Finding Unknown Malice in 10 Seconds: Mass Vetting for New Threats at the Google-Play Scale

Kai Chen‡,†, Peng Wang†, Yeonjoon Lee†, XiaoFeng Wang†, Nan Zhang†, Heqing Huang§, Wei Zou†, Peng Liu§

†Indiana University, Bloomington
‡Institute of Information Engineering, Chinese Academy of Sciences
§College of IST, Penn State University

http://www.appomicsec.com
Background

• Android Malware
  – Billions of mobile computing devices. 70% are Android.
  – In 2014, 99% of mobile malware targets Android system

• Current Approaches
  – Signature-based detection & Behavior-based detection

• Are they effective in malware detection?
Are they effective?

• Signature-based detection
  – Cannot detect new malware: *Over 160,000 new malware samples created every day (Panda Security, 2014)*
  – Code obfuscation, e.g., DroidChameleon (AsiaCCS 2013)

• Behavior-based Detection
  – *Heavyweight* information-flow analysis
  – Require *known suspicious behaviors* (e.g., Dynamic code loading)
Can we design an approach that is:

- Highly efficient
- Detect malware with unknown behaviors

We achieve this goal using neither signatures nor behaviors. But only code comparison.
Observation: a unique business model

Attackers like to attach the same attack payload to legitimate apps.
Results of Repackaging

Compare related apps, check “different” code
Results of Repackaging

Detect code intersection in apps with unrelated apps
Our approach: DiffCom Analysis

Sim-View Analysis

Yes

Diff Analysis

No

Com Analysis

Suspicious?
Sim-View Analysis: An example
Sim-View Analysis: An example
Sim-View Analysis: An example
Sim-View Analysis: An example
Sim-View Analysis: An example
Sim-View Analysis: View graph
Sim-View Analysis: View Graph

Another Entry Point

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener

OnClickListener
Sim-View Analysis: Compare View Graphs

VS!
Can we avoid graph isomorphism analysis?

$O(n^2 \cdot M^2)$

“Enemy” for scalability

Goal

$O(c \cdot M)$
Sim-View Analysis: Challenge

- **Challenge 1: A Graph edge = abstract relation**
  - The abstract relation could have arbitrary length

- **Challenge 2: Switching branches changes node positions**

Original Graph  
Challenge 1  
Challenge 2
Our idea: Fix the nodes in the graph

• Step 1: view graph → 3D-view-graph → v-core
• Step 2: Scalable comparison
Sim-View Analysis: v-core

Step 1: Accurate mapping: view graph $\rightarrow$ 3D-view-graph $\rightarrow$ v-core

3D-View-Graph is a View Graph in which each node has a unique coordinate.

- The coordinate is a vector $<x,y,z>$
- $x$ is the sequence number in the view graph
- $y$ is the number of outgoing edges of the node
- $z$ is the depth of loop of the node
Sim-View Analysis: v-core

Step 1: Accurate mapping: view graph → 3D-view-graph → v-core

A <1, 1, 0>; B <2, 2, 1>; C <3, 2, 1>; D <4, 1, 1>; E <5, 1, 0>; F <6, 0, 0>
Sim-View Analysis: v-core

Step 1: Accurate mapping: view graph $\rightarrow$ 3D-view-graph $\rightarrow$ v-core

$$vC_i = \frac{\sum_{e(p,q) \in G_i} (w_p \vec{c}_p + w_q \vec{c}_q)}{\sum_{e(p,q) \in G_i} (w_p + w_q)}$$
Sim-View Analysis: v-core

Step 2: Scalable comparison
- First, sub-graph-level comparison
  \[ |vC_i - vC_t| \leq \tau \]
- Second, app-level comparison
  \[ \sum_l |G_{i(l)}| / \sum_i |G_i| \geq \theta \]

**Feature 1:** The similarity between two graphs is monotonically correlate to the “distance” between two v-cores.

**Feature 2:** V-cores are sortable. We only need to compare a v-core with its neighbors, but not all v-cores.
Diff Analysis

• For apps having **the same view** and **different signatures**, the different methods between the two apps may be malicious.

• Challenge 1: How to quickly compare two apps and find the different methods?

• Challenge 2: Are the different methods malicious?
Diff Analysis

• Challenge 1: How to quickly compare two apps and find the different methods?
• Centroid on methods:
  Control flow graph (CFG) $\rightarrow$ 3D-CFG $\rightarrow$ m-core

![Diagram of control flow graph and 3D model](image)
Diff Analysis

• Challenge 2: Are the different methods malicious?
  – Ads and other libraries
  – Updated code (from the same author)
  – Unharmful code

• Solution
  – White-list of libraries
  – Stand-alone analysis
  – Sensitive APIs
    • e.g., GetSimSerialNumber
    • Avoid heavy-weight information flow analysis
Com Analysis

• For the apps with different views, find the common code

• Challenge 1: Are the two apps really unrelated?
• Challenge 2: Is the common code really malicious?
Com Analysis

• Challenge 1: Is the two apps really unrelated?
• Correlation check
  – Similar ideas with “Diff”

Rovio Entertainment
Com Analysis

• Challenge 2: Is the common code really malicious?
  – Library code: Ads, third-party libraries
  – Code reuse: templates

• Approach
  – White-listing popular libraries
  – Training set: the methods not viewed as malicious by virustotal

• Report suspicious code: the method with dangerous APIs
Measurement – Scale of study

• Total apps collected: 1.2 million apps
  – Duplicates removed using MD5

• App markets covered: 33

• # of apps collected from different markets and region
  – GooglePlay: 400,000+ apps
  – Chinese app markets: 596,437 apps
  – European app markets: 61,866 apps
  – Other US stores: 27,047 apps
Measurement – False Positive

- Flagged apps by MassVet: 127,429 apps (10.93%)
- **FDR** (false-positive VS all detected): 4.73%
- **FPR** (false-positive VS all apps analyzed): < 1%
- Manually studied: 20/40 malware

FDR: 4.73%
Measurement – Coverage

- 2700 Randomly sampled apps
  - Virustotal: 281 apps
  - MassVet: 197 apps (70.11%)
  - NOD32: 171 apps (60.85%)
  - McAfee: 45 apps (16.01%)
  - 21 apps (11%) apps missed by Virustotal

<table>
<thead>
<tr>
<th>AV Name</th>
<th># of Detection</th>
<th>% Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (MassVet)</td>
<td>197</td>
<td>70.11</td>
</tr>
<tr>
<td>ESET-NOD32</td>
<td>171</td>
<td>60.85</td>
</tr>
<tr>
<td>VIPRE</td>
<td>136</td>
<td>48.40</td>
</tr>
<tr>
<td>NANO-Antivirus</td>
<td>120</td>
<td>42.70</td>
</tr>
<tr>
<td>AVware</td>
<td>87</td>
<td>30.96</td>
</tr>
<tr>
<td>Avira</td>
<td>79</td>
<td>28.11</td>
</tr>
<tr>
<td>Fortinet</td>
<td>71</td>
<td>25.27</td>
</tr>
<tr>
<td>AntiVir</td>
<td>60</td>
<td>21.35</td>
</tr>
<tr>
<td>Ikarus</td>
<td>60</td>
<td>21.35</td>
</tr>
<tr>
<td>TrendMicro-HouseCall</td>
<td>59</td>
<td>21.00</td>
</tr>
<tr>
<td>F-Prot</td>
<td>47</td>
<td>16.73</td>
</tr>
<tr>
<td>Sophos</td>
<td>46</td>
<td>16.37</td>
</tr>
<tr>
<td>McAfee</td>
<td>45</td>
<td>16.01</td>
</tr>
<tr>
<td>DrWeb</td>
<td>45</td>
<td>16.01</td>
</tr>
<tr>
<td>Baidu-International</td>
<td>44</td>
<td>15.66</td>
</tr>
<tr>
<td>AVG</td>
<td>40</td>
<td>14.23</td>
</tr>
<tr>
<td>Comodo</td>
<td>32</td>
<td>11.39</td>
</tr>
<tr>
<td>Cyren</td>
<td>29</td>
<td>10.32</td>
</tr>
<tr>
<td>F-Secure</td>
<td>22</td>
<td>7.83</td>
</tr>
<tr>
<td>AhnLab-V3</td>
<td>20</td>
<td>7.12</td>
</tr>
<tr>
<td>Tencent</td>
<td>16</td>
<td>5.69</td>
</tr>
<tr>
<td>Symantec</td>
<td>15</td>
<td>5.34</td>
</tr>
<tr>
<td>Alibaba</td>
<td>15</td>
<td>5.34</td>
</tr>
<tr>
<td>Commtouch</td>
<td>13</td>
<td>4.63</td>
</tr>
<tr>
<td>GData</td>
<td>10</td>
<td>3.56</td>
</tr>
</tbody>
</table>
Measurement – Performance

• A server with 260 GB memory, 40 cores at 2.8 GHz and 28 TB hard drives

• 9 seconds from the submission of the app to the completion of the whole process on it.

<table>
<thead>
<tr>
<th># Apps</th>
<th>Pre-Processing analysis</th>
<th>v-core database search</th>
<th>differential</th>
<th>m-core database search (Intersection)</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5.84</td>
<td>0.15</td>
<td>0.33</td>
<td>1.80</td>
<td>8.12</td>
</tr>
<tr>
<td>50</td>
<td>5.85</td>
<td>0.15</td>
<td>0.34</td>
<td>1.99</td>
<td>8.33</td>
</tr>
<tr>
<td>100</td>
<td>5.85</td>
<td>0.14</td>
<td>0.35</td>
<td>2.23</td>
<td>8.57</td>
</tr>
<tr>
<td>200</td>
<td>5.88</td>
<td>0.16</td>
<td>0.35</td>
<td>3.13</td>
<td>9.52</td>
</tr>
<tr>
<td>500</td>
<td>5.88</td>
<td>0.16</td>
<td>0.35</td>
<td>3.56</td>
<td>9.95</td>
</tr>
</tbody>
</table>
Measurement – Landscape

• 35,473 (north America), 4,852 (Europe), 87,104 (Asia)
• Apps from Google Play: 7.61% are potentially harmful
• Virustotal confirmed 91,648 malware
  – 4.1% were alarmed by at least 25 out of 54 scanners
Measurement – Existing defense

• Existing defense: Google Play indeed makes effort to mitigate the malware threat

• Most malware we discovered were uploaded in the past 14 months
Measurement – Disappeared apps

• After uploading 3,711 apps to Virustotal (scan mode)
  – **40 days later:** 250 of them disappeared
  – **90 days later:** another 129 apps disappeared
  – Among the 379 disappeared apps, 54 apps (14%) are detected by Virustotal
Measurement – Disappeared apps

• Track 2,265 developers of the 3,711 apps (2014/11~2015/02)
  – Additional 2014 apps disappeared (all detected by MassVet)
  – We did NOT check them by virustotal
    • Google Play also looked into their common malicious components under the same developers, but not across the whole market (may take long time).
    • Our work is just the one can help them (in several seconds).

• Reappeared apps
  – 604 confirmed malware (28.4%) showed up in Google Play unchanged
  – 829 apps showed up using different names
Measurement – Impact

• Distribution of downloads for malicious or suspicious apps in GooglePlay

400 apps: 1,000,000+

2000 apps: 50,000+
Measurement – Impact

• The distribution of rating for malicious or suspicious apps in GooglePlay

3000 apps: 3.9
### Measurement – Signatures

- Top 5 signatures used in apps

<table>
<thead>
<tr>
<th>Signature</th>
<th># of malicious apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>c673c8a5f021a5bdc5c036ee30541dde</td>
<td>1644</td>
</tr>
<tr>
<td>a2993eae31e3c2bcad4769cb79f1556</td>
<td>1258</td>
</tr>
<tr>
<td>3be7d6ee0dca7e8d76ec68cf0ccd3a4a</td>
<td>615</td>
</tr>
<tr>
<td>f8956f66b67be5490ba6ac24b5c26997</td>
<td>559</td>
</tr>
<tr>
<td>86c2331f1d3bb4af2e88f485ca5a4b3d</td>
<td>469</td>
</tr>
</tbody>
</table>
Measurement – Identities

- Top 5 signatures used by different identities

<table>
<thead>
<tr>
<th>Signature</th>
<th># of different identities</th>
</tr>
</thead>
<tbody>
<tr>
<td>02d98ddfbcde202b13c49330182129e05</td>
<td>604</td>
</tr>
<tr>
<td>a2993eaecf1e3c2bcad4769cb79f1556</td>
<td>447</td>
</tr>
<tr>
<td>82fd3091310ce901a889676eb4531f1e</td>
<td>321</td>
</tr>
<tr>
<td>9187c187a43b469fa1f995833080e7c3</td>
<td>294</td>
</tr>
<tr>
<td>c0520c6e71446f9ebdf8047705b7bda9</td>
<td>145</td>
</tr>
</tbody>
</table>
Conclusion

• We propose a new technique for efficient vetting of apps for unknown malware
  – Compare an app with all other apps on a market (DiffCom Analysis)
  – Light-weight code analysis compared with other approaches

• We implemented MassVet and apply it to analyze 1.2 million apps.

• MassVet found 127,429 malware (20 likely to be zero days)
MassVet Available Now

http://www.appomicsec.com

Thank You! Questions