Sibling-First Recursive Graph Drawing for Java Bytecode

Md. Jawaherul Alam, Michael T. Goodrich, and Timothy Johnson

Donald Bren School of Information and Computer Sciences
University of California, Irvine

(Full paper)

Overview

We describe a tool, the JVM abstracting abstract machine (Jaam) Visualizer, or "J-Viz" for short, which is intended for use by security analysts to find algorithmic vulnerabilities through the exploration of graphs derived from Java bytecode.

Our tool accomplishes this by constructing a graph representing the control flow of the program and displaying it interactively using a canonical ordering we call the *sibling-first recursive ordering*.

We use the following algorithm:

1. Construct the control flow graph using 1-CFA.
2. Draw a spanning tree using the sibling-first recursive ordering, putting each branch in a separate lane.
3. Draw edges that are not in spanning tree.
4. Highlight suspicious areas of the graph.
5. Group nodes hierarchically.

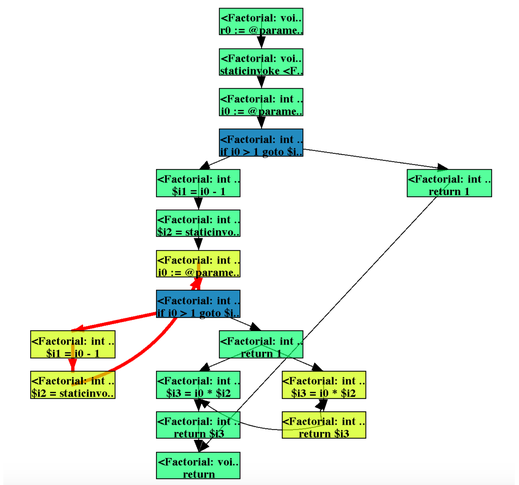
Control Flow Analysis (CFA)

Our desired properties for a control flow graph are:

- Soundness: Including every possible execution path
- Precision: Excluding impossible execution paths

The *k*-CFA hierarchy [1] provides a range of algorithms for constructing a sound control flow graph of a program, trading off increased precision against increased size of the graph produced.

For a given *k*, each state stores a single instruction and the most recent *k* function calls.



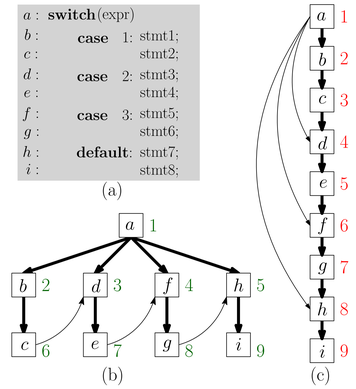
A recursive factorial function analyzed using 1-CFA. The two highlighted nodes represent the same line of code, but are called in different contexts.

The SFR ordering

Algorithm DFS(v):
Number v as vertex n
Increment n
for each directed edge (v, w) **do**
  if w is not numbered **then**
    Identify w as a child of v
    DFS(w)

Algorithm SFR(v):
for each directed edge (v, w) **do**
  if w is not numbered **then**
    Identify w as a child of v
    Number w as vertex n
    Increment n
for each directed edge (v, w) **do**
  if w is a child of v **then**
    SFR(w)

The DFS and SFR algorithms

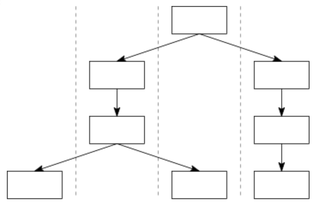


(a) A code segment with a switch statement
(b) Tree based on SFR ordering
(c) Tree based on DFS ordering

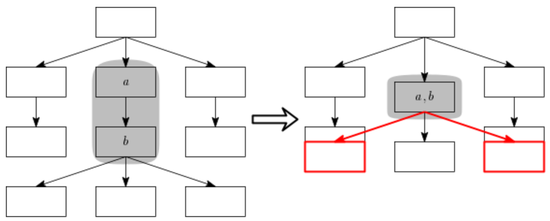
Tree layout

We require that a node can only be drawn directly below another node if it is a descendant in the SFR tree. This divides our graph into lanes, as in the figure below.

This preserves the user's mental model by maintaining the relative positioning between nodes, even when arbitrary connected subsets of nodes are collapsed or expanded.



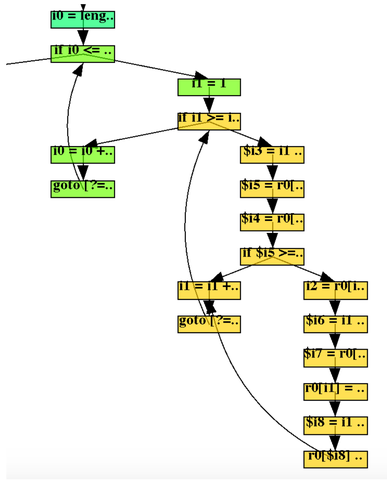
Neglecting this condition can produce collisions, such as in the following example where nodes *a* and *b* are collapsed to a single node.



Final processing

Edges not in our SFR tree are drawn as curves if they point upward, and straight lines if they point downward.

We then highlight the nodes based on the number of nested loops containing them, using an algorithm from [2].



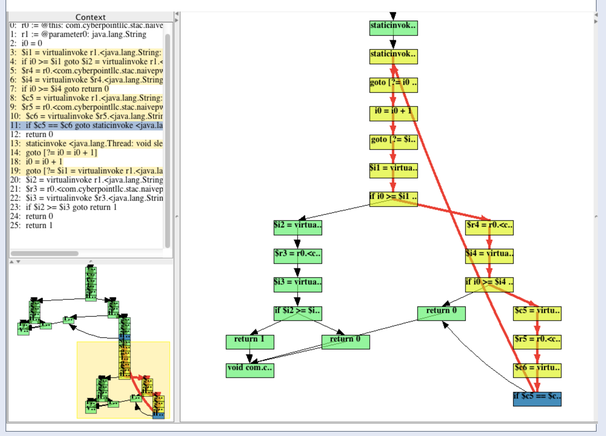
Lastly, we collapse groups of nodes in three ways:

- Chains
- Methods
- Chains of methods

Case study 1

This program checks a password, one character at a time. But it exits as soon as it finds the first incorrect character, allowing for a timing attack that can determine the password with only a handful of guesses.

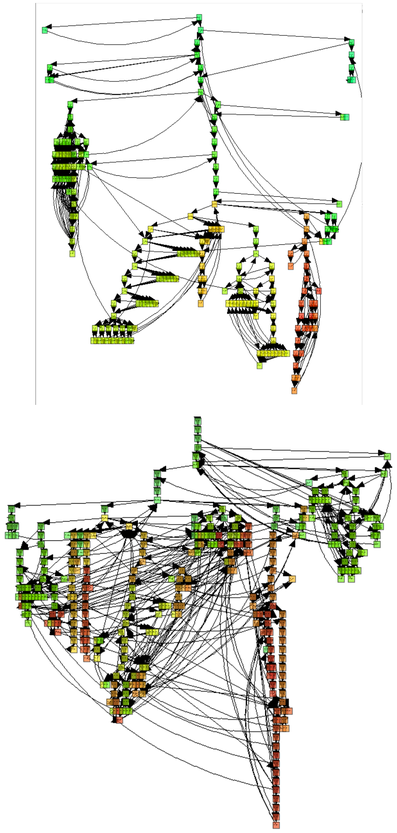
This can be found quickly because our graph shows all of the possible exits from the loop.



Case study 2

The next two programs were provided as part of the DARPA challenge. They are much larger, containing thousands of lines of code.

But in each one, the vulnerability appears in the red portion of the graph, allowing the human analyst to check only a small fraction of the code.



References

1. Shivers, O.G.: Control-Flow Analysis of Higher-Order Languages. Ph.D. thesis, Carnegie Mellon University (1991)
2. Wei, T., Mao, J., Zou, W., Chen, Y.: A new algorithm for identifying loops in decompilation. In: Static Analysis, pp. 170–183. Springer (2007)

Acknowledgments

This article reports on work supported by the Defense Advanced Research Projects Agency under agreement no. AFRL FA8750-15-2- 0092. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This work was also supported in part by the U.S. National Science Foundation under grants 1228639 and 1526631. In addition, we would like to thank David Eppstein, Matthew Might, William Byrd, Michael Adams, and Guannan Wei for helpful discussions regarding the topics of this paper.

Email contacts: {alamm1,goodrich,tujohnso}@uci.edu