A Curated Dataset of Security Defects in Scientific Software Projects

Justin Murphy
Tennessee Technological University
Cookeville, Tennessee
jdmurphy43@students.tntech.edu

Elias T. Brady
Tennessee Technological University
Cookeville, Tennessee
etbrady42@students.tntech.edu

Shazibul Islam Shamim
Tennessee Technological University
Cookeville, Tennessee
mshamim42@students.tntech.edu

Akond Rahman
Tennessee Technological University
Cookeville, Tennessee
arahman@tntech.edu

CCS CONCEPTS

- **Security and privacy → Software security engineering**.

KEYWORDS

dataset, defects, julia, security, scientific software

ACM Reference Format:
Justin Murphy, Elias T. Brady, Shazibul Islam Shamim, and Akond Rahman. 2020. A Curated Dataset of Security Defects in Scientific Software Projects. In *Hot Topics in the Science of Security Symposium (HotSoS '20), April 7–8, 2020, Lawrence, KS, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3384217.3384218

1 INTRODUCTION

Scientific software is defined as software that is used to explore and analyze data to investigate unanswered research questions in the scientific community [6]. The domain of scientific software includes software needed to construct a research pipeline such as software for simulation and data analysis, large-scale dataset management, and mathematical libraries [4]. Programming languages such as Julia [1] are used to develop scientific software efficiently and achieve desired program execution time. Julia was used in Celeste [1], a software used in astronomy research. Celeste was used to load 178 terabytes of astronomical image data to produce a catalog of 188 million astronomical objects in 14.6 minutes [2]. The Celeste-related example provides anecdotal evidence on the value of studying Julia-related projects from a cybersecurity perspective.

We focus on scientific software projects because these projects have real-world implications for scientific research, finance-based institutions, and the energy sector. Unlike general purpose programming languages, we focus on Julia because Julia is a dedicated programming language for creating scientific applications [1]. As a hypothetical example, if security defects appear in Julia source code files used by finance-based institutions such as Aviva, BlackRock, and Federal Reserve Bank of New York, then the software is susceptible to software breaches that can impact millions of end-users.

The goal of the paper is to help researchers in performing cybersecurity research in the domain of scientific software through a curated dataset of security defects observed in scientific software projects.

We answer the following research question: ***How frequently do security defects appear in scientific software projects?***

Our contribution is a curated dataset of security defects in scientific software projects developed with Julia.

2 METHODOLOGY

Answering our research question involves two steps: repository curation and qualitative analysis of commits.

2.1 Repository curation

We use open source software (OSS) repositories to construct our dataset. As advocated by prior research [7], OSS repositories need to be curated. Software developers often use OSS repositories to store personal projects that are not reflective of the professional software development. We apply the following criteria to curate our collected repositories:

- **Criteria-1**: At least 1% of the files in the repository must be Julia source code files. By using a cutoff of 1% we assume to collect repositories that contain sufficient amount of Julia source code files for analysis.
- **Criteria-2**: The repository must be available for download.
- **Criteria-3**: The repository is not a clone.
- **Criteria-4**: The repository must have at least two commits per month. Munaiah et al. [7] used the threshold of at least two commits per month to determine which repositories have enough software development activity. We use this threshold to filter repositories with little activity.
- **Criteria-5**: The repository has at least 5 contributors. Our assumption is that the criteria of at least 5 contributors may help us to filter out irrelevant repositories. Previously, researchers have used the cutoff of at least nine contributors [8].
- **Criteria-6**: The repository uses continuous integration (CI) tool. Munaiah et al. [7] observed that professionally developed scientific software projects use CI tools such as Travis CI [3]. Our

[1] https://www.hpcwire.com/off-the-wire/julia-joins-petaflop-club/
[2] https://juliacomputing.com/case-studies/celeste.html
[3] https://travis-ci.org/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

HotSoS '20, April 7–8, 2020, Lawrence, KS, USA

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-7561-0/20/04.
https://doi.org/10.1145/3384217.3384218

assumption is that repositories that use CI are indicative of software projects that are professionally developed. We included projects that use Travis CI and have at least one month of data before and after adoption of Travis CI.

2.2 Qualitative analysis of commits

We use commits from the collected OSS repositories obtained from Section 2.1. We use commits because commits summarize changes that are made to a source code file and could identify the types of changes that are being performed on a source code file. We apply qualitative analysis [9] on the collected commits to determine which commit is related to a security defect. We apply qualitative analysis using a rater who is well-versed on software security. The rater determined each of the collected commits to be security-related by performing the following activities:

- *Activity-1*: The rater observes if any of the following keywords appear in the commit message: ‘race’, ‘racy’, ‘buffer’, ‘overflow’, ‘stack’, ‘integer’, ‘signedness’, ‘widthness’, ‘underflow’, ‘improper’, ‘unauthenticated’, ‘gain access’, ‘permission’, ‘cross site’, ‘css’, ‘xss’, ‘htmlspecialchar’, ‘denial service’, ‘dos’, ‘crash’, ‘deadlock’, ‘sql’, ‘sqli’, ‘injection’, ‘format’, ‘string’, ‘printf’, ‘scanf’, ‘request forgery’, ‘csrf’, ‘xsrf’, ‘forged’, ‘security’, ‘vulnerability’, ‘vulnerable’, ‘hole’, ‘exploit’, ‘attack’, ‘bypass’, ‘backdoor’, ‘threat’, ‘expose’, ‘breach’, ‘violate’, ‘fatal’, ‘blacklist’, ‘overrun’, and ‘insecure’. We collect these keywords from prior work [2].
- *Activity-2*: The rater determines a commit to be a security-related defect if the message indicates that an action was taken to address a security concern for the software of interest. The rater determines a commit message to be related to security concern if any of the following security objects are violated: confidentially, integrity, or availability. We apply this step because only relying on keyword search could generate false positives.
- *Activity-3*: The rater's categorization is verified with another rater's categorization. A subset of the collected commit messages is given to the other rater. Cohen's Kappa [3] is recorded and interpreted using Landis and Koch's guidelines [5] to measure agreement between the raters.

Upon completion of the above-mentioned activities we obtain a dataset where each commit is labeled as a security defect or not. If the commit is related to a security defect then the label is ‘INSECURE’. Otherwise the commit is labeled as ‘NEUTRAL’. We answer our research question by reporting the count and proportion of commits that are labeled as ‘INSECURE’.

3 RESULTS

Using our filtering criteria mentioned in Section 2.1 we obtain 20 repositories. A complete breakdown of how many repositories are satisfied using each criterion is listed in Table 1. We download these repositories on August 30, 2019.

The second author of the paper performed the qualitative analysis described in Section 2.2 to determine what commits are related to security defects. The process took 117 hours for 7,024 commit messages. The second author's categorization is verified by using the last author as another rater, who also applied qualitative analysis on the randomly-selected subset of the 50 commit messages.

Table 1: OSS Repositories Satisfying Criteria (Sect. 2.1)

Initial Repo Count	3,405,303
Criteria-1 (1% Julia files)	3,866
Criteria-2 (Available)	3,115
Criteria-3 (Not a clone)	2,173
Criteria-4 (Commits/Month ≥ 2)	2,173
Criteria-5 (Contributors ≥ 5)	253
Criteria-6 (CI)	20
Final Repo Count	20

The subset includes 50 commit messages. The Cohen's Kappa is 1.0, which is ‘almost perfect’, according to Landis and Koch [5].

We identify 308 commits in the collected 20 repositories to be insecure. The proportion of security defects is 4.4%. The labeled dataset is available online [4]. The dataset consists two CSV files: ‘HOTSOS2020_SCI_SOFT_SECU’ has mapping of security-related labels to commits, whereas ‘HOTSOS2020_SCI_SOFT_META’ contains the repository links of commits. Both files can be imported using standard tools such as Python Pandas [5].

4 CONCLUSION

The cybersecurity research community might benefit from a curated dataset where commits mined from scientific software projects are labeled as security defects. We constructed a curated security defect dataset by mining 7,024 commits from 20 scientific software projects. Our dataset can be beneficial for cybersecurity researchers in two ways: (i) use the dataset to conduct security defect categorization and prediction research; and (ii) find undiscovered security defects in scientific software projects.

REFERENCES

[1] [n.d.]. The Julia Language. https://docs.julialang.org/en/v1/.

[2] Amiangshu Bosu, Jeffrey C. Carver, Munawar Hafiz, Patrick Hilley, and Derek Janni. 2014. Identifying the Characteristics of Vulnerable Code Changes: An Empirical Study *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 257–268. https://doi.org/10.1145/2635868.2635880

[3] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46. https://doi.org/10.1177/001316446002000104 arXiv:http://dx.doi.org/10.1177/001316446002000104

[4] George Thiruvathukal Jeffrey. Carver, Neil Hong. 2016. *Software Engineering for Science* (1st ed.). CRC Press, NY, NY, USA.

[5] Richard Landis and Gary Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. http://www.jstor.org/stable/2529310

[6] E. S. Mesh and J. S. Hawker. 2013. Scientific software process improvement decisions: A proposed research strategy. In *2013 5th International Workshop on Software Engineering for Computational Science and Engineering (SE-CSE)*. 32–39. https://doi.org/10.1109/SECSE.2013.6615097

[7] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. 2017. Curating GitHub for engineered software projects. *Empirical Software Engineering* (2017), 1–35. https://doi.org/10.1007/s10664-017-9512-6

[8] Akond Rahman, Amritanshu Agrawal, Rahul Krishna, and Alexander Sobran. 2018. Characterizing the Influence of Continuous Integration: Empirical Results from 250+ Open Source and Proprietary Projects *(SWAN 2018)*. ACM, New York, NY, USA, 8–14. https://doi.org/10.1145/3278142.3278149

[9] Johnny Saldaña. 2015. *The coding manual for qualitative researchers.* Sage.

[4] http://tiny.cc/hotsos20_scisoft

[5] https://pandas.pydata.org/