Assignment 2– Instrumentation for Fuzzing and Symbolic Execution

HSS
Fall 2021

In this assignment, you will work on instrumentation techniques that will help fuzzing and symbolic execution.

Part 1 - Execution Prioritization Pass (50 points)

Write an instrumentation pass that will allow only those executions that *can* effectively test the target function. Specifically, given a program `p` and a function `f`, instrument `p` such that any execution that *cannot* reach `f` or *cannot* be reached from `f` will exit. Consider the following example:

```
1  int func1(int i) {
2    ...
3  }
4  int func2(int i) {
5    exit(-1);
6    ...
7  }
8  int bar(int i) {
9     ...
10    func1(j);
11    ...
12 }
13
14 void foo(int i) {
15   if (i < 0) {
16     exit(-1);
17     func2(i);
18   } else {
19     bar(i);
20   }
21
22 }
23
24 int main(int argc, char **argv) {
25    int h;
26    if (argc > 2) {
27       bar(argc);
28    } else {
29      exit(-1);
30      return -1;
31    }
32    scanf("%d", &h);
33    foo(h);
34    exit(-1);
```

```
35     return 0;
36 }
```

Here, lets say our target function is `bar`. Then text in red shows the `exit` calls that should be inserted by your pass.

Few things to note here:

- We should not exit from `func1` because it is called from our target function `bar` and thus helps in effectively executing `bar`.
- See that in function `foo`, we only exit the true branch of the `if` condition.
- The case is same for function `main`.

Implementation: First, start from the target function and find all the functions reachable from the target function. These functions should not be instrumented. Next, for all the other functions, insert `exit` call into all basic blocks that cannot reach the target function. The skeleton of the pass is present in `https://github.com/purs3lab/hssllvmsetup/tree/main/assignment2-skeleton/FunctionPrioritization`. It has a basic structure, and there are some TODOs that need to be filled. Your task is to complete them.

Part 2 - LimitLoopIterations Pass (35 points)

Here, write an instrumentation pass that will limit the maximum number of loop iterations to a given number.

Implementation: First, create a local variable (loop counter) for each loop. Second, initialize the loop counter to 0 at the beginning of the loop. Third, increment loop counter for every iteration. Finally, check the loop counter to be less than the given maximum number. If yes, continue else, exit the loop. The skeleton of the pass is present in `https://github.com/purs3lab/hssllvmsetup/tree/main/assignment2-skeleton/LimitLoopIterations`. It has a basic structure, and there are some TODOs that need to be filled. Your task is to complete them.

Part 3 - Implications of LimitLoopIterations pass (15 points)

Is LimitLoopIteration pass sound? i.e., can the instrumented program result in a crash that is not possible in the original function? (Yes or No). Justify your answer with an example. Specifically, if yes, provide a proof argument; otherwise, provide a program where the instrumentation will make the program unsafe, i.e., cause the instrumented program to crash on certain inputs.

1 Submission

Create a `tar.gz` with the completed `assignment2-skeleton` folder and pdf containing answer for the part 3.

Extracted folder should have the following structure:

```
extracted_folder:
 -assignment2-skeleton
 -solution.pdf
```

We should be able to build passes by following the below instructions:

```
cd extracted_folder/assignment2-skeleton
mkdir build
cd build
cmake ..
```

Submit the `tar.gz` to brightspace.