CuriousDroid:

Automated User Interface Interaction for Android Application Analysis Sandboxes

Patrick Carter, Collin Mulliner, Martina Lindorfer, William Robertson, Engin Kirda

02/23/2016
• Most popular mobile OS
  – 84.7% of 2015 Q3 mobile device sales
  – 48.6% 2014 total device sales

* Gartner
Android Malware

- Apps appear normal to user
  - Malicious functionality hidden from user

- Russian banking malware
  - Send SMS
  - Capture images
  - Record Audio
  - Track GPS
  - Address book
  - List of recent calls
  - Etc.
Android Security

• Google Play Store
  – Google Bouncer
  – Doesn’t protect against 3rd party sources

• Anti-Malware applications
  – Generally looking for malware signatures

• User defenses
  – Permissions
  – Avoid 3rd party sources

• A more robust malware analysis is necessary
Malware Analysis

• Static analysis
  – Safely approximates all behaviors
  – False positives more likely

• Dynamic Analysis
  – High-fidelity results
  – Coverage is hard!
Android Dynamic Malware Analysis

• Coverage is even harder!
  – All Android apps are event/GUI based

• Exercising application UIs is imperative for increased coverage
  – Cannot drive execution of application forward without exercising the UI
Android Test Generation

• De facto tools for exercising application UIs are the Monkey and MonkeyRunner (Google)
  – Monkey: fuzzer
  – MonkeyRunner: requires source code and knowledge of application to build test applications

• Other exercisers require either source code (instrumentation) or take a long time to generate exploration paths
CuriousDroid

- Android UI stimulation for malware sandbox environments
  - Fully automated: No human in loop
  - No source code or prior knowledge of application is necessary
  - Runs on devices in addition to emulators
    - Needs root

- Emulates human interactions
Dynamic Dalvik Instrumentation

- Method for injecting arbitrary code into a running process
  - Add additional class files to Dalvik VM

- Allows us to overwrite application and framework methods:
  - Application code is not modified
  - No need to disassemble
System Overview

Three Phases of CuriousDroid

UI Decomposition
- Extract hierarchy of UI elements
- Label interactive elements

Input Inference
- Determine what type of input each element takes (if any)
- Determine order of interaction

Input Generation
- Translate inputs to physical interactions
- Inject inputs into application/OS
Android UI

• *Activity* class is a way for a user to interact with an application
  – Provides window and contains the UI elements

• UI composed of different elements:
  – Containers
  – *Views*
    • Interactive: Buttons, text fields, etc
    • Non-interactive: text labels, etc
User Interface Decomposition

- Overwrite Activity method `onWindowFocusChanged()`
  - Called *after* Views drawn to screen
- Starting with the root view, recursively examine each sub-view until all views are examined
  - As each view is examined compile list of interactive views or “widgets”
Input Inference
Input Inference

- Examine each widget to determine type of interaction
  - Text fields take crafted input
  - Buttons take taps, etc.
Input Inference

- Use hints to determine context
  - Text labels or textfield “hints”
  - Compare to list of keywords
- Draw from list of predefined input values
Input Inference

• Determine order to interact with widgets
  – Top-down left-right
  – nextFocus property

• Always press buttons last!
Input Generation

• Translate ordered inputs into physical interactions
  – Generate data representing gesture
• Separate process writes data directly to input driver
Evaluation

- Does better input generation improve dynamic analysis?
  - Dynamic behavior
  - Activity Coverage

- In total 38,572 applications tested
  - Apps pulled from Andrubis database
  - Compare results generated by Andrubis where input generation system is varied
Andrubis

• Android malware analysis system:
  – Static and Dynamic analysis
    • Static: requested permissions, services, broadcast receivers. API calls used.
    • Dynamic: data leaks, filesystem activity, Phone and SMS, dynamic code loading, JNI

• Assigns score (0 - 10) for each application:

www.anubis.org
Results: Borderline Classification

- 8827 Apps chosen with score from 4-5
- Majority of apps reclassified to benign
- Change in score driven by increase in number of dynamic features generated
Results: Dynamic Behaviors

- Applications chosen for each category contain bytecode for a given behavior that was not exercised by Monkey
- These behaviors often seen in malware

<table>
<thead>
<tr>
<th>Category</th>
<th># Apps</th>
<th># Triggered</th>
<th>% Triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS</td>
<td>6871</td>
<td>440</td>
<td>6.40%</td>
</tr>
<tr>
<td>Dynamic Code</td>
<td>8371</td>
<td>358</td>
<td>4.28%</td>
</tr>
<tr>
<td>Native Code</td>
<td>7669</td>
<td>1945</td>
<td>25.36%</td>
</tr>
<tr>
<td>Networking</td>
<td>7134</td>
<td>2650</td>
<td>37.15%</td>
</tr>
</tbody>
</table>
Results: Activity Measurements

- Activity coverage:
  - Some applications have high number of Activities (up to 287)
  - Some Activities only triggered under certain circumstances
    - SMS received, network data
- How Activities triggered is more important!
  - Valid form data passed from one to another
Conclusion

• CuriousDroid: a tool for automated execution of Android Applications in an intelligent and human-like fashion
• Geared towards high-volume malware analysis systems that require no prior knowledge of apps
• Our results show improved performance over black-box fuzzing
Questions?
Test Application Execution

1. MainActivity
   curiosdroid12345
   ............
   neu.curiosdroid@gmail.com
   1234567890
   banana
   Agree
   Ok Clear

2. My Message
   Username: curiosdroid12345
   Password: asdfASDF1234
   Email: neu.curiosdroid@gmail.com
   Phone: 1234567890
   Message: banana,

3. MainActivity
   curiosdroid12345
   ............
   neu.curiosdroid@gmail.com
   1234567890
   banana
   Agree
   Ok Clear

4. Clear Messages
   All Fields Cleared!!

5. MainActivity
   Username
   Password
   email
   Phone
   Enter Phrase
   Agree
   Ok Clear
Input Generation

• Event injection mechanism running in separate process
  – Takes output from Input Generator
  – Writes directly to the touchscreen input driver

• Mimics actual touch events which are then passed to applications through the Android framework

• OS cannot tell difference between real and simulated touch events