Project Milestone Report
Memory Safety Bug Hunting with BAP

Tiffany Bao (tiffanybao@cmu.edu)
Daming Chen (ddchen@cmu.edu)
Rijnard van Tonder (rvt@cmu.edu)

http://ddcc.github.io/15745-s15-project

15-745 S15 Optimizing Compilers
April 15, 2015

Major Changes: No, there have not been any major changes to the goals or implementation of our project.

What You Have Accomplished So Far: We have accomplished the following achievements, which are in line with our 75% goals toward our 100% goals

1) We implemented an intraprocedural analysis that tracks data flow dependencies between instructions. It is based on the idea of use-def chains obtained from a reaching definition analysis. Linking these use-def chains produces a data flow dependency graph (see website for specific results).

2) We implemented an ABI interface under BAP which gives allows us to query the registers associated with arguments and return values for any given function. This is important for checking the return value of potentially memory unsafe functions, such as *strcpy, malloc,* and *memcpy.*

Meeting Your Milestone: Yes, we have met the major 75% milestone goals, as enumerated above; namely producing the data dependency graph and developing the infrastructure for detecting unchecked allocations. Although we are currently in the process of implementing the detection of unchecked allocations, this should be a straightforward exercise. The delay was largely due to synchronizing individual components of our implementation between group members.

Surprises: There have been no major surprises, although we have run into a number of snags. In particular, we have developed a better understanding of the advantages and limitations of the current BAP framework. For instance, we needed to implement a custom ABI interface for inferring the registers associated with arguments and return values for library functions, since they are not actually located in the PLT and cannot be directly analyzed. Another snag surfaced when implementing a reaching definitions analysis in the BAP intermediate representation, in that it may produce basic blocks containing conditional statements from a given disassembly.

Revised Schedule: Describe what each member of your group is going to accomplish between now and May 2nd to successfully complete your project. If you are stuck on anything, please let us know. (In fact, if you are stuck, please send us email immediately rather than waiting for us to read this report.)

- Rijnard: Continue working on 100% goal to “Implement basic intraprocedural detection of buffer overflows in stack”. By May 2nd, also produce a suite of test cases where this detection works (and doesn’t work).
- Dominic: Continue cleaning up the dataflow framework implementation, and begin implementing intraprocedural detection of heap buffer overflows starting April 20th.
- Tiffany: Complete detection of unchecked memory allocations.

Resources Needed: Yes, we have all the resources we need.