Black-box Fuzzing Problem Set

Description

The goal of this problem set is to identify potentially exploitable crash bugs in the program `/usr/bin/pdftotext` using black-box mutational fuzzing.

To complete the problem set, you will need to ssh to your container at `$user@amplifier.ccs.neu.edu:$port`, where `$user` is your gitlab username and `$port` is your assigned ssh port (https://seclab-devel.ccs.neu.edu/snippets/6). Authentication is performed using any of your uploaded ssh public keys in gitlab.

Important Information	
Available	Thu 09 Apr 20:00 EST
Submission Deadline	Fri 17 Apr 18:00 EST

Test Harness

To fuzz the target program, create a test harness that can run the program on an input and collect any resulting crash dump for later analysis. That is, your harness should be able to execute a target program in a child process, monitor the child for abnormal termination, and in that case store the child's core dump.

Input Generation

Your test harness will require an input generation strategy. In particular, given an initial configuration value – i.e., a random 32 bit unsigned integer – your input generator should create a mutant input from the input seed located at `/usr/local/share/gpg_print.pdf`. Input generation should be *deterministic* given an initial configuration value; that is, two fuzzing runs with a given configuration value should produce identical sequences of mutated inputs.

Crash Triage

Write a program to cluster the crash dumps your fuzzer produces, grouping crashes according to whether they represent the same bug in the target program.

Answer Submission

Create a repository in gitlab at `git@seclab-devel.ccs.neu.edu:$user/prset06.git`. Commit your fuzzer to `fuzzer/`, and include an executable script at `fuzzer/fuzz` that runs your fuzzer with the following command-line interface:

```
$ ./fuzz $init_config_value $input_seed
```

NOTE: Your fuzzer *must* be executable using the above interface from a fresh git checkout of your repository to receive full credit.

Also, commit a `README.md` that describes in as much detail as possible the following:

- The design of your test harness
- The input generation strategy you used
- The criteria you used to cluster your crash dumps
- How many bugs did you find, and of what kind?
- Which bugs were easier to trigger, which were more difficult, and why?

Finally, commit a `solution.json` with the following format:

```
{
    "clusters": [
        [
            <init_config_value_0>,
            <init_config_value_1>,
            <init_config_value_2>,
            // ...
        ],
        // additional clusters of initial configuration values
    ]
}
```