Open Source Vulnerability Notification

Brandon Carlson[1], Kevin Leach[2(✉)], Darko Marinov[1], Meiyappan Nagappan[3], and Atul Prakash[2]

[1] University of Illinois at Urbana-Champaign, Urbana, USA
{blcrlsn2,marinov}@illinois.edu
[2] University of Michigan, Ann Arbor, USA
{kjleach,aprakash}@umich.edu
[3] University of Waterloo, Waterloo, Canada
mei.nagappan@uwaterloo.ca

Abstract. The use of third-party libraries to manage software complexity can expose open source software projects to vulnerabilities. However, project owners do not currently have a standard way to enable private disclosure of potential security vulnerabilities. This neglect may be caused in part by having no template to follow for disclosing such vulnerabilities. We analyzed 600 GitHub projects to determine how many projects contained a vulnerable dependency and whether the projects had a process in place to privately communicate security issues. We found that 385 out of 600 open source Java projects contained at least one vulnerable dependency, and only 13 of those 385 projects had a security vulnerability reporting process. That is, 96.6% of the projects with a vulnerability did *not* have a security notification process in place to allow for private disclosure. In determining whether the projects even had contact information publicly available, we found that 19.8% had no contact information publicly available, let alone a security vulnerability reporting process. We suggest two methods to allow for community members to privately disclose potential security vulnerabilities.

Keywords: Vulnerable dependency · Security disclosure · Open source

1 Introduction

Open source project maintainers often ignore or overlook important preventative maintenance tasks, including security scans for vulnerabilities in libraries [16], even with automated upgrades of libraries [19]. Neglecting such scanning tasks can impact both the end user and the software maintainer. "Using Components with Known Vulnerabilities" [23] was listed among the top ten application security risks in 2017.

Bug bounty programs can incentivize security disclosures [15] but have not taken hold as a standard policy. The National Institute of Standards and Technology is constructing a new process for receiving, analyzing, and responding

© IFIP International Federation for Information Processing 2019
Published by Springer Nature Switzerland AG 2019
F. Bordeleau et al. (Eds.): OSS 2019, IFIP AICT 556, pp. 12–23, 2019.
https://doi.org/10.1007/978-3-030-20883-7_2

to vulnerability disclosures [25], providing at least a guideline for open source projects to establish their own policies. Such vulnerability disclosure policies are growing in importance due to the continued increase of information breaches [2]. For example, Equifax recently fell victim to a high-profile breach, resulting from an unpatched open source dependency (Apache Struts) [24]. In this paper, we suggest approaches to address this shortcoming in the open source community.

This paper makes the following contributions:

- An empirical study on how many open source projects are using vulnerable dependencies, contain security policies, and ways for individuals to contact the open source project.
- A quantitative analysis of the findings from the data collection process using open source Java projects from GitHub.
- Recommendations for improving security of open source projects through improved communication among the open source community, security researchers, and the open source repository providers.

2 Motivating Experience

We were originally motivated by the Equifax breach [24], which was related to a vulnerable dependency (Apache Struts). GitHub already provides a dependency scanning tool that will alert project owners if vulnerabilities are discovered in those dependencies [7]. However, project owners are not required to address such reports. Indeed, project owners could decide the vulnerability is unexploitable through their project or that it is not worth the effort to address. In such a scenario, project owners that ignore reports from dependency scanning tools pose a risk to the community at large. Thus, we sought to investigate the prevalence of projects requiring vulnerable versions of Apache Struts.

We used Snyk [26], a dependency scanning tool, to analyze various open source projects on GitHub for vulnerable dependencies. Indeed, we found many projects depended on a vulnerable version of Apache Struts. We visited each such repository to determine if a bug bounty program or disclosure process was documented. In the majority of cases, projects contained no guidelines for reporting vulnerabilities. We attempted to communicate this vulnerable dependency through a combination of emails to project owners, opening issues, and submitting pull requests.

Project owners offered myriad responses. While some did not respond and some thanked us for the report, one owner requested private communications over public pull requests or open issues. Unfortunately, attempting personal contact for other projects resulted in our GitHub account being flagged for spam. These diverse and sometimes discouraging responses suggest the open source community could benefit from some standardized practice among open source projects for reporting such vulnerabilities without facing retribution. Motivated by this need, we chose to systematically analyze a corpus of open source projects for vulnerable dependencies and corresponding policies for reporting them.

3 Project Selection and Data Collection

To learn whether open source projects contain both easy to access contact information and a security vulnerability reporting policy we selected open source Java projects from GitHub using seven different sources: (1) Libraries.io's public dataset [21]; (2) Legunsen et al. [17]; (3) Munaiah et al. [20]; (4) GitHub trending Java projects [11]; (5) GitHub and Government list of government-sponsored open source projects [9]; (6) GitHub and Government list of government-funded, research-related open source projects [10]; and (7)GitHub and Government list of civic hackers open source projects [8]. These sources entail a combination of public repositories (1 and 4), previously published work (2 and 3), and government-funded and whitehat hacker projects (5, 6, and 7). We focused on Java projects because it is a widely used language [12] amenable to our software scanning infrastructure [26]. Table 1 summarizes the statistics of the projects from each of these sources.

Table 1. Projects selected by resource

Resource	Projects	Stars	Forks	Issues	Commits	Contributors
Libraries.io [21]	209	141	57	12.6	608.6	12.6
Legunsen et al. [17]	126	244	103	14.5	333.1	11.8
Munaiah et al. [20]	83	2195	685	59.1	545.2	16.6
GitHub trending Java projects [11]	71	2400	923	52.1	1537.5	27.0
Government-sponsored [9]	68	8	7	3.0	648.1	8.1
Government-funded research [10]	22	8	5	2.9	152.0	3.4
Government civic hackers [8]	21	3	3	2.6	2536.4	6.4
Total projects	600					

4 Evaluation

We seek to answer the following research questions (RQs):

- *RQ1: How prevalent are vulnerable dependencies among our projects?*
- *RQ2: How common are security notification policies in open source projects?*
- *RQ3: How available is contact information for open source projects?*

4.1 RQ1: Prevalence of Known Vulnerable Dependencies

Finding Vulnerable Dependency Libraries. We used Snyk [26] to scan each of our 600 Java projects. Snyk maintains a database of libraries and corresponding vulnerabilities for each version of each library. These vulnerabilities are associated with a Common Vulnerabilities and Exposures (CVE) ID or a

Common Weakness Enumeration (CWE) ID. We say a given project contains a *direct vulnerability* if it uses a vulnerable library as reported by Snyk. Similarly, we say that a project has a *transitive vulnerability* if a project's dependency itself uses a subsequent dependency with a vulnerability as reported by Snyk. We also used the GitHub API to record statistics (e.g., stars, open issues) for each project.

Results. In the 600 open source projects we examined, we found an average of 7.8 direct or transitive vulnerabilities per project. Further, we used the severity scale based on the severity rating provided by Snyk as part of the scanning process (derived from CVE and CWE reports). The average number of high severity vulnerabilities was 4.1 per project, compared to medium severity level vulnerabilities which averaged only 3.6 per project, and low severity which averaged 0.1 vulnerabilities per project.

We further categorize these open source projects containing vulnerabilities based on the severity level of vulnerabilities. A single project may contain multiple vulnerabilities that range from low to high severity. Overall, we obtained the following numbers of projects:

- 266 projects used at least one dependency with a high severity vulnerability;
- 202 projects used at least one dependency with a medium severity;
- 39 projects used at least one dependency with a low severity;
- 215 had no known vulnerability.

Note that some of these sets overlap because some projects contained multiple vulnerabilities with different severity levels.

Overall, 64.2% (385 of 600) of the projects we examined used at least one vulnerable dependency. This result is consistent with a recent report that Java applications are likely to include at least one vulnerable library [29]. Although a vulnerable library does not necessarily mean that a project using that library can be exploited, it does represent a risk to the community overall.[1] Additionally, OWASP's 2017 report [23] described how such uncertainty could facilitate adapting known exploits to many projects.

We note that not all vulnerabilities are exploited equally—CVE reports for a vulnerability contain a Common Vulnerability Scoring System (CVSS) score [22], a measure of how exploitable that vulnerability is. We leave the consideration of CVSS and exploitability for future work.

Based on our results, the open source community is in need of a standard process of reporting potential security vulnerabilities to open source project owners. This process would allow the open source community and cyber security researchers to privately disclose potential security issues in a standardized way. Such a process could improve the quality and software of open source projects.

[1] This effect can be viewed as a dual of herd immunity with immunization—the more projects use vulnerable libraries, the more risk there is to the community as a whole.

4.2 RQ2: How Common Are Security Notification Policies?

Determining Whether the Open Source Project Has a Security Policy. We first scanned the project's `README` and `CONTRIBUTING.md` files for the keywords `security`, `vulnerability`, `reporting`, or `disclosure`. We also looked for files whose name contained a keyword. If no such files or keywords existed, we scanned the owner's user or group account page on GitHub for links to company or user webpages or public Wikis. When such pages existed, we (manually) scanned them for bug bounty or security disclosure policies. In cases where no such information could be gleaned, we considered that project to have no policy for reporting security vulnerabilities.

Results. Recall from Sect. 4.1 that 385 of 600 open source projects contained at least one vulnerable dependency. Of those projects, only 3.4% (13 of 385) had a security vulnerability reporting process. The other 96.6% (372 of 385) open source projects which had a vulnerability had no publicly available security vulnerability reporting process. Overall, out of 600 open source projects, only 3.2% (19 of 600) had some type of security vulnerability reporting process. The remaining 96.8% (581 of 600) had no security vulnerability reporting process based on the aforementioned method.

For the remaining 581 projects, there is no standard recourse for reporting vulnerabilities. Recall from Sect. 2 that opening issues, submitting pull requests, or attempting private contact can result in unpredictable outcomes (including losing a GitHub account to the spam flagging system).

For the 19 open source projects that contained a security notification policy, we manually read through the policy. We broadly categorize these policies as: bug bounty program, email address, or web form. First, 4 of 19 projects had a bug bounty program administered through HackerOne [14], which outlined the process, rules, and scope for an individual reporting a vulnerability. Second, 11 of 19 projects provided an email address to contact in case of a security vulnerability. Some projects provided a more specific security reporting policy that a security researcher might follow in reporting potential issues. Third, 1 of 19 projects contained a web form for submitting vulnerabilities. This project provided a detailed security reporting process. The remaining 3 projects had unique notification policies that did not fit into these three categories.

Security Policies at Scale. Next, we considered projects from popular hosting platforms GitHub, GitLab, and BitBucket. We used two curated lists, one for BugCrowd [1] and one for HackerOne [14], that contained a list of current bug bounty programs [28]. We searched the bug bounty lists for projects contained in the Libraries.io's dataset [21]. Specifically, we used the repository owner name and project name from Libraries.io to find Bug Bounty programs.

We found that, of the 30,705,634 repositories in Libraries.io's dataset, only 6,645 open source projects have a bug bounty program. We interpret this as a sign that the open source community does not have a standard process to report

Table 2. Location of contact information

Location of contact information	Count
Account page	339
README	130
Other locations	12
None	119

security vulnerabilities. Additionally, our analysis suggests that significant effort is required to find security policies in projects where they do exist. This has the potential of inducing failures to report vulnerabilities appropriately or at all.

4.3 RQ3: Is Contact Information Available for Open Source Projects?

Approximating Effort to Discern Point of Contact. We manually inspected each open source project using the following steps:

1. Check the project's `README` file
2. Check the `CONTRIBUTING.md` file
3. Check the GitHub Wiki page
4. Check on the repository's account or group page
5. Check any provided website for the project (e.g., in the project description)
6. Check any provided website on the repository's account or group page
7. Check whether the Top Contributor for the project has their contact information *publicly* available (not just email addresses in Git commits)

We consider a project to have no contact information available if none of the above steps yield contact information. We used a stopwatch to measure the approximate time taken to find (or fail to find) contact information.

Results. We discovered 19.8% (119 of 600) of open source projects contained no publicly available contact information. Among the remaining 481 open source projects that provided contact information, 27.0% (130 of 481) contained contact information in the `README` file. The remaining 351 projects required more thorough investigation to determine contact information.

Table 2 shows the breakdown of where we found contact information for open source projects. For the majority, we found the contact information on the repository group's or top contributor's account page on GitHub. The next most popular location was within the `README` file contained in the repository. For the remaining open source projects, we found the contact information in a variety of locations described above.

Table 3 shows the breakdown of different forms of communication we found in these open source projects. In several cases we found multiple forms of communication but no preference or priority associated with each form. The majority

Table 3. Type of contact information found

Form of contact information	Count
Email	312
Website	108
Gitter	29
Google group or forum	22
Twitter	17
IRC	6
Slack	5
LinkedIn	3
Mailing list	3
List of individual contacts	2
Discordapp	1

of cases had an email address. We note a heterogeneity of communication forms associated with open source projects, adding to the potential communication burden associated with reporting vulnerabilities.

We approximated effort required to find contact information by measuring the time taken to search projects as described above. It took us an average of 44 s (±2.6 s with 95% confidence) to search a project for contact information. Times ranged from 7 s to 300 s. We observed a bimodal effect with times: projects would either take a very short time (e.g., if the top of the `README` happened to contain contact information) or a very long time (e.g., if it was not clear without searching through many files in the project).

While 44 s may not seem like a significant burden, Liu et al. [18] showed that the first 10 s that users observe a newly loaded web page are critical. These results suggest that a person may lose interest before successfully finding contact information for an open source project.

5 Recommendations and Discussion

In this section, we discuss two potential approaches to addressing the reporting of security vulnerabilities of open source projects. First, we suggest introducing a standardized `SECURITY.md` file to projects that describes basic contact information and disclosure processes for vulnerabilities. Second, we discuss a potential addition to hosting platforms (such as GitHub) to support private or hidden pull requests that enable developers or security researchers to disclose vulnerabilities.

5.1 SECURITY.md Mechanism for Vulnerability Notification

Given the dearth of security reporting policies among open source projects, we recommend the creation of a `SECURITY.md` file in open source repositories.

This file would contain contact information and the disclosure policy of an open source project. Of the 19 open source projects that contained such a policy, only one of those projects described the policy in the repository itself, while the remaining 18 projects required additional effort to find the relevant information.

The creation of a `SECURITY.md` file would provide a solution to the open source community that is currently lacking a standard process as shown by both our research and in the Snyk's report [27]. Additionally, lacking public information can make it difficult to assess the overall commitment to security from an open source project and to understand how to disclose newly discovered vulnerabilities to open source project owners [27].

We suggest an adaptation of an existing RFC, "A Method for Web Security Policies" [6], modified for an open source repository. We suggest using the `SECURITY.md` file in the root of a repository to contain basic contact information (email addresses) and optionally contain text describing the security policy, encryption, and contribution guidelines for the project. Such a file could help inform the community and cyber security researchers with an effective way to report vulnerabilities as they are discovered.

Adding `SECURITY.md` provides a beneficial method for standardization of vulnerability reporting processes. This recommendation also helps to fix an issue that was discovered in the 2017 Open Source Survey by GitHub where one of the largest issues was "Incomplete or confusing documentation" [13]. Additionally, by including this file in an open source project's repository, it shows that the project has a commitment to improving the security of the project.

Furthermore, Williams and Dabirsiaghi [29] suggest using a vetting process when choosing whether to use a project's library or source code. Considering the majority of open source projects currently do not have a security reporting process, the addition of the `SECURITY.md` would fulfill Williams and Dabirsiaghi's recommendation [29] by providing a way to vet projects.

5.2 Adapting Hosts to Facilitate Security Disclosures

As an alternative to `SECURITY.md`, open source hosting platforms could provide new features to facilitate communication of vulnerabilities. For example, based on our experience described in Sect. 2, we suggest the creation of a "verified researcher" tags to be associated with an account that has built a reputation for submitting pull requests or issues that disclose security vulnerabilities. These tags would allow project owners to evaluate contributor account reputation. As another example, hosting platforms could provide features for submitting private/hidden pull requests and issues that would be visible to project owners but not fully public.

Our recommendations seek to increase the interaction between the open source communities and vulnerability research communities. Establishing better interaction would reduce burden on reporting vulnerabilities and vetting contributors (e.g., by focusing on the requests from verified accounts). The increased interaction could help improve security in open source projects.

6 Related Work

We draw inspiration for standardized security policies from several sources.

RFC2142 [3] suggests that organizations maintain a security@domain mailbox that is used for security bulletins and questions. However, it does not specify where or how such information should be made known publicly. We build upon this work by suggesting standardized locations for security policy information to be placed in an open source software repository.

Foudil and Shafranovich [6] suggest placing a `security.txt` file in the root of a web server that allows websites to define security policies. Such information could define how owners can be contacted with security concerns or how a bug bounty program would work. We build on this work by suggesting `SECURITY.md` be added to open source repositories.

Snyk's report [27] investigated the top 400,000 public repositories on GitHub to see if there was any documentation for basic security information for the open source projects. No details were made available in the report about the process they used to look for such information, which programming languages they considered, or a definition for basic security information. Additionally, the report did not propose a solution to this problem, but instead highlighted a current problem in the open source community. In our work we focused on one programming language, Java, and outlined how we selected open source projects and gathered information. Additionally, we proposed a recommendation on how to improve the current lack of security vulnerability reporting process in open source projects.

Decan et al. [4] mined statistics about security vulnerabilities from over 600,000 open source projects. They found that 50% of security vulnerabilities survive to 30 months after being introduced. However, they also found that 50% of vulnerabilities were fixed within 1–2 months after discovery. This statistic suggests that having a mechanism in place to disclose vulnerabilities could help contribute to more rapidly fixing vulnerabilities.

GitHub already informs project owners about potentially vulnerable dependencies included in a project by examining commit messages and CVEs [7]. However, GitHub's technique only works for those projects that adopt GitHub's file format for describing dependencies, which is language-specific and limited to Java, JavaScript, .NET, Python, and Ruby. It still does not address the issue of how researchers or other developers (who may have developed other ways to detect vulnerable dependencies) could privately disclose potential vulnerabilities.

7 Limitations

Our recommendation to add a `SECURITY.md` file may be a burden for project developers to properly maintain. Additionally, older or abandoned open source projects may not add new files. In this case, the only option for a security researcher is to leave a public issue or pull request on the open source project so that future users of the project are made aware of the security vulnerability

and have a way to fix it prior to using the project. We note there is currently a limit—but no transparency!—on how many issues or pull requests a security researcher can make before an account is flagged for spam.

Our other recommendation is that open source hosting platforms provide new features for reporting security vulnerabilities. The limitation is that open source project owners cannot do it just themselves as with the `SECURITY.md` file.

8 Conclusion

Both the open source community and the providers of open source repositories need a method to communicate security vulnerabilities, which would both give a voice to project owners about how they want disclosures to occur, and give service providers (e.g., GitHub) better understanding of user needs.

In this study, we evaluated three different aspects of 600 open source Java projects: (1) how frequently open source projects use known vulnerable libraries, (2) whether open source projects contain security vulnerability reporting policies and what kind of policy is it, and (3) how much effort is required to find contact information for an open source project. Our findings showed a high ratio of the open source projects currently contain at least one vulnerable dependency. Although this does not guarantee the vulnerability can be exploited, it does show that there is at least some risk to the project and its users until the dependency can be updated to a non-vulnerable version. Additionally, with the exception of 19 open source projects, the majority of projects lacked some kind of security vulnerability reporting process that was easily accessible. Finally, the majority of open source projects studied did have some kind of contact information, but it was often not easy to find, and there is no guarantee that contact information found would be the correct person or group to contact about potential security vulnerabilities.

To address the current shortcomings of security reporting policies in open source projects, we proposed one recommendation to create a `SECURITY.md` file that would contain the necessary details about the open source project's security reporting policy and who to contact when (or if) a potential security vulnerability is discovered. To encourage the adoption of `SECURITY.md`, we could create a website in similar to securitytxt.org [5] to gain feedback from the open source community to improve the concept of `SECURITY.md`. Another recommendation is for open source hosting platforms to provide new features for private disclosure of vulnerabilities. Many challenges remain in the process of reporting potential security vulnerabilities to the open source community.

Acknowledgments. We thank Snyk [26] for providing us access to their tool and data. This material is based upon work partially supported by the US Air Force Research Laboratory under Contract FA8750-15-2-0075 and US National Science Foundation under Grant Nos. CNS-1646305, CNS-1646392, CNS-1740897, and CNS-1740916.

References

1. BugCrowd: Bugcrowd. https://www.bugcrowd.com
2. Cavusoglu, H., Cavusoglu, H., Raghunathan, S.: Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. IEEE TSE **33**, 171–185 (2007)
3. Crocker, D.: Mailbox Names for Common Services, Roles and Functions. RFC 2142, Internet Engineering Task Force (1997). http://www.rfc-editor.org/rfc/rfc2142.txt
4. Decan, A., Mens, T., Constantinou, E.: On the impact of security vulnerabilities in the npm package dependency network. In: MSR (2018)
5. Foudil, E., Shafranovich, Y.: securitytxt.org. https://securitytxt.org
6. Foudil, E., Shafranovich, Y.: A method for web security policies. Technical report, Internet Engineering Task Force (2018). https://datatracker.ietf.org/doc/html/draft-foudil-securitytxt-03
7. GitHub: About security alerts for vulnerable dependencies. https://help.github.com/en/articles/about-security-alerts-for-vulnerable-dependencies
8. GitHub: GitHub and government civic hackers projects. https://government.github.com/community/#civic_hackers
9. GitHub: GitHub and government open source projects. https://government.github.com/community/
10. GitHub: GitHub and government research projects. https://government.github.com/community/#research
11. GitHub: GitHub trending Java open source projects. https://github.com/trending/java
12. GitHub: Octoverse. https://octoverse.github.com/projects#languages
13. GitHub: Open source survey. https://opensourcesurvey.org/2017
14. HackerOne: HackerOne. https://hackerone.com
15. HackerOne: Vulnerability disclosure policy basics: 5 critical components. https://www.hackerone.com/blog/Vulnerability-Disclosure-Policy-Basics-5-Critical-Components
16. Kula, R.G., German, D.M., Ouni, A., Ishio, T., Inoue, K.: Do developers update their library dependencies? ESE **23**, 384–417 (2018)
17. Legunsen, O., Hassan, W.U., Xu, X., Roşu, G., Marinov, D.: How good are the specs? A study of the bug-finding effectiveness of existing Java API specifications. In: ASE (2016)
18. Liu, C., White, R.W., Dumais, S.: Understanding web browsing behaviors through Weibull analysis of dwell time. In: SIGIR (2010)
19. Mirhosseini, S., Parnin, C.: Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In: ASE (2017)
20. Munaiah, N., Kroh, S., Cabrey, C., Nagappan, M.: Curating GitHub for engineered software projects. ESE **22**, 3219–3253 (2017)
21. Nesbitt, A., Nickolls, B.: Libraries.io open source repository and dependency metadata (2017)
22. NIST: National vulnerability database (2018). https://nvd.nist.gov
23. OWASP Foundation: Top ten security risks. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project
24. Podjarny, G.: Open source vulnerabilities tripped Equifax, how can you defend yourself? https://snyk.io/blog/equifax-breach-vulnerable-open-source-libraries
25. Rapid7: NIST cyber framework updated with coordinated vuln disclosure processes. https://blog.rapid7.com/2017/12/19/nist-cyber-framework-revised-to-include-coordinated-vuln-disclosure-processes

26. Snyk: Snyk. https://snyk.io
27. Snyk: The state of open source (2017). https://snyk.io/stateofossecurity
28. Tetelman, A.: bounty-targets-data (2018). https://github.com/arkadiyt/bounty-targets-data
29. Williams, J., Dabirsiaghi, A.: The unfortunate reality of insecure libraries. https://www.contrastsecurity.com/the-unfortunate-reality-of-insecure-libraries