任务

Develop a fuzzing tool, that:

- Fuzzing files, memory, APIs, web applications, web browsers, protocols, etc. Either is fine
- Including test data generation, test execution, test monitoring
- You must be successful in finding bugs, either artificial or real is OK
- Submission: source code of your fuzzing tool, running screenshots, tested program (source/binary), analysis reports of the finding bugs
- Reminder: real, unknown bugs in complicated software get more credits

文件介绍

- 待测程序：源代码位于 `ExifFuzzer/exif`，其 Github 仓库
- 项目工具：源代码位于 `ExifFuzzer/src`，已上传至 Github
- 测试环境：Ubuntu 18.04, Python 3.9

项目简介

Exif文件格式与JPEG文件格式相似，Exif会向JPEG中插入一些图像/数字信息以及缩略图信息，所以可以像查看JPEG文件一样，使用兼容JPEG的浏览器、图像查看器或者图像修改软件来查看Exif格式的图像文件。

本项目对 Exif 图形阅读器 ExifFuzzer/exif 做 fuzzing。

安全策略

典型的安全策略往往将 `fatal signal` （例如段错误）视为违规测试，该策略易于实现，因为操作系统允许此类异常情况在没有任何检测的情况下被 fuzzer 捕获，但该策略并不能检测所有的内存漏洞，例如堆栈缓冲区溢出，覆写了指针指向另一个有效地址，程序可能仅出现无效结果，而不会崩溃终止，因此 fuzzer 无法检测到该漏洞。

本项目采用谷歌的 `Sanitizers` 识别其他错误。

变异

本项目变异策略：

- 使用 **算数突变** 的变异方法，将种子转化为字节序列作为整数 i ，并对该值进行简单的算数运算 $i \pm r$，其中 r 由用户自定义配置，默认 $0 < arithmetic\ range < 35$
- 结合 **位翻转** 思路，提供变异比 `mutation_ratio`，仅对给变异比的字节进行突变

评估

本项目构建线程池，使用变异生成的测试用例执行 PUT，本次实验将触发安全策略的测试用例保存到 `testcase` 中，未触发的将被删除。

`Triage` 用于分析和报告违反策略的安全用例，通分为：重复数据删除，优先级和测试用例最小化。

在本实验中，仅对输出结果做简单判断，并将 `Sanitizer` 的输出保存到 result 中，文件后缀格式含义如下：

- HBO：heap buffer overflow
- SBO：stack buffer overflow

- ML：memory leaks
- SEGV：sigsegv(e.g. segmentation fault)

使用方法

0. 环境依赖

```
Ubuntu 18.04 4核 4G内存
Python 3.9.23
```

1. 安装项目依赖

```
pipenv install
```

2. 编译待测程序

```
git clone https://github.com/mkttanabe/exif.git
gcc -fsanitize=address -ggdb -o exif_ASan sample_main.c exif.c
```

3. 自定义 fuzzer 核心配置 `config.ini`

```
[generation]
mutation_ratio = 0.01     # 变异率
arithmetic_range = 0,35   # 字节随机范围，借鉴AFL: 0 < r <35
[evaluation]
round = 500               # fuzzing 多少轮
thread_count=5            # 线程数，请设置: cpu核数+1
```

此外，请将 `config.ini` 中的路径都改为自己的对应路径。

4. 开始 fuzzing

```
python src/main.py
```

执行分析

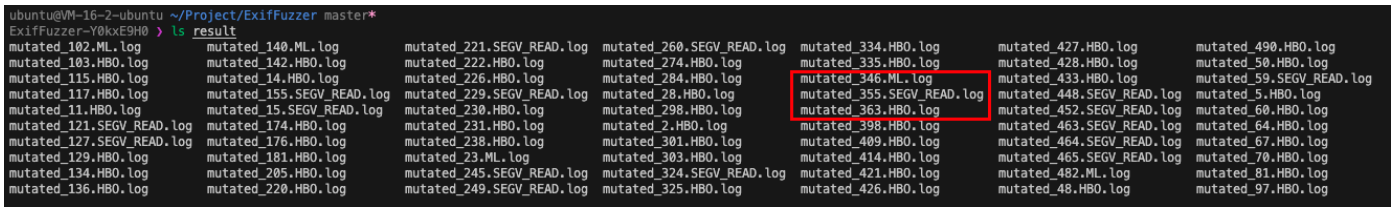
执行 ExifFuzzer，默认变异率为 1%，突变范围随机 $0 < r < 35$，4 核 5 线程，执行 500 轮 fuzzing，结果如下：

```
CaveFuzzing-vyolCl_H ❯ python src/main.py
2023-05-24 16:35:55.848 | INFO     | __main__:main:17 - [ExifFuzzer] Round: 500, Thread Count: 5, Start Success
2023-05-24 16:36:01.726 | INFO     | __main__:main:25 - [ExifFuzzer] Finished in 5.877215147018433 Second.
```

共耗时 5.87s，在 testcase 文件夹下存放着触发安全策略的测试样例，如图：

```
ubuntu@VM-16-2-ubuntu ~/Project/ExifFuzzer master*
ExifFuzzer-Y0kxE9H0 ❯ ls testcase
mutated_102.jpg  mutated_129.jpg  mutated_15.jpg   mutated_222.jpg  mutated_245.jpg  mutated_2.jpg    mutated_346.jpg  mutated_426.jpg  mutated_464.jpg  mutated_5.jpg
mutated_103.jpg  mutated_134.jpg  mutated_174.jpg  mutated_226.jpg  mutated_249.jpg  mutated_301.jpg  mutated_355.jpg  mutated_427.jpg  mutated_465.jpg  mutated_60.jpg
mutated_115.jpg  mutated_136.jpg  mutated_176.jpg  mutated_229.jpg  mutated_260.jpg  mutated_303.jpg  mutated_363.jpg  mutated_428.jpg  mutated_482.jpg  mutated_64.jpg
mutated_117.jpg  mutated_140.jpg  mutated_181.jpg  mutated_230.jpg  mutated_274.jpg  mutated_324.jpg  mutated_398.jpg  mutated_433.jpg  mutated_48.jpg   mutated_67.jpg
mutated_11.jpg   mutated_142.jpg  mutated_205.jpg  mutated_231.jpg  mutated_284.jpg  mutated_325.jpg  mutated_409.jpg  mutated_448.jpg  mutated_490.jpg  mutated_70.jpg
mutated_121.jpg  mutated_14.jpg   mutated_220.jpg  mutated_238.jpg  mutated_28.jpg   mutated_334.jpg  mutated_414.jpg  mutated_452.jpg  mutated_50.jpg   mutated_81.jpg
mutated_127.jpg  mutated_155.jpg  mutated_221.jpg  mutated_23.jpg   mutated_298.jpg  mutated_335.jpg  mutated_421.jpg  mutated_463.jpg  mutated_59.jpg   mutated_97.jpg
```

在 result 文件夹中存放着对应的分析报告，如图：

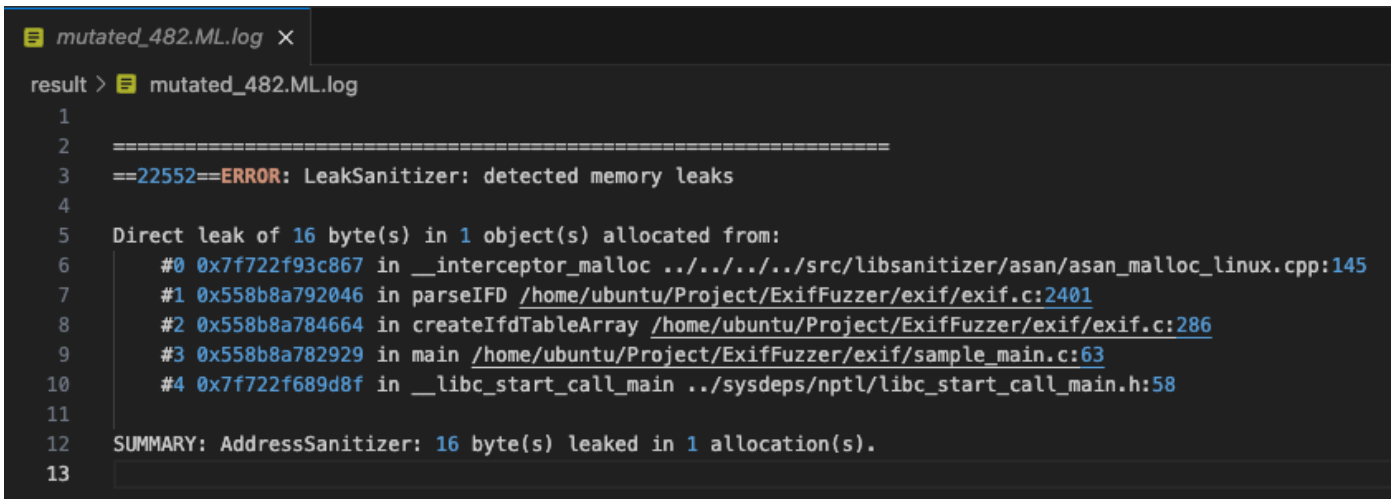


可以看到，主要分为三类漏洞：

- ML：memory leaks
- HBO：heap buffer overflow
- SEGV：sigsegv(e.g. segmentation fault)

经过人工分析，许多文件都指向同一个漏洞，本项目并未在 Triage 实现重复数据删除，后续将进行优化。

memory leaks



进入 `exif.c`，查看：

```
2401 array = (unsigned int*)malloc(allocSize);
2410 if (len <= 4) {
       ...
       } else {
       if (seekToRelativeOffset(fp, tag.offset) != 0 ||
           ...
2439       continue;
         }
       }
2439 free(array);
```

`line 2401` 执行 `malloc()` 后，`line 2439` 直接执行 `continue` 分支，跳过了 `line 2439` 行的 `free()` 方法，导致了内存泄露。

该内存泄露并不会引发程序崩溃退出，此处笔者采用不使用 `Sanitizer` 编译的 exif 执行测试用例：

```
exif/exif testcase/mutated_482.jpg
```

并不会触发异常。

heap buffer overflow

mutated_414.HBO.log

result > mutated_414.HBO.log

```
1  ==================================================================
2  ==22414==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000016 at pc 0x7fe001c22f89 bp 0x7fffe461a5d0 sp 0x7fffe4619d48
3  READ of size 7 at 0x602000000016 thread T0
4      #0 0x7fe001c22f88 in printf_common ../../../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors_format.inc:553
5      #1 0x7fe001c24bd5 in __interceptor_vsnprintf ../../../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors.inc:1668
6      #2 0x55645b514d2f in PRINTF /home/ubuntu/Project/ExifFuzzer/exif/exif.c:2666
7      #3 0x55645b5069e6 in _dumpIfdTable /home/ubuntu/Project/ExifFuzzer/exif/exif.c:469
8      #4 0x55645b505fe0 in dumpIfdTable /home/ubuntu/Project/ExifFuzzer/exif/exif.c:409
9      #5 0x55645b503be4 in main /home/ubuntu/Project/ExifFuzzer/exif/sample_main.c:93
10     #6 0x7fe0019c6d8f in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
11     #7 0x7fe0019c6e3f in __libc_start_main_impl ../csu/libc-start.c:392
12     #8 0x55645b5035e4 in _start (/home/ubuntu/Project/ExifFuzzer/exif/exif_ASan+0x55e4)
13
14 0x602000000016 is located 0 bytes to the right of 6-byte region [0x602000000010,0x602000000016)
15 allocated by thread T0 here:
16     #0 0x7fe001c79867 in __interceptor_malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:145
17     #1 0x55645b50b025 in addTagNodeToIfd /home/ubuntu/Project/ExifFuzzer/exif/exif.c:1546
18     #2 0x55645b5126d4 in parseIFD /home/ubuntu/Project/ExifFuzzer/exif/exif.c:2336
19     #3 0x55645b50546e in createIfdTableArray /home/ubuntu/Project/ExifFuzzer/exif/exif.c:271
20     #4 0x55645b503929 in main /home/ubuntu/Project/ExifFuzzer/exif/sample_main.c:63
21     #5 0x7fe0019c6d8f in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
```

```
2666    cnt = vsnprintf(buf, sizeof(buf)-1, fmt, args);
```

理论上来说 vsnprintf 不会出现缓存区溢出的问题，猜测是 False Positives。

SEGV

mutated_355.SEGV_READ.log

result > mutated_355.SEGV_READ.log

```
1  AddressSanitizer:DEADLYSIGNAL
2  =================================================================
3  ==22287==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x55a44fde2a9f bp 0x7ffe213942b0 sp 0x7ffe21392050 T0)
4  ==22287==The signal is caused by a READ memory access.
5  ==22287==Hint: address points to the zero page.
6      #0 0x55a44fde2a9f in parseIFD /home/ubuntu/Project/ExifFuzzer/exif/exif.c:2449
7      #1 0x55a44fdd4b8c in createIfdTableArray /home/ubuntu/Project/ExifFuzzer/exif/exif.c:334
8      #2 0x55a44fdd2929 in main /home/ubuntu/Project/ExifFuzzer/exif/sample_main.c:63
9      #3 0x7f36d5153d8f in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
10     #4 0x7f36d5153e3f in __libc_start_main_impl ../csu/libc-start.c:392
11     #5 0x55a44fdd25e4 in _start (/home/ubuntu/Project/ExifFuzzer/exif/exif_ASan+0x55e4)
12
13 AddressSanitizer can not provide additional info.
14 SUMMARY: AddressSanitizer: SEGV /home/ubuntu/Project/ExifFuzzer/exif/exif.c:2449 in parseIFD
15 ==22287==ABORTING
```

此类测试用例输入时，会使原 PUT 崩溃退出：

```
ubuntu@VM-16-2-ubuntu ~/Project/ExifFuzzer master*
ExifFuzzer-Y0kxE9H0 ❯ exif/exif testcase/mutated_355.jpg
[1]    29597 segmentation fault (core dumped)  exif/exif testcase/mutated_355.jpg
```

```
        if (tag) {
2449        thumbnail_ofs = tag->numData[0];
        }
```

优化点

- 预处理阶段：Seed Selection、Seed Trimming
- 配置更新阶段：每次迭代后更新配置、维护种子池并且不断进化种子
- 评估阶段：Triage 删除重复数据等
- 白盒/灰盒测试：获取更多信息，如覆盖率