USENIX Security '26 Artifact Appendix: CULPA: Universal Detection of Memory-Safety Bugs in Unsafe Rust Through the Lens of Safety Requirements

Hung-Mao Chen, Bo Lu, Xu He, Xiaokuan Zhang, Kun Sun

A Artifact Appendix

A.1 Abstract

In this paper, we present CULPA, a universal detector for memory-safety bugs in Rust programs. The key insight of CULPA is to detect the root cause of such bugs: the violation of safety requirements in unsafe Rust contexts. CULPA covers existing bug classes addressed by four static analyzers and identifies additional memory-safety vulnerabilities beyond their scope. We evaluate CULPA on the top 1,000 Rust packages. CULPA uncovers 55 previously unknown (zero-day) memory-safety bugs, 29 of which have been confirmed by developers. Most of these vulnerabilities are missed by four state-of-the-art Rust static analyzers and one LLM-based tool. The data artifacts of this paper include source code, detected bugs, and assigned RustSec/CVE IDs.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

We take ethics seriously in this project. All Rust repositories we tested in the paper are publicly accessible on Github. During evaluations of our work, CULPA identified several previously unknown vulnerabilities in widely used software. In each case, we followed a responsible disclosure policy, and reported our discovered vulnerabilities to the developers. We also submitted our findings to the CVE program and the RustSec Advisory Database. We did not disclose those issues to anyone else. All the examples mentioned in the paper are the issues that have been acknowledged and fixed. The RustSec IDs are: RUSTSEC-2025-0137, RUSTSEC-2025-0140, RUSTSEC-2025-0143, RUSTSEC-2026-0001, RUSTSEC-2026-0008; The CVE ID is CVE-2026-0810.

A.2.2 How to access

The artifact can be downloaded from Zenodo: https://zenodo.org/records/20740755

A.2.3 Hardware dependencies

We deploy CULPA on a server with 48-core Intel Xeon CPU ES-2630 and 256 GB memory. A regular PC can support our tool if the user doesn't run the detector with multiple threads.

A.2.4 Software dependencies

All the software dependencies can be installed automatically in the docker container. If setting up in local environment, users are required to install `rustc` with specific version `1.87.0-nightly` and the components including `rust-src`, `rustc-dev`, `llvm-tools-preview`, `miri`. The instructions of installing software dependencies are provided in the file `README.md`.

A.2.5 Benchmarks

Following datasets are included in our artifact:

First, 1000 Rust packages (`top_1000_crates.txt`), which were downloaded before December 1, 2025.

Second, reports that include all CULPA detection results on 1000 packages (`culpa_findings.csv`): The dataset includes all true positives, false positives, FFI (which is out of our scope and should not be count into true/false positives). Based on same results, we also compare whether four state-of-the-art static analyzers and one LLM-based tool can identify the same bugs (`zero_day_comparison_add_s4u`): "yes" means they can detect the same bugs at same location, "no" means they detect the same bugs.

A.3 Set-up

A.3.1 Installation

We provide docker to automatically install and compile all the dependencies and CULPA into the container.

A.3.2 Basic Test

The basic test can be started by setting up the docker we provide in artifacts. The instructions for running a simple functionality test are provided in the section: *Sample usage* of the file `README.md`.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): CULPA can uncover 55 new bugs. This is proven by the experiment E1.

(C2): CULPA can detect more new bugs than baseline tools (4 static analyzers and 1 LLM-based tool). This is proven by the experiment E2.

A.4.2 Experiments

(E1): 10 human-minutes + 10 compute-minutes + 50GB disk

How to: First, verify **Preparation** has been done. Second, follow **Execution** to run the docker script. Third, Compare the **Results** to verify whether it matches the data of `culpa_findings.csv`.

Preparation: Build the docker image with `docker build -t culpa:eval .`.

Execution: Run the docker script with `docker run -it culpa:eval`.

Results: The standard output will show all true positives that should be found and the corresponding number in each crate. It should show "TP (new bug) rows reproduced: 55 / 55" at the end.

(E2): 60 human-minutes + 2 compute-hour + 50GB disk

How to: The source code of the packages on existing RUSTSEC bugs have been included in the directory of `rustsec`. First, follow **Preparation** to install the tools as benchmarks,

Preparation: Create the shell with CULPA by `docker run -it culpa:eval bash`. In the shell, install the five baseline tools: TYPEPULSE, SafeDrop, MirChecker, RUDRA, Safe4U.

Execution: Following the instructions of baseline tools to check whether they can detect the same bugs.

Results: The detection results should match the ones in `zero_day_comparison_add_s4u.csv`.

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://zenodo.org/records/20740755.