

LIBERATOR artifact

Flavio Toffalini[†]
Ruhr-Universität Bochum; EPFL
flavio.toffalini@rub.de

Nicolas Badoux
EPFL
nicolas.badoux@epfl.ch

Zurab Tsinadze
EPFL
zurab.tsinadze@epfl.ch

Mathias Payer
EPFL
mathias.payer@nebelwelt.net

I. ARTIFACT APPENDIX

In this Appendix, we provide the requirements, instructions, and further details necessary to reproduce the experiments from our paper *Liberating libraries through automated fuzz driver generation: Striking a Balance Without Consumer Code* (DOI: 10.1145/3729365).

A. Description and requirements

The artifact contains the material to reproduce the results: RQ1: t_{gen} vs t_{test} trade-off analysis (Section 7.1—Figure 3), RQ2: How does LIBERATOR test libraries? (Section 7.2—Figure 4 & Figure 5), RQ3: Comparison with state-of-the-art (Section 7.3—Table 4, LIBERATOR part), and RQ5: Ablation study (Section 7.5—Table 9 & Table 10). This material is released under the Apache License 2.0.

- 1) *Accessing the artifact*: We release the artifact on a public GitHub repository. While the main branch contains the latest version of the code, the `fse-25-artifact` tag contains the exact version of the code that was submitted for review in the artifact evaluation. Additionally, the code is available on Zenodo with the DOI:10.5281/zenodo.14888072.
- 2) *Hardware dependencies*: The artifact requires a machine with at least 64GB of RAM and a 1TB disk.
- 3) *Software dependencies*: The artifact was tested on Ubuntu 20.04 and require the ability to run Docker containers. An active internet connection is also necessary for fetching the target libraries. Additionally, `curl`, `git`, `docker`, and `pip` should also be installed.

B. Artifact installation

The initial step is to clone the repository and build the Docker image. Additionally, please run `./preinstall.sh` from inside the repo to install dependencies. Notice that this script will also install and compile LLVM, this operation may take a few hours depending on your machine.

Each experiment is encapsulated in one or multiple Docker container. The Dockerfile is available at the root of the artifact repository. We do not provide support for running the experiments locally.

C. Experiment workflow

Our artifact aims at reproducing the results from four experiments presented in the paper. The first aims at exposing

the trade-off of testing vs creating drivers and compare our best results with our competitors. The second experiment evaluates of LIBERATOR drivers explore a library. Lastly, the third experiment demonstrate the value of each component of LIBERATOR by conducting an ablation study.

We propose to run these experiments sequentially as they are presented in the paper. The artifact provides scripts to run the experiments and collect the results. The scripts will also generate tables similar to the ones presented in the paper.

More complete and detailed instructions on how to test a library is available in the README file of the repository.

D. Major claims


- (C1) *Trade-off analysis & Comparison with state-of-the-art*: LIBERATOR exposes the trade-off between creating and testing driver. This is showcased in experiment E1 which runs LIBERATOR with four different balance of testing and driver creation duration.

The trade-off results are presented in Figure 3 while the comparison with the state-of-the-art is in Table 4, 5, & 6.

Due to time constraints, we do not provide automatic scripts to run and extract the data from our competitors (i.e., Hopper, UTOPIA, FuzzGen, OSS-Fuzz and OSS-Fuzz-Gen) that would be necessary for Table 4, Table 5, & Table 6. We, however, provide instructions on how to run these experiments.

- (C2) *Library exploration capability*: By diversifying the driver set, LIBERATOR explores different regions of a library. Our experiment E2 highlights on the 15 libraries we tested. In the paper, the results are available in Figure 4 & Figure 5. The plot can slightly differ from the ones in the paper due to the different hardware configurations, but we expect the global trend across the benchmark to remain consistent.
- (C3) *Ablation study*: To highlight the contributions of each module of LIBERATOR, we conduct an ablation study to assess the cost and the gained coverage due to each. The results are presented in Table 9 & 10 and can be reproduced through the experiment E3.

E. Evaluation

In this section, we provide the detailed steps to run the experiments and process the results to get the tables presented in the paper. Overall, this process requires around  two to three days of computation time on a multi-core server.

[†]Corresponding author.

While we conducted five repetitions for our evaluation, to reduce the necessary resources for the artifact evaluation, we run only one repetition.

We recommend running these experiments in `tmux` or screen session to avoid interruptions.

Experiment 1 (E1) - Claim (C1): Trade-off analysis:

[2 humans minutes + 60 compute-hours] The experiment leverages LIBERATOR to explore how each target library behave along the trade-off between t_{gen} and t_{test} . The experiment consists of running four campaigns with each a different ratio of t_{gen} to t_{test} and plotting the results.

[Preparation] Ensure that to have the results from the static analysis which are provided in the repository.

```
./install_analysis_result.sh
# Otherwise the static analysis will be run as
↪ part of the other commands but can take
↪ more than a day to complete.
```

```
./run_campaign_artifact.sh
# Expected output: You should see multiple
↪ docker builds and have logs for the
↪ different driver created and run.
```

[Execution] Run the commands:

```
# Kick the the campaigns necessary. We proceed
↪ first to  $t_{\text{gen}}$  of 24h and reuse partial
↪ results for smaller  $t_{\text{gen}}$  values.
./fig3.sh
./tab4.py
```

[Results] Upon completion, the script will generate a PNG image identical to Figure 3 as `fig3.png`. Additionally, it will output a table similar to the first five columns of Table 4. There might be slight variations due to the different underlying machine and the inherent stochastic nature of fuzzing.

Experiment 2 (E2) - Claim (C2): Library exploration capability: [2 humans minutes + 10 compute-hours]

This experiment evaluates the capability of LIBERATOR to explore a library by computing the code and API function coverage across a 24h generation session. It reuses the results from experiment E1.

[Preparation] Make sure to have run the preparation for experiment E1.

[Execution] In a shell, run the following command:

```
# Compute the cumulative coverage across the
↪ drivers generated as part of E1.
# Expected output: Two plots for each library:
↪ edge_coverage.png and
↪ api_function_coverage.png.
./fig4.sh
./fig5.sh
```

[Results] Upon completion, the script will generate figures similar to the ones in Figure 4 & 5. There might be slight variations due to the different underlying machine and the inherent stochastic nature of fuzzing.

Experiment 3 (E3) - Claim (C3): Ablation study: [2 humans minutes + 24 compute-hours] This experiment evaluates the contribution of each module of LIBERATOR by conducting an ablation study.

[Preparation] Make sure to have run the preparation of experiment E1 first to have the numbers for *full* LIBERATOR in Table 9.

[Execution] In a shell, run the following command:

```
# Run libErator without field bias.
# Expected output: Tables similar to Table 9
↪ and 10 in the paper will be printed.
./tab9.sh
./tab10.sh
```

[Results] Upon completion, the script will print tables similar to Table 9 & 10. There might be slight variations due to the different underlying machine and the inherent stochastic nature of fuzzing.