Categorizing Static Analysis Results

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Linghui Luo (Paderborn University) and Prof. Eric Bodden (Paderborn University & Fraunhofer IEM)

linghui.luo@uni-paderborn.de @LinghuiLuo

Motivation

Problem:

- Static analysis tools implement only may-analysis.
- Reported warnings are inconclusive.
- Developers are forced to identify and prioritize excessive warnings manually.

Observation:

There are different reasons for may-analysis results.

Reason 1 – Program Input

```
String secret = source();
String input = bufferedReader.readLine();
if(input.contains(“address“)){
        print(secret); // depends on input
}
```

Example 1: Definite data leak under certain user inputs

Reason 2 – Environment Configuration

```
String secret = source();
boolean a = confi.getOption(“A“).isOn();
if(a){
        print(secret); // depends on option A
}
```

Example 2: Definite data leak under certain configuration

Reason 3 – Analysis Limitation

```
String secret = source();
if(isPrimeNumber(5754853343)){
        print(secret); // branch uncertain
}
```

Example 3: Possible data leak

Goal

Hypothesis: Warnings reported by static analysis tools can be categorized according to those three reasons. Based on this categorization, warnings can be optimized or prioritized.

Research Questions:

RQ1: Does such a classification exist for the warnings according to the categories we defined?

RQ2: To what extent are the categories mutally exclusive?

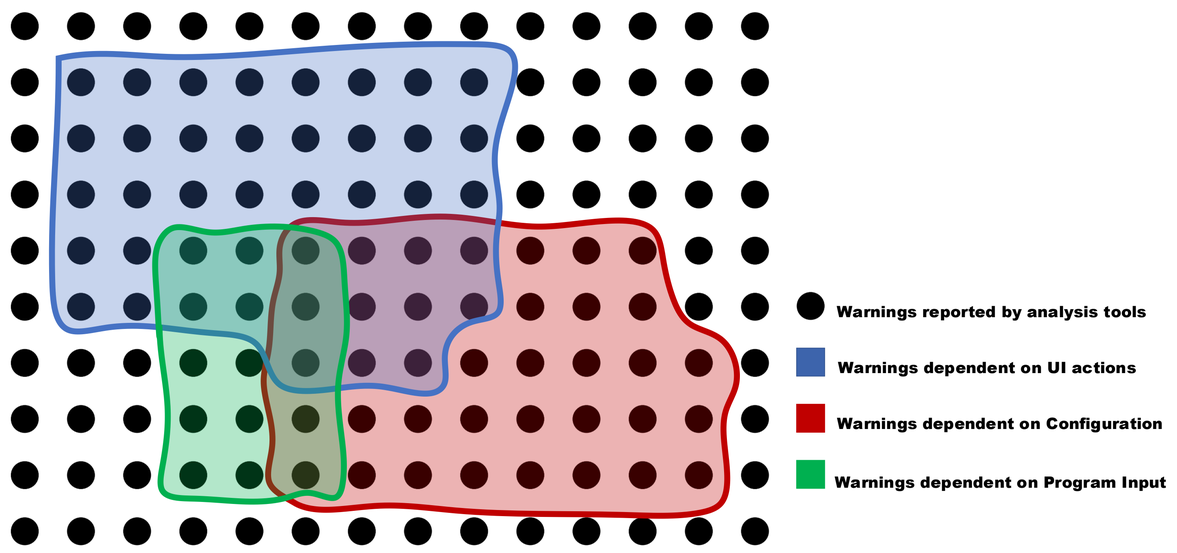

Figure 1. Categorization of warnings reported by static analysis tools

Approach

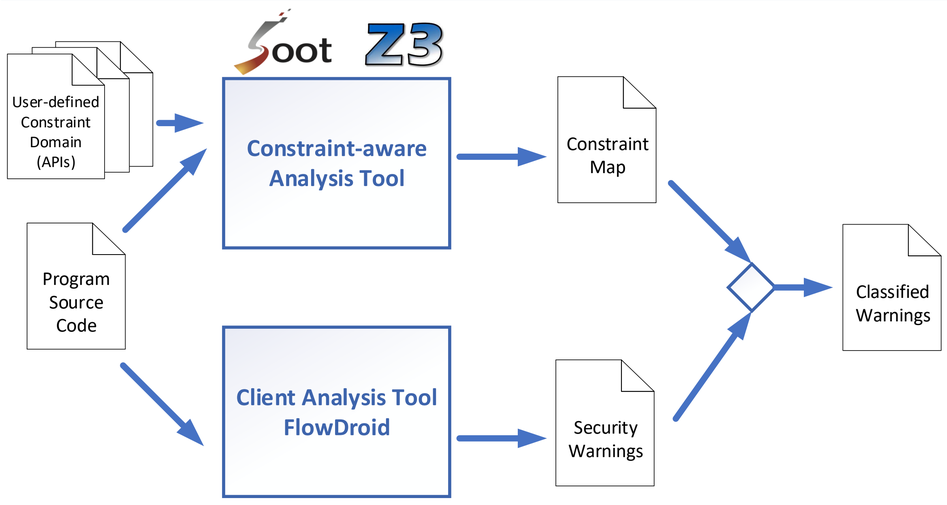

Figure 2. Workflow of classifying the security leaks reported by FlowDroid

```
public class LeakyApp extends Activity {
  private TelephonyManager tm;
  @Override
  protected void onCreate(Bundle bundle) {
    ...
    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        String model= Build.MODEL;
        if (model.equals("HTC Desire")){
          String deviceID= tm.getDeviceId();
          send(deviceID);
        }
      });
  }
  private void send(String deviceID){
    int sdk = Build.VERSION.SDK_INT;
    if (sdk < 25) {
      SmsManager sms = SmsManager.getDefault();
      sms.sendTextMessage("+4912", null, deviceID, null, null);
    }
  }
}
```

CLICK
CLICK
Source CLICK ^ im(MODEL)
CLICK ^ im(MODEL)
CLICK ^ im(MODEL) ^ (SDK < 25)
CLICK ^ im(MODEL)
CLICK ^ im(MODEL)
CLICK ^ im(MODEL) ^ (SDK < 25)
Sink CLICK ^ im(MODEL) ^ (SDK < 25)

Figure 3. Example of reasoning reported security leak

Results

Empirical Study:

- Implemented a constraint-aware analysis tool based on Soot, VASCO, Z3 and Boomerang, supporting analysis for Java and Android applications.
- Extended FlowDroid with this tool and analyzed 421 real apps from app stores, classified 6924 leaks.