User-Centric Dependence Analysis For Identifying Malicious Mobile Apps

Karim Elish, Danfeng (Daphne) Yao, Barbara G. Ryder

Department of Computer Science
Virginia Tech
Legitimate or Malicious: an app-classification problem

Problem: How to classify unknown apps as benign or malicious?

Source: http://news.cnet.com/8301-1009_3-57328575-83/androids-a-malware-magnet-saysmcafee/?tag=mncol;topStories
An Anomaly Detection Approach

**Can one enforce properties of legitimate programs, as opposed to chasing malware patterns?**

**Challenge: what is norm?**

**Attack model:** stealthy malware accesses system resources without user’s awareness or consent

**Our key observations:**
- legitimate critical system events are typically initiated by user inputs/actions
- Android mobile apps require a lot of user interactions

**Our Goal:**

*Use static program analysis to track the dependence between the definition and use of user-generated data in programs*
Example of Android Malware: HippoSMS

This malware sends SMS messages to a hard-coded premium-rated number without the user’s awareness

```
public class MessageService{
    .....;
    public void onStart(){
        sendsms("1066156686", "8", "");
    }
    public void sendsms(String param1, String param2, String param3){
        .....;
        localSmsManager.sendTextMessage(param1, param2, param3);
    }
}
```

A Data Dependence Graph

Malicious code
What is the norm? How to enforce it?

Requests to access system resources should be based on user inputs / actions

Our approach:

Identify the dependency relation between critical system events and user-initiated events in programs

Resources to protect from malicious programs:

- File system access
- Network access
- Sensitive/personal data

User inputs/actions

ReadFile() ✓

User inputs/ actions

SendSMS() ✗
Our User-Centric Dependence Based Anomaly Detection Approach

Our Static Analysis Tool:
- We utilize def-use structures provided by Soot (a static analysis toolkit for Java)
- Inter-procedural and context-sensitive analysis
Most malware apps do not satisfy our data dependence requirement (FP = 0, FN = 1)

<table>
<thead>
<tr>
<th>App/Malware Name</th>
<th># of User Inputs/Actions (Source)</th>
<th>% of Sensitive Func. Calls without User Inputs</th>
<th>Library of Sensitive Function Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Legitimate</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SendSMS</td>
<td>3</td>
<td>0%</td>
<td>android.telephony.gsm</td>
</tr>
<tr>
<td>BMI Calculator</td>
<td>2</td>
<td>0%</td>
<td>android.app.Activity</td>
</tr>
<tr>
<td>BluetoothChat</td>
<td>2</td>
<td>0%</td>
<td>java.io.OutputStream</td>
</tr>
<tr>
<td>SendMail</td>
<td>4</td>
<td>0%</td>
<td>android.app.Activity</td>
</tr>
<tr>
<td>Tip Calculator</td>
<td>4</td>
<td>0%</td>
<td>android.widget</td>
</tr>
<tr>
<td><strong>Malicious</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GGTracker.A</td>
<td>0</td>
<td>100%</td>
<td>org.apache.http.impl.client</td>
</tr>
<tr>
<td>HippoSMS</td>
<td>0</td>
<td>100%</td>
<td>android.telephony.gsm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>android.content.ContentResolver</td>
</tr>
<tr>
<td>Fakeneflic</td>
<td>3</td>
<td>0%</td>
<td>org.apache.http.impl.client</td>
</tr>
<tr>
<td>GoldDream</td>
<td>0</td>
<td>100%</td>
<td>android.content.Context</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>java.io.FileOutputStream</td>
</tr>
<tr>
<td>Walk &amp; Text</td>
<td>0</td>
<td>100%</td>
<td>android.content.ContentResolver</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>org.apache.http.impl.client</td>
</tr>
<tr>
<td>RogueSPPush</td>
<td>0</td>
<td>100%</td>
<td>android.telephony.gsm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>android.content.ContentResolver</td>
</tr>
<tr>
<td>Dog Wars</td>
<td>0</td>
<td>100%</td>
<td>android.telephony.gsm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>android.content.ContentResolver</td>
</tr>
</tbody>
</table>
## Security Analysis

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>phishing apps / social engineering apps</td>
<td>site authentication and user education</td>
</tr>
<tr>
<td>using superfluous user inputs and actions</td>
<td>easy to detect by using our approach to track the dependency</td>
</tr>
<tr>
<td>Code obfuscation or Java reflection</td>
<td>dynamic taint analysis</td>
</tr>
</tbody>
</table>
Related Work

- Static program analysis for malware detection
  - [Wagner ’01, Bhatkar ’06, …]
- Malware apps detection based on power consumption
  - [Liu ’09, Dixon ’11]
- Identify leakage of sensitive information
  - Panorama [Yin ’07], TaintDroid [Enck ’10]
    - Dynamic taint analysis
    - Paths between source and sink are expressed as violations
Conclusion and Future Work

- We proposed and implemented a promising approach for malware identification based on user-centric data dependence analysis.
  - Our solution allows the detection of suspicious Android apps and Java programs.

- Future Work
  - To enhance our analysis to handle inter-application Intents.
  - To utilize dynamic taint analysis in addition to our current static program analysis.
  - To extend our evaluation with more mobile apps including Android-based apps and Java ME-based apps.
Acknowledgement

- This work has been supported in part by Security and Software Engineering Research Center (S²ERC), an NSF Industry/University Cooperative Research Center (I/UCRC)
Questions?
kelish@vt.edu