Project 3
Security In Software Applications

Giovanni Pica
1816394

1 Introduction

This project is based on **Fuzz Testing** and the tool used is **American Fuzzy Lop** that is a security-oriented fuzzer that employs a novel type of compile-time instrumentation and genetic algorithms to automatically discover clean, interesting test cases that trigger new internal states in the targeted binary. The software that is tested is **ImageMagick** that is an image manipulation software, in particular in this project I decide to test the 6.7.0-10 version. I decide also to use **ASAN** which is a memory error detector for C/C++.

2 Setup

At the beginning I decide to develop this project on my **Macbook Air M1**, but I notice that AFL doesn't support this CPU and when I tried to compile the makefile it gives to me many errors, so I decide to use my old **Lenovo** with a VM based on **Linux Mint**. The steps to correctly install AFL were:

- get AFL from github with
  `git clone https://github.com/google/AFL.git`
- make the MakeFile of AFL and install it

Now the AFL is installed and afl-clang or afl-gcc (afl-clang++ and afl-g++ for C++ programs) can be used for the fuzz testing of any kind of software. To perform AFL to ImageMagick the steps were:

- get ImageMagick from https://sourceforge.net/projects/imagemagick/files/old-sources/
- change the version of the ImageMagick with
- set the CC=afl-gcc, CXX=afl-g++ ,CFLAGS="-fsanitize=address -g" and also for CXXFLAGS for ASAN and execute
  `./configure --prefix=/path/to/ImageMagick`

- make and install the file created by the instruction above and setting AFL_USE_ASAN=1
- create the afl_in and afl_out directory for input files and output
- finally perform fuzz testing with -d for quick and dirty mode (skips deterministic steps) and -m none for ASAN
  `afl-fuzz -d -m none -i afl_in -o afl_out -- ./bin/convert @@ /dev/null`

3 Results

Number of files used is **1652** and I take the downloadable archive from https://lcamtuf.coredump.cx/afl/demo/ and I decide to use **PNG** format. Results are summarized in the figure 1:

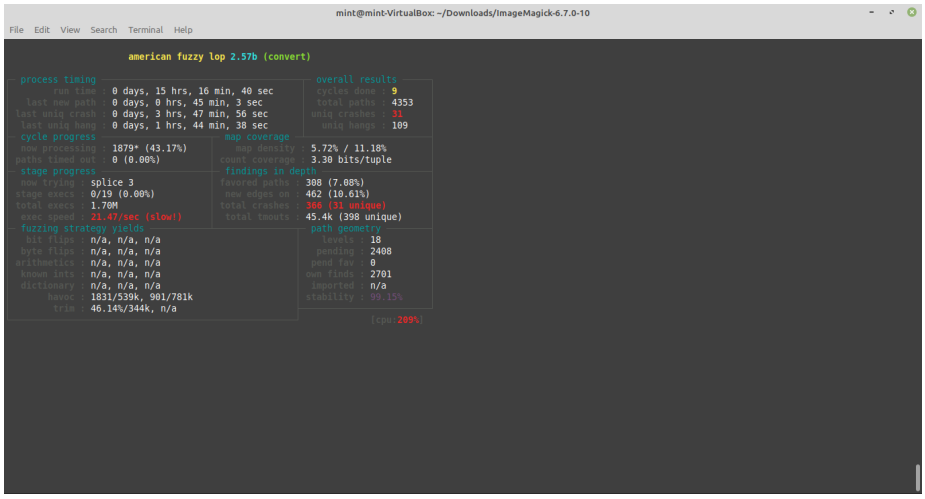

Figure 1: Results of AFL

3.1 Flaws Found are known CVEs?

For flaws research I ran "convert" function of ImageMagick of the files that were in the folder "crashes" (that are 31) of "afl_out" folder and I discovered that all of those crashes are caused by Heap-Buffer-Overflow and in this site there are the CVEs based on this flaw https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=heap+buffer+overflow. In the figure 2 I show how the ASAN works to find those flaws. I also handled the "hangs" and I discovered that some of those timeouts were caused by memory leaks that the LeakSanitizer found or unexpected end-of-file or ivalid pixels or pixel cache allocation failed and so on. In figure 3 I give an example of how LeakSanitizer works.

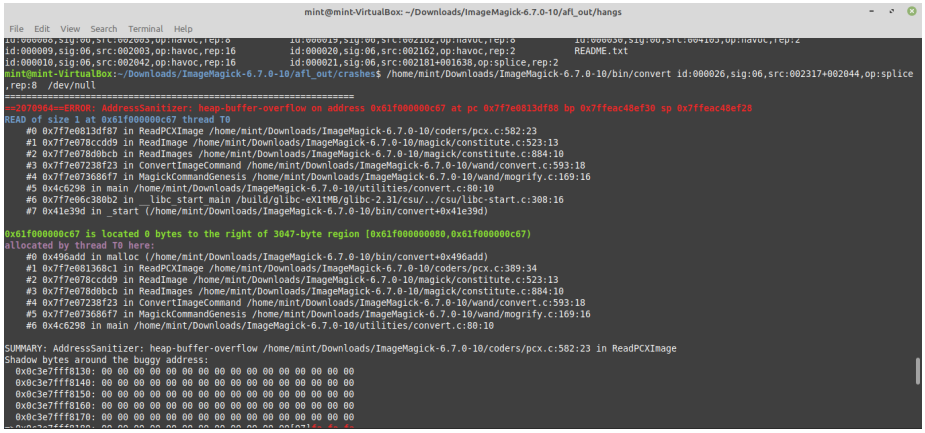

Figure 2: ASAN heap buffer overflow

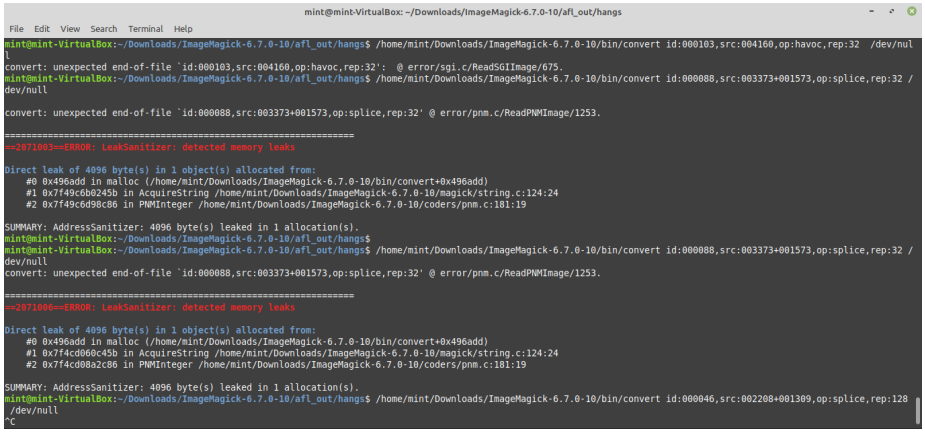

Figure 3: Hangs memory leaks