JS-Fuzzer

Javascript fuzzer for stand-alone shells like D8, Chakra, JSC or Spidermonkey.

Original author: Oliver Chang

Building

This fuzzer may require versions of node that are newer than available on ClusterFuzz, so we use pkg to create a self contained binary) out of this.

Prereqs

You need to intall nodejs and npm. Run `npm install` in this directory.

Fuzzing DB

This fuzzer requires a fuzzing DB. To build one, get the latest `web_tests.zip` from gs://clusterfuzz-data/web_tests.zip and unzip it (note https://crbug.com/v8/10891 for making this data publicly available). Then run:

```
$ mkdir db
$ node build_db.js -i /path/to/web_tests -o db chakra v8 spidermonkey WebKit/JSTests
```

This may take a while. Optionally test the fuzzing DB with:

```
$ node test_db.js -i db
```

Building fuzzer

Then, to build the fuzzer,

```
$ ./node_modules/.bin/pkg -t node10-linux-x64 .
```

Replace "linux" with either "win" or "macos" for those platforms.

This builds a binary named `ochang_js_fuzzer` for Linux / macOS OR `ochang_js_fuzzer.exe` for Windows.

Packaging

Use `./package.sh`, `./package.sh win` or `./package.sh macos` to build and create the `output.zip` archive or use these raw commands:

```
$ mkdir output
$ cd output
$ ln -s ../db db
$ ln -s ../ochang_js_fuzzer run
$ zip -r /path/output.zip *
```

NOTE: Add `.exe` to `ochang_js_fuzzer` and `run` filename above if archiving for Windows platform.

Development

Run the tests with:

```
$ npm test
```

When test expectations change, generate them with:

```
$ GENERATE=1 npm test
```

Generating exceptional configurations

Tests that fail to parse or show very bad performance can be automatically skipped or soft-skipped with the following script (takes >1h):

```
$ WEB_TESTS=/path/to/web_tests OUTPUT=/path/to/output/folder ./gen_exceptions.sh
```

Experimenting (limited to differential fuzzing)

To locally evaluate the fuzzer, setup a work directory as follows:

```
$ workdir/
$ workdir/app_dir
$ workdir/fuzzer
$ workdir/input
$ workdir/output
```

The `app_dir` folder can be a symlink or should contain the bundled version of `d8` with all files required for execution. Copy the packaged `ochang_js_fuzzer` executable and the `db` folder to the `fuzzer` directory or use a symlink. The `input` directory is the root folder of the corpus, i.e. pointing to the unzipped data of `gs://clusterfuzz-data/web_tests.zip`. The `output` directory is expected to be empty. It'll contain all output of the fuzzing session. Start the experiments with:

```
$ # Around ~40000 corresponds to 24h of fuzzing on a workstation.
$ NUM_RUNS = 40000
$ python tools/workbench.py $NUM_RUNS
```

You can check current stats with:

```
$ cat workdir/output/stats.json | python -m json.tool
```

When failures are found, you can forge minimization command lines with:

```
$ MINIMIZER_PATH = path/to/minimizer
$ python tools/minimize.py $MINIMIZER_PATH
```

The path should point to a local checkout of the minimizer.