Accelerating JavaScript Static Analysis via Dynamic Shortcuts (Artifact Evaluation)

Joonyoung Park*
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
gmb55@kaist.ac.kr

Jihyeok Park*
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
jhpark0223@kaist.ac.kr

Dongjun Youn
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
f52985@kaist.ac.kr

Sukyoung Ryu
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
sryu.cs@kaist.ac.kr

ABSTRACT

We present *dynamic shortcuts*, a new technique to flexibly switch between abstract and concrete execution during JavaScript static analysis in a sound way. SAFE_{DS} is the actual instance of dynamic shortcuts (DS) based on Jalangi2 in order to accelerate the static analyzer SAFE. It can significantly improve the analysis performance and precision by using highly-optimized commercial JavaScript engines (V8 in Node.js in our setting) and lessen the modeling efforts for opaque code.

We apply our artifact for the Reusable badge and Artifacts Available badge. Our artifact provides the reproducible experimental environment, the full results of the experiments presented in the paper, and the commands of SAFE_{DS} to analyze a new input program. A user can reproduce the experiments presented in the paper that is the comparison of analysis performances of SAFE and SAFE_{DS} on Lodash4 tests. There are script files to juxtapose experimental results for each RQs with numbers in the paper. The `README` file on the root directory describes the above in detail.

This package is forked from SAFE and imports Jalangi2 as a git submodule. The license of this package is under the BSD license. We added the option "ds" to the original SAFE to trigger dynamic shortcuts. When the option is turned on, SAFE_{DS} communicates with the Node.js server in the `dynamic-shortcut` directory and the server runs Jalangi2 for dynamic analysis of functions in the target program on the concrete engine. The requirements of SAFE_{DS} are inherited from SAFE and Jalangi2 and specified in the `REQUIREMENTS` file. The `INSTALL` file will guide to initialize the submodule, SAFE, and Jalangi2.

KEYWORDS

JavaScript, static analysis, dynamic analysis, dynamic shortcut, sealed execution

1 ARTIFACT DESCRIPTION

SAFE_{DS} is an instance of **D**ynamic **S**hortcuts on JavaScript static analyzer **SAFE**, which is a new technique to flexibly switch between abstract and concrete execution during JavaScript static analysis in a sound way. This artifact extends SAFE for abstract execution and Jalangi2 for concrete execution. Jalangi2 is forked and its extension is imported as a git submodule.

*Both authors contributed equally to the paper.

We added the option "ds" to the original SAFE to trigger dynamic shortcuts. When the option is turned on, SAFE_{DS} communicates with the Node.js server in the dynamic-shortcut directory and the server runs Jalangi2 for dynamic analysis of functions in the target program on the concrete engine.

2 GETTING STARTED GUIDE

The artifact is open-source can be obtained by cloning the following git repository:

```
$ git clone \
  https://github.com/kaist-plrg/safe-ds.git
$ cd safe-ds
```

To build and execute the artifact, you should follow the instrucitons in the `INSTALL` file in the artifact. Since our artifact is based on SAFE written in Scala, it requires `sbt`, which is an intereactive build tool for Scala. Moreover, for concrete execution by Jalangi2, you also need to install Node.js LTS version.

Additionally, we packaged the artifact in a docker container. If you want to skip the environment setting, we recommend you to use it. You can install the docker by following the instruction in https://docs.docker.com/get-started/ and downlaod our docker image with the following command:

```
$ docker pull gmbale/fse-21-safe-ds
$ docker run -w /home/ubuntu/safe-ds -it \
  -m=16g --rm gmbale/fse-21-safe-ds
```

WARNING: The docker image is 1.7GB large thus be patient when you download it and please assign more than 16GB memory for the docker engine.

3 BASIC COMMANDS

You can run the artifact with the script files in the eval directory:

```
$ cd \$SAFE_HOME/eval
```

3.1 Analyze a JavaScript file with dynamic shortcut and print results

You can analyze a JavaScript program with `-ds` option as follows.

```
$ ./run -w -ds <INPUT_JS_FILE>
```

You can print analysis results as follows.

```
$ cat ds-safe.result ds-jalangi.result
```

3.2 Clean the result files

```
$ ./clean
```

3.3 Reproduce the experiments on Lodash4 tests (it takes about 24 hours)

```
$ ./experiment
```

4 STEP-BY-STEP INSTRUCTIONS

4.1 Reproduce the experiments on Lodash4 tests

```
$ cd $SAFE_HOME/eval
$ ./experiment
```

The above script generates experimental results in the `raw_data` directory.

Note that experimental results are not deterministic due to execution time and the timeout settings: 5m for total analysis time for each test case and 5s for each trial of dynamic analysis.

4.2 RQ1. Analysis Speed-up

Execute the following script to check the average analysis speed-up for each benchmark.

```
$ node RQ1.js
```

Also, the following file shows the analysis speed-up for each test case.

```
$ cat RQ1_speed-up.json
```

It only shows the average analysis speed-up described in Section 6.1. To fully calculate Figure 6, 7, and 8, please see 3rd, 4th, 5th, and 6th columns in the `raw_data/<BENCH-no?-DS>/summary.tsv` files.

4.3 RQ2. Precision Improvement

To reproduce the slope result in Figure 9, please type the following command:

```
$ node RQ2.js
```

Moreover, it produces coordinates in Figure 9 except (0, 0) in the following file.

```
$ cat RQ2_precision.json
```

4.4 RQ3. Opaque Function Coverage

The following script prints and produces the result in Table 1 respectively.

```
$ node RQ3.js
$ cat RQ3_opaque_function.json
```

4.5 RQ scripts for the result of the paper

The directory `fse21_results` contains the numbers presented in the paper. You can copy the directory into `raw_data` and run the RQ scripts on it.

```
$ cp -r fse21_results raw_data
$ node RQ*.js
```