DroidEagle: Seamless Detection of Visually Similar Android Apps

Mingshen Sun, Mengmeng Li and John C.S. Lui

Department of Computer Science and Engineering
The Chinese University of Hong Kong

June 25, 2015
Introduction – Android Malware is Coming
Android malware samples accounted for 98% of all mobile threats
99% came from many third-party markets
trojan, fake and phishing apps: can you tell which one is real Taobao, one of the largest online shopping platform in China?
86% of Android malware are using the *repackaging technique*
- disassemble a legitimate app using some well-known tools
- hackers can add or modify logics of the original apps, and then assemble it back
- distribute them in third-party markets

malicious functions using repackaging technique
- crack paid apps to bypass payment functions
- replace developers’ advertisement IDs
- acquire sensitive information: account password and credit card number
Related Work

- instruction sequences
  - fuzzy hashing
  - sensitive to instruction sequence obfuscation

- semantic information
  - call reference graph
  - hacking tricks to bypass existing disassembling tools
Observations:

- repackaged apps should have similar appearance as original one
- phishing malware relies on similar appearances as banks or shopping apps to deceive users
- by comparing visual similarity, one can quickly determine potential repackaged malware or phishing malware

**DroidEagle** is based on visual characteristics to detect similar Android apps.

- detect visually similar apps in app repository and Android device respectively
- RepoEagle and HostEagle implementation
Android app user interface

- **View**: objects on the screen which can interact with users and display objects (e.g., ImageView, EditText)
- **ViewGroup**: define the layout arrangement of its elements (e.g., ScrollView, LinearLayout)

- layout files in `/res/layout*` directory
Detection Methodologies – Layout Tree

■ Overview
  ■ accuracy, efficiency, scalability and flexibility
  ■ visual resources in an app: layout files and drawable images

■ A layout tree is a tree data structure over a layout file where:
  ■ A node in the layout tree represents an element in the layout file.
  ■ The parent/child relationship of nodes in a layout tree is the same as that in the layout file.
  ■ attributes for each node
1. layout tree defines the visual structure of an app’s user interface
2. repackaged malware and phishing malware rely on same layout tree to deceive users
3. detailed attributes in layout tree accurately describe visual appearance
4. layout tree is easily obtained in android package file
5. modifications on layout files can mess up appearance

Diagram:

```
LinearLayout
  orientation="vertical"
  layout_height=...
  layout_width="fill_parent"

ToolBar
ScrollView
  background=...

LinearLayout

TextView
  text="Password"
  layout_marginBottom="2.0dip"

EditText

CheckBox
```
RepoEagle: Repository Analysis

**Repository Analysis**

- RepoEagle can analyze all apps in an app repository to discover all visually similar apps
  - layout tree extractor, certificate extractor, similarity comparator, certificate verifier
Similarity Comparison: layout edit distance (LED)

- measure the similarity between two layout trees
- LED is the *minimum number of operations* required to transform from one layout tree to another tree
Host-based Detection

- safety and authenticity of apps from unknown sources
- host-based detection system for Android
- HostEagle can detect repackaged malware and phishing malware in-device
  - layout hashing, certificate extractor, detector, local detection & cloud detection
HostEagle: Host-based Detection – Layout Hashing

- leaf pruning: all leaves in the layout tree
  - View objects in leaves
  - layout skeleton of the user interface
- tree hashing

(1) leaf pruning
LinearLayout
  Toolbar
  ScrollView
  RelativeLayout
    CheckBox
    LinearLayout
      TextView
      EditText

(2) hashing
LinearLayout
  orientation="vertical" layout_width="fill_parent" ...
  background="@color/bg_grounped_list" ...
RelativeLayout
  layout_width="fill_parent" layout_height="..."
LinearLayout
  orientation="vertical" id="..."
Layout Hashing: 12B39EF
We crawled and collected 100,126 apps from

- Google official market
- third party markets
- public cloud storage

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>URL</th>
<th># of Apps</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Official</td>
<td>Google Play</td>
<td>play.google.com</td>
<td>500</td>
<td>7.0 GB</td>
</tr>
<tr>
<td>Third-party</td>
<td>appchina</td>
<td>appchina.com</td>
<td>34,989</td>
<td>238 GB</td>
</tr>
<tr>
<td></td>
<td>appfun</td>
<td>appfun.cn</td>
<td>12,427</td>
<td>154 GB</td>
</tr>
<tr>
<td></td>
<td>hiapk</td>
<td>apk.hiapk.com</td>
<td>5,287</td>
<td>87 GB</td>
</tr>
<tr>
<td></td>
<td>android.d.cn</td>
<td>android.d.cn</td>
<td>4,064</td>
<td>163 GB</td>
</tr>
<tr>
<td></td>
<td>jimi168</td>
<td>jimi168.com</td>
<td>23,723</td>
<td>76 GB</td>
</tr>
<tr>
<td></td>
<td>anzhi</td>
<td>anzhi.com</td>
<td>18,736</td>
<td>118 GB</td>
</tr>
<tr>
<td>Cloud Storage</td>
<td>Baidu</td>
<td>pan.baidu.com</td>
<td>200</td>
<td>3.5 GB</td>
</tr>
<tr>
<td></td>
<td>Huawei</td>
<td>dbank.com</td>
<td>200</td>
<td>3.1 GB</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>100,126</td>
<td>849.6 GB</td>
</tr>
</tbody>
</table>

Mingshen Sun (CUHK)
RepoEagle statically analyze apps in the repository.

- most of repackaged apps are cracked games with unlocked in-app paid markets
- malware samples are found in public cloud storage
  - hide identity
  - spread in forum and social media

<table>
<thead>
<tr>
<th>Market</th>
<th># of Visually Similar Apps (Percentage)</th>
<th># of Malware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third-party Market</td>
<td>1159 (1.6 %)</td>
<td>10</td>
</tr>
<tr>
<td>Cloud Storage (Baidu)</td>
<td>50 (10.0 %)</td>
<td>0</td>
</tr>
<tr>
<td>Cloud Storage (Huawei)</td>
<td>89 (17.8 %)</td>
<td>15</td>
</tr>
</tbody>
</table>
## Experiment – Repackaged Apps

### Case study of repository for Andry Bird app
- 8 apps which are visually similar with the official app “Angry Bird” from the Google Play
- different certificate issuers
- detection results: DroidKungFu malware can contact remote server and download malware, gain the root privilege and prevent uninstalling.

<table>
<thead>
<tr>
<th>App ID</th>
<th>LED</th>
<th>LH</th>
<th>Certificate Issuer</th>
<th>Cert</th>
<th>Rpkg</th>
<th>Market</th>
<th>Detection Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>22d3</td>
<td>0</td>
<td>1cd5</td>
<td>Rovio Mobile Ltd.</td>
<td>5557</td>
<td></td>
<td>Google Play</td>
<td></td>
</tr>
<tr>
<td>233c</td>
<td>0</td>
<td>1cd5</td>
<td>Rovio Mobile Ltd.</td>
<td>5557</td>
<td></td>
<td>appfun.com</td>
<td></td>
</tr>
<tr>
<td>5803</td>
<td>0</td>
<td>1cd5</td>
<td>Rovio Mobile Ltd.</td>
<td>5557</td>
<td></td>
<td>appfun.com</td>
<td></td>
</tr>
<tr>
<td>666c</td>
<td>0</td>
<td>1cd5</td>
<td>Virtuous Ten Studio</td>
<td>A925</td>
<td>✓</td>
<td>appfun.com</td>
<td>Android.Adware. Jumptap.a</td>
</tr>
<tr>
<td>7a43</td>
<td>0</td>
<td>1cd5</td>
<td>Rovio Mobile Ltd.</td>
<td>5557</td>
<td></td>
<td>appfun.com</td>
<td></td>
</tr>
<tr>
<td>9ee4</td>
<td>0</td>
<td>1cd5</td>
<td>databin</td>
<td>FC00</td>
<td>✓</td>
<td>appfun.com</td>
<td></td>
</tr>
<tr>
<td>22d3</td>
<td>0</td>
<td>1cd5</td>
<td>Rovio Mobile Ltd.</td>
<td>5557</td>
<td></td>
<td>android.d.cn</td>
<td>Android.Adware. Dowgin</td>
</tr>
<tr>
<td>0f6f</td>
<td>0</td>
<td>1cd5</td>
<td>android-debug</td>
<td>264B</td>
<td>✓</td>
<td>android.d.cn</td>
<td></td>
</tr>
<tr>
<td>3b52</td>
<td>0</td>
<td>1cd5</td>
<td>keystore3</td>
<td>990B</td>
<td>✓</td>
<td>dbank.com</td>
<td>Android.Trojan. DroidKungFu</td>
</tr>
<tr>
<td>ed0e</td>
<td>2</td>
<td>1cd5</td>
<td>Rovio Mobile Ltd.</td>
<td>5557</td>
<td></td>
<td>jimi168.com</td>
<td></td>
</tr>
</tbody>
</table>
Experimental result of layout hashing for phishing malware (fake apps).

- FakeAV masquerades as the “Avast” anti-virus software
- FakeMart masquerades as the Google official market “Google Play”
- Agent: network agent utility

<table>
<thead>
<tr>
<th>Name</th>
<th># of Samples</th>
<th>Layout File</th>
<th>LH</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FakeAV</td>
<td>8</td>
<td>activity_scanning.xml</td>
<td>5a7e</td>
<td>0.466</td>
</tr>
<tr>
<td>FakeMart</td>
<td>3</td>
<td>main.xml</td>
<td>d41d</td>
<td>0.355</td>
</tr>
<tr>
<td>Agent</td>
<td>5</td>
<td>activity_main.xml</td>
<td>9344</td>
<td>0.501</td>
</tr>
</tbody>
</table>

* Generation time of LH in second on Nexus 5.
The impact of nodes number in layout files in the repository analysis system.

- number of qualified apps
- detection accuracy of repackaged “Fruit Ninja” and “PPS” respectively
Evaluation – Detection Efficiency

Time of hash value generation on Android device for different sizes of DEX file.

- traditional fuzzy hashing method should disassemble the source code, which will be affected by dex file size
- HostEagle will not be affected
- relationship between threshold and the analysis time
- pre-processing time for different numbers of apps
Conclusion

- repackaged malware and phishing malware
- detection based on visual resource: layout file
- 3 hours, 1298 visually similar apps with 25 malware

Questions?