EDGEMINER: AUTOMATICALLY DETECTING IMPLICIT CONTROL FLOW TRANSITIONS THROUGH THE ANDROID FRAMEWORK

Yinzhi Cao, Yanick Fratantonio, Antonio Bianchi, Manuel Egele, Christopher Kruegel, Giovanni Vigna, and Yan Chen

Columbia, UCSB, BU, NU
Introduction

- Static analysis has been used for security and privacy.
- Many analyses rely on the control flow graph.
- Challenge in Android: the framework is 8.6 million lines of code.
- Ignoring the framework -> incorrect control flow graph of Android apps.
  - Common cause for imprecision: “callbacks”, e.g., onClick
A Motivating Example

```java
public class MainClass {
  static String url;
  public static void main(String[] args) {
    MalComparator mal = new MalComparator();
    MainClass.value = 42;
    Collections.sort(list, mal);
    sendToInternet(MainClass.value);
  }
}
```

```java
public class MalComparator implements Comparable<Object> {
  public int compare(Object arg0, Object arg1) {
    MainClass.value = GPSCoords;
    return 0;
  }
}
```

Privacy leakage is up to the value of MainClass.value.
Existing Approaches

- Whole program analysis
  - State explosion
  - Pushing existing static analysis systems to their limits
  - Redundant Efforts (slow-down of static analysis)

- Summary-based analysis
  - Manual summarization: impossible due to the high volume of callbacks
  - Heuristic summarization: inaccurate
EdgeMiner: Usage Scenario

- Summarize framework: list of registration-callback pairs
Concepts

- **Callback**
  - Necessary condition: a framework method that can be overridden by an application method

- **Registration**
  - Necessary condition: a framework method that is invokable from the application space
1 public class Collections {
2     public static void sort(List list, Comparator comparator) {
3         ...
4         comparator.compare(e1, e2);
5     }
6 }

An object with the callback
System Architecture

Android Framework → Pre-processing → EdgeMiner
  - IR Form
  - Class Hierarchy
  - Call Graph
  - Metainfo

Potential Callbacks Callsites → Backward Dataflow Analysis
  - Callback 1
  - Callback 2
  - ...

Framework Summary
Implementation

- ROP intermediate representation (IR)
  - Well-suited for static analysis
  - In SSA form
  - Integral part of Android SDK
- EdgeMiner
  - Built on top of ROP
  - Performs backward dataflow analysis
  - Summarizes implicit control flows through framework
Results

- Number of registrations and callbacks

<table>
<thead>
<tr>
<th>Android Version</th>
<th># Registrations</th>
<th># Callbacks</th>
<th># Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3 (API 10)</td>
<td>10,998</td>
<td>11,044</td>
<td>1,926,543</td>
</tr>
<tr>
<td>3.0 (API 11)</td>
<td>12,019</td>
<td>13,391</td>
<td>2,606,763</td>
</tr>
<tr>
<td>4.2 (API 17)</td>
<td>21,388</td>
<td>19,647</td>
<td>5,125,472</td>
</tr>
</tbody>
</table>

- Results for Android 4.2 at

http://edgeminер.org
Accuracy

- False negative
  - Compare with dynamic approach
    - Incomplete but accurate
  - 8,195 randomly selected applications
  - 6,906 registration-callback pairs
  - EdgeMiner finds all pairs

- False positive
  - Manual inspection
  - Eight false positives out of 200 pairs
Improving FlowDroid

- Integration with FlowDroid
  - Synchronous callbacks: inline invocation
    - E.g., Collections.sort and Comparator.compare
  - Asynchronous callbacks: delayed invocation
    - E.g., setOnClickListener and onClick

<table>
<thead>
<tr>
<th>Pattern</th>
<th># FlowDroid</th>
<th># EdgeMiner</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Listener</em></td>
<td>155</td>
<td>576</td>
</tr>
<tr>
<td><em>Callback</em></td>
<td>19</td>
<td>319</td>
</tr>
<tr>
<td><em>On</em></td>
<td>3</td>
<td>509</td>
</tr>
<tr>
<td>None of the above</td>
<td>4</td>
<td>18,243</td>
</tr>
<tr>
<td>Total</td>
<td>181</td>
<td>19,647</td>
</tr>
</tbody>
</table>
Improving FlowDroid – Accuracy

<table>
<thead>
<tr>
<th>Tool</th>
<th>FlowDroid</th>
<th>FlowDroid + EdgeMiner</th>
</tr>
</thead>
<tbody>
<tr>
<td># Apps with ≥ 1 privacy leak</td>
<td>285</td>
<td>294 (285 + 9)</td>
</tr>
<tr>
<td># Privacy leaks (in total)</td>
<td>46,586</td>
<td>51,418</td>
</tr>
<tr>
<td># Apps timeout</td>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>

- Run 9 new apps in TaintDroid
  - 4 verified, 2 crash, and 3 no leak
- Incorrect call graph -> missed privacy leaks
- Performance
  - 34.7 seconds one-time loading
  - Only 1.85% overhead added to FlowDroid
Conclusion

- EdgeMiner summarizes implicit control flows in Android framework
- EdgeMiner identifies registration-callback pairs through backward data flow analysis
- Results improve state-of-the-art static Android analyses
  - FlowDroid detected 9 additional apps with leaks
Thank you!

Questions?

Results are available at http://www.edgeminer.org