AirBag: Boosting Smartphone Resistance to Malware Infection

Chiachih Wu  Yajin Zhou  Kunal Patel
Zhenkai Liang and Xuxian Jiang

North Carolina State University  National University of Singapore
Popularity of Smartphones

40 years ago

Nowadays

Source: theatlantic.com

Source: sandiway.blogspot.com
Popularity of Android Phones

Data Source: http://www.gartner.com
Popularity of Android Phones

Year 2013

Data Source: http://www.gartner.com
Popularity of Android Phones

Year 2013

78.4%

Data Source: http://www.gartner.com
Apps Are Becoming Popular
Apps Are Becoming Popular
Apps Are Becoming Popular
Apps Are Becoming Popular
Apps Are Becoming Popular
Apps Are Becoming Popular
So Are the Malicious Apps
So Are the Malicious Apps

Report: Malware-infected Android apps spike in the Google Play store

Zach Miners
@zachminers
Feb 19, 2014 2:03 PM

The number of mobile apps infected with malware in Google’s Play store nearly quadrupled between 2011 and 2013, a security group has reported.

In 2011, there were approximately 11,000 apps in Google’s mobile marketplace that contained malicious software capable of stealing people’s data and committing fraud, according to the results of a study published Wednesday by RiskIQ, an online security services company. By 2013, more than 42,000 apps in Google’s store contained spyware and information-stealing Trojan programs, researchers said.

Apps designed to personalize people’s Android-based phones were most susceptible, as well as entertainment and gaming apps. Some of the most malicious apps in the Google Play store downloaded since 2011 were Wallpaper Dragon Ball, a wallpaper app, and the games Finger Hockey and Subway Surfers Free Tips.
So Are the Malicious Apps

Malware makers 'tailor' Android threats geographically

Cyber thieves who target Android phones are getting more sophisticated, suggests a report.

Malware makers are tailoring their creations to make the most of conditions in each territory, said the report by mobile security firm Lookout.

In some places such as Russia, Android users were far more likely to encounter malicious code, it said.

The report comes as analysis of apps on Google’s Play store shows a
Server-side Solutions

- Google Play: Bouncer
Server-side Solutions

- **Google Play: Bouncer**

Google's Android platform has become the most popular mobile operating system both among consumers and malware writers, and the company earlier this year introduced the **Bouncer** system to look for malicious apps in the Google Play market. Bouncer, which checks for malicious apps and known malware, is a good first step, but as new work from researchers Jon Oberheide and Charlie Miller shows, it can be bypassed quite easily and in ways that will be difficult for Google to address in the long term.

Oberheide and Miller, both well-known for their work on mobile security, went into their research without much detailed knowledge of how the Bouncer system works. Google has said little publicly about its capabilities, preferring not to give attackers any insights into the system's inner workings. So Oberheide and Miller looked at it as a challenge, an exercise to see how much they could deduce about Bouncer from the outside, and, as it turns out, the inside.

*It can be bypassed quite easily and in ways that will be difficult for Google to address in the long term.*
Server-side Solutions

- Google Play: Bouncer

  Google's Android platform has become the most popular mobile operating system both among consumers and malware writers, and the company earlier this year introduced the Bouncer system to look for malicious apps in the Google Play market. Bouncer, which checks for malicious apps and known malware, is a good first step, but as new work from researchers Jon Oberheide and Charlie Miller shows, it can be bypassed quite easily and in ways that will be difficult for Google to address in the long term.

  Oberheide and Miller, both well-known for their work on mobile security, went into their research without much detailed knowledge of how the Bouncer system works. Google has said little publicly about its capabilities, preferring not to give attackers any insights into the system's inner workings. So Oberheide and Miller looked at it as a challenge, an exercise to see how much they could deduce about Bouncer from the outside, and, as it turns out, the inside.

- Third-party app markets

  It can be bypassed quite easily and in ways that will be difficult for Google to address in the long term.
Client-side Solutions

- Android app sandbox
- Security app
- In-app reference monitor
AirBag

- A light-weight solution to effectively isolate untrusted apps
System Design

Native Android Runtime

User

Kernel

Linux Kernel
System Design

Native Android Runtime

User

Kernel

Linux Kernel

App

App

Malicious App
System Design

Native Android Runtime

Malicious App

User

Kernel

Linux Kernel

App

App
System Design

- App
- App
- Malicious App

Native Android Runtime

User

Kernel

Linux Kernel
System Design

- App
- App
- Malicious App

Native Android Runtime

User

Kernel

Linux Kernel
System Design

![Diagram showing system design with various components such as App, Malicious App, Native Android Runtime, Trusted App, User, Native Android Runtime, Kernel, and Linux Kernel.](image-url)
System Design

- App
- App
- Malicious App

Native Android Runtime

User

Native Android Runtime

Kernel

Linux Kernel

Trusted App
Trusted App

AirBag

Untrusted App

AIR

Malicious App

Linux Kernel
System Design

- App
- App
- Malicious App

Native Android Runtime

User

- Trusted App
- Trusted App

Trusted App

Native Android Runtime

AIR

- Untrusted App

AirBag

Kernel

Linux Kernel

Linux Kernel
Key Techniques

- **App**
- **Native Android Runtime**
- **Context-aware Device Virtualization**
- **Linux Kernel**
- **AIR**
- **Untrusted App**
- **AirBag**

Diagram showing the relationship between these components.
Key Techniques

- Decoupled app isolation runtime (AIR)

Diagram: 
- App
- Native Android Runtime
- Context-aware Device Virtualization
- Linux Kernel
- Untrusted App
- AirBag
- User
- Kernel
Key Techniques

- Decoupled app isolation runtime (AIR)
- Namespace and filesystem isolation
Key Techniques

- Decoupled app isolation runtime (AIR)
- Namespace and filesystem isolation
- Context-aware device virtualization
App Isolation Runtime (AIR)

- Separated (and customized) Android runtime for untrusted apps

- Benefits
  - Isolation: compromised AIR does not affect native Android runtime
  - Customization: different running modes
  - Privacy-awareness: prevent stealthy actions
Namespace and Filesystem Isolation

- Separated name space
  - Benefit: apps inside AirBag cannot interact with outside ones

- Separated filesystem: all modifications are inside AirBag
  - Benefit: does not affect original Android system
  - Bonus: easy to provide “restore to default” feature
Context-aware Device Virtualization

- Multiplexing system resources between AIR and native Android runtime
## Implementation

<table>
<thead>
<tr>
<th>Device</th>
<th>Kernel</th>
<th>AIR based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Nexus One</td>
<td>2.6.35.7</td>
<td>Cyanogenmod 7.1.0 Stable Release</td>
</tr>
<tr>
<td>Google Nexus 7</td>
<td>3.1.10</td>
<td>Cyanogenmod 9 Nightly Build</td>
</tr>
<tr>
<td>Samsung Galaxy S3</td>
<td>3.0.8</td>
<td>Cyanogenmod 9.1.0 Stable Release</td>
</tr>
</tbody>
</table>

Porting for each device is done within one week!

- Three Android devices with different kernel versions
- Less than 2,000 lines of kernel patch
## Context-aware Device Virtualization

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>Audio Playback and Capture</td>
</tr>
<tr>
<td>Framebuffer</td>
<td>Display Output</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processor</td>
</tr>
<tr>
<td>Input</td>
<td>Touchscreen and Buttons</td>
</tr>
<tr>
<td>IPC</td>
<td>Binder IPC Framework</td>
</tr>
<tr>
<td>pmem</td>
<td>Physical Memory Allocator</td>
</tr>
<tr>
<td>Networking</td>
<td>WiFi Network Interface</td>
</tr>
<tr>
<td>Power</td>
<td>Suspend/Resume</td>
</tr>
<tr>
<td>RTC</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>Sensors</td>
<td>Temperature, Accelerometer, GPS</td>
</tr>
<tr>
<td>Telephony</td>
<td>Cellular Radio</td>
</tr>
</tbody>
</table>
## Context-aware Device Virtualization

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>Audio Playback and Capture</td>
</tr>
<tr>
<td>Framebuffer</td>
<td>Display Output</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processor</td>
</tr>
<tr>
<td>Input</td>
<td>Touchscreen and Buttons</td>
</tr>
<tr>
<td>IPC</td>
<td>Binder IPC Framework</td>
</tr>
<tr>
<td>pmem</td>
<td>Physical Memory Allocator</td>
</tr>
<tr>
<td>Networking</td>
<td>WiFi Network Interface</td>
</tr>
<tr>
<td>Power</td>
<td>Suspend/Resume</td>
</tr>
<tr>
<td>RTC</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>Sensors</td>
<td>Temperature, Accelerometer, GPS</td>
</tr>
<tr>
<td>Telephony</td>
<td>Cellular Radio</td>
</tr>
</tbody>
</table>

*Basic devices to bring up AirBag*
Input

- Keeping the namespace info while registering evdev
- Dispatching input events to the active runtime
Evaluation

- Dataset: malware samples from 20 families
- Results: malicious operations are isolated
Case Study: HippoSMS

Infected Video Browser running inside AirBag

A pop-up alert on background SMS behavior
Performance Overhead

### Performance Overhead

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Version</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NenaMark2</td>
<td>2.3</td>
<td>GPU</td>
</tr>
<tr>
<td>SunSpider</td>
<td>0.9.1</td>
<td>CPU/IO</td>
</tr>
<tr>
<td>BrowserMark</td>
<td>2.0</td>
<td>CPU/IO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Version</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NenaMark2</td>
<td>2.3</td>
<td>GPU</td>
</tr>
<tr>
<td>SunSpider</td>
<td>0.9.1</td>
<td>CPU/IO</td>
</tr>
<tr>
<td>BrowserMark</td>
<td>2.0</td>
<td>CPU/IO</td>
</tr>
</tbody>
</table>

**Around 2%**

AnTuTu Benchmark | 0.1% - 5.6%

**Nexus 7**

**CPUINT**

**CPUFP**

**2D**

**3D**
## Power & Memory

### Power consumption

<table>
<thead>
<tr>
<th></th>
<th>Stock Nexus 7 (battery level)</th>
<th>Nexus 7 with AirBag (battery level)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle for 24hrs</td>
<td>91%</td>
<td>89%</td>
</tr>
<tr>
<td>Playing music for 24hrs</td>
<td>66%</td>
<td>63%</td>
</tr>
</tbody>
</table>

### Memory use

<table>
<thead>
<tr>
<th></th>
<th>Stock Nexus 7 (in-use memory)</th>
<th>Nexus 7 with AirBag (in-use memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle for 4hrs</td>
<td>59.31%</td>
<td>60.87%</td>
</tr>
<tr>
<td>Playing music for 4hrs</td>
<td>60.25%</td>
<td>63.70%</td>
</tr>
</tbody>
</table>
Conclusion

- AirBag: a light-weight solution to effectively and efficiently isolate untrusted apps
Yajin Zhou
http://yajin.org