COM S 413/513: Homework 9 [Project]
Analyzing software changes and versions

November 2, 2023

Learning Objectives:

In this homework, you will

1. gain experiences with open source program analysis tools LLVM and MVICFG
2. understand the challenges of analyzing program changes and versions
3. develop extensions and contribute to existing program analysis tools

Instructions:

1. Total points: 29 pt
2. Early Deadline: Nov 15 (Wed) 11:59PM
3. Deadline: Nov 17 (Fri) 11:59PM
4. This homework is accomplished by the team of 2 students.
5. Only one team-member need to make a submission on Canvas. Each submission must have the team-members names.
6. How to submit:
   - Submit 2 files, a PDF and a Zip. Don't bundle the PDF within the Zip file.
   - PDF should contain the written answers and screenshots.
   - Zip should contain all the code should have a README to explain which files/folders you have modified and how to run the modified version to produce the answers.

1 Description

In this homework, you are going to work with one of the prominent program analysis platforms LLVM and one of the frontier static analysis tools, called multi-version control flow graphs MVICFG. Here are the set of instructions that can guide you through the work.

1. Read the paper (you can focus on only Sections 1-4): "Patch Verification via Multiversion Interprocedural Control Flow Graphs".
2. Get access to the MVICFG repo: https://github.com/iowastateuniversity-programanalysis/hydrogen.
3. Consider the test cases given below:

(a) Initial Version:

```c
#include <stdio.h>
int main() {
  float a, b, area;
  scanf("%f %f", &a, &b);
  area = .5 * a * b;
  if (area >= 1)
    printf("The area is %.4f.", area);
  else
    printf("The area is %.4f.", -area);
  return 0;
}
```

(b) Second Version:

```c
#include <stdio.h>
int main() {
  float a, b, area;
  scanf("%f %f", &a, &b);
  area = .5 * a * b;
  if (area >= 0)
    printf("The area is %.4f.", area);
  else
    printf("The area is %.4f.", -area);
  return 0;
}
```

4. Follow the README of the MVICFG tool and install the docker and get the MVICFG running with the test case given above.

5. Extend the MVICFG framework to report the paths that will be impacted by the changes:

   - Which path(s) in old version are affected?
   - Which path(s) in the new version are added?
   - Report the change in paths using `instructionID` from `Graph_Instruction` class. For example for the Buggy and Correct versions present in test programs, the output would be as follows:

```
[root@9e1663fc6239 BuildNinja]#  ./Hydrogen.out ../TestPrograms/Buggy/ProgV1.bc ../TestPrograms/Correct/ProgV2.bc :: ../TestPrograms/Buggy/Prog.c :: ../TestPrograms/Correct/Prog.c
Newly removed paths:
Path 1: 48 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 47 49
Path 2: 48 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 34 35 36 37 38 39 40 41 42 43 44 45 46 47 49
Number of newly removed paths: 2
Newly added paths:
Path 1: 48 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 55 56 57 58 59 60 61 62 63 64 65 66 47 49
Path 2: 48 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 67 68 69 70 71 72 73 74 75 76 77 78 79 47 49
Number of newly added paths: 2
Finished Building MVICFG in 3ms
```

6. Can this change impact the output of the code?

2 Deliverables

1. (5 pt) In the PDF, please submit the screenshot to show that you can successfully generate an MVICFG for the test programs (Q4).

2. (9 pt) In the PDF, please submit the answers of Q5 and Q6 and provide screenshots to show that your code can compute Q5.

3. (15 pt) Please submit the source code as well as the binary of your modified Hydrogen tool in the Zip submission. As appendix, please include the diff generated using `diff -U 0` for all the files you modified.