Fuzz Testing

Jason Greenlaw

CS1632 – DELIVERABLE 6

https://github.com/jasongreenlaw/deliverable6

Summary

For the last deliverable I decided to do something called Fuzz Testing. Fuzz Testing is a form of random testing that is often used to test security issues. I stumbled across this after browsing the internet a bit for different topics to use for my final project. I’ve had a small but intriguing interest in security which led me to stumble upon a couple youtube videos through the recommendation of a friend for this project, and then found fuzz testing. The testing involves injecting random data into the inputs of different programs to see how they react. In particular we are testing to see if the program crashes, but you can also look for assertions, memory leeks, and other issues. It can be used for white-box testing, but it is more commonly used as a black box testing technique.

I decided to black box fuzz test programs used to open .docx, .pdf and .txt files. I took files that would normally open correctly in these programs, and change random parts of the files to random data, and then attempt to open them. Ideally each of these programs would have a proper way of handing this issue of trying to open a now corrupt file, and not crashing.

The Fuzzer program takes a couple command line arguments. The first is the filename that the test will run on, and the actual program tested is the default program for opening that file type. I then take a fuzzFactor and number of iterations. The file, starting from the original state again every time, will be tested the number of iterations received on input. For iteration, I will randomly change a random number of bytes between 1 and the file length divided by the fuzz factor, to random byte values, and then attempt to open the file. I check for and log any errors generated by the process while opening it. There is also a FuzzerDriver file that does all of this for multiple files of the types I listed before.

Testing Concerns/Concerns Going Forward

For the most part there were no major issues when writing the tests. Working with the files and paths java libraries was new and challenging at first, but once we learned them properly, we were able to use them where necessary. When actually running the tests, our driver is attempting to open hundreds of files through programs on the computer in a fairly short amount of time. This didn’t cause any major issues, but did seem to run slow and at some points seemed overwhelming.

Based on the experiences thus far with these programs, and with fuzz testing, I would not expect any major issues going forward with testing these programs. Doing more investigating on performance with these programs (especially with Microsoft Word, which is a huge program, can feel slow on occasion) and incorporate that with more fuzz testing would be interesting to look into further. Regarding the security aspect of fuzz testing, finding trust boundaries and looking to further testing the security capabilities of the programs would also be another type of tests that could be run on the system.

Assessment of Quality

There were no failed tests from our fuzz testing. Technically in this circumstance, the programs being tested were Microsoft Word, Preview, and TextEdit. I would classify all of these programs as very high quality. Having been approved by Microsoft and Apple for production, I was expecting my fuzz test to produce these positive results. Project code located here: https://github.com/jasongreenlaw/deliverable6

Test Results