USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Robust, Efficient, and Widely Available Greybox Fuzzing for COTS Binaries with System Call Pattern Feedback

Jifan Xiao, *Key Laboratory of High Confidence Software Technologies, Peking University;* Peng Jiang, *Southeast University;* Zixi Zhao, Ruizhe Huang, Junlin Liu, and Ding Li, *Key Laboratory of High Confidence Software Technologies, Peking University*

https://www.usenix.org/conference/usenixsecurity25/presentation/xiao-jifan

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: Robust, Efficient, and Widely Available Greybox Fuzzing for COTS Binaries with System Call Pattern Feedback

Jifan Xiao[1], Peng Jiang[2], Zixi Zhao[1], Ruizhe Huang[1], Junlin Liu[1], and Ding Li[1]

[1] *Key Laboratory of High Confidence Software Technologies, Peking University*
[2] *Southeast University*

A Artifact Appendix

A.1 Abstract

This document provides instructions to execute the artifacts. This is a prelinimary version of SPFuzz and SPFuzz++. The artifacts are provided as a Virtual Machine (VM) image. The VM is configured with all the dependencies and the source codes of SPFuzz and SPFuzz++, with necessary data for benchmarks. Please exactly follow the instructions in this document to execute SPFuzz and SPFuzz++.

A.2 Description & Requirements

To use the artifacts, you first need to create a VM with the provided harddisk image. The image is in vdi format and can be loaded using Oracle VirtualBox. The VM is configured with 2 cores and 4GB of RAM. The VM is running Ubuntu 18.04 LTS. (Since the current version of NoDrop module does not support docker environments, we choose to share our artifacts in this way.)

A.2.1 Security, privacy, and ethical concerns

We have reported all the vulnerabilities discovered by SPFuzz to the authors/owners of the vulnerable software.

A.2.2 How to access

We publish the artifacts through the following link: https://doi.org/10.5281/zenodo.15209966.

Once you have configured and created the VM successfully, you can login to the VM using the following credentials:

Username: usenix
Password: security

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

For this version, the software environment should be identical with the provided VM image, especially the OS kernel version.

A.2.5 Benchmarks

Benchmarks are contained in the VM image disk.

A.3 Set-up

A.3.1 Installation

- Step 1: Configure Core Pattern For Fuzzers

```
sudo -s
echo core >/proc/sys/kernel/core_pattern
su usenix
```

- Step 2: Load NoDrop Kernel Module

```
cd ~/SPFuzz/NoDrop/build
bash remake.sh
make
make load
```

- Step 3: Make SPFuzz & SPFuzz++

  Take SPFuzz++ as an example (SPFuzz is similar) :

```
cd ~/SPFuzz/SPFuzz++
make clean
make
```

  Then, an executable named *afl-fuzz* will be generated in the current directory.

- Step 4: Prepare Benchmarks

  All benchmarks are held in the `~/SPFuzz/benchmarks` directory. Each benchmark is a directory with a binary file, an *input.zip* file containing the inputs, and another

.zip file containing the source codes if this benchmark is open-source.

To prepare the benchmark, you need to unzip the seeds from the *inputs.zip* file, and copy the binary to a new binary named *toTest* in the same directory. For example, to prepare the *perlbench_r* benchmark:

```
cd ~/SPFuzz/benchmarks/perlbench_r
unzip inputs.zip
cp perlbench_r toTest
```

A.3.2 Basic Test

Take SPFuzz++ as an example (SPFuzz is similar). First, make sure the "NoDrop" kernel module is loaded. Then, execute the following command:

```
cd ~/SPFuzz/SPFuzz++
AFL_SKIP_BIN_CHECK=1 ./afl-fuzz \
-t 1000 -m none \
-i ~/SPFuzz/benchmarks/NAME/inputs \
-o ~/SPFuzz/benchmarks/NAME/output \
-- ~/SPFuzz/benchmarks/NAME/toTest @@
```

Note to replace the NAME with the benchmark you want to test. An example using *perlbench_r* is stored in the `~/example.sh` file.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *SPFuzz and SPFuzz++ are able to fuzz all the benchmark binaries without source codes, and the efficiency is comparable with traditional methods.*

A.4.2 Experiments

(E1): *[Optional Name] [1 human-hour + 24 X 29 compute-hour + 3GB X 29 disk]:*
How to: First, prepare the tools and benchmarks according to Section A.3, then start the fuzzing campaign and wait for 24 hours. This procedure applies to all the 29 benchmarks, both open-source and close-source.
Results: Follow the instructions in step-6 of the README.md file contained in the repo to calculate coverage. SPFuzz and SPFuzz++ should be able to run all the fuzzing tasks. SPFuzz++ should obtain comparable speeds and coverage data with traditional fuzzers.

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/usenixsec2025/.