Finding Unknown Malice in 10 seconds: Mass Vetting For New Threats at the Google Play Scale

Paper By:
Kai Chen, Peng Wang, Yeonjoon Lee, Xiao Feng Wang, Nan Zhang, Heqing Huang, Wei Zou and Peng Liu

Summary by: Abhishek Aravind Kulkarni
Need for Automated detection of Malware

- Huge number of user-generated apps, numbering in millions.
- 70% of all mobile devices are android.
- 99% of all mobile malware is on android.
- Hidden malicious behavior behind legitimate software.
- Most malware is simply repackaged version of popular apps.

Number of malicious apps over period of 5 years[1]
An Existing Solution – Google Bouncer

- Analyses an app before it is even published to the market.
- Emulates the android environment.
- The static system cannot detect new malware.
- The working of the bouncer is very predictable.
- Heavyweight dynamic system which may be circumvented using obfuscation.
MassVet: Architecture
MassVet: Architecture

Elements of MassVet

- Pre-processor
- DiffCom Analyzer
- Feature Database System
Pre-processor

Static analysis/Code Injection

- Convert app to intermediate code
- Remove whitelisted code

Disassemble (baksmali)  
.smali

Image taken from paper[2]
Pre-processor

- Extract v-cores and m-cores
- Create a CFG using the v-cores

Image taken from paper[^1]
Feature Database System

- Apps on the store analyzed.
- The m-cores and v-cores are extracted to be matched with suspicious apps.
- Heavy on memory and storage.
Diff-com Analyzer

1. APP
2. CHECK FOR SIMILAR VIEWS IN APP STORE
   - SIMILAR FOUND
     - DIFF ANALYSIS
   - SIMILAR
     - NONE
3. COM ANALYSIS
Diff-com Analyzer

- V-core of suspect app compared to feature DB.
- The measure of similarity is by the use of centroids.
- If similarities > threshold, difference module is applied, else common module is applied.

\[ vC_i = \frac{\sum_{e(p,q) \in G_i} (w_p \overrightarrow{c}_p + w_q \overrightarrow{c}_q)}{\sum_{e(p,q) \in G_i} (w_p + w_q)} \]

- First, sub-graph-level comparison
  \[ |vC_i - vC_t| \leq \tau \]

- Second, app-level comparison
  \[ \sum_l |G_{i(l)}| / \sum_i |G_i| \geq \theta \]
Evaluation – Total Detections

- VirusTotal – a service using 56 different anti virus systems detected 89% of all malware samples.
- MassVet scored the best among individual scanners (~70%)
Evaluation – Errors in detection

- MassVet is incredibly accurate in detections using its v-core matching.
- It has very low false positive and false negative detection rates.
Evaluation - Performance

- MassVet uses a server with 280 GB RAM, 40 cores and 28 TB HDD.
- It detects malware with an average delay of around 9 seconds.

<table>
<thead>
<tr>
<th># apps</th>
<th>Pre processing</th>
<th>Cores</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5.84</td>
<td>1.95</td>
<td>8.12</td>
</tr>
<tr>
<td>50</td>
<td>5.85</td>
<td>2.14</td>
<td>8.33</td>
</tr>
<tr>
<td>100</td>
<td>5.85</td>
<td>2.37</td>
<td>8.57</td>
</tr>
<tr>
<td>200</td>
<td>5.88</td>
<td>3.29</td>
<td>9.52</td>
</tr>
<tr>
<td>500</td>
<td>5.88</td>
<td>3.72</td>
<td>9.55</td>
</tr>
</tbody>
</table>

Source: Paper[1]
Observations - I

- Google Bouncer is capable of detecting a fair amount of malware.
- Most repackaged malware reappear after being deleted.
- Malware Code is often localized by region.
- Just as UI is reused, virus code is almost always reused.
Observations - II

- MassVet has very low false positive and low false negative rates with repackaged code.
- MassVet is completely automated.
- It uses light-weight comparison strategies.
- MassVet is capable of identifying violations of IPR.
Observations - III

- MassVet will fail in case of new UI design or new method construct.
- MassVet can not handle obfuscated code.
References


THANK YOU :)