Fuzz Testing

Michael Stoltz, Adam Darr

CS1632 – FINAL DELIVERABLE

https://github.com/adamdarr/fuzzer

Summary

For our project we decided to do Fuzz Testing. Fuzz Testing is a form of random testing that is often used to test security issues. 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. Although it can be used for white-box testing, it is more commonly used as a black box testing technique.

We decided to black box fuzz test programs used to open .docx, .pdf and .txt files. Our tests use the default program on the machine where it is running, so in our case that was a mac, and the programs are Microsoft Word, Preview, and TextEdit. We will take 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.

Our Fuzzer program takes a few 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. Then we also take a fuzzFactor and number of iterations. The file, starting from the original state again every time, will be tested the number of iterations inputed. For each of those iterations, we 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. We check for and log any errors generated by the process while opening it. We also have 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 we did not encounter any 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 could be overwhelming. Many of the programs had dialog boxes and other user involved ways of handling their inability to open the files, which was our intended finding instead of the programs crashing or error out in some way, but the dialog boxes piled up very quickly and caused a mess on the computer where the tests were run.

Based on our 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. With 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 we would like to run on the system.

Assessment of Quality

We did not have any 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, we were expecting our fuzz to test to produce these positive results. It would then be my recommendation that they would be released, if they hadn’t been already ☺. Project code located here: https://github.com/adamdarr/fuzzer

Test Results