Contextual Policy Enforcement in Android Applications with Permission Event Graphs

Kevin Chen, Noah Johnson, Vijay D’Silva, Shuaifu Dai, Kyle MacNamara, Tom Magrino, Edward Wu, Martin Rinard*, and Dawn Song

University of California, Berkeley
*Massachusetts Institute of Technology
Android

Figure: Google Play App Market Growth
Android Malware

"2577% growth over 2012"  -Cisco Security Report 2013

"Android malware cases to hit 1 million in 2013"
-Trend Micro Annual Threat Report

Figure: Google Play App Market Growth
Android Malware Detection

Our approach
Undetected Malware Example

Do you want to install this application?

Allow this application to:

⚠️ Storage
modify/delete SD card contents

⚠️ Hardware controls
record audio

Install  Cancel
User Intended Policy

"The recording can only be started by clicking the REC button, and it will be stopped when the user clicks the STOP button."
A representation that summarizes the event dependencies and their API/permission level behaviors (*The Permission Event Graph*), and a policy language based on that.
"The recording can only be started by clicking the REC button, and it will be stopped when the user clicks the STOP button."
PEG: States

0

initialization sub-graph

1

2

finalization sub-graph
PEG: States

State $s$: $\{\text{true}, \text{false}\} \wedge \text{ModeVar}$

Predicate abstraction of event states.

e.g.

- `Button.registered`,
- `Activity.foreground`,
- `API.called`
PEG: Transitions
PEG: Labels

initialization sub-graph

onCreate

onResume

startService

Start-Recording

Stop-Recording

onCreate

onResume

startService

finalization sub-graph
Sound Recorder: The Good Part

0
---
initialization
sub-graph

onCreate

onResume
startService

1
---

onClick
Stop-Recording

onClick
Start-Recording

2

finalization
sub-graph
The Complete PEG

Recorder Activity

0

onCreate

initialization

onResume

startService

1

Recorder Activity

2

Recorder Service

3

onCreate

4

onStart

Timer.run

Start-Recording

5

Timer.run

Stop-Recording

6

finalization
PEG: Context of the Benign Use

Recorder Activity

0

initialization

1

REC.onClick
Start-Recording

2

finalization

3

Recorder Service

4

finalization

5

6
"The recording can only be started by clicking the REC button, and it will be stopped when the user clicks the STOP button."

$(\neg\text{Start-Recording} \cup \text{REC.onClick}) \land (\text{Stop-Recording} \iff \text{STOP.onClick})$
Approach Overview

Abstraction Phase

Permission Event Graph

Verification Phase

Conformance or counter-examples

Approach Overview

Apps

Abstraction Phase

Permission Event Graph

Verification Phase

Conformance or counter-examples

Policies

\[ (\neg \text{Start-Recording} \lor \text{REC.onClick}) \land (\text{Stop-Recording} \iff \text{STOP.onClick}) \]
Case Study: Geotag

"Mark location of your photos"
Case Study: Geotag

"Mark location of your photos"

Access-GPS \( \checkmark \) Locate.onClick
Case Study: Geotag

"Mark location of your photos"

\[ \neg \text{Access-GPS} \cup \text{Locate.onClick} \]

\[ \neg \text{Access-GPS} \cup \left( \lor \text{Locate.onClick} \land \text{AdTask.onPreExecute} \right) \]
Case Study: SMS Replicator Secret

A spyware that secretly forwards every SMS to another number.
Case Study: SMS Replicator Secret

A spyware that secretly forwards every SMS to another number.

- `Send-SMS Ù Button.onClick`
LIFE OF PEG

Abstraction Phase → Verification Phase

Permission Event Graph

Conformance or counter-examples

Apps

Policies

(¬Start-Recording U REC.onClick)
∧ (Stop-Recording ⇔ STOP.onClick)
Abstraction
Abstraction: The Android Trinity

Application

Application Code

System Code

Event System  Sys. Libraries
Abstraction: The Android Trinity

- **Application**
- **Event System**
- **Sys. Libraries**
- **States**

Application Code

System Code

Call event handler
Abstraction: The Android Trinity

Application

Event System

Sys. Libraries

Call event handler

Call API

States

Application Code

System Code
Abstraction: The Android Trinity

Diagram:

- Application
  - Call event handler
  - Call API

- Event System
  - Register handler

- Sys. Libraries

States
Abstraction: The Algorithm

Abstract Interpretation on \((P(\text{aState}) \times P(\text{API}) \times P(\text{aState}))\). Interprocedural CFG with a partially context sensitive points-to analysis

- 1200+ APIs
- 63 Kinds of Events
- Event Semantics Engine
- API Semantics Engine
- Application Analyzer
- SrcStates
  - Partial Valuation of the vars in ModeVar
- (SrcStates, DstStates) Pairs
  - Partial Valuation of the vars in ModeVar
- Event Handler
- Summary-based
- 63 Kinds of Events
- Summary-based
- 1200+ APIs

PEG
Verification: BFS for Conformance

Policies:

\[ \neg \text{Start-Recording} \lor \text{REC.onClick} \]
\[ \land (\text{Stop-Recording} \iff \text{STOP.onClick}) \]

Permission Event Graph

Verification Phase

Conformance or counter-examples
Evaluation: PEG size (# states, CDF)

* 269 applications. Binary code sizes vary from 4KB to 6MB
Evaluation: Abstraction Time (CDF)
Evaluation: Verification Time (CDF)

* Always terminate within 3.6 hours
Conclusion

- Permission event graph: event-dependencies and their API/permission-level behaviors

- Contextual policies based on event sequences enable the detection and analysis of complex malicious behaviors (user-oriented security)

- Enriches the set of detection techniques used by security analysts
Questions ?
Kevin Chen <kevinchn@cs.berkeley.edu>
Backup Slides
Backup Slides
Native Code

- Known
  - The API Semantics Engine

- Unknown
  - Do NOT support
Rewriting

- **Barriers for static analysis:**
  - Unresolved Reflection: 15.5%
  - Unresolved Dynamic dispatching: 42.9%

- **Solutions:**
  - Insert runtime checks
  - More in the paper
API Frequency in 95,000 Apps
## Specification Constructs

\[ \neg \text{Start-Recording} \cup \text{REC.onClick} \]
\[ \land (\text{Stop-Recording} \iff \text{STOP.onClick}) \]

<table>
<thead>
<tr>
<th>Information Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>System status variables (Mode variables)</td>
<td>STOPButton.registered, MyActivity.inBackground</td>
</tr>
<tr>
<td>System APIs and their arguments</td>
<td>android.location.Location: double getLatitude(),</td>
</tr>
<tr>
<td></td>
<td>&quot;content://com.android.contacts/contacts&quot;</td>
</tr>
<tr>
<td>Permissions</td>
<td>&quot;android.permission INTERNET&quot;</td>
</tr>
</tbody>
</table>
Specification Checker Interface

Bounded BFS for conformance analysis
Write the specification FSM using the following interfaces:

```java
public int getStateId();
public void restoreFromStateId(int id);
public ListenerResult stateListener(ModelState state);
public ListenerResult actionListener(EventModality action);
public ListenerResult methodListener(PathItemMethod method);
```
# Evaluation

<table>
<thead>
<tr>
<th>Sensitive Operation</th>
<th>Malicious</th>
<th>Benign</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NoUI</td>
<td>Total</td>
</tr>
<tr>
<td>GPS</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>SD card</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>SMS</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>
Applications

• Usage scenarios:
  ○ Extra semantics-based filter for malware screening
  ○ Diagnostic tool for security analysts
  ○ Fine-grained information about permission use for the user
• emphasis: a new representation to detect a specific category of malicious behaviors. ... in addition to the traditional Android malware detection techniques.
• 2/18 for the specific set of behaviors
• no longer binary,
  ○ should be a continues answer.
  ○ Geotag