USENIX Security '25 Artifact Appendix: Stack Overflow Meets Replication: Security Research Amid Evolving Code Snippets

Alfusainey Jallow, Sven Bugiel

A Artifact Appendix

A.1 Abstract

The study examines the impact of Stack Overflow code evolution on the stability of prior research findings derived from Stack Overflow data and provide recommendations for future studies. We systematically reviewed papers published between 2005–2023 to identify key aspects of Stack Overflow that can affect study results, such as the language or context of code snippets. Our analysis reveals that certain aspects are non-stationary over time, which could lead to different conclusions if experiments are repeated at different times. We replicated six studies using a more recent dataset to demonstrate this risk. Our findings show that four papers produced significantly different results than the original findings, preventing the same conclusions from being drawn with a newer dataset version. The code and data artifact are permanently hosted on Zenodo, allowing verification of the claims made in this study.

A.2 Description & Requirements

Our artifact provides the necessary components to reproduce the experiments and results presented in our paper. It consists of two main parts: the source artifact, which includes the code and environment setup, and the data artifact, which contains the datasets used in our experiments.

To ensure reproducibility, we have included a *requirements.txt* file that lists all Python dependencies used in the project. Evaluators should use this file to create a conda environment, ensuring the exact setup used for the experiments. The source artifact is stored on GitLab, and the data artifact is available on Zenodo.

For reproducibility, evaluators should download the source code and data from Zenodo. The artifact has been designed to be run without any security, privacy, or ethical concerns, and we have ensured that the setup is consistent and reliable for evaluation.

A.2.1 Security, privacy, and ethical concerns

Our artifact does not pose any security for the evaluators. There are no destructive steps involved in its execution, nor are any security mechanisms disabled during its operation.

A.2.2 How to access

The complete artifact for this paper is available on Zenodo at the following URL: `https://zenodo.org/records/14733196`. Download the archive file and extract its contents to a local directory.

A.2.3 Hardware dependencies

To run the scripts in the *sources/**/analysis* directory, we tested on a machine with 756 GB of RAM and a CPU with 64 cores, along with 3.5 TB of disc space. However, we did not use all the 3.5 TB space and we believe that a disc space of 1.5TB would also work. However, to reproduce the major claims described in Section A.4.1, you only need to run the scripts in the `sources/**/analysis` directory. The instructions for running each experiment to reproduce the major claims are described in Section A.4.2.

A.2.4 Software dependencies

Our artifact relies on the following software dependencies:

- **virtual environment**: A software for creating python virtual environment, e.g., Conda or Venv, is required.
- **Docker** for creating the MySQL docker container.
- **7-Zip** for extracting the artifact archive file stored on Zenodo.

A.2.5 Benchmarks

We have released the data set required for the experiments reported in our paper on Zenodo. The data set can be accessed and downloaded from the following stable URL: `https://zenodo.org/records/14733196`.

A.3 Set-up & Installation

Create a fresh virtual environment and install the required packages using *pip*. The example below shows how to create a new conda environment named 'usenix-ae' and install the required packages:

```
$ conda create -n usenix-ae python=3.12
$ conda activate usenix-ae
$ pip install -r requirements.txt
```

Afterward, set up a Docker container with a MySQL server, according to the following steps, that holds the `sotorrent22` database (and the required tables) needed to replicate claims **C1**, **C7**, **C13**, **C14**, **C15**, and **C16** described in Section A.4.1.

```
1. cd into the directory
cd mysql-docker

2. Build the Docker image
$ docker build -t paper403 .

2. Run the container (in detached mode)
$ docker run --name mysql-server -d \
  -p 3306:3306 paper403

3. Connect to the DB:
$ docker exec -it mysql-server mysql \
  -u sotorrent -psotorrent
```

Finally, follow the instructions provided in Step 5 of the Zenodo description to import the necessary database tables.

A.3.1 Basic Test

Once the MySQL server docker container is running, connect to the server and list all the databases on the server. Confirm that the *sotorrent22* database is listed:

```
1. Connect to the server
docker exec -it mysql-server mysql -u sotorrent -p

2. List the databases. Confirm that sotorrent22 is listed
$ show databases

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| sotorrent22        |
+--------------------+
3 rows in set (0.01 sec)
```

Finally, confirm that all the database tables have been successfully imported into the *sotorrent22* database:

```
mysql> use sotorrent22
mysql> show tables;

+--------------------+
| Tables_in_sotorrent22 |
+--------------------+
| Posts              |
| PostHistory        |
| Users              |
| CppCheckWeakness   |
| CrawledPapers      |
| DicosResults       |
| Violations         |
| PostReferenceGH    |
+--------------------+
```

A.4 Evaluation workflow

Below, we outline the key claims in the paper along with the corresponding experiments to reproduce each claim. The evaluators identified minor discrepancies for **C9** (observed: 30,359, expected: 30,343) and **C18**: *no-dupe-keys* under **Parse Errors (PE)** violations (observed 874 vs. expected 2,070). These discrepancies have been addressed in the extended version of the paper, which is available on Arxiv[1].

A.4.1 Major Claims

(C1, Optional): *Our literature search yielded 42 relevant studies. This is proven by the experiment (E1) described in Section 3.2 whose results are reported in Table 1.*

(C2): *Certain programming languages have exhibited distinct trends regarding the overall number of added snippets and the ratio of security-relevant edits, with languages like C/C++ being relatively stable, while others like JavaScript and Python have fluctuated over time. This claim is demonstrated by the monthly trends in the addition of new code snippets shown in **Figure 3** and the analysis of post edits in **Figure 4** of Section 5.*

(C3): *The ratio of security-relevant comments on Stack Overflow has steadily increased, suggesting an ongoing community effort to improve the quality of shared content. This claim is backed by the data shown in **Figure 5**, highlighting the increasing percentage of security-relevant commits and comments over time.*

(C4, page 8): *Code snippets with weaknesses (i.e., $Code_w$) increased (11,748 ↗ 30,254), indicating a growth rate of 157.5%.*

(C5, page 8): *The authors identified 12,998 CWE instances in the latest versions of 7,481 answers, whereas we found 7,679 instances in 5,721 answers.*

(C6, page 8): *We noticed a shift in the ranking of CWE types: CWE-758 climbed 6th ↗ 2nd; CWE-401 dropped 2nd ↘ 3rd; CWE-775 fell 3rd ↘ 7th.*

(C7, page 8): *As revisions increased from 1 to 3 or more, the proportion of improved $Code_w$ only rose from 3.1% to 7.4%. This claim is shown on the right-hand side of Table 7 in Appendix A.*

[1] https://arxiv.org/abs/2501.16948

(C8, page 8 and page 9): *We found that the fraction of users with vulnerable C/C++ snippets more than doubled compared to the original findings. Moreover, the number of users that contributed just one vulnerable snippet also increased. Further, the authors reported that users who contributed multiple vulnerable snippet versions repeatedly contributed the same CWE type. We found that these users contribute different types of CWE with the same likelihood.*

(C9, page 10): *Using the same approach on our dataset of 1,046,052 posts, we discovered 30,359 (2.9%) insecure posts, i.e., an* ***increase of 52%****. Among these, 4,887 (16.1%) insecure posts contained all three features, i.e., an* ***increase of 155%****, while the remaining 25,472 posts contained two features.*

(C10, page 10): *We found that* DICOS *had an 11% precision (compared to the authors' 91%), 32% accuracy (versus 89%), and an 87% recall (versus 93%). These results are shown in Table 4. This low precision and accuracy indicates that the ability of* DICOS *to detect insecure code snippets has decreased significantly.*

(C11, page 11): *Using the newer version of the SOTorrent dataset, we observed a higher ratio of insecure posts between accepted (7.72%) and non-accepted (6.61%) answers. Figure 10 in Appendix B compares the original and replication results.*

(C12, page 11): *The types of security weaknesses detected in C/C++ code snippets have changed over time, with memory leaks becoming the most prevalent, replacing undefined behavior as the dominant issue. This claim is supported by the comparison of the original and replication results for RQ3, as seen in Section 6.2.3. Figure 11 in Appendix B compares the original and replication results.*

(C13, page 12): *After applying the authors' filtering criteria, we obtained 12,095 questions containing 72,202 answers, of which 10,140 were accepted answers. This means the number of code snippets matching the authors' filtering criteria has dropped since 2018: 529,054 ↘ 239,575.*

(C14, page 12): *Our findings regarding the number of questions with at least one insecure answer differ significantly: 18.1% (out of 10,861) dropped to 4.9% (out of 12,095). Similarly, the percentage of accepted answers containing at least one insecure snippet also decreased: 9.8% (out of 7,444) ↘ 2.2% (out of 10,139).*

(C15, page 12): *Although the vulnerability rankings from the original study remain unchanged, we observed a shift in the number of affected snippets: code injection increased (2,319 ↗ 5,734), while insecure cipher (564 ↘ 356), insecure connection (624 ↘ 276), and data serialization (153 ↘ 140) all dropped. Like the original study, no snippets were impacted by XSS vulnerabilities.*

(C16, page 12): *Like the original study, we also found no significant difference between the two user groups. However, we observed different $p-value$ (0.9 ↗ 6.2) and Cliff's delta (0.01 ↗ 0.03) values.*

(C17, page 13): *The number of violations in JavaScript code snippets increased from 5,587,357 to 7,385,044, with the average violations per snippet rising from 11.94 to 28.8. This claim is supported by the results in Section 6.4.3 (****Revisiting RQ1 Findings****).*

(C18, page 13): *Due to Stack Overflow's evolution, the number of violations in each category has risen. Further, the ranking of violations within certain categories has changed. This claim is supported by the results in Section 6.4.3 (****Revisiting RQ2 Findings****).*

A.4.2 Experiments

For all the experiments described below, make sure to create and activate a Python virtual environment based on the *requirements.txt* file (see Section A.3). All the experiments require the virtual environment to be set up and activated. Not all of the experiments listed below require the MySQL server Docker container to be running. The claims that do require the Docker container are **C1**, **C7**, **C13**, **C14**, **C15**, and **C16**. The other claims do not require the container for verification.

(E1 (Optional)): *[Table 1, Section 3.2] [10 mins human-hour + ≈20 mins compute-time]:* The goal of this experiment is to reproduce **C1**, which pertains to the number of research studies identified through our systematic literature review process outlined in Figure 2. The relevant studies discovered are presented in Table 1, and the aim of this experiment is to replicate the studies listed in Table 1.

Preparation: *Ensure that the MySQL Server Docker container is running and that you can connect to the server as outlined in Section A.3.1. You will also need access to an OPENAI API Key. The script requires the API Key and the* `DB_HOST` *to be set in the environment where it is being executed. For example, when running the script from the terminal, the* `OPENAI_API_KEY` *should be configured in the environment. If the default database user and password created during the creation of the docker container is not changed, then the script will use those credentials to connect to the DB server on localhost. Otherwise, configure the* `DB_HOST` *environment variable with the server's IP address.*

Execution: *Run* `python relevant_studies.py`

Results: *The script will display all the relevant studies described in Table 1.*

(E2): *[Section 5] [10 mins human-hour + ≈15 mins compute-time]: The goal of this experiment is to reproduce claims* ***C2*** *and* ***C3****, which pertains to the statistics about the evolution of code and their surrounding context (i.e., comments and commit messages) on Stack Overflow.*

The claims and statistics supporting them are described in Section 5.

Preparation: *Ensure that the virtual environment is activated and contains all the packages listed in the requirements.txt file. Also ensure that the **data** directory is located in its proper location and contains all necessary files. All related scripts for this experiment are located in the* `sources/5_evolution_of_stackoverflow` *directory.*

Execution: *Run all the code cells in the* `Evolution.ipynb` *and* `PCS.ipynb` *Jupyter notebooks, which is found in the* `5_evolution_of_stackoverflow` *folder. The notebooks include direct quotes from the paper in markdown cells, with the corresponding code to reproduce each quote provided right after it.*

Results: *Each code cell in the notebook, once executed, will reproduce the claims C2 and C3 described in Section 5 of the paper.*

(E3): *[Section 6.1] [10 mins human-hour + ≈20 mins compute-time]: The goal of this experiment is to demonstrate how claims C4, C5, C6, C7, and C8 described in Section 6.1 can be reproduced.*

Preparation: *Ensure that the virtual environment is activated and contains all the packages listed in the requirements.txt file. The* `table_7.py` *script connects and queries the sotorrent22 database running in the MySQL Server docker container. All related scripts for this experiment are located in the* `case_study_1/analysis` *directory.*

Execution: *Execute the following scripts to reproduce the findings for each claim.*

C4: *Execute the* `main()` *function inside the* `table_2.ipynb` *Jupyter notebook.*

(C5, C6): *Execute all code cells inside the* `RQ1_RQ2.ipynb` *Jupyter notebook.*

C7 *Run* `python table_7.py`

C8 *Study and execute all code cells of the* `Zhang.ipynb` *Jupyter notebook. The notebook includes direct quotes from the paper, with the corresponding code to reproduce each quote provided right after it.*

Results: *Executing the scripts described above will reproduce claims C4, C5, C6, C7 and C8.*

(E4): *[Section 6.2] [10 mins human-hour + ≈20 mins compute-time]: The goal of this experiment is to demonstrate how claims C9, C10, C11 and C12 described in Section 6.2 can be reproduced.*

Preparation: *Ensure that the virtual environment is activated and contains all the packages listed in the requirements.txt file. All related scripts for this experiment are located in the* `case_study_2/analysis` *directory. The results for Claim C10 were obtained through manual labeling conducted by two researchers. The* `data/dicos_discovery_accuracy.xlsx` *CSV file includes multiple tabs, each containing the coding label results for each task. The **Discovery Accuracy** tab includes the results for the precision, accuracy, and recall calculations.*

Execution:

(C9, C11, C12): *Run all the code cells in the* `main.ipynb` *Jupyter notebook, which is found in the* `case_study_2/analysis` *directory. The notebook includes direct quotes from the paper in markdown cells, with the corresponding code to reproduce each quote provided right after it.*

(C10): There is no Python script to run in order to verify this claim. Instead, the results of this experiment can be found in the Discovery Accuracy tab of the `data/dicos_discovery_accuracy.xlsx` CSV file.

Results: *Each execution of the python script will reproduce the findings for each claim.*

(E5): *[Section 6.3] [10 mins human-hour + ≈10 mins compute-time]: The goal of this experiment is to demonstrate how claims C13, C14, C15, and C16 described on page 12 in Section 6.4 can be reproduced.*

Preparation: *This experiment utilizes scripts that require DB server credentials to connect to and query the sotorrent22 database. The scripts expect the environment variables* `DB_USER` *and* `DB_PASSWORD` *to be set, containing the database username and password. These variables will be used by the script to connect to the MySQL Server Docker container, with the default host set to localhost (127.0.0.1). All related scripts for this experiment are located in the directory* `case_study_3/analysis`*.*

Execution: *Execute the following scripts to reproduce the findings for each claim.*

C13: *Execute the code cells in* `general_stats.ipynb`

C14: `python revisiting_RQ1_findings.py insecure_post_stats`

C15 `python revisiting_RQ1_findings.py rankings`

C16 `python revisiting_RQ2_findings.py`

Results: *Each execution of the python script will reproduce the findings for each claim.*

(E6): *[Section 6.4] [10 mins human-hour + ≈10 mins compute-time]: The goal of this experiment is to reproduce claims **C17** and **C18**, described on page 12 in Section 6.4.*

Preparation: *Ensure that the virtual environment is activated and contains all the packages*

listed in the requirements.txt file. This experiment uses the `violations.feather` *and* `rule_categories.feather` *files located in the* `data/feather_files` *directory.*
Execution: *Run all the code cells in the* `main.ipynb` *Jupyter notebook, which is found in the* `case_study_4/analysis` *directory. The notebook includes direct quotes from the paper in markdown cells, with the corresponding code to reproduce each quote provided right after it.*
Results: *Each code cell in the notebook, once executed, will reproduce claims C17 and C18.*

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.