Out of the few open source tools available, *Retire.js* and *Snyk* are among the top. *Retire.js*, is recommended by OWASP (Open Web Application Security Project) for scanning node and JavaScript vulnerabilities. *Snyk* on the other hand is more feature rich than most other tools and has an extensive vulnerability database. Both these tools come with CLI (command line interface) integration, which provides a benchmark standard for comparison. This paper 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 will focus on parameters like user-friendliness, vulnerability database, extent of scanning and features of command line interface of two tools. Result-based comparison will directly compare the result of scans and the vulnerability database of two tools.

Key Words: Vulnerability Scanner, Web Security, Components with known vulnerabilities, *Retire.js*, *Snyk*, OWASP Top 10

1. INTRODUCTION

The OWASP Top Ten exposes typical software security flaws of software systems. In 2017, Using Components with Known Vulnerabilities is listed as number 9 in this list [1]. The security threats arise from the development teams being unaware of all the open source components and not maintaining the dependencies. Open source software usually refers to software whose source code is "open" and available to anyone to study, use and adapt [2]. Often the developer uses a dependency and then later doesn't keep the dependency updated to its latest secure version (latest release or patch). Components often run with the same privileges as the application, so any security flaw in the component can lead to serious issues in the entire

application. Number of security vulnerabilities in web application has grown with the tremendous growth of web application in last two decades [3]. One of the perceived values of open source software is the idea that many eyes can increase code quality and reduce the amount of bugs [4]. But a manual observation of components can be a sheer wastage of resource. OWASP acknowledges that to avoid any security issue the simplest way is to develop the dependencies in-house and avoid use of open source components.

But such approach will cost any organization a lot of resources, manpower, and timeframe. The best way to avoid such issue is to continuously monitor the vulnerabilities using National Vulnerability Database (NVD), security advisories and issue trackers.

According to WhiteSource's research, 91% of software projects contain indirect open source dependencies. The average project relies on no less than 64 different libraries with 8 different licenses. In addition, in 65% of the cases, open source components bring with them additional dependencies that are subject to a different license. [5]

It is recommended that organizations recognize that libraries are a critical part of their software infrastructure and ensure they have the level of awareness and the necessary tooling within their organization to generate appropriate assurance [6]. There are a lot of dependencies, libraries and license variations. Keeping track of all such components and finding the corresponding CVE (Common Vulnerabilities and Exposures) number is a difficult task. That is why there is a need for automated open source management system for any modern organization. Therefore, security society actively develops automated approach to finding security vulnerability [7]. The importance of using vulnerability scanners to unveil flaws in web applications before they are deployed has been realized by many organizations today [8]. The system should be able to continuously monitor the software and generate informative reports about the vulnerable components, insecure versions, security threats and CVE numbers. Web applications have become the primary source of security vulnerabilities [9]. This paper reviews two open source

scanning tools for node dependencies. The tools under consideration are Retire.js and Snyk.

1.1 Retire.js

The goal of Retire.js is to help detect use of version with known vulnerabilities. It scans both the node modules as well as JavaScript libraries. Retire.js has a separate GitHub repository where it keeps an updated list of all the vulnerable node modules and JavaScript libraries. These are in accordance with NVD, CVE, npm advisory, HackerOne and other such vulnerable dependency management databases. Being an open source tool, new vulnerabilities can be reported in a format specified by Retire.js.

Retire.js can be used in different ways:

1. Command line scanner
2. Grunt plugin
3. Chrome extension
4. Firefox extension
5. Burp and OWASP zap plugin

For reviewing the scanners this paper uses command line tool, and the report generated by that.

For detecting an insecure version, it uses filename or URL. If that doesn't work, it downloads the file and looks for comments inside that. It can also use hashes for minified files. If all the above method fails, Chrome plugin tries to run the code in a sandbox and detects the name and version of the component. Command line scanner does not use this method because running arbitrary JavaScript files in node environment can lead to complications and unwanted consequences.

1.2 Snyk

Snyk also offers a feature-rich command line tool and a more user-friendly UI for finding and reporting security vulnerabilities. Snyk command line tool scans through the project's node modules. Unlike Retire.js it doesn't scan the JavaScript libraries. Snyk also maintains its own vulnerability database and since it's an open source tool, newer vulnerabilities if discovered can be reported to the Snyk community. Snyk tool provides a feature of GitHub integration where the project hosted on GitHub can be automatically scanned for vulnerabilities through pull request. The GitHub support allows for a continuous open source management system for a modern organization/company. For common review of both the tools command line interface of both are considered in this paper.

2. FEATURE BASED COMPARISON OF RETIRE.JS AND SNYK

This section compares the two open source tools based on their features. Here we compare the extent of scanning, ease

of access, CLI options, types/format of report generated and informativeness of the report (Table 1).

Table -1: Feature Based comparison of Retire.js and Snyk

Feature Based comparison of Retire.js and Snyk	
Retire.js	Snyk
Retire.js has no GitHub integration.	Snyk supports GitHub Integration (i.e. automatically checks for new vulnerabilities introduced through pull requests).
Retire.js scans both node as well as JavaScript libraries. Retire.js command line tool has the option of scanning only node or only JavaScript library if required. Accomplished by <code>-n/--node</code> and <code>-j/--js</code> .	Snyk scans only node modules.
Retire.js reports contain key important fields that describe the vulnerability such as CVE id, Severity, info/references, version used, insecure versions, but are not as informative as Snyk reports.	Snyk reports are more informative than Retire.js reports. For example: <ol style="list-style-type: none"> 1. "functions" field lists out the functions in the code where the vulnerable component was used. 2. "patches" field lists out available patches of that component. 3. "from" field lists out the path through which the component gets introduced. 4. "isPatchable", "isUpgradable" fields state the possibility of node module being upgradable/patchable.
Retire.js generates reports in json/text. (No readable html format)	Snyk command line tool generates a more readable html report (using <code>snyk-to-html</code>)
Continuous monitoring feature in Retire.js does not exist. Scanning	Snyk monitor alerts when new vulnerabilities are disclosed.

<p>is done by manually running retire.js commands.</p>	
<p>Automatic fix feature in Retire.js does not exist. It only scans and generated reports.</p>	<p>Snyk will not only report vulnerabilities in components, but it will also offer to fix them with their wizard tool (using snyk wizard and snyk protect).</p>
<p>Both Retire.js and Snyk have ignore option that ignores certain dependencies. That means if you have a certain dependency and you know its vulnerable, but you still want to use that, this feature will ignore that component.</p> <p>Retire.js does that by using the command --ignore <paths></p>	<p>Snyk performs the ignore process by using the command ignore</p>
<p>Both Retire.js. and Snyk have option to set severity threshold i.e., you can set it to medium then only vulnerabilities which are >= medium severity are reported.</p> <p>Retire.js accomplishes that by --severity <level></p>	<p>Snyk accomplishes severity threshold by --severity-threshold</p>
<p>Retire.js directly mentions the vulnerable component file path not how it gets introduced (i.e. no description of dependency path).</p>	<p>Snyk's report includes dependency paths of each vulnerable component.</p>

```

{
  "CVSSv3": "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L/E:P/RL:O/RC:C",
  "alternativeIds": [],
  "creationTime": "2019-06-20T09:34:56.241544Z",
  "credit": [
    "Snyk Security Team"
  ],
  "cvssScore": 7.3,
  "description": "## Overview\n\n[mixin-deep](https://www.npmjs.com/package/mixin-deep)",
  "disclosureTime": "2019-06-19T09:34:10Z",
  "exploit": "Proof of Concept",
  "fixedIn": [
    "2.0.1",
    "1.3.2"
  ],
  "functions": [
    {
      "functionId": {
        "className": null,
        "filePath": "index.js",
        "functionName": "module.exports.copy"
      },
      "version": [
        "<1.1.1"
      ]
    }
  ],
}

```

Fig -1: Vulnerable node module mixin-deep reported by Snyk

```

"moduleName": "mixin-deep",
"packageManager": "npm",
"packageName": "mixin-deep",
"patches": [],
"publicationTime": "2019-06-20T09:34:08Z",
"references": [
  {
    "title": "GitHub Commit",
    "url": "https://github.com/jonschlinkert/mixin-deep/commit/8f464c8ce9761a8c9c2b3457eae9e9d404fa7af9"
  }
],
"semver": {
  "vulnerable": [
    ">=2.0.0 <2.0.1",
    "<1.3.2"
  ]
},
"severity": "high",
"title": "Prototype Pollution",
"from": [
  "owasp-nodejs-goat@1.3.0",
  "forever@2.0.0",
  "forever-monitor@2.0.0",
  "chokidar@2.1.8",
  "braces@2.3.2",
  "snapdragon@0.8.2",
  "base@0.11.2",
  "mixin-deep@1.3.1"
],

```

Fig -2: Vulnerable node module mixin-deep reported by Snyk

3. RESULT BASED COMPARISON OF RETIRE.JS AND SNYK

This section makes use of both the tools and draws comparison between them based on vulnerability report they generate. A common node-based project was taken, and both the scanners were run on the same project.

3.1 Methodology

The vulnerable node-based project for scanning is hosted on GitHub. The repository was cloned on the local system, and then the dependencies in package.json file were installed. After installation, the command line scanners (both Retire.js and Snyk) were run. Both scanners generated a report in json format which was then converted into readable format. Snyk command line has existing feature snyk-to-html for such conversion. For Retire.js scanner, third party json to html converter was used.

Retire.js Scan: [10]

retire --path <path-to-repository> --outputformat json --outputpath report.json

Retire.js command line scan with options of path (of the cloned repository), output format (json) and output path (path to the generated report). After this the generated report was converted into a html file using create-html npm package.

Snyk Scan: [11]

snyk test --json | snyk-to-html -o result.html

Snyk command line scan with options of output format (json) and simultaneous passing of json report to snyk-to-html which has the option of output path for the generated html report.

3.1.1 Project Under Scanner

OWASP NodeGoat:

Developed by OWASP to provide an environment for learning how OWASP Top 10 security risks apply to node-based web application.

3.1.2 Common vulnerabilities found by both scanners

Table -2: Common Vulnerabilities found by Snyk and Retire.js.

Common Vulnerabilities found by Snyk and Retire.js.		
npm module	Version	Severity Level
adm-zip	0.4.4	high
growl	1.9.2	high
hoek	0.9.1	low
mixin-deep	1.3.1	high
utile	0.2.1	low
uglify-js	2.3.24	medium
extend	3.0.0	critical
stringstream	0.0.5	medium
brace-expansion	1.1.6	medium
lodash	2.4.2	low
request	2.36.0	medium
tough-cookie	2.2.2	high

hawk	1.0.0	medium
qs	0.6.6	Medium

Table 2. lists out the vulnerable node modules reported by both the tools in NodeGoat project. Both tools reported the same version of component as vulnerable showing they have a consistent database for these modules.

```

{
  "file": "node_modules\\mixin-deep\\package.json",
  "results": [
    {
      "component": "mixin-deep",
      "version": "1.3.1",
      "vulnerabilities": [
        {
          "info": ["https://snyk.io/vuln/SNYK-JS-MIXINDEEP-450212"],
          "below": "1.3.2",
          "severity": "high",
          "identifiers": {},
          "summary": "Prototype pollution attack",
          "CVE": ["CVE-2019-10746"]
        }
      ]
    }
  ]
}

```

Fig -3: Vulnerable node module mixin-deep reported by Retire.js

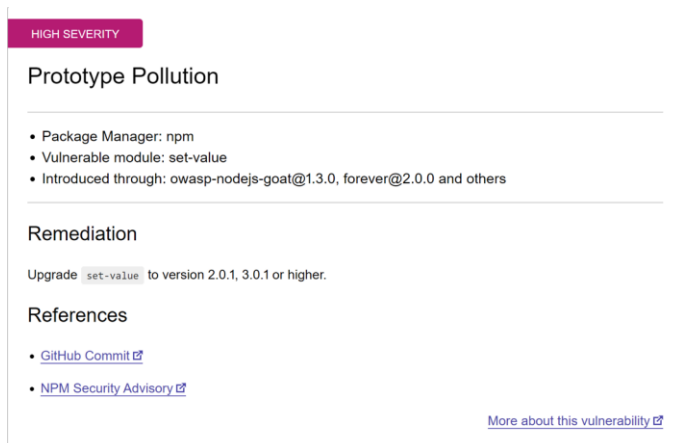
Fig. 1 and Fig. 2 (Snyk scan for mixin-deep module) show the descriptive fields like “cvssScore”, “functions” and “from” which are not present in Fig. 3 (Retire.js scan of same mixin-deep module).

3.1.3 Vulnerabilities found only by Snyk

Table -3: Vulnerabilities found only by Snyk

Vulnerabilities found only by Snyk		
npm module	Version	Severity Level
set-value	2.0.0	high
mongodb	2.2.36	high
bbson	1.0.9	high
minimist	different versions in package-lock.json	medium
helmet-csp	1.2.2	medium

Fig. 4 and Fig. 5 are the scan results from Snyk tool which show the vulnerabilities in set-value and mongodb modules. Among others listed in Table 3, these modules went undetected under Retire.js scanner.



HIGH SEVERITY

Prototype Pollution

- Package Manager: npm
- Vulnerable module: set-value
- Introduced through: owasp-nodejs-goat@1.3.0, forever@2.0.0 and others

Remediation

Upgrade `set-value` to version 2.0.1, 3.0.1 or higher.

References

- [GitHub Commit](#)
- [NPM Security Advisory](#)

[More about this vulnerability](#)

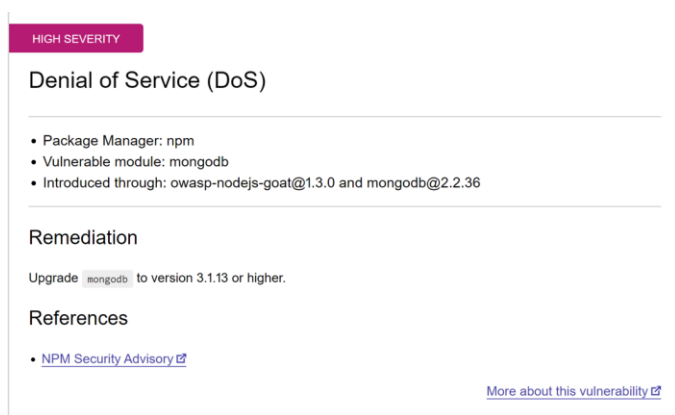
```

{
  "file": "C:\\Users\\aanupam\\Desktop\\dummy\\gitrepos\\node_modules\\grunt-retire\\test-files\\dojo.js",
  "results": [
    {
      "version": "1.4.2",
      "component": "dojo",
      "detection": "filecontentreplace",
      "vulnerabilities": [
        {
          "info": [
            "https://github.com/dojo/dojo/pull/387",
            "https://dojotoolkit.org/blog/dojo-1-14-released"
          ],
          "below": "1.10.10",
          "severity": "medium",
          "identifiers": { "PR": "387" }
        },
        {
          "info": ["https://dojotoolkit.org/blog/dojo-1-14-released"],
          "below": "1.14",
          "severity": "medium",
          "identifiers": { "CVE": ["CVE-2018-15494"] }
        }
      ]
    }
  ]
}

```

Fig -4: Vulnerability reported by Snyk (set-value module)

Fig -6: Vulnerable JavaScript library dojo.js reported by Retire.js



HIGH SEVERITY

Denial of Service (DoS)

- Package Manager: npm
- Vulnerable module: mongodb
- Introduced through: owasp-nodejs-goat@1.3.0 and mongodb@2.2.36

Remediation

Upgrade `mongodb` to version 3.1.13 or higher.

References

- [NPM Security Advisory](#)

[More about this vulnerability](#)

Fig -5: Vulnerability reported by Snyk (mongodb module)

4. CONCLUSIONS

It is established that for any organization there is a need for automated node.js vulnerability scanning tools. Software maintainers and auditors would benefit from a tool to help them focus their attention on functions that are likely to be the source of security vulnerabilities [12]. With a thorough analysis of the two open source scanners Retire.js and Snyk, it can be concluded that Snyk scanner is more advantageous because of its user-friendliness, feature-rich command line scanner, GitHub integration support and informative report generation. Retire.js scanner has an upper hand when it comes to reporting vulnerable JavaScript libraries as Snyk doesn't report JavaScript libraries. Also, it was found that Snyk was able to report more vulnerable node modules than Retire.js, hence, Snyk has a vast and extensive vulnerability database. Therefore, for any open source security management system, it should have the benefits of both the scanners, extensive informative node module scan like Snyk and vulnerable JavaScript library detection like Retire.js.

REFERENCES

- [1] Mircea Cadariu, Eric Bouwers, Joost Visser, Arie van Deursen, "Tracking Known Security Vulnerabilities in Proprietary Software Systems", 2015, IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)
- [2] Fortify Security Research Group and Larry Suto, "Open Source Security Study" <http://www.fortify.com/landing/oss/oss-report.jsp>, July 2008.
- [3] S. Rafique, M. Humayun, B. Hamid, A. Abbas, M. Akhtar and K. Iqbal, "Web application security vulnerabilities detection approaches: A systematic mapping study", 2015, 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)
- [4] H. Sajnani, V. Saini, J. Osher, and C. V. Lopes. Is popularity a measure of quality? An analysis of maven components. In Proc. of ICSME'14, pages 231-240. IEEE, 2014.

3.1.4 Vulnerable JavaScript libraries found by Retire.js

Table 4 lists out the vulnerable JavaScript libraries found by Retire.js command line scanner, Snyk scanner does not have this feature. Fig. 6 provides a sample report for a vulnerable JavaScript library generated by Retire.js.

Table -4: Vulnerable JavaScript libraries found by Retire.js

Vulnerable JavaScript libraries found by Retire.js		
JavaScript library	Version	Severity level
jQuery	1.4.4	medium
dojo	1.4.2	medium
bootstrap	3.0.0	high
handlebars	4.0.5	high
jquery-ui-dialog	1.8.10	medium
tinyMCE	4.0.26	medium

- [5] Rami Sass, "OWASP A9: Using Components With Known Vulnerabilities - Why You Can't Ignore It." June 27, 2018. <https://resources.whitesourcesoftware.com/blog-whitesource/owasp-a9-using-components-with-known-vulnerabilities>
- [6] J. Williams and A. Dabirsiaghi. The unfortunate reality of insecure libraries. *Asp.Sec.*, pages 1–26, 2012.
- [7] Dr. T. Pandikumar, Tseday Eshetu, "Detecting Web Application Vulnerability using Dynamic Analysis with Penetration Testing", 2016, International Research Journal of Engineering and Technology (IRJET)
- [8] Balume Mburano, Weisheng Si, "Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark", 2018 26th International Conference on Systems Engineering (ICSEng)
- [9] J. Walden, M. Doyle, G. A. Welch and M. Whelan, "Security of open source web applications," 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, 2009, pp. 545-553.
- [10] Retire.js [GitHub](https://github.com/RetireJS/retire.js) repository. <https://github.com/RetireJS/retire.js>
- [11] K. Elissa, "Title of paper if known," unpublished. Snyk CLI-reference. <https://support.snyk.io/hc/en-us/articles/360003812578-Our-full-CLI-reference>
- [12] K. Elissa, "Title of paper if known," unpublished. D. DaCosta, C. Dahn, S. Mancoridis and V. Prevelakis, "Characterizing the 'security vulnerability likelihood' of software functions," International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings., Amsterdam, The Netherlands, 2003, pp. 266-274.