SE-4930: Developing Secure Software

Lab 8: Fuzz Testing

Due by 23:00 February 11, 2013

Introduction

In class, we looked at the process of fuzz testing. In this weeks lab, you are going to actually do a bit of fuzz testing as well as fix an application which suffers from poor fuzz testability.

Step 1 – Reading about Fuzz testing

IBM Developer works has a nice article explaining the concept of fuzz testing at http://www.ibm.com/developerworks/java/library/j-fuzztest/index.html. Before starting this lab, you will want to understand the concepts as well as take a look att he sample code that is present there.

Step 2 – Understanding the Application

On the course website, there is a simple Java program. This java program essentially reads and concatenates a set of XML files which represent bug reports. The program takes a set of command line arguments, the first of which is the output file and the rest are the individual bug reports that are to be concatenated. From this, it creates an XML file which is the effective concatenation of the bug reports. The program has not been significantly tested, and simply throws any exception which occurs back to the user.

Step 3 – Lab Process and Writing a fuzzer

Based on the IBM article and the code provided, you are to do two things. First off, you are to construct a tool for fuzz testing this application. In essence, the tool will take a valid XML file, mutate it in some manner, and try to parse it with the tool. If the tool hangs for a given period of time or throws an exception, then you can consider the program to have crashed. If the program does not hang or throw an exception, than you can consider the output to be valid.

As you are developing the fuzzer, you goal also is to make the application more robust by handling exceptions. (Note: No, a global catch of any exception is NOT acceptable…) Your fuzzer should keep track of the number of times it runs successfully as well as the number of times it fails and should be able to generate a simple output log showing its test execution. (Note: Simplicity is key here… Don't get too fancy.)

Deliverables

Each team shall submit a report in pdf format with the following information

1. *Title Page* - Name of all team members, course, and date details.

2. *Introduction* - Summarize what you did in lab and the problem your were trying to solve.
3. Discussion
    a. How did you design your fuzzer? Did you integrate it with JUnit or something else?
4. Source Code differences
    a. Include a difference report between the initial source code for the application provided by the instructor and the final version with bugs fixed. You may generate this with Beyond Compare or any other tool that you desire.
5. *Things gone right / Things gone wrong* –
    a. Discuss the things which went well with this lab, as well as the problems you had, either with the tools, the process, etc.
6. *Conclusions*
    a. What have you learned from this experience

In addition, upload your java code and tester in a zip file archive.