How Long Do Vulnerabilities Live in the Code? A Large-Scale Empirical Measurement Study on FOSS Vulnerability Lifetimes

Nikolaos Alexopoulos, Manuel Brack, Jan Philipp Wagner, Tim Grube, and Max Mühlhäuser, *Technical University of Darmstadt*

https://www.usenix.org/conference/usenixsecurity22/presentation/alexopoulos

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

This artifact includes software that covers all 3 basic functionalities needed to reproduce the results of our paper "How Long Do Vulnerabilities Live in the Code? A Large-Scale Empirical Measurement Study on FOSS Vulnerability Lifetimes". The three main functionalities are (a) data collection, (b) heuristic execution (code analysis), and (c) experiments (analysis incl. plots, tables, etc. that are presented in the paper). The artifact shows that the process described in the paper is reproducible and the results of executing the artifact (tables, plots) should be similar to the results reported in the paper (by executing the artifact now, new CVEs will be added to the analysis so results are expected to differ slightly from the ones reported in the paper).

The artifact is shipped as a docker image (the repository includes a Dockerfile that can be used to create the image). We tested on Docker version 18.09.1 on Debian GNU/Linux 10. Function (b) "heuristic execution" is CPU-intensive and parallelized, so we recommend using a machine with many CPU cores to speed up the process. The docker container requires ~60GB of disk space, mainly to download the repositories of the projects in the study. Although we tested the artifact on a "big" 128-core machine, we expect it to run without problems on "smaller" machines. We offered reviewer of the AE Committee of USENIX Security '22 ssh access to the machine we used for testing.

A.2 Artifact check-list (meta-information)

- **Data set:** The "ground-truth" dataset (CVE to VCC mappings) is included as a set of files in the repository. The process to create the main dataset is performed by the artifact.
- **Run-time environment:** The artifact is intended to be executed in a docker container, so docker is required (which generally also implies root privileges on the machine).
- **Hardware:** No specific hardware is required, although the execution of the experiments can be accelerated with the use of multiple CPU cores (heuristic execution) and good bandwidth (cloning repositories). Since the execution can take long to complete, a dedicated machine or server (with internet access) is required.
- **Output:** The output consists of the tables and plots included in the paper.
- **Experiments:** The artifact includes a "run_all.sh" bash script that performs all the actions required. Alternatively the user can run the commands in this script manually.
- **How much disk space required (approximately)?:** 60GB
- **How much time is needed to prepare workflow (approximately)?:** 1-5 min.
- **How much time is needed to complete experiments (approximately)?:** 10-80 hrs (mainly depending on number of cores available and bandwidth)
- **Publicly available (explicitly provide evolving version reference)?:** https://github.com/manuelbrack/VulnerabilityLifetimes/tree/usenix_ae
- **Code licenses (if publicly available)?:** GPL-3.0
- **Data licenses (if publicly available)?:** CC BY 4.0
- **Archived (explicitly provide DOI or stable reference)?:** https://github.com/manuelbrack/VulnerabilityLifetimes/tree/usenix_v1.0

A.3 Description

A.3.1 How to access

https://github.com/manuelbrack/VulnerabilityLifetimes/tree/usenix_ae

A.3.2 Hardware dependencies

The process requires ~60GB of disk space. This is mainly because the artifact looks into the repositories of some big projects, such as chromium and Linux. More cores will mean the artifact will run faster.

A.3.3 Software dependencies

Git to clone the repository and docker to build and run the image.

A.3.4 Data sets

The required datasets are either included in the repository or created by the artifact.

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

Clone the repository[1] and follow the instructions in the readme[2] to build and run the docker image (docker usually implies that root access is required on the machine you are using).

[1] `git clone --branch usenix_v1.0` https://github.com/manuelbrack/VulnerabilityLifetimes

[2] https://github.com/manuelbrack/VulnerabilityLifetimes/blob/usenix_ae/README.md

A.5 Experiment workflow

The artifact implements the 3 main functionalities required to reproduce the results of our paper: (a) dataset creation, (b) heuristic execution, and (c) results analysis. All required steps are included in a "run_all.sh" bash script with comments that explain the purpose of each step. At the end we recommend that you copy the contents of the "/project/VulnerabilityLifetimes/out/" directory[3] to a machine with a GUI so you can inspect the plots.

A.6 Evaluation and expected results

Note that this is a measurement paper and that the artifact also implements the critical part of dataset creation. The ability of this code to create an up-to-date dataset automatically is a key contribution of this artifact. Given that the dataset collection for the paper was executed some months before publication, you should expect some variation of the results caused by new data points. You can get exactly the same results as the ones reported in the paper by importing the mappings from https://figshare.com/s/4dd1130c336f43f6e18c and running the analysis scripts but there is no real reproduction value there. The main claims of the paper can be summarized by the following:

1. The heuristic provides good estimates for vulnerability lifetimes
    - Check the output of the `/out/heuristic.csv` file for the content of Table 2 of the paper. Here you should note that the numbers reported are similar; especially, the numbers in the last 2 columns of the file are smaller than the respective numbers of the previous columns.
    - Check the plots under `./out/year_trends/year_trend_linux_gt_comp.pdf`, `./out/distributions/distribution_gtdata_gt.pdf`, `./out/distributions/distribution_gtdata_heuristic.pdf`, as well as the qqplots in the same directory. They should be similar to Figures 3 and 4, showing that the results of the heuristic are close to the ground truth data.
2. *Section 5.1:* Look into `./out/lifetimes_table.csv` for results similar to the ones reported in Table 3. You should be able to observe big differences between projects and a higher average/mean value than median. This table can also be used as a check to see if the experiment has been executed successfully. If results in your file are similar to the ones reported in the paper, then you can be pretty sure that the experiment ran correctly.
3. *Section 5.2:* Look into the plots at `./out/distributions/distribution_All_pdf.pdf` and `./out/distributions/qq_plot.pdf`. for similar results to Figures 5 and 6. Here you should be able to observe that the exponential distribution is a good fit to the data.
4. *Section 5.3:* Inspect plots with the naming convention `./out/year_trends/year_trend_{project}.pdf` for similar results to Figure 7 (increasing trends except for Firefox).
5. *Section 5.4:* Look into the plots at the directory `./out/regular_code_age/` for similar results to Figure 9. Here you should be able to observe the correlation between vulnerability lifetimes and code age and, especially for Chromium, that lifetimes are increasing slower than code age.
6. *Section 5.6:* Look at the plot at `./out/year_trends/year_trend_kernel_mem_vs_others.pdf` for similar results to Figure 10. You should be able to note that the trend is increasing both for memory vulnerabilities and for other types.

Apart from the main results listed above, you can find many more results (both presented in the paper and additional material in the directories referenced above). Also, some results, such as the statistical tests for vulnerability types are printed in stdout. You can find these results in the log file of the execution.

A.7 Experiment customization

The code is written in a way that new projects and data sources can be added with relatively little additional effort (although not trivially). See the readme in the main branch of the repository for more information. However, the scripts for the artifact evaluation do not support seamless addition of projects to analyze. This could be a point for future work.

A.8 Notes

Warnings during the execution of the artifacts are not suppressed and are to be expected. Here is a short explanation:

- 'Cannot add or update a child row...': a fixing commit-CVE mapping has been identified by text mining techniques but the CVE is not in the list of CVEs that affect the project as identified by cpe.
- 'CVE search: 89it [00:14, 37.71it/s]...': The CVE entry is not complete.
- '62f4f82ad39f177538f733b37cdd5dabd8f333de could not be saved...': Commit message includes a picture (emoji) – see https://github.com/chromium/chromium/commit/62f4f82ad39f177538f733b37cdd5dabd8f333de. This can be fixed in a future version of the tool.

[3] e.g. `docker cp` and then `scp` if you are connected to a server

- ‘warnings.warn("Commit not found 0".format(commitsha))‘: a fixing commit-CVE mapping has been identified by text mining techniques but the commit is not in the repository.
- ‘ValueWarning: omni_normtest‘: some projects have few points and such warnings are natural.
- ‘UserWarning: no blames‘: No commits were blamed by the heuristic for a given fixing commit, e.g. because it changed only non C/C++ files.
- ‘WARNING:root:SKIPPED powernorm distribution (taking more than 30 seconds)‘: Expected warning from the fitter package (https://fitter.readthedocs.io/en/latest/)

A.9 Version

Based on the LaTeX template for Artifact Evaluation V20220119.