Assignment 3 — More Fuzzing

2018-10-07

This assignment consists of two parts. The submission is a single `.tar` or `.tgz` file. Opening the file should create a single folder whose name is your a-number. Inside the folder there is one PDF file and two subfolders. The PDF file, `answers.pdf`, contains the answers to both parts of this assignment. The subfolders, named `part-1` and `part-2` contain additional information and solutions for Parts 1 and 2.

We will provide a test script that checks these and some other requirements. If your submission does not meet the requirements above we will not test it and your mark will be 0.

Part 1 (20%) — Getting Here.

Break the code that hides the URL of this assignment. This is a part of the assignment even if you reached here without breaking the code! Submit a description of how you broke the code. (Not more than one page of text.) If you used any software to break the code, submit it in the subfolder `part-1`. Such software can be in any programming language. It will not be evaluated for coding or style quality. Its purpose is only to support your description.

Part 2 (80%) — Advanced Fuzzing.

So far we have used fuzzing to find crashes. Fuzzing can also be used for finding logical bugs. The idea is that you need to compare the output of the program under test with the ground truth, and force a crash, e.g. by calling `abort()`, if the outputs do not match. This is a slightly limited scenario because you should be able to generate the ground truth, however, this is still useful in some scenarios, for example to verify that new versions of software produce the same output as the old versions.

In this assignment we provide a buggy bignum library. You can get it by running `tar xf assignment3.pdf ybn`. To avoid naming conflicts, we have changed all the functions and type definitions to start with `ybn_` instead of `bn_`, however, the expected semantics of the functions is the same. Your task is to build a test environment that allows you to fuzz the code we provide, identify the bugs, and fix them. To get the ground truth, use your own `libbn.a`. (Instead of using your own code, you can use any of the examples provided as part of Assignment 2 at a penalty of 10% to this part of the assignment, i.e. 8% of the mark of the assignment. Your `answers.pdf` *must* disclose a use of another student's code.) Adapting your `calc` program could come handy. You are encouraged to also use the sanitisers discussed in class, and can use code inspection.

The folder `part-2` in your submission should contains two subfolders: `ybn` which contains the *fixed* library, and `fuzz` which contains the environment you used to build test and fuzz `ybn`. It may also include a copy of `ybn` or its contents. We are not evaluating the contents of `fuzz` for coding quality or correctness. The main reason we need it is to be able to better understand how you built the test environment.

In `answers.pdf` please submit a description of the test environment (30%) and a description of each bug you found, including how you found it, what the problem is, and how you fixed it (50%).