Lawrence Oks

CAP6135

March 11, 2022

Programming Assignment 2: Fuzz Testing

1. **Design**

   In this programming assignment we find software vulnerabilities in a simple target program “jpgbmp” by implementing a fuzzer. Jpgbmp is an image conversion program that converts jpg files to bmp files. A fuzzer detects security vulnerabilites in a program by randomly mutating a valid input for that program to generate many test cases, some of which may find a bug. In this case, we created a fuzzer that takes a valid jpg file for jpgbmp and mutates it 10,000 times to produce as many input images that can be tested on the program. I produce the provided “cross.jpg” file 5000 times each in one of two ways:

   1. Alter a single random byte of cross.jpg with a random value between 0-255.
   2. Alter a random 1-4 byte value in cross.jpg to a magic number[1][2]

2. **Implementation**

   For each of the 10,000 iterations, the provided cross.jpg file is read and converted to a list of bytes, so that we can updated the list with a mutated byte(s). For each half of the iterations, we alter the image using one of the mutation techniques listed above, and save it in case it triggers a bug in jpgbmp. We then run the mutated image through the program, and analyze the output of the program to determine if it triggered a bug or not. If it did, we update the count of bugs triggered, then rename the mutated file to represent the bug it triggered. If the mutated image did not produce a bug, we delete both the input and output files as they are of no use to us. Additionally, we keep track of which bugs have

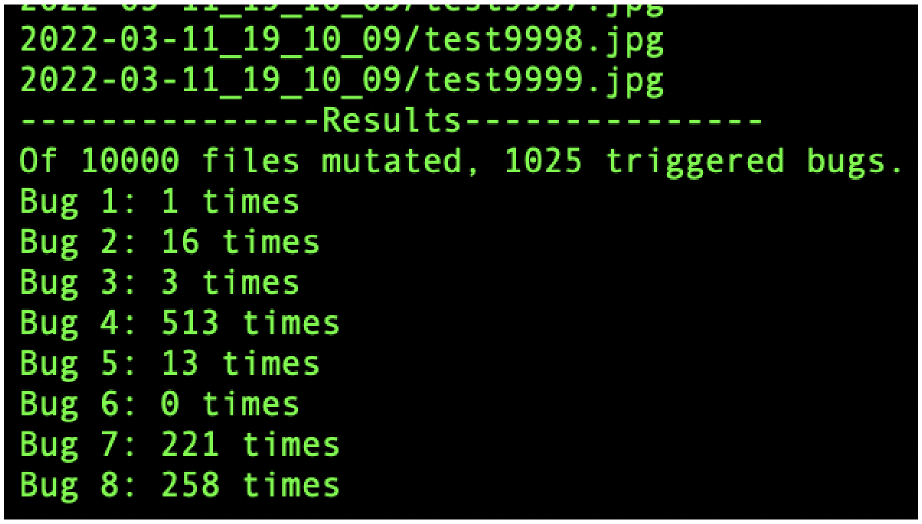
[1] https://en.wikipedia.org/wiki/Magic_number_(programming)

[2] Mutation Code for method 2 borrowed from https://h0mbre.github.io/Fuzzing-Like-A-Caveman/

been triggered so that if all 8 bugs are found before all iterations complete, we stop the fuzzer early.

3. **Results and Analysis**

Not all mutations produced bugs, in fact only 1,025 of the 10,000 total iterations did so. We can see these results below in Figure 1:



```
2022-03-11_19_10_09/test9998.jpg
2022-03-11_19_10_09/test9999.jpg
---------------Results---------------
Of 10000 files mutated, 1025 triggered bugs.
Bug 1: 1 times
Bug 2: 16 times
Bug 3: 3 times
Bug 4: 513 times
Bug 5: 13 times
Bug 6: 0 times
Bug 7: 221 times
Bug 8: 258 times
```

Figure 1: Results from running the fuzzer

My mutate implementations were not able to trigger bug 6, but did trigger all others, especially bug 4 most commonly. Notably both mutation methods were able to trigger all bugs except bug 1, which was only able to be triggered by method 1. The magic numbers could not trigger bug 1. We use Table 1 to demonstrate the number of occurrences for each bug:

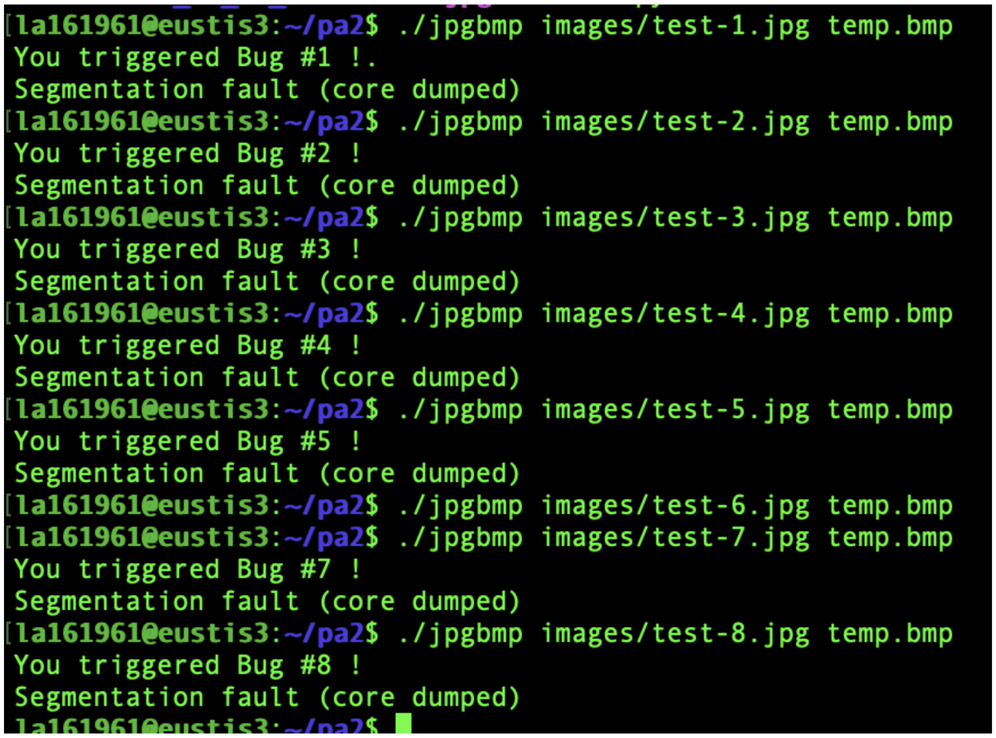
Bug	Count
1	1
2	16
3	3
4	513
5	13

6	0
7	221
8	258

Table 1: Occurrences for each bug produced

4. **Output**

   To demonstrate that each of my mutated images can trigger the bug when input into the jpgbmp program, a screenshot is provided in Figure 2 of the bugs being produced once again. Additionally, each of these images have been attached to the submission.



```
la161961@eustis3:~/pa2$ ./jpgbmp images/test-1.jpg temp.bmp
You triggered Bug #1 !.
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$ ./jpgbmp images/test-2.jpg temp.bmp
You triggered Bug #2 !
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$ ./jpgbmp images/test-3.jpg temp.bmp
You triggered Bug #3 !
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$ ./jpgbmp images/test-4.jpg temp.bmp
You triggered Bug #4 !
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$ ./jpgbmp images/test-5.jpg temp.bmp
You triggered Bug #5 !
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$ ./jpgbmp images/test-6.jpg temp.bmp
la161961@eustis3:~/pa2$ ./jpgbmp images/test-7.jpg temp.bmp
You triggered Bug #7 !
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$ ./jpgbmp images/test-8.jpg temp.bmp
You triggered Bug #8 !
Segmentation fault (core dumped)
la161961@eustis3:~/pa2$
```

Figure 2: Bugs triggered again by each of the mutated images

5. **Instructions to run**

This report was submitted in a file named pa2.zip. Assuming this zip file has already been decompressed so that you can read this report, the following steps need to be run in eustis3 to reproduce my results:

1. cd pa2/
2. python fuzzer.py

Any mutation jpgs that trigger bugs will be saved in a folder that is named by the current timestamp.