CIS4361-Programing project: Fuzzing

Due: Apr 15, 2018

Overview

The goal of this project is to implement a “fuzzer”, or fuzz tester. Fuzz testing is one way of discovering security vulnerabilities in any code that processes potentially malicious input. The concept of a fuzzer is simple; it repeatedly runs a program which is being evaluated on a series of automatically generated inputs. If some input causes the program being tested to crash with a memory access violation, the program has failed the test and may have an exploitable security vulnerability. By generating inputs either partly or completely randomly, the fuzzer can rapidly run large numbers of tests without human supervision and also may discover bugs which occur under unusual inputs which the developer may have overlooked.

A mutation-based fuzzer takes a valid input for the target program, and works by creating random mutations or changes to generate new test cases. Mutation-based fuzzers are application independent, and so they do not have any knowledge about input format (protocol format) accepted by the target program.

In teams of maximum four, you will implement a mutation-based fuzzer capable of finding bugs in real-world programs. Each team may implement their fuzzers in the programming language(s) of their choice. As part of the project, you introduce a number of vulnerabilities into a JPEG to PDF converter program (called jpg2pdf) which we will provide to you its source code, “jpg2pdf.c”.

How to save generated image files that trigger some bugs?

In most cases, your generated image file (based on the sample.jpg) cannot trigger any bug. Since you need to generate tens of thousands mutated image file as input to the jpeg2pdf for fuzzing. In your fuzzer, only when you determine that the jpeg2pdf generates signal 11 (segmentation fault) then you save the generated image file.

Project Report Requirement

Please submit a project report on 3-5 pages including sections such as Analysis, Design and Implementation.

Design

Briefly explain your strategy used to implement the fuzzing test. Specifically, you should point out how you mutate the sample.jpg, including which byte or bytes you modify, what values you use, and which bug is triggered by modifying which format field (if you use jpeg format), etc.

Empirical results

Show one or several graphs or tables describing the results from running the fuzzer. Show how many inputs you have tried in fuzzing, the number of bugs you have found (which bugs you have found). For each bug you found, how many different image input files have triggered the bug.

When you find an input modified image file 'test.jpg' that could trigger Bug #1, name the file as 'test-1.jpg', save it (for each bug, you only need to save one image input for final submission; of course, during fuzzing test, you may have saved multiple image files that generate the same bug).

Then when you run the program: (using command line: *jpeg2pdf sourceFile destinationFile* **)**

The jpegconv should crash and print out:

BUG 1 TRIGGERED

Show the screenshot image(s) of each bug you have triggered.

Delivery:

Submit a .zip file through UCF Canvas. The zip file should contain:

1. Source code file of your fuzzer. You can include multiple fuzzer codes where each of them is used to discover one or several specific bugs.

2. The input files (modified jpg images based on the sample.jpg) that could trigger each of the bugs you can find. For each bug, just provide one input image file. Name the input file as "test-x.jpg" for Bug # x. In this way, I can easily check whether you really found Bug #x by running it by myself!

3. A 3-5 page project report (in PDF).

Grading

(70 points) There should be at least 10 bugs in the sample program. You need to find them to receive full credit. A bug is confirmed to be found only if you have a screenshot image showing the crash and the printout text by jpeg2pdf showing triggering of the bug.

(10 points) Empirical results presented in the project report. Report contains clear description.

(10 points) Submitted compressed file contains image files that could trigger each of found bugs (e.g., if you found 6 bugs, you should have 6 test-x.jpg files in your submission).

(10 points) Project report contains the screenshot images showing the "BUG X TRIGGERED" for each found bug.

You can find list of some open source fuzzer in the following link:

https://www.peerlyst.com/posts/resource-open-source-fuzzers-list