A Artifact Appendix

A.1 Abstract

Our artifact includes three major parts: hardware-free device driver fuzzer, modified PANDA/QEMU with full-system concolic tracing support and concolic code exploration scripts. The code require a 64-bit x86 system with clean Ubuntu 20.04 install.

A.2 Artifact check-list (meta-information)

- **Compilation: It's best to use the default Ubuntu 20.04 compilers**
- **Run-time environment: Linux (preferably Ubuntu 20.04)**
- **Hardware: 64-bit x86 computer**
- **How much disk space required (approximately)?: 40G**
- **How much time is needed to prepare workflow (approximately)?: 30 min**
- **How much time is needed to complete experiments (approximately)?: several hours to days depending on number of drivers and duration of fuzzing session you want to run**
- **Publicly available (explicitly provide evolving version reference)?: https://github.com/messlabnyu/DrifuzzProject/tree/d0b9edfa364c2f9fe45d4b63c0ad9f62dca0bfc9**

A.3 Description

A.3.1 How to access

Clone source from https://github.com/messlabnyu/DrifuzzProject/. Or you can obtain our docker image from docker hub https://hub.docker.com/repository/docker/buszk/drifuzz-docker.

A.3.2 Hardware dependencies

64-bit x86 machine.

A.3.3 Software dependencies

Ubuntu 20.04 for building from source. Any Linux distro should be fine for running the Docker image.

A.4 Installation

```
git clone https://github.com/messlabnyu/↙
    DrifuzzProject.git
cd DrifuzzProject \&\& ./build.sh 2>\&1 |tee ↙
    build.log
```

A.5 Experiment workflow

In top-down perspective, our work invokes a golden seed search script to generate quality initial seeds. Then, we can run the fuzzing tool with the generated seed to increase coverage. Because our initial seed has solved many roadblocks, using it tends to find better coverage tank starting with random seed (e.g. Agamotto's approach). To find the golden seeds, we leverage concolic execution and forced execution to find and tackle roadblocks incrementally to increase coverage.

A.6 Evaluation and expected results

Evaluation should show that Drifuzz is able to perform concolic tracing in device driver execution and our golden seed search algorithm is able to provide a good initial seed resulting more code coverage.

A.6.1 Prerequisite

After installation, please check if the following files are created correctly. If any of the file was not created properly, please check the build log and script to triage the problem.

```
cd ~/DrifuzzRepo/Drifuzz
ls image/buster.img
ls panda-build/x86_64-softmmu/panda-system-↙
    x86_64
ls panda-build/x86_64-softmmu/panda/plugins/↙
    panda_taint2.so
ls linux-module-build/vmlinux
```

A.6.2 Concolic Tracing

Note: USB targets are supported with "–usb" flag in ./snapshot_helper.py and ./concolic.py.

```
cd ~/DrifuzzRepo/drifuzz-concolic

# Create a driver specific snapshot
./snapshot_helper.py ath9k
ls work/ath9k/ath9k.qcow2 # should exists

# Run concolic script with random input
head -c 4096 /dev/urandom >rand
./concolic.py ath9k rand
cat work/ath9k/drifuzz_path_constraints # path ↙
    constraints
cat work/ath9k/drifuzz_index # accessed MMIO/DMA
ls work/ath9k/out # generated inputs with ↙
    flipped branch

# Understand the concolic result
head work/ath9k/drifuzz_path_constraints
# Get the program counter of the first symbolic ↙
    branch
head work/ath9k/drifuzz_path_constraints |grep ↙
    PC | awk '{print␣$6}'
# Use addr2line script to get stack trace
./addr2line.py ath9k [program counter]
```

A.6.3 Golden seed

Note: you may encounter a bug that consumes all available disk space. In that case, run du tool to find and remove the files. If you

use provided docker image, deleting the container and retrying a new random seed might solve the problem.

Note: if you run into an AssertionError for “Cannot find a feasible path for given model" for the first ./concolic.py run, it seems that PANDA’s concolic tracing is not work. Please double check that you have a snapshot in good standing and are able to run concolic tracing. If that fails, changing a seed is reported to work in the situation.

Note: USB targets are supported with “–usb” flag in ./search_greedy.py.

```
cd ~/DrifuzzRepo/drifuz-concolic
# Run the golden seed script (takes a hour or so)
./search_greedy.py ath9k rand 2>&1|tee search_ath9k.log
ls work/ath9k/out/0 # generated seed
```

Fields below can be derived from generated log to compare with Table 2 from paper.

- $\#blocking\ branches = \#iterations - 1$
- $\#symbolic\ branches = sizeof(last\ branches\ list)$

A.6.4 Fuzzing

```
cd ~/DrifuzzRepo/Drifuzz

# Fuzz ath9k with random seed on 4 cores
fuzzer/drifuzz.py -D -p 4 seed/seed-random work/ath9k ath9k
# Ctrl^C once to stop

# Reproduce a generated input
scripts/reproduce.sh ath9k work/ath9k/ work/ath9k/corpus/payload_1

# Process stacktrace when you see a crash
scripts/decode_stacktrace.sh crash.log
```

We also provide some helpful scripts to combine our golden seed and concolic support with our fuzzer. Note: You need to run the golden seed generation script before running some of the following scripts.

```
cd ~/DrifuzzRepo/Drifuzz

# Fuzzing random input without concolic support
scripts/run_random.sh ath9k
# Fuzzing golden seed with concolic support
scripts/run_conc_model.sh ath9k
```

A.6.5 Coverage comparison

Get the coverage metric from the fuzzing sessions. The result should show that the second session should have better coverage than the first. The detailed number will differ because of time of fuzzing period and non-determinism in fuzzing.

```
tail -n1 work/work-ath9k-random/evaluation/data.csv |awk -F';' '{print $16}'
tail -n1 work/work-ath9k-conc-model/evaluation/data.csv |awk -F';' '{print $16}'
```

A.7 Notes

Details of how to run each part of Drifuzz are shown in GitHub page.

A.8 Version

Based on the LaTeX template for Artifact Evaluation V20220119.