4. Exercises

ASI36

2019

1 Magic bytes

Let's consider the following program.

```c
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>

void crash()
{
     raise (SIGSEGV);
}

#define BUFSIZE 1024

int main(int argc, char* argv[])
{
     char inp[BUFSIZE] = { 0 };
     if (argc > 1)
     {

          int f = open(argv[1], O_RDONLY);
          read(f, inp, BUFSIZE);
          int in = atoi(inp);
          if (in == 0xdeadbeef) {
               printf("Aaargh!\n");
               crash();
          }
          printf("You lose\n");
          return 0;
     }
     printf("Please, at least one arg !\n");
     return 0;
}
```

1. Fuzz this program for 5 minutes *with an empty seed*. Did you find a crash?
2. Fuzzers include a fair bit of randomization, maybe you just were not lucky. Now rerun this for 5 more minutes (and maybe once more). Did you find a crash this time?
3. Rewrite the program so that it is semantically equivalent to the original program (no loss of functionality) but so that the fuzzer can reach the buggy path.

2 Hard-to-find events

```c
# include <stdio.h>
# include <stdlib.h>
```

```c
# include <string.h>
# include <unistd.h>
# include <fcntl.h>

int f, *p, *p_alias;
char inp[10], *buf[5];

void bad_func(int *p) {
    p = malloc(sizeof(int));
    free(p); // exit() is missing
    *p = 1;
}

int benign_func(int *p) {
    if (inp[2] == 'F' && inp[3] == 'o' && inp[4] == 'o') {
        free(p);
        return -1;
    }
    return 0;
}

void func() {
    if (inp[1] == 'A') {
        bad_func(p);
        if (inp[2] == 'F' && inp[3] == 'u' && inp[4] == 'z') {
            *p = 1;
        } else {
            p = malloc(sizeof(int));
            p_alias = p;
            if (benign_func(p_alias) == -1) {
                return;
            }
            *p_alias = 1;
            free(p);
        }
    }
}

int main (int argc, char *argv[]) {
    f = open(argv[1], O_RDONLY);
    read(f, inp, 10);

    if (inp[0] == 'U') {
        p = malloc(sizeof(int));
        p_alias = p; // p_alias points to the same area as p
        func();
    }
    return 0;
}
```

1. Run the fuzzer multiple times on the above program. Did you find any crash ? If not, why do you think it is so ?

2. Recompile your program using `AddressSanitizer`, and fuzz it again. Are any crashing inputs found ?