UE SCLAM
Software Security
Lab on Fuzzing techniques

```
Download this tar file from the Moodle web page
Read the instructions to use the fuzzing tools ...
```

Part 1 – Grey box fuzzing with AFL

Exercise 1: handling AFL

Go to the directory `AFL-example` and look at the file example1.c.
This code takes an input file (as a command line argument) containg an integer and use this integer value as an upper bound to access a fix-sized buffer.

Compile it (with AFL instrumentation) and run it using the following command:
```
run-afl.sh example1.c
```

Look at the directory out/crashes containing the input file(s) leading to crashe(s).

Exercise 2: push it to the limits ...

Write a few examples of (small) vulnerable C code of your own and check if AFL is able (or not) to spot the problems. Try do understand and to provide some explanation when it is not the case.

You can consider for instance code examples containing a vulnerability which is not a "direct" stack buffer overflow. It could be for instance:

- a heap-based buffer overflow
- a use-after-free
- a "non-obvious" memory error (e.g., resulting from some artithmetic overflow or unexpected type cast)

Exercise 3: run it on larger example

The code given in directory `JsonParser` implements a Json parser, allowing to parse a Json file and pretty-prints its content on the screen. Json is a human-readable file format for data interchange. Json syntax and examples of Json files can be found here.

The code provided contains several security vulnerabilities, the objective here being to find and understand them using AFL.

Q0. Compile the code provided (using the command make). You should get 2 executable files:

- `jsonparser` (instrumented for AFL)
- `jsonparser_ASAN` (instrumented for AFL and AddressSanitizer) .

Run (without AFL) the jsonparser executable giving as argument the few examples provided in directory regular_input, e.g.: `./jsonparser regular-input/ex1`

Q1. Have a look at the code to briefly understand its structure and try to spot by hand potential vulnerability issues (locate where and how memory is allocated and accessed, possible buffer overflows, arithmetic overflows, etc.).

Q2. Try to (randomly) find crashes running some hand-made examples (you may actually find easily some crashes !). Keep your "winning" inputs (the ones leading to a crash)

Q3. Try now a more effective crash detection technique using AFL using either:

- `./run-json-afl_ASAN.sh` (for the ASAN version)
- ./`run-json-afl.sh` (for the non ASAN one)

Q4 (crash analysis). For each (unique) crash found:

- without using AFL run the program (instrumented with AdSan) with the corresponding crash input
- identify the bug
- is this bug a security vulnerability ? Is it likely to be exploitable ?

Rk: You can also use a debugger (like `gdb`) to get more information about the crashes.

Other possible example : alternatively you can also try to fuzz this small C library provided in the Example 2 of this tarfile (to automatically retrieve the vulnerability you found by hand).

Part 3 – Application to a concrete example

You can **choose** between the two following examples:

a) Grub2 Linux Bootloader

The link below describes two vulnerabilities (leading to a CVE) found in a part of the code of the Grub2 Linux Bootloader.

The objective is to see how much a fuzzing tool like AFL and/or Klee may help to find such vulnerabilities. To do so:

1. Read the explanations provided on the link above ...
2. get the code of the grub_username_get fuction provided in the previous link
3. make it executable:
   - replace "grub" library calls by standart ones (grub_memset by memset, grub_isprint by is_print, grub_printf by printf, etc.)
   - provide a main function allowing to read a string from an input file, store it on a fixed-size buffer and call grub_username_get

When you code can be compiled and executed properly on simple examples, try to fuzz it using AFL in order to find the vulnerabilities mentionned (or other ones !)

b) The user management account example

This is one of the examples used in the beginning of the semester ...