AppsPlayground: Automatic Security Analysis of Smartphone Applications

Vaibhav Rastogi, Yan Chen, and William Enck†
Lab for Internet and Security Technology, Northwestern University
†North Carolina State University
Android Threats

• Privacy leakage
  – Users often have no way to know if there are privacy leaks
  – Even legitimate apps may leak private information without informing user

• Malware
  – Number increasing consistently
  – Need to analyze new kinds

flickr.com/photos/panda_security_france/
Requirements

• Large number of apps in online app stores
  – Google Play has over 700,000 apps
  – This number is constantly increasing
• Offline analysis is important to protect users
• Need a *scalable* and *automatic* approach to tackle threats
• Possible techniques: dynamic analysis and static analysis
## Dynamic vs. Static

<table>
<thead>
<tr>
<th></th>
<th>Dynamic Analysis</th>
<th>Static Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Coverage</strong></td>
<td>Some code not executed</td>
<td>Mostly sound</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>False negatives</td>
<td>False positives</td>
</tr>
<tr>
<td><strong>Dynamic Aspects</strong></td>
<td>Handled without additional effort</td>
<td>Possibly unsound for these</td>
</tr>
<tr>
<td>(reflection, dynamic loading)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Execution context</strong></td>
<td>Easily handled</td>
<td>Difficult to handle</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Usually slower</td>
<td>Usually faster</td>
</tr>
</tbody>
</table>
AppsPlayground

• A system for offline dynamic analysis
  – Includes multiple detection techniques for dynamic analysis

• Challenges
  – Techniques must be light-weight
  – Automation requires good exploration techniques
Outline

• Architecture

• Applications and Results

• Related Work

• Conclusion and Future Work
Outline

• Architecture

• Applications and Results

• Related Work

• Conclusion and Future Work
Architecture

AppsPlayground
Virtualized Dynamic Analysis Environment

Exploration Techniques
- Event triggering
- Intelligent input
- Fuzzing

Detection Techniques
- Kernel-level monitoring
- Taint tracking
- API monitoring

Disguise techniques
Architecture

Event triggering

Intelligent input

Fuzzing

Kernel-level monitoring

Taint tracking

API monitoring

Disguise techniques

Contributions

Exploration Techniques

AppsPlayground

Virtualized Dynamic Analysis Environment

Detection Techniques
Kernel-level Monitoring

• Useful for malware detection
• Most root-capable malware can be logged for vulnerability conditions
• Rage-against-the-cage
  – Number of live processes for a user reaches a threshold
• Exploid / Gingerbreak
  – Netlink packets sent to system daemons
Intelligent Input

• Fuzzing is good but has limitations
• Another black-box GUI exploration technique
• Capable of filling meaningful text by inferring surrounding context
  – Automatically fill out zip codes, phone numbers and even login credentials
  – Sometimes increases coverage greatly
Disguise Techniques

• Make the virtualized environment look like a real phone
  – Phone identifiers and properties
  – Data on phone, such as contacts, SMS, files
  – Data from sensors like GPS
  – Cannot be perfect
Outline

• Architecture

• Applications and Results

• Related Work

• Conclusion and Future Work
Privacy Leakage Results

• AppsPlayground automates TaintDroid

• Large scale measurements - 3,968 apps from Android Market (Google Play)
  – 946 leak some info
  – 844 leak phone identifiers
  – 212 leak geographic location
  – Leaks to a number of ad and analytics domains
Malware Detection

• Case studies on DroidDream, FakePlayer, and DroidKungfu

• AppsPlayground’s detection techniques are effective at detecting malicious functionality

• Exploration techniques can help discover more sophisticated malware
Exploration Effectiveness

• Measured in terms of code coverage
  – 33% mean code coverage
    • More than double than trivial
    • Black box technique
    • Some code may be dead code
    • Use symbolic execution in the future

• Fuzzing and intelligent input both important
  – Fuzzing helps when intelligent input can’t model GUI
  – Intelligent input could sign up automatically for 34 different services in large scale experiments
Outline

• Architecture

• Applications and Results

• Related Work

• Conclusion and Future Work
Related Work

• Google Bouncer
  – Similar aims; closed system

• DroidScope, Usenix Security’12
  – Malware forensics
  – Mostly manual

• SmartDroid, SPSM’12
  – Uses static analysis to guide dynamic exploration
  – Complementary to our approach
Conclusions and Future Work

• AppsPlayground is a system for large-scale, automatic dynamic analysis of Android apps
  – Multiple detection, exploration, and disguise techniques

• Future work
  – Symbolic execution
  – Improve disguise techniques

• Release
  – Check back soon at http://list.northwestern.edu/mobile.html