Android Intent ICC

- Android applications are composed of four types of components that can communicate with one another.

- Intents and Intent Filters are the main inter-component communication (ICC) mechanism. Components can communicate within single applications or between different applications.

- We developed Epicc, a static analysis tool for Android ICC that determines possible component interaction, within and between applications. It is available at: [http://siis.cse.psu.edu/epicc/](http://siis.cse.psu.edu/epicc/)

IDE Data Flow Analysis

- The IDE (Interprocedural Distributive Environment) framework is used for interprocedural data flow analysis problems. A program is represented as a supergraph.

- Example ICC code

- Variables have values associated with them ("environment"). Environment transformers represent the influence of statements on these values.

- A data flow IDE problem is such that environment transformers are distributive, i.e.:

  \[ t(A \cup B) = t(A) \cup t(B) \]

- IDE problems can be solved efficiently using existing algorithms (see "Precise Interprocedural Dataflow Analysis with Applications to Constant Propagation" by Sagiv et al.).

An IDE Model for ICC

- Example for Bundle: We define intermediate Bundle values

  \[ \{(E_1, O_1), \ldots, (E_n, O_n)\} \]

  They represent the value of a Bundle, taking into account any possible Bundle composition (using method `putAll(Bundle)`).

  We also define Bundle transformers

  \[ \beta^b = (n_1^{n_1}, n_1^{n_1}, c1, \Theta_1), \ldots, (n_m^{n_m}, n_m^{n_m}, c_m, \Theta_m) \]

  They represent the influence of program statement on intermediate Bundle values.

  ![Pointwise Bundle environment transformers](chart.png)

  - In a first step, we find intermediate Bundle values using the IDE solver. In a second step, we resolve all composition relationships and find final Bundle values.
  
  - The models for Intents and Intent Filters are similar, with more data fields.

Implementations and Results

- Android applications are retargeted using our Dare tool (available at [http://siis.cse.psu.edu/dare/](http://siis.cse.psu.edu/dare/)). Epicc, our static ICC analysis tool, is implemented on top of the Soot framework using the Heros solver (https://github.com/Sable).

  - We ran Epicc on 1,200 applications. We were able to find precise ICC specifications in over 93% of cases. Computation took on average 113 seconds per application, making it scalable to a large number of applications. Precision and computation time were very variable between applications.

  - A first study of ICC showed that ICC specifications are relatively narrow, with most ICC objects having a single possible type. Also, key-value pairs are widely used to communicate data between components.

  - We performed an ICC vulnerability study of our application corpus using Epicc and ComDroid (introduced in "Analyzing Inter-Application Communication in Android" by Chin et al.). We looked for potential vulnerabilities in entry points (e.g., public components) and in exit points (e.g., implicit Intents). We found that Epicc is more precise, flagging 24% fewer code locations than ComDroid.

Publications