
Sam Malek
Associate Professor
Institute for Software Research
University of California, Irvine
malek@uci.edu

ISR Research Forum, May 27, 2016
Typical App Developer
Many Low Quality Apps

Source: Research2Guidance
Potentially have Access to Lots of Private Data

- Camera
- Microphone
- Accelerometer
- Gravity sensor
- Linear acceleration sensor
- Magnetic field sensor
- Orientation sensor
- Gyroscope
- Light sensor
- Proximity sensor
- Temperature sensor
- Pressure sensor
Perfect Storm

• App markets → best tool ever known to attackers for delivering malicious payload

• Market operators are challenged by the limitations of program analysis
  – Halting problem

• Lots of riches to be gained
  – Premium numbers
  – Adware
  – ...

Malicious Android Apps

• Immense number of Android malware apps
  – 342% growth in 2015

Source: Calyptix Security
Malware Family

• GingerMaster
  – First Android malware using root exploit
  – Steal sensitive info (IMEI, SIM card number, etc.)

• DroidJack
  – No root access required
  – Remote Access Tool
  – Update itself
  – Record phone calls and audio
  – Steal sensitive info

• ...
Countermeasures

• Detection and removal
Countermeasures

• Detection and removal **is not enough**
Countermeasures

• Detection and removal is not enough---identify families
Countermeasures

• Detection and removal is not enough---identify families
• Malware likes to hide
Countermeasures

• Detection and removal is not enough---identify **families**
• Malware likes to **hide**
• Catch them **fast**
Our Research

1. Is it possible to learn what makes an app malicious?

2. If so, is it possible to automatically learn the family of malicious apps?
RevealDroid

- A machine learning-based approach for malware detection and family identification
  - Accurate
  - Highly efficient
  - Obfuscation-resilient
Classifier Construction for Malware Detection

Training Apps

Feature Extraction

Supervised Learning

Malware Classifier
## App Representation for Supervised Learning

<table>
<thead>
<tr>
<th>App</th>
<th>Feature1</th>
<th>Feature2</th>
<th>Feature3</th>
<th>Feature4</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Malicious App Icon]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Malicious</td>
</tr>
<tr>
<td>![Malicious App Icon]</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Malicious</td>
</tr>
<tr>
<td>![Benign App Icon]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Benign</td>
</tr>
<tr>
<td>![Benign App Icon]</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Benign</td>
</tr>
</tbody>
</table>
Supervised Learning for Malware Detection
Supervised Learning for Family Identification
# Feature Selection

<table>
<thead>
<tr>
<th></th>
<th>Perm</th>
<th>Comp</th>
<th>IFilters</th>
<th>Flows</th>
<th>UAPI</th>
<th>PAPI</th>
<th>SAPI</th>
<th>IActions</th>
<th>Reflection</th>
<th>Native</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Obfuscation</strong></td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Feature Examples: Package API (PAPI)

• Numbers of Android API methods invoked by app per package
  – android.telephony
    • TelephonyManager.getCellLocation()
    • CellIdentityLte.getCi()
Feature Examples: Reflective Calls

• Apps may dynamically load libraries/classes through reflection
  – Used frequently to obfuscate malicious behavior

```java
ClassLoader cl = MyClass.getClassLoader();
try {
    Class c = cl.loadClass("MyActivity");

    ... Method m = c.getMethod("onPause", ...);

    ...
    m.invoke(...);
} catch { ... }
```
Feature Examples: Native Calls

- Apps can make system calls and calls to native binaries
  - Analysis of native binaries requires disassembly of ELF files

```asm
1 99ec: e59d0010  ldr  r0, [sp, #16]
2 99f0: e59f13c0  ldr  r1, [pc, #960]
3 99f4: ebfffc3e  bl   8af4 <chmod@plt>
```

Code segment where `chmod` is invoked in `GingerBreak` malware
Labeling and Classifier Selection

• Classifier for detection
  – 2-way classifier with labels “benign” or “malicious”
  – Support Vector Machine (SVM)

• Classifier for family identification
  – $n$-way classifier where $n = \text{the number of families}$
  – Classification and Regression Trees (CART)
Experiments
Experimental Setup

• Prototype built using open-source software
  – Java-based
• Over 23,300 benign and 28,100 malicious apps
  – Collected from Malware Genome, Drebin, and Virus Share repositories
• 68 different malware families
Detection accuracy on non-obfuscated apps

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>95%</td>
<td>85%</td>
<td>90%</td>
</tr>
<tr>
<td>Malicious</td>
<td>89%</td>
<td>96%</td>
<td>92%</td>
</tr>
<tr>
<td>Average</td>
<td>92%</td>
<td>91%</td>
<td>91%</td>
</tr>
</tbody>
</table>

Greater than 90% precision and recall
Family identification accuracy on non-obfuscated apps

<table>
<thead>
<tr>
<th></th>
<th>No. Apps</th>
<th>No. Families</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware Genome</td>
<td>1,250</td>
<td>49</td>
<td>92%</td>
</tr>
<tr>
<td>Virus Share</td>
<td>18,065</td>
<td>68</td>
<td>87%</td>
</tr>
</tbody>
</table>

A random classifier would obtain only 1.5% correct classification rate
Detection accuracy on obfuscated apps

- Testing apps were obfuscated using DroidChameleon
  - Shown to evade all commercial antivirus products
  - String/Array encryption, class renaming, call indirection, etc.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>96%</td>
<td>70%</td>
<td>81%</td>
</tr>
<tr>
<td>Malicious</td>
<td>82%</td>
<td>98%</td>
<td>89%</td>
</tr>
<tr>
<td>Average</td>
<td>89%</td>
<td>84%</td>
<td>85%</td>
</tr>
</tbody>
</table>
## Family identification accuracy on obfuscated apps

<table>
<thead>
<tr>
<th>Malware Genome</th>
<th>1,188</th>
<th>49</th>
<th>94%</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>No. Apps</th>
<th>No. Families</th>
<th>Correct Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware Genome</td>
<td>1,188</td>
<td>49</td>
</tr>
</tbody>
</table>
Performance

<table>
<thead>
<tr>
<th>No. of apps</th>
<th>Feature Extraction</th>
<th>Classification (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 randomly selected</td>
<td>18 (s)</td>
<td>31 (s)</td>
</tr>
</tbody>
</table>

It takes around 30 seconds to run RevealDroid on an app
Department of Homeland Security

- Available for use through the SWAMP portal – https://continuousassurance.org/
Conclusion

• **RevealDroid**
  – A machine-learning based approach for malware detection and family identification
  – Highly accurate, obfuscation resilient, and fast

• **Acknowledgement**
  – Joshua Garcia
  – Mahmoud Hammad
  – Kari Nies
Backup
Mobile Software Ecosystems

• Successful software platforms open themselves to third party developers, resulting in massive product lines
  – E.g., Android app ecosystem
1,200 different Android devices
46 different versions of OS