Grenoble INP **UGA**
Master CyberSecurity

Software Security

Lab on Symbolic Execution with PathCrawler

For this lab you should use the online version of PathCrawler:
http://pathcrawler-online.com:8080/

Remarks:

- do not forget to upload again the tar file to run it with PatchCrawler after each change on your machine
- you can put several functions in a same file and fuzz them independently

In the following we will work with examples provided on directory `PathCrawler-Examples`

Exercise 1

Q1. Upload the tar file `Example1.tar` and fuzz function `f` located in file `example1.c`

Take the time you need to understand the various ouptuts produced under the "Test Case" menu, namely:

- the **test cases**, and their corresponding execution paths in the source code;
- the **path predicates** associated to each of these test cases
- the **test input**, i.e. the concrete values found by the solver satisfying each of these path predicates.

Q2. Modify this example in order to add it a *non feasible* execution path, that will be not covered/explored by any test case. Check the result by running this example with PathCrawler.

Exercise 2

The file `example2.c` contains several functions containing code variants accessing a buffer T depending on the value on their input parameter i. Depending on the value of i these functions may (or may not) contain a buffer overflow.

Q1. Execute successively all these functions with PathCrawler and try to understand the results provided (with respect to the bug detection).

Exercise 3

Try examples from your own corresponding to the following scenarios:

1. a function containing a vulnerability induced by an arithmetic overflow
2. a function containing a vulnerabilty not found by PathCrawler because of a non decidable constraints …

3. a function containing a vulnerability inside a loop body
4. a function containing a vulnerability occurring after a "for" loop with a fixed number of iteration (i.e., not depending on the function parameters)
5. a function containing a vulnerability occurring after a "for" loop with a variable number of iteration (i.e., depending on a function parameter).