MOAT: Towards Safe BPF Kernel Extension

Hongyi Lu, *Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, and Hong Kong University of Science and Technology;* Shuai Wang, *Hong Kong University of Science and Technology;* Yechang Wu and Wanning He, *Southern University of Science and Technology;* Fengwei Zhang, *Southern University of Science and Technology and Research Institute of Trustworthy Autonomous Systems*

https://www.usenix.org/conference/usenixsecurity24/presentation/lu-hongyi

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

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

USENIX Security '24 Artifact Appendix: MOAT: Towards Safe BPF Kernel Extension

Hongyi Lu[1,2,3], Shuai Wang[3,†], Yechang Wu[2], Wanning He[2], Fengwei Zhang[2,1,†]

[1]*Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology*
[2]*Department of Computer Science and Engineering, Southern University of Science and Technology*
[3]*Department of Computer Science and Engineering, Hong Kong University of Science and Technology*

A Artifact Appendix

A.1 Abstract

MOAT's artifact contains the source code and the benchmarks used in the evaluation part. We outline the steps to retrieve the artifact and reproduce the experiments in the paper.

A.2 Description & Requirements

The artifact contains the following components.

1. MOAT's Linux (ver. 6.1.38) source
2. A set BPF programs with various functionality
3. A set of user-space programs used to stress the MOAT's kernel and the BPF programs

A.2.1 Security, privacy, and ethical concerns

We make moderate modification to the original kernel that might not be compatible with certain user-space programs and could potentially cause data loss. Therefore, we suggest evaluating MOAT in a clean-state machine or a VM[1].

A.2.2 How to access

The artifact is in https://github.com/jwnhy/MOAT-Open/tree/b1cfea3114ddf237c2100bc1bdc53f4030f4780b.

A.2.3 Hardware dependencies

We evaluate MOAT on a target machine with an Intel 8505 CPU, 8 GiB of RAM and an I226 network controller. A host machine is deployed to send network packets to test the target machine. The host machine has an Intel 12700K CPU, 32 GiB of RAM and a RTL8125B network controller.

A.2.4 Software dependencies

We use no proprietary software in our evaluation.

A.2.5 Benchmarks

We use `iperf3`, `nginx`, `wrk`, `sysfilter`, `Phoenix Test Suite` and `UnixBench` as user-space workloads. We use modified `libbpf` (included in our repository) as BPF workloads.

A.3 Set-up

A.3.1 Installation

Prepare `rootfs` MOAT must be installed on a clean-state Linux `rootfs`. Depending on the platform (virtual/physical machine), there are the two ways to prepare MOAT's `rootfs`.

If you are evaluating MOAT with physical machine (e.g., Intel 8505), you can install arbitrary Linux distro and replace its kernel with MOAT's. As a reference, we use Gentoo when conducting the experiments in the paper.

If you are evaluating MOAT with virtual machine, you can use `debootstrap` to create a Debian `rootfs` from scratch.

Install MOAT Once you have a usable Linux, you can install MOAT's kernel. As a reference, one can run `sudo make -j$(nproc) && sudo make install`.

A.3.2 Basic Test

One can load `dropfilter.bpf.c` using the `libbpf` as a basic test for MOAT. The expected result is that all the packets in the target network interfaces are dropped. For other tests, one can refer to the `README` in our repository.

A.4 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/.

[†] Shuai Wang and Fengwei Zhang are the corresponding authors.

[1] VM is only suitable for functional evaluation and does not reflect the actual performance of MOAT.