Nondeterministic Results from DataFlowSolver Flow Insensitive.

New issue

Closed

Hello,

I have recently been using Flowdroid and noticed what I think could be an issue in the option -ds FLOWINSENSITIVE.

Description

Originally, I was investigating the Flowdroid commit but I have confirmed that a similar behaviour is present in the more recent commit When running the command line Flowdroid with the command java -jar ../soot-infoflow-cmd-jar-with-dependencies.jar -a 'apk' -s .... -p ... -ds FLOWINSENSITIVE where the apk is produced from this (this project is actually AnonymousClass1 from Droidbench, modified slightly to build with Gradle and should produce an equivalent APK to the original) - I receive nondeterministic results from Flowdroid. I have observered this behaviour across multiple machines, on both Mac and Ubuntu (and in Docker containers).

Results

Running Flowdroid (on Ubuntu 22.04) 20 times with the above configuration results in

- 8 runs output finding 1 leak from getLatitude() -> log.i() in onResume()
- 11 runs output finding 1 leak from getLongitude() -> log.i() in onResume()
- 1 run outputs finding 2 leaks (both the previous flows together)

Thoughts

All of these flows are True Positives in the AnonymousClass1 project, I don't think this is the expected behaviour but I could be wrong, is this expected?

I cannot pinpoint exactly what is causing this behaviour because I have only a basic understanding of the solver code. From my investigation, whenever I modified Flowdroid's code (in the older commit, have not tried this on the later commit) to ignore using threads the results then became deterministic (20 runs & 20 outputs that were all getLongitude() -> log.i() in onResume()). This is, naturally, expected because using only 1 thread makes the order we propagate taint consistent but I am struggling to even understand how this could be the case. I would think that the order we process edges does not matter, we should still end up with the same flows at the end?

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

Notifications Customize

Subscribe

You're not receiving notifications from this thread.

Member

FlowDroid should deliver deterministic results even in the multi-threaded case. However, we know that there is a strange bug somewhere. **@timll** has been investigating a flaky test case that sometimes returns one result (which is correct), and sometimes returns no result at all. Unfortunately, we have not yet fully understood why this happens.

What we know so far:

- The problem also happens in the context-sensitive variant of the FastSolver.
- If we synchronize the path reconstruction, the problem still shows. That means the different must be in the input to the path reconstructor, i.e., the taint graph itself, i.e., the output of the IFDS solver.
- The taints that arrive at the sink are indeed different, but that is expected.

Some variability in the taints is expected. Each abstraction records its predecessor and a set of neighbors, each of which also has a predecessor. That means you can have equivalent taint graphs by moving between predecessors of the taint at the sink and predecessors of neighbors of that sink. It's conceptually the same, but the structure (i.e., what becomes "main" abstraction and what becomes a neighbor) is timing-dependent.

This makes debugging the analysis harder, because we can't just check for graph equivalence. We have different graphs, and sometimes traversing one of these graphs leads to fewer paths between sinks and sources (reconstruction is bottom-up). We're still thinking about a good way to really understand what the differences are (aside from the expected re-organization I mentioned). Only then, we can look into why these differences occur.

If we reduce the thread count to one, we deterministically get the same taint graph. However, this graph (as you have also found) seems to be correct consistently, i.e., doesn't help us with understanding the problem at hand.

If you have a new APK file (which is not yet part of our test suite) that exhibits a similar behavior, we can try to look into the problem again based on your APK.

Merged

StevenArzt closed this as completed