Morpheus: Automatically Generating Heuristics to Detect Android Emulators

Yiming Jing†, Ziming Zhao†, Gail-Joon Ahn†, and Hongxin Hu†
“You take the blue pill – the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill – you stay in the Wonderland, and I show you how deep the rabbit hole goes. Remember: all I’m offering is truth. Nothing more”.

Morpheus to Neo, The Matrix
Outline

• Android malware, emulators, and red pills
• Design & Implementation: Morpheus
• Discovered red pills
• Evaluation
• Conclusion
Background
Application Stores and Android Malware

1,300,000 apps
100,000,000 users
Background
Emulator-based Dynamic Analysis Systems

• Emulator-based dynamic analysis
  – Google Bouncer
  – Andrubis, SandDroid, TraceDroid ...
Background

Red Pills

• What if malware takes red pills
  – Split personalities [Balzarotti *et al.*, NDSS’10]
  – Dynamic external code loading [Poeplau *et al.*, NDSS’14]
• Example
  – `getDeviceID()` returns “0000...0000”
  – True → emulators; False → real devices
  – IMEI hardcoded in the emulated baseband
Problem Statement

• Existing approaches to discover red pills
  – Ad-hoc
  – Manual

• To comprehensively discover red pills, it is imperative to develop **systematic** and **automated** approaches.
Our Expected Red Pills

- No native code, no root exploit
- Zero or few popular permissions
- Few lines of Java code

```java
final TelephonyManager tm = (TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);
if (tm.getDeviceID().contains("0000000000"))
    stay_dormant();
else
    do_evil();
```
Morpheus Work Flow

- Analyze DAC and MAC policies
- Identify **observable artifacts**
- Retrieve artifacts in emulators and real Android devices
- Extract artifacts whose *presence or contents* indicate discrepancies
- Rank candidate red pills
Sandbox Analyzer
for identifying sources of observable artifacts

• Analyze the DAC and MAC policies to find artifacts that can be observed by apps

• Observe = stat or read

• Rules to identify observable artifacts
  1. DAC rule: objects that are world-readable or under world-listable directories
  2. MAC rule: objects that are accessible by the untrusted_app domain using read-like or stat-like operations
Artifact Retriever
for retrieving artifacts in emulators and devices

• Send a probe into emulators and devices
  – Request all non-signature permissions
  – 3 modules
    1. Directory walker
    2. Reflection-based function caller
    3. Binder IPC caller
Heuristic Extractor
for extracting candidate red pills

• Extract two types of red pills

  – Type E(mulator)
    • True in >50% emulators and <50% real devices

  – Type D(evice)
    • False in >50% emulators and <50% real devices
Heuristic Extractor
for extracting candidate red pills

• Extract two types of red pills

  – Type E(mulator)
    1. `/sys/qemu_trace` exists
    2. The return value of `getDeviceID` contains “0000000000”

  – Type D(evice)
    3. `/proc/uid_stat` exists
    4. The value of `ro.build.tags` contains “release-key”
Heuristic Selector
for ranking candidate red pills

• Mimic a feature selection process
  – Labeled training set: emulator / real device
  – Binary features: extracted candidate red pills

  – Apply random forests to
    • Generate a score for each candidate red pill
    • Remove redundant red pills
Discovered Red Pills

• Experimental setup
  – 3 sources of observable artifacts
    • Files: /proc and /sys
    • Android APIs that have no parameters
    • Android system properties
  – Dataset
    • 16 instances of Android SDK emulators (QEMU)
    • 11 instances of Genymotion™ emulators (VirtualBox)
    • 25 distinct Android smartphones and tablets
## Discovered Red Pills

<table>
<thead>
<tr>
<th></th>
<th>File</th>
<th>API</th>
<th>Property</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>QEMU + VirtualBox</td>
<td>2,121</td>
<td>81</td>
<td>82</td>
<td>2,284</td>
</tr>
<tr>
<td>QEMU Only</td>
<td>2,961</td>
<td>163</td>
<td>132</td>
<td>3,256</td>
</tr>
<tr>
<td>VirtualBox Only</td>
<td>4,782</td>
<td>150</td>
<td>160</td>
<td>5,092</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>9,864</td>
<td>394</td>
<td>374</td>
<td>10,632</td>
</tr>
</tbody>
</table>
Discovered Red Pills

• Most red pills we identified are related to software-emulated hardware
Discovered Red Pills

• Red pills for both QEMU and VirtualBox
  – Network: net.eth0.dns1, isTetheringSupported()
  – Power: /sys/..../cpu1, /sys/.../scaling_driver
  – Audio: /sys/class/i2c-dev
  – USB: /sys/class/usb_device, /sys/module/usbcore
  – Radio: gsm.version.baseband, getDeviceID()
Discovered Red Pills

• QEMU red pills
  – QEMU: qemu.sf.fake_camera
  – Goldfish: /sys/.../goldfish-battery
  – Unsupported HW: /sys/module/bluetooth

• VirtualBox red pills
  – VirtualBox: /sys/module/vboxguest
  – PC HW: /proc/acpi, /sys/module/libata
Evaluation

• Experimental setup
  – Top 10 File, API, and Property red pills
  – 9 dynamic analysis tools/services
    • DroidBox 2.3 and 4.1
    • Andrubis, CopperDroid, SandDroid, TraceDroid
    • Qihu 360, NVISO ApkScan, ForeSafe Scanner
  – 128 distinct Android phones/tablets
Evaluation

• 3 red pills with 100% accuracy
  – /proc/ioports, “\0ff\0:”, E
  – /sys/devices/system/cpu/cpu0/cpufreq,D
  – /sys/devices/virtual/misc/android_adb,D

• 17 red pills with >90% accuracy
Evaluation

• Binder IPC message returned by getDeviceID() contains “\0\0\03”
  – Accuracy = 45.3%
  – Sensitivity = 66.7%, Specificity = 43.8%

  – DroidBox 4.1
    • Dummy IMEI can be bypassed

  – Smartphones sold in China
    • IMEIs do not start with “35”
## Evaluation

<table>
<thead>
<tr>
<th>Category</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>97.8%</td>
</tr>
<tr>
<td>API</td>
<td>62.9%</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>89.5%</td>
</tr>
<tr>
<td>TYPE E(MULATOR)</td>
<td>89.9%</td>
</tr>
<tr>
<td>TYPE D(EVICE)</td>
<td>79.4%</td>
</tr>
<tr>
<td>ARTIFACT ONLY</td>
<td>95.3%</td>
</tr>
<tr>
<td>CONTENT+ARTIFACT</td>
<td>77.0%</td>
</tr>
</tbody>
</table>
Conclusion

• We have proved the threat of malware detecting Android emulators

• We have proposed Morpheus, a framework for systematically discovering red pills

• We have discovered more than 10,000 red pills, characterized them, and measured their accuracies
Dataset and source code to be released at: http://honeynet.asu.edu/morpheus

Yiming Jing
ymjing@asu.edu
http://sefcom.asu.edu

This work was supported in part by the National Science Foundation under Grant CNS-0916688 and National Research Foundation under Grant NRF-2014K1A1A2043029.