USENIX Security '25 Artifact Appendix: Serverless Functions Made Confidential and Efficient with Split Containers

Jiacheng Shi, Jinyu Gu, Yubin Xia, Haibo Chen
Institute of Parallel and Distributed Systems (IPADS), SEIEE, Shanghai Jiao Tong University
Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, China

A Artifact Appendix

A.1 Abstract

The artifact includes the following components: (1) the source code of CoFunc system, including the CVM OS code (`cvm_os`), the shadow container code (`shadow_container`) and the patches for the host Linux/QEMU (`patches`); (2) the serverless functions utilized for the evaluation (`testcases/testcases`); (3) the scripts for conducting the experiments (`scripts` and `testcases/tools`). Users of this artifact can evaluate the performance of serverless functions under different container runtimes, including split containers (CoFunc), CVM-based Kata Containers (Kata-CVM), and native lean containers (Native).

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact poses no harm to machine security or data privacy.

A.2.2 How to access

The artifact is available at Github (https://github.com/shijc-sjtu/cofunc-artifact) and Figshare (https://doi.org/10.6084/m9.figshare.28234346.v4).

A.2.3 Hardware dependencies

The artifact requires AMD CPUs with SEV-SNP support. It has been tested on an EPYC-7T83 machine. At least 96 CPU cores and 180GB of memory are required.

A.2.4 Software dependencies

The artifact works on an SEV-SNP version of the Linux kernel (https://github.com/AMDESE/linux.git, branch `svsm-preview-hv-v2`), along with the modifications in `patches/linux.patch`. The following dependencies are required for building the artifact and running the experiments: Docker, screen, Python 3 (with matplotlib, numpy, boto3, pandas, CouchDB) and gcc.

A.2.5 Benchmarks

The Dockerfiles for building the serverless functions are available at `testcases/testcases`.

A.3 Set-up

A.3.1 Installation

For reviewers using the test server, the artifact is already installed, and no further steps are required. Other users need to build the artifact components and set up the host environment with the steps described in the README.

A.3.2 Basic Test

Run the script `scripts/run_simple.sh`, which executes a serverless function using CoFunc. The script will output the function's running time.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): For the 28 evaluated functions, CoFunc demonstrates significant performance improvements (up to 60×) compared with Kata-CVM, while incurring <14% performance overhead compared with Native. This is proven by the experiment (E1) described in Section 7.1 whose results are illustrated in Figure 11.

(C2): CoFunc outperforms Kata-CVM on FINRA application by 31× when 200 auditing functions start concurrently. This is proven by the experiment (E2) described in Section 7.4 whose results are reported in Section 7.4.

A.4.2 Experiments

(E1): [1.5 compute-hours]: Evaluate the end-to-end latencies (handling a single request) of the functions using CoFunc, Kata-CVM, and Native.
Preparation: None.
Execution: Run the script `scripts/run_fig11.sh`. This script executes the functions with different runtimes and outputs the latencies to the `log` directory.

Results: The script will generate a table at `plots/fig11.txt` that contains the function latencies and the overhead/optimization of CoFunc compared with Native/Kata-CVM. Additionally, the script will generate a figure at `plots/fig11.pdf`, which can be compared with Figure 11.

(E2): [10 compute-minutes]: Evaluate the end-to-end latency of FINRA application with 200 concurrent auditing functions using CoFunc and Kata-CVM.
Preparation: None.
Execution: Run the script `scripts/run_finra.sh`. This script executes FINRA application with different runtimes and outputs the latencies to the `log` directory.
Results: The end-to-end application latencies and the optimization of CoFunc compared with Kata-CVM can be found in `plots/finra.txt`.

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/.