Demo of ExtractFix

This document contains instructions for obtaining and executing a demo version of ExtractFix.

Please note the following:

1. This is an incomplete demo version of ExtractFix. The demo is primarily designed for testing purposes, and does not reflect the final design of ExtractFix.
2. The demo should be considered alpha-quality software.
3. The demo is distributed via a Docker image (the size is around 6.5GB).

Description

In ExtractFix, the main components, such as *Propagation Engine*, are mainly implemented using C/C++, and linked using Python. The executable of ExtractFix is ExtractFix.py. The basic usage of the demo tool is as follows:

```
Usage: ExtractFix.py [-h] -s SOURCE_PATH -t TESTS [TESTS ...] -c RUN_COMMAND -b
BUG_TYPE -n BINARY_NAME [-v]

Optional arguments:
 -h, --help                 show this help message and exit
 -s SOURCE_PATH, --source-path SOURCE_PATH
                            the path of the project
 -t TESTS [TESTS ...], --tests TESTS [TESTS ...]
                            the test input
 -c RUN_COMMAND, --run-command RUN_COMMAND
                            the command to compile the target program
 -b BUG_TYPE, --bug-type BUG_TYPE
                           type of the crash/vulnerability (supported type:
                               [buffer_overflow, integer_overflow, null_pointer,
                                assertion_failure, divide_by_0, api_specific])
 -n BINARY_NAME, --binary-name BINARY_NAME
                           the binary name
 -v, --verbose             show debug information
```

Run the Demo

In order to run the ExtractFix demo, please create a temporary docker container using the following command:

```
$ docker run --rm -ti gaoxiang9430/extractfix:demo /bin/bash
```

The docker image will be automatically downloaded from docker hub and a docker container will be created by executing a bash shell. Once the shell is launched, execution the following commands to run the demo:

```
# cd ExtractFix
# source setup.sh        ### this is used to setup the environment variables.
# cd build
# ./ExtractFix.py -s ../demo/libtiff-5321 -t test_case -c driver -b buffer_overflow -n tiffcrop -v
```

The above command will try to fix the buffer overflow in *demo/libtiff-5321*, and that can be triggered by *demo/libtiff-5321/test_case*.

Output

ExtractFix will output the generated patch and some intermedia results.

In the /ExtractFix/demo/result0 directory, there are three files:

- constraints.txt: the constraints provided by KLEE
- fix_stm.txt: the statement that is patched
- patch: the generated patch (which includes some noises caused by instrumentation)

While /ExtractFix/demo/logs provides all the intermedia results:

- cfc: the crash free constraints generated by Sanitizer
- fixlocalization.json: the fix localization results

The generated patch in this example is:

```
992c994
<       for (s = 0; s < spp; s++)
---
>       for (s;(((s)<(spp))&&((s)<=(7))); s++)
```