1 Artifact Appendix

1.1 Abstract

The evaluated artifact includes the prototype implementation of ELASTICLAVE that we have presented in the paper. We have publicly released it on GitHub.

Running the full set of experiments take significant long time and relies on the AWS EC2 platform. Therefore, we also provide the option to run them with QEMU, which, though inaccurate for performance evaluation, serves well as a quick way to test the system functionally. This option only requires an x86-64 Linux system with Docker installed.

1.2 Artifact check-list (meta-information)

- **Program:** IOZone (included)
- **Compilation:** GCC cross compiler targeting RISC-V 64 (included)
- **Run-time environment:** Linux (Ubuntu 20.04 LTS recommended) with Docker
- **Hardware:** x86-64
- **Metrics:** Execution time
- **Output:** Performance numbers (execution time) in console log files. Results are the differences in the performance among different solutions
- **Experiments:** Run the benchmarks with the scripts we have prepared. Compare the performance numbers produced with different solutions and the difference should be on the same order of magnitude as the results reported in the paper
- **How much disk space required (approximately)?:** 20 GB
- **How much time is needed to prepare workflow (approximately)?:** 1 hour
- **How much time is needed to complete experiments (approximately)?:** 30 hours
- **Publicly available?:** Yes

1.3 Description

1.3.1 How to access

This artifact is publicly released on GitHub[1]. The commit hash of the evaluated version is 29aab39.

1.3.2 Hardware dependencies

It is necessary to run the artifact on an AWS EC2 and use FireSim to obtain accurate performance data. For evaluation of the functionality, any modern x86-64 Linux platform should suffice. The required disk space is approximately 20 GB.

[1]The main repository (which references more repositories as submodules): https://github.com/jasonyu1996/elasticlave

1.3.3 Software dependencies

This artifact is expected to run on any GNU/Linux distribution with Docker installed.

1.4 Installation

We provide two options to run the artifact.

The first option requires running FireSim on an AWS EC2 F1 instance, and hence can incur significant monetary cost. In addition, it takes much longer to run the experiments than the second option. Since this is the option that provides cycle-accurate simulation, it is necessary if the goal is to evaluate the performance of the system and reproduce the experimental results.

The second option is to emulate the system on QEMU. It does not incur extra cost and consumes less execution time. This option is unable to produce accurate performance data and is only suitable for testing the functionality.

We have automated most part of the installation process. Below are the installation instructions for both options.

FireSim. On your AWS EC2 instance with FireSim set up, clone the repository and checkout to the evaluated snapshot:

```
git clone https://github.com/jasonyu1996/elasticlave.git
cd elasticlave
git checkout 29aab39
```

Pull the submodules recursively:

```
git submodule update --init --recursive
```

Build:

```
./docker.sh
./docker-run.sh ./make-firesim.sh
./docker-run.sh ./make-firesim.sh image
```

Launch the simulated system:

```
./run-firesim.sh
```

It might help to understand what happens under the hood in the script executed above.

The script first launches FireSim:

```
firesim launchrunfarm && firesim infrasetup && \
    firesim runworkload
```

After this is finished, it logs into the newly launched F1 instance:

```
ssh RUNFARM_IP
```

, where `RUNFARM_IP` is the IP address of the F1 instance as reported in `firesim runworkload`.

The script then connects to the terminal of the simulated system:

```
screen -r fsim0
```

When the simulation ends, the script terminates the F1 instance: Terminate the F1 instance:

```
firesim terminaterunfarm
```

Below are instructions for use inside the connected terminal of the simulated system.

The login is `root` and the password is `sifive`.

To run the benchmarks, execute the following inside the shell of the prototype system:

```
insmod keystone-driver.ko
./tests.ke
```

There are also individual benchmarks that are not included in `tests.ke`. To run them, execute the scripts in the individual folders.

After the benchmark execution completes, you can end the simulation through:

```
poweroff -f
```

A log of the data on the terminal can be found inside `FIRESIM_FOLDER/deploy/results-workload`.

QEMU. Install Docker following the instructions on the official website.

Clone the repository and run the build scripts:

```
git clone https://github.com/jasonyu1996/elasticlave.git
cd elasticlave
git checkout 29aab39
git submodule update --init --recursive
./docker.sh
```

Run the artifact:

```
./docker-run.sh ./run.sh
```

The login is `root` and the password is `sifive`. The password is `sifive`.

To run the benchmarks, execute the following inside the shell of the prototype system:

```
insmod keystone-driver.ko
./tests.ke
```

There are also individual benchmarks that are not included in `tests.ke`. To run them, execute the shell scripts (`*.sh`) in the individual folders.

1.5 Experiment workflow

As described in Section 6, The experiments involve a range of benchmarks which are run on our prototype ELASTICLAVE implementation. The benchmarks involve data sharing across enclave boundaries, and the total running time with the ELASTICLAVE model is compared against that with the traditional spatial isolation model, as well as the running time when they are run in a native Linux environment without the protection of a TEE.

The prototype system runs an unmodified Linux kernel (with a driver for enclave management). Each benchmark includes both enclaves and untrusted code which runs as the host process and launches the enclaves. For using the spatial isolation model, the untrusted code is also responsible for marshalling messages.

1.6 Evaluation and expected results

The key claims made in this paper include:

1. Compared to the spatial ShMem model, our ELASTICLAVE implementation achieves 1–2 orders of magnitude better performance for data sharing. The overhead of ELASTICLAVE is about 10% compared with native execution without a TEE;
2. ELASTICLAVE incurs modest TCB and hardware complexity impact.

This artifact can be used to verify the following key results that support the above claims:

1. The performance comparison among ELASTICLAVE, the spatial ShMem model, and native execution for data sharing on synthetic benchmarks and IOZone (corresponding to Figures 6, 7, 8, 10, and 11). This supports Claim 1 above.
2. The TCB increase of ELASTICLAVE over Keystone (corresponding to Table 4). This supports Claim 2 above.

1.6.1 Performance

To obtain accurate performance numbers, it is necessary to run the benchmarks using FireSim. See Section A.4 for details. The results obtained from this artifact are expected to reflect the same patterns as in Figures 6, 7, 8, 10, and 11 as well as the associated descriptions in Section 6.1.

IOZone. Set `TESTS=iozone` in `tests/tests/mkconfig.mk`, rebuild the benchmarks with `./docker-run.sh ./make-firesim.sh image`. and execute `./tests.ke` in simulation. This runs IOZone with ELASTICLAVE. To run it with the baseline spatial ShMem model or a native Linux setting, set `NATIVE_TESTS` or `BASELINE_TESTS` to `iozone` instead.

Thread synchronization. Set `EXTRA_TESTS` to `lock` (spinlock with ELASTICLAVE), `lock-futex` (futex with ELASTICLAVE, `lock-spatial` (spinlock with the spatial ShMem model), `lock-native` (futex without TEE) and rebuild the benchmarks. Then execute `./tests.ke <thread-count> <work-amount>` in simulation. To get the numbers reported in the paper, supply 2 as `thread-count` and vary `work-amount` from 12800 to 3276800.

Data sharing patterns. The names of the corresponding synthetic benchmarks start with `icall-`, followed by the names of the patterns (`consumer`, `server`, and `proxy-3`). Name endings indicate whether the benchmarks are run with the spatial ShMem baseline (`spatial`), ELASTICLAVE without exclusivity support (`ne`), or full ELASTICLAVE (otherwise). To run the benchmarks, open the file `tests/tests/mkconfig.mk`, add the benchmark names in the line that starts with `EXTRA_PACKS`, and rebuild the benchmarks. The available benchmark names can be viewed in `tests/tests`.

1.6.2 TCB Increase.

To measure the TCB increase over Keystone, download the revision of the original security monitor and enclave runtime from Keystone[2]. Use `diff -x '.*' -Nwr <old-dir> <new-dir> | diffstat` to compare the directories `riscv-pk` and `sdk/rts/eyrie` with them and pipe the results to. The sums of insertion and modification numbers are below the numbers reported in Table 4.

1.7 Experiment customization

You can adjust the benchmarks to be included in each run of the artifact. To achieve this, edit the file `tests/tests/mkconfig.mk`, and add the names of the benchmarks you want to run.

[2] https://github.com/keystone-enclave/riscv-pk/tree/5b3d71 and https://github.com/keystone-enclave/keystone-runtime/tree/87351c