# When Threads Meet Interrupts: Effective Static Detection of Interrupt-Based Deadlocks in Linux

Chengfeng Ye, Yuandao Cai, and Charles Zhang,
*The Hong Kong University of Science and Technology*

https://www.usenix.org/conference/usenixsecurity24/presentation/ye

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

# USENIX Security ’24 Artifact Appendix: When Threads Meet Interrupts: Effective Static Detection of Interrupt-Based Deadlocks in Linux

Chengfeng Ye Yuandao Cai Charles Zhang
*The Hong Kong University of Science and Technology*
*{cyeaa, ycaibb, charlesz}@cse.ust.hk*

## A Artifact Appendix

### A.1 Abstract

This appendix describes the software artifact that implements algorithms and evaluation proposed in the ’When Threads Meet Interrupts: Effective Static Detection of Interrupt-Based Deadlocks in Linux.’

Specifically, the artifact includes the implementation of ARCHERFISH in the form of binary, LLVM Bitcode compiled from Linux v6.4 (benchmark), and several scripts to reproduce the experiment described in §7.1 and §7.2, corresponding to **RQ1** and **RQ2** of the paper. We also prepare a docker image to simplify the workflow.

### A.2 Description & Requirements

#### A.2.1 Security, privacy, and ethical concerns

The artifact includes information about detected bugs, but there is no need to address any ethical concerns because they have all been reported to the developers.

#### A.2.2 How to access

Artifact is available at https://doi.org/10.5281/zenodo.13117816. Docker image is also provided, based on Ubuntu 20.04, at **cyeaa/archerfish:1.0**. The Docker image includes all the necessary libraries and environments required to run the experiments.

#### A.2.3 Hardware dependencies

The experiments were run on a server with two 20-core Intel(R) Xeon @2.20GHz CPUs with 256GB of DRAM and 128GB of swap space. Having a similar hardware setup (which includes memory available to docker) is suggested to run the experiments.

Additionally, at least 20GB of disk space is required.

#### A.2.4 Software dependencies

Linux-based operating system (e.g., Ubuntu) with docker installed.

#### A.2.5 Benchmarks

LLVM Bitcode compiled from Linux v6.4 stable branch.

### A.3 Set-up

#### A.3.1 Installation

**To obtain the docker image :**

```
$ docker pull cyeaa/archerfish:1.0
```

**To create a docker container with the docker image :**

```
$ docker run -it cyeaa/archerfish:1.0
/bin/bash
```

**Enter the artifact folder :**

```
$ cd ARCHERFISH_ARTIFACT
```

This will create a docker container with the docker image, and you will get a bash terminal to run the experiments. Note that the ARCHERFISH implementation is provided in the form of binary, and all the benchmarks have been compiled into LLVM IR. Thus, no extra compilation effort is required.

#### A.3.2 Basic Test

If everything is correct, the evaluator should be inside the docker environment, and the current directory should be ’ARCHERFISH_ARTIFACT’ including a README.md file, several shell scripts, and several sub-folders for experiment purposes.

### A.4 Evaluation workflow

#### A.4.1 Major Claims

**(C1):** ARCHERFISH has been used to uncover 76 new interrupt-based deadlocks in the Linux kernel. This is proven by the experiment described in §7.1 of the paper.

**(C2):** Several critical design can effectively improve ARCHERFISH. This is proven by the ablation evaluations described in §7.2 of the paper.

#### A.4.2 Experiments

**(E1):** *[Interrupt-Based Deadlock Detection] [20 human-minutes + 30 compute-minutes]:*
**Description**: The experiment involves running ARCHERFISH on Linux kernel v6.4 LLVM Bitcode to detect interrupt-based deadlocks, corresponding to § 7.1 of the paper. The statistical data related to the bug detection would be printed out, and the bug reports would be generated inside corresponding folders.
**Workflow**:

- (30 compute-minutes) Execute the script:
  `$ ./1_run_archerfish.sh`
- Next, execute the script:
  `$ ./2_show_archerfish_statistics.sh`

**Result**: The script `./1_run_archerfish.sh` is to perform the bug detection. This step produces bug reports with the name "deadlocks.txt" inside folders corresponding to different Linux kernel subsystems. The reported bugs include those described in Table 5 of the paper appendix. The script `./2_show_archerfish_statistics.sh` is to print out some statistical data related to bug detection, corresponding to most data in Table 1 of the paper. Also, a summary of the detected bugs would be printed out. Furthermore, detailed information on the reported bugs (LKML links and Linux commits) can be found in the file `reported_bug_detail.xlsx` inside the artifact main folder.

**(E2):** *[LLVM-Assisted ISR Identification] [10 human-minutes + 30 compute-minutes]:*
**Description**: The experiment involves evaluating the accuracy of LLM-Assisted ISR specification and evaluating its effect on interrupt-based deadlock detection, corresponding to §7.2.1 of the paper. For ease of evaluation, we skip the GPT query process and directly provide the generated specification.
**Workflow**:

- Execute the script:
  `$ ./3_spec_accuracy.sh`
- (30 compute-minutes) Execute the script:
  `$ ./4_llm_comparision.sh`

**Result**: The script `./3_spec_accuracy.sh` would produce the accuracy of the specification corresponding to Table 2 of the paper. The script `./4_llm_comparision.sh` would turn off the LLM-assisted ISR specification and perform interrupt-based bug detection again, then print out the differential result corresponding to Figure 12 of the paper.

**(E3):** *[Effectiveness of Preemption Unit] [5 human-minute + 1 compute-hour]:*
**Description**: The experiment involves evaluating the effectiveness of the Preemption Unit in increasing the analysis performance, corresponding to §7.2.2 of the paper. To do so, we execute an ablation version of ARCHERFISH in which the Preemption Unit is disabled and compare the time and memory consumption.
**Workflow**:

- (60 compute-minutes) Execute the script:
  `$ ./5_preempt_unit_comparison.sh`

**Result**: The script would disable the Preemption Unit and perform interrupt-based bug detection again, then printed out the corresponding differential data described in the section **Ablation Study 2: Preemption Unit** of §7.2.2 in the paper.

#### A.4.3 Notes

- The three experiments should be performed in the order they are described in this documentation.
- If the experiment is interrupted due to unexpected reasons, you can use the script `clear.sh` to clean up all the results and then perform the experiment again.
- Slight variability is to be expected in the results since some experiments are sensitive to execution environments, such as memory and CPU, and we recommend running all the experiments in the same environment for consistent results.

### 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/usenixsec2024/.