Exercise 06 – Fuzzing

General Directions

Please submit –

1. Your fuzzer code as a file named "fuzzer.py".
2. README file explaining what you did.
3. The "bits" file for each original file provided in the exercise, in the naming convention described above, in the same directory structure you got the exercise in.

General note: Running your fuzzer on all the input files may take a while – **run it in advance to submit on time!**

Exercise

1. Unpack the ex06.bin ex-pack as usual.
2. You will find several sample images in various formats in the base_images directory.
3. In this exercise, you will fuzz the ImageMagick toolset using these base images (you can read about the various ImageMagick tools online or on the man pages).
   We will focus on the convert program.
4. Your goal is to make the command "convert [modifiedsample image] [output.bmp]" exit with an error, by flipping one bit at a time in the image data.

5. Fuzzer specification and hints –
   - **5.1. Write the fuzzer in python.**
   - **5.2. The fuzzer you write will only need to fuzz the first 1000 bytes of each file, and the last 500.**
   - **5.3. Please explain - Why not fuzz the entire file? Why are those parts of the file chosen? (you may need to read a little about the file formats)**
   - 5.4. Your script will be called in the form of :
     "./fuzzer.py [path to base image]".
   - 5.5. After your fuzzer has completed its run, there should be a new file, named "[path to base image].bits" (the original path, concatenated with ".bits"), which has the locations of each bit that caused the convert program to crash or exit with an error, one location per line, in the format of:
     [Byte address in hex].[bit number, 0-7]\n
   - 5.6. Learn how to launch the regular 'convert' program from your fuzzer. Read about the subprocess module and subprocess.Popen() / subprocess.call().
     Any other **reliable** method will also be fine.
   - 5.7. Read up further to understand how to monitor the child process.
   - 5.8. You will need to loop over the bit locations. Each round begins by creating a modified base image and feeding it to the "convert"program. It's best to write this temp image to /tmp/. Make sure to preserve the file extension (file type).
   - 5.9. To edit the image files, you may use direct binary manipulation or the 'struct'

module. Other ways that work will also be accepted.

5.10. After testing each file, make sure if a converted file was generated, and if you found an error or a crash – print the location to the output “bits” file.

6. **Bonus (+25 points):**

Generate a file that actually **crashes** (seg-faults) the convert program. This might not be possible with the technique specified here (that’s why it’s a bonus).