USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

TypePulse: Detecting Type Confusion Bugs in Rust Programs

Hung-Mao Chen and Xu He, *George Mason University;*
Shu Wang, *George Mason University and Palo Alto Networks, Inc.;*
Xiaokuan Zhang and Kun Sun, *George Mason University*

https://www.usenix.org/conference/usenixsecurity25/presentation/chen-hung-mao

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix
TYPEPULSE: Detecting Type Confusion Bugs in Rust Programs

Hung-Mao Chen, Xu He, Shu Wang, Xiaokuan Zhang, Kun Sun

A Artifact Appendix

A.1 Abstract

In this paper, we develop a static analysis tool called TYPEPULSE to detect three main categories of type confusion bugs in Rust including misalignment, inconsistent layout, and mismatched scope. We run TYPEPULSE on the top 3,000 Rust packages and uncover 71 new type confusion bugs, exceeding the total number of type confusion bugs reported in RUSTSEC over the past five years. We have received 32 confirmations from developers, along with one CVE ID and six RUSTSEC IDs. The artifact includes the source code of TYPEPULSE, dataset with GitHub issue links, and the detection results provided in the paper.

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, TYPEPULSE identified several previously unknown type confusion 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 issued are: RUSTSEC-2023-0046,RUSTSEC-2023-0047, RUSTSEC-2024-0408, RUSTSEC-2024-0424, RUSTSEC-2024-0426, RUSTSEC-2024-0431; The CVE ID is currently in the reserved status at the time of writing and will be released later.

A.2.2 How to access

The artifact can be downloaded from Zenodo: https://zenodo.org/records/14862485.

A.2.3 Hardware dependencies

We deploy TYPEPULSE 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 `1.72.0-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

Three datasets are included in our artifact:

First, 3000 Rust packages (`top3kscanned.txt`), which were downloaded before September 1, 2024.

Second, reports that include all TYPEPULSE detection results on 3000 packages (`report3k.csv`). The dataset includes all true positives, false positives, the corresponding function names, and the links of reported GitHub issues if true positives.

Third, existing RUSTSEC bugs for our evaluation (The directory of `rustsec`). The code of packages are provided in the directory by cases on RUSTSEC. In the subdirectories, we provide `README.md` to show all detection results. TYPEPULSE can also used to scan the packages to reproduce the detection results.

A.3 Set-up

A.3.1 Installation

We provide two ways to run TYPEPULSE:

First, execute dockerfile to automatically install and compile all the dependencies and TYPEPULSE into the container.

Second, to manually set up TYPEPULSE in the local environment, the user can also follow instructions in sections: *Requirements*, *First time to set up environment*, and *install with install.sh* in the file `README.md`.

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: *Usage Tutorial: Run TypePulse in your package* of the file `README.md`. The instructions will provide the steps to download the package `candle-core`, which is included in Section 5.5, and reproduce the detection results.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We claim that TYPEPULSE can uncover 71 new type confusion bugs in Table 3 (while Table 3 only showed 50 bugs due to disclosure policies). This is proven by the experiment E1.

(C2): We claim that TYPEPULSE can achieve higher precision by enabling interprocedural analysis, as shown in Table 4. This is proven by the experiment E2.

(C3): We claim that TYPEPULSE can detect more type confusion bugs than Clippy and Rudra on existing RUSTSEC bugs, as shown in Table 5. This is proven by the experiment E3.

A.4.2 Experiments

Before running the following experiments, we need to ensure the required packages are downloaded. To download the source code of packages, we can run `crates_downloader.py` and `unarchive.sh`. We have detailed instructions in README.md.

(E1): 30 human-minutes + 1 compute-hour + 50GB disk
How to: First, verify **Preparation** has been done. Second, follow **Execution** to run the python script. Third, Compare the **Results** saved in `report.txt` to verify whether it matches the data of `report3k.csv`.
Preparation: Make sure the packages are all downloaded to the path `/home/crates/sources/`.
Execution: Use python3 to run the script `fast_typepulse.py`.
Results: The true positive and false positive detection results with interproceudural analysis from the packages in Table 3 will be saved in the `/home/crates/sources/[package-name]/report.txt`.

(E2): 30 human-minutes + 1 compute-hour + 50GB disk
How to: First, verify **Preparation** has been done. Second, follow **Execution** to run the python script. Third, Compare the **Results** saved in `report.txt` to verify whether it matches the data of `report3k.csv`.
Preparation: Make sure the packages are all downloaded to the path `/home/crates/sources/`.
Execution: Use python3 to run the script `fast_typepulse.py`.
Results: The true positive and false positive detection results without interproceudural analysis from the packages in Table 3 will be saved in the `/home/crates/sources/[package-name]/zreport.txt`.

(E3): 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: Install Clippy (version 0.1.72) and Rudra. Second, follow *Execution* to run TYPEPULSE and scan packages. Third, Compare the *Results* with corresponding `README.md` in `rustsec/rs-**`.
Execution: In each directory of `rustsec/rs-**`, run `cargo typepulse`.
Results: Detection results will be shown as standard output.

A.5 Notes on Reusability

A.6 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/14862485.