Malware Detection for Android Platforms

J. Morris Chang
Department of Electrical and Computer Engineering
Iowa State University
July, 2014
Outline

• Android Fast Facts
• Reflection Classification
  – Reflection case examples
  – Classifying malware and benign apps
• Enhance Dynamic Analysis with Instrumentation and Customized Android APIs
  – Instrumentation of Android APKs
  – Customizing Android APIs
Trends in Cybersecurity

• Growing Workforce
  – Pentagon is planning to expand its cybersecurity force from 900 to 4,000 over the next several years (Washington Post, Jan. 2013)
  – Cybersecurity education programs and faculty positions have increased sharply.

• Expanded Research
  – Moving away from traditional security areas.

• Mobile Coverage
  – Policies for Bring Your Own Devices (BYOD)
  – DARPA Automated Program Analysis for Cybersecurity (APAC)
Android Fast Facts

Android had 79% market share on smartphones in June 2013. Android also leads the tablet market with 56.5% market share.

According to Google, as of September 3, 2013, more than 1 billion Android devices have been activated and over 48 billion apps have been installed from the Google Play store.

# Android: Most Popular Mobile OS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>997.7</td>
<td>80.2%</td>
<td>1,401.3</td>
<td>77.6%</td>
<td>12%</td>
</tr>
<tr>
<td>iOS</td>
<td>184.1</td>
<td>14.8%</td>
<td>247.4</td>
<td>13.7%</td>
<td>10%</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>43.3</td>
<td>3.5%</td>
<td>115.3</td>
<td>6.4%</td>
<td>28.1%</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>9.7</td>
<td>0.8%</td>
<td>4.6</td>
<td>0.3%</td>
<td>-25%</td>
</tr>
<tr>
<td>Others</td>
<td>9.3</td>
<td>0.7%</td>
<td>37.7</td>
<td>2.1%</td>
<td>31.5%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1,244.1</strong></td>
<td><strong>100%</strong></td>
<td><strong>1,806.3</strong></td>
<td><strong>1005</strong></td>
<td><strong>12.3%</strong></td>
</tr>
</tbody>
</table>

**Worldwide Smartphone Forecast (Units in Millions)**

Source: IDC Worldwide Mobile Phone Tracker, May 28, 2014
### Android API Development

<table>
<thead>
<tr>
<th>Version</th>
<th>Code Name</th>
<th>API Level</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>May 10 2010</td>
</tr>
<tr>
<td>2.3</td>
<td>Gingerbread</td>
<td>9</td>
<td>Dec 6 2010</td>
</tr>
<tr>
<td>2.3.7</td>
<td></td>
<td>10</td>
<td>Feb 9 2011</td>
</tr>
<tr>
<td>3.0</td>
<td>Honeycomb</td>
<td>11</td>
<td>Feb 22 2011</td>
</tr>
<tr>
<td>3.1</td>
<td></td>
<td>12</td>
<td>May 10 2011</td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td>13</td>
<td>Jul 15 2011</td>
</tr>
<tr>
<td>4.0</td>
<td>Ice Cream Sandwich</td>
<td>14</td>
<td>Oct 19 2011</td>
</tr>
<tr>
<td>4.0.4</td>
<td></td>
<td>15</td>
<td>Dec 16 2011</td>
</tr>
<tr>
<td>4.1</td>
<td>Jelly Bean</td>
<td>16</td>
<td>Jul 9 2012</td>
</tr>
<tr>
<td>4.2</td>
<td></td>
<td>17</td>
<td>Nov 13 2012</td>
</tr>
<tr>
<td>4.3</td>
<td>KitKat</td>
<td>18</td>
<td>Jul 24 2013</td>
</tr>
<tr>
<td>4.4</td>
<td></td>
<td>19</td>
<td>Oct 31 2013</td>
</tr>
</tbody>
</table>

As of Nov 6, 2013, Android API 19 (KitKat) contains 2200 classes and 19000 methods.

The number of known malware samples increased almost five times between July 2012 (about 45,000 samples) and June 2013 (about 273,000 samples).
Reflection Usage in Android

• Reflection is heavily used in Android apps
  – A recent study of 120K apps shows 70% use reflection (2013)
  – Our study of
    • AMGP shows 73% of apps use reflection
    • Top-chart apps from Play Store shows that 98% use reflection

• Reflection is used for
  – Support anticipated features
  – Maintain backward compatibility
  – Support plugins
  – **Deliver malicious code or hide intents??**
OBad Malware

- Named “the most sophisticated Android malware” in 2013 by many antivirus companies
- Steals sensitive user’s information
- Sends SMS to premium services
- Extensively uses reflection and encryption to achieve anti-analysis
- Reflection targets are unknown at compile-time
- Classes are dynamically loaded

Create a need to develop techniques to address malware analysis challenges posed by reflection usage!
Executive Summary

• Four Java semantics for reflection usage
  – Study of nearly 1800 apps
    • 1258 malware apps from AMGP
    • 378 recent malware
    • 126 most downloaded apps from Play Store
  
• Building a dynamic analysis system for Android
  – Instrumentation of Android APKs
  – Customizing Android APIs

Use our system to tackle Obad Malware
Outline

• Reflection Classification
  – Reflection case examples
  – Classifying malware and benign apps
• Enhance Dynamic Analysis with Instrumentation and Customized Android APIs
• Demo: analyzing two malware apps
## Reflection Classification

<table>
<thead>
<tr>
<th>Classification</th>
<th>Reflection target</th>
<th>Classloader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (a)</td>
<td>Constant</td>
<td>Default</td>
</tr>
<tr>
<td>1 (b)</td>
<td>Constant</td>
<td>Custom</td>
</tr>
<tr>
<td>2 (a)</td>
<td>String variable</td>
<td>Default</td>
</tr>
<tr>
<td>2 (b)</td>
<td>String variable</td>
<td>Custom</td>
</tr>
</tbody>
</table>
Reflection Case Examples

Class clazz=Class.forName("TargetClass:com.example.fast:FastSend");
Object obj= clazz.newInstance();
Method method=clazz.getDeclaredMethod("sendMail", String.class,String.class, String.class);
method.invoke(obj, address,subject,content);

classloader = DexClassLoader/libpath/Dir, null, getloader());
classloader.loadClass("FordImage");
getMethod("sendMail");
invoke();
Reflection Usage in tested Android Apps

Total count: 1258

2011 and before *

- Apps with Reflection: 929 (74%)

2012 and after **

- Apps with Reflection: 171 (45%)

Google Play Top-chart Apps

- Apps with Reflection: 3 (98%)

Total count: 378

Total count: 126

* Malware collection from Android Malware Genome Project
** Self collected malwares
*** Google play top-chart apps: mostly 50 million+ downloads
Reflection Usage in Malwares and Top-chart Apps

**Reflection usage in malware collection**

<table>
<thead>
<tr>
<th>Reflection Classification</th>
<th># of reflective calls</th>
<th>Percentage</th>
<th># of reflective calls</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (a)</td>
<td>6357</td>
<td>78.7%</td>
<td>946</td>
<td>20%</td>
</tr>
<tr>
<td>1 (b)</td>
<td>2</td>
<td>0.02%</td>
<td>7</td>
<td>0.1%</td>
</tr>
<tr>
<td>2 (a)</td>
<td>1664</td>
<td>20.6%</td>
<td>3613</td>
<td>76.2%</td>
</tr>
<tr>
<td>2 (b)</td>
<td>56</td>
<td>0.7%</td>
<td>176</td>
<td>3.7%</td>
</tr>
</tbody>
</table>

**Total APKs**

<table>
<thead>
<tr>
<th></th>
<th>1258</th>
<th>378</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2010-2011</td>
<td>2012-2014</td>
</tr>
</tbody>
</table>

**Reflection usage in Top-chart Apps from Google play**

<table>
<thead>
<tr>
<th>Reflection Classification</th>
<th># of reflective calls</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (a)</td>
<td>1774</td>
<td>53.9%</td>
</tr>
<tr>
<td>1 (b)</td>
<td>31</td>
<td>0.9%</td>
</tr>
<tr>
<td>2 (a)</td>
<td>1315</td>
<td>40%</td>
</tr>
<tr>
<td>2 (b)</td>
<td>170</td>
<td>5.2%</td>
</tr>
</tbody>
</table>

**Total APKs**

<table>
<thead>
<tr>
<th></th>
<th>126</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2014</td>
</tr>
</tbody>
</table>

**Year**

<table>
<thead>
<tr>
<th></th>
<th>2014</th>
</tr>
</thead>
</table>
Reflection Density (average)

<table>
<thead>
<tr>
<th>Malware Collection</th>
<th>Total # of classes</th>
<th># of classes with reflections</th>
<th>Density (per class)</th>
<th>Total # of Methods</th>
<th># of methods with reflections</th>
<th>Density (per method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMGP</td>
<td>299126</td>
<td>3015</td>
<td>1.01%</td>
<td>1718380</td>
<td>4494</td>
<td>0.26%</td>
</tr>
<tr>
<td>2012-2014</td>
<td>118267</td>
<td>1975</td>
<td>1.67%</td>
<td>839109</td>
<td>3248</td>
<td>0.39%</td>
</tr>
<tr>
<td>Google play top-chart</td>
<td>372985</td>
<td>4317</td>
<td>1.16%</td>
<td>2364347</td>
<td>7423</td>
<td>0.31%</td>
</tr>
</tbody>
</table>
Reflection Density (per Class)

**AMGP Reflection Density (per Class)**

- >50%: 0
- 10% - 50%: 10
- 1% - 10%: 568
- <1%: 680

Total count: 1258

**2012-2014 Malware Reflection Density (per Class)**

- >50%: 4
- 10% - 50%: 17
- 1% - 10%: 107
- <1%: 250

Total count: 378
Reflection Density (per Class)

Google Play Top-chart Reflection Density (per Class)

<table>
<thead>
<tr>
<th>Density Interval</th>
<th>APK Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1%</td>
<td>53</td>
</tr>
<tr>
<td>1% - 10%</td>
<td>73</td>
</tr>
<tr>
<td>10% - 50%</td>
<td>0</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>0</td>
</tr>
</tbody>
</table>

Total count: 126
Reflection Density (per Method)

AMGP Reflection Density (per Method)

- >50%: 0
- 10% - 50%: 10
- 1% - 10%: 73
- <1%: 1185

Total count: 1258

2012-2014 Malware Reflection Density (per Method)

- >50%: 4
- 10% - 50%: 20
- 1% - 10%: 11
- <1%: 346

Total count: 378
Reflection Density (per Method)

Total count: 126

<table>
<thead>
<tr>
<th>Reflection Density</th>
<th>APK count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1%</td>
<td>53</td>
</tr>
<tr>
<td>1% - 10%</td>
<td>73</td>
</tr>
<tr>
<td>10% - 50%</td>
<td>0</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>0</td>
</tr>
</tbody>
</table>

Google Play Top-chart Reflection Density (per Method)
Enhanced Dynamic Analysis

• Instrumentation of Android APKs
  – Instrument classes.dex to precisely collect data of interest: e.g. Reflection analysis, identify reflection target

• Customize Android APIs
  – Provide enriched functionality of original API to support dynamic analysis: e.g., DexClassLoader(), PathClassLoader() to retrieve .dex class file
Instrumenting Android Apps

- Using Soot to generate Jimple code from APK
- Look for reflection calls based on the four usage patterns (e.g., 1(a), 2(b))
- After each reflection call, add new statements to collect reflection target name and assign ID for each reflection
Customize Android APIs

• Java API vs Android API
  – Some useful Java APIs are not fully implemented in Android
    e.g., `Java.lang.ClassLoader.getResource()` or `getResourceAsStream()`
  – Customize class loaders `DexClassLoader()` and `PathClassLoader()` to support retrieving class files at runtime
Analysis Framework with Instrumentation
Reflection case study: Tackle Obad.apk

• Named “the most sophisticated Android malware” in 2013 by many anti-virus companies
• Extensive encryption and reflection usage (reflection class density > 80%)
• Features of Obad:
  – 24 permissions requested, every reflection target is encrypted, request for administrative control, hides itself, sends data to remote server, drains battery
  – No GUI interface
Instrumentation Workflow

- Obad APK
- Analysis/Instrumentor
- SOOT
- Reflection calls

Instrumented Obad APK
Demo

• Demonstrate the effectiveness of our approach to pinpoint reflection usage in malware
  – *Obad* (7 instances of 1(a) and 1119 instances of 2(a))
    • to identify and obtain reflection targets by instrumenting dex files
  – Our own malware (instances of 1(b) and 2(b))
    • to get the class file by customizing Android API *DexClassLoader()*,
      *PathClassLoader()*
Conclusion

• Heavy use of reflection creates challenges for malware analysts
  • Reflection can be used to hide intent or deliver malicious code
  • Yet it is not easy to analyze statically

• We present and demonstrate a framework that can help analysts to dynamically analyze malwares with reflection usage
  – Identify reflection targets and collect information of interest through instrumentation
  – Retrieve class files at runtime through custom Android API
Thank You!

Q & A
Hybrid Analysis Framework
Instrumenting Obad

- Jimple snippet before instrumentation

```java
$s1 = staticinvoke <com.android.system.admin.OlclClC: java.lang.String C01C000o(int,int,int)>(20, -137, 291);
s$r7 = staticinvoke <java.lang.Class: java.lang.Class forName(java.lang.String)>($s1);
s$r1 = staticinvoke <com.android.system.admin.OlclClC: java.lang.String C01C000o(int,int,int)>(38, -147, 611);
s$r6 = newarray (java.lang.Class)[1];
s$r0 = staticinvoke <com.android.system.admin.OlclClC: java.lang.String C01C000o(int,int,int)>(20, -138, 436);
s$r5 = staticinvoke <java.lang.Class: java.lang.Class forName(java.lang.String)>($r0);
s$r6[0] = $r5;
s$r8 = virtualinvoke $r7.<java.lang.Class: java.lang.reflect.Method getMethod(java.lang.String,java.lang.Class[])>($r1, $r6);
virtualinvoke $r8.<java.lang.reflect.Method: java.lang.Object invoke(java.lang.Object,java.lang.Object[])>($s14, $r4);
```
Instrumenting Obad

- Jimple snippet after instrumentation

```java
$r7 = staticinvoke $<java.lang.Class: java.lang.Class $forName(java.lang.String)>(sr1);
stringBuilder = new java.lang.StringBuilder;
specialInvoke stringBuilder.$void <init>(java.lang.String)>("IID=1176,f,");
stringBuilder = virtualInvoke stringBuilder. $java.lang.StringBuilder $append(java.lang.String)>(sr1);
className = virtualInvoke stringBuilder. $java.lang.String $toString>()();
outPrint = $<java.lang.System: java.io.PrintStream $out>();
virtualInvoke outPrint. $java.io.PrintStream $void $println(java.lang.String)> (className);

$r8 = virtualInvoke sr7.$java.lang.Class: java.lang.reflect.Method $getMethod(java.lang.String,java.lang.Class[])>(sr1, sr6);
stringBuilder = new java.lang.StringBuilder;
specialInvoke stringBuilder. $void <init>(java.lang.String)>("IID=1178,g,");
stringBuilder = virtualInvoke stringBuilder. $java.lang.StringBuilder $append(java.lang.String)>(sr1);
methodName = virtualInvoke stringBuilder. $java.lang.String $toString>()();
outPrint = $<java.lang.System: java.io.PrintStream $out>();
virtualInvoke outPrint. $java.io.PrintStream $void $println(java.lang.String)> (methodName);
decClass = virtualInvoke $r8.$java.lang.reflect.Method: java.lang.Class $getDeclaringClass>()();
className = virtualInvoke decClass. $java.lang.Class $getName>()();
methodName = virtualInvoke sr8.$java.lang.reflect.Method: java.lang.String $getName>()();
outPrint = $<java.lang.System: java.io.PrintStream $out>();
stringBuilder = new java.lang.StringBuilder;
specialInvoke stringBuilder. $void <init>(java.lang.String)>("IID=1179,l,");
stringBuilder = virtualInvoke stringBuilder. $java.lang.StringBuilder $append(java.lang.String)> (className);
stringBuilder = virtualInvoke stringBuilder. $java.lang.StringBuilder $append(java.lang.String)> (";
stringBuilder = virtualInvoke stringBuilder. $java.lang.StringBuilder $append(java.lang.String)> (methodName);
className = virtualInvoke stringBuilder. $java.lang.String $toString>()();
virtualInvoke outPrint. $java.io.PrintStream $void $println(java.lang.String)> (className);
```