Mobile OS Security

CSE 451 – December 2, 2016

Franziska (Franzi) Roesner
franzi@cs.washington.edu
Today’s Goals

• Introduce some OS security concepts through a case study of mobile OSes, particularly Android.
• Along the way, highlight that it matters how these systems interface with people (users & devs).
Smartphone (In)Security

Users accidentally install malicious applications.

Over 60% of Android malware steals your money via premium SMS, hides in fake forms of popular apps

By Emil Protalinski, Friday, 5 Oct ‘12, 05:50pm
Smartphone (In)Security

Even legitimate applications exhibit questionable behavior.

Hornyack et al.: 43 of 110 Android applications sent location or phone ID to third-party advertising/analytics servers.
Malware in the Wild

Android malware is growing.

FIGURE 2. MOBILE MALWARE SAMPLES SINCE JUNE 2012

KINDSIGHT SECURITY LABS MALWARE REPORT - H1 2014
ALCATEL-LUCENT
What does Mobile Malware Do?

• Unique to phones:
  – Premium SMS messages
  – Identify location
  – Record phone calls
  – Log SMS

• Similar to desktop/PCs:
  – Connects to botmasters
  – Steal data
  – Phishing
  – Malvertising
Mobile Malware Examples

- **DroidDream** (Android)
  - Over 58 apps uploaded to Google app market
  - Conducts data theft; send credentials to attackers

- **Zitmo** (Symbian, BlackBerry, Windows, Android)
  - Poses as mobile banking application
  - Captures info from SMS – steal banking 2\textsuperscript{nd} factors
  - Works with Zeus botnet

- **Ikee** (iOS)
  - Worm capabilities (targeted default ssh password)
  - Worked only on jailbroken phones with ssh installed
Mobile Malware Examples

“ikee is never going to give you up”
## (Android) Malware in the Wild

**What does it do?**

<table>
<thead>
<tr>
<th></th>
<th>Root Exploit</th>
<th>Remote Control</th>
<th>Financial Charges</th>
<th>Information Stealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Net</td>
<td>SMS</td>
<td>Phone Call</td>
</tr>
<tr>
<td># Families</td>
<td>20</td>
<td>27</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td># Samples</td>
<td>1204</td>
<td>1171</td>
<td>1</td>
<td>256</td>
</tr>
</tbody>
</table>

Why all these problems with mobile malware?
Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) design:
1. There may be multiple users who don’t trust each other.
2. Once an application is installed, it’s (more or less) trusted.
Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) design:
1. There may be multiple users who don’t trust each other.
2. Once an application is installed, it’s (more or less) trusted.

```
 FranziBook:Desktop franzi$ whoami
 franzi

 FranziBook:Desktop franzi$ id
 uid=501(franzi) gid=20(staff) groups=20(staff),401(com.apple.sharepoint.group.1),502(access_bpf),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),33(_appstore),100(_lpoperator),204(_developer),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh)

 FranziBook:Desktop franzi$ ls -l hello.txt
 -rw-r--r-- 1 franzi staff 0 Nov 29 10:08 hello.txt

 FranziBook:Desktop franzi$ chmod 700 hello.txt

 FranziBook:Desktop franzi$ ls -l hello.txt
 -rwx------ 1 franzi staff 0 Nov 29 10:08 hello.txt
```
Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) design:
1. There may be multiple users who don’t trust each other.
2. Once an application is installed, it’s (more or less) trusted.

Apps can do anything the UID they’re running under can do.
What’s Different about Mobile Platforms?

• Applications are isolated
  – Each runs in a separate execution context
  – No default access to file system, devices, etc.
  – Different than traditional OSes where multiple applications run with the same user permissions!

• App Store: approval process for applications
  – Market: Vendor controlled/Open
  – App signing: Vendor-issued/self-signed
  – User approval of permissions
More Details: Android

• Based on Linux
• Application sandboxes
  – Applications run as separate UIDs, in separate processes.
  – Memory corruption errors only lead to arbitrary code execution in the context of the particular application, not complete system compromise!
  – (Can still escape sandbox – but must compromise Linux kernel to do so.) ← allows rooting
Rooting and Jailbreaking

• Allows user to run applications with root privileges
  – e.g., modify/delete system files, app management, CPU management, network management, etc.

• Done by exploiting vulnerability in firmware to install su binary.

• Double-edged sword...

• Note: iOS is more restrictive than Android
  – Doesn’t allow “side-loading” apps, etc.
Challenges with Isolated Apps

So mobile platforms isolate applications for security, but...

1. **Permissions**: How can applications access sensitive resources?
   - the rest of today’s lecture

2. **Communication**: How can applications communicate with each other?
   - specific communication APIs (there may be vulnerabilities in how apps use them)
Permission Granting Problem

Smartphones (and other modern OSes) try to prevent such attacks by limiting applications’ access to:

- System Resources (clipboard, file system).
- Devices (camera, GPS, phone, ...).

How should operating system grant permissions to applications?

Standard approach: Ask the user.
State of the Art

Prompts (time-of-use)
State of the Art

**Prompts** *(time-of-use)*

**Manifests** *(install-time)*

*Disruptive, which leads to prompt-fatigue.*
State of the Art

**Prompts** (time-of-use)

- **Disruptive**, which leads to prompt-fatigue.

- In practice, both are **overly permissive**: Once granted permissions, apps can misuse them.

**Manifests** (install-time)

- **Out of context**; not understood by users.
Are Manifests Usable?

Do users pay attention to permissions?

24 observed installations

- 42% looked at permissions
- 42% didn’t look, but aware
- 17% unaware of permissions

... but 88% of users looked at reviews.
Are Manifests Usable?

Do users understand the warnings?

<table>
<thead>
<tr>
<th>Permission</th>
<th>n</th>
<th>Correct Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_CALENDAR</td>
<td>101</td>
<td>46</td>
</tr>
<tr>
<td>CHANGE_NETWORK_STATE</td>
<td>66</td>
<td>26</td>
</tr>
<tr>
<td>READ_SMS1</td>
<td>77</td>
<td>24</td>
</tr>
<tr>
<td>CALL_PHONE</td>
<td>83</td>
<td>16</td>
</tr>
<tr>
<td>WAKE_LOCK</td>
<td>81</td>
<td>27</td>
</tr>
<tr>
<td>WRITE_EXTERNAL_STORAGE</td>
<td>92</td>
<td>14</td>
</tr>
<tr>
<td>READ_CONTACTS</td>
<td>86</td>
<td>11</td>
</tr>
<tr>
<td>INTERNET</td>
<td>109</td>
<td>12</td>
</tr>
<tr>
<td>READ_PHONE_STATE</td>
<td>85</td>
<td>4</td>
</tr>
<tr>
<td>READ_SMS2</td>
<td>54</td>
<td>12</td>
</tr>
<tr>
<td>CAMERA</td>
<td>72</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. \( n \) is the number of respondents. (Internet Survey, \( n = 302 \))
Are Manifests Usable?

Do users act on permission information?

“Have you ever not installed an app because of permissions?”

25 interview responses

- Yes: 20%
- No: 72%
- Probably: 8%
Over-Permissioning

- Android permissions are badly documented.
- Researchers have mapped APIs $\rightarrow$ permissions.

www.android-permissions.org (Felt et al.), http://pscout.csl.toronto.edu (Au et al.)
Permission Re-Delegation

• An application without a permission gains additional privileges through another application.

• **Demo video**

• Settings application is **deputy**: has permissions, and accidentally exposes APIs that use those permissions.
Android 6.0: Prompts!

- **First-use prompts** for sensitive permission (like iOS).
- **Big change!** Now app developers need to check for permissions or catch exceptions.
Improving Permissions: AppFence

Today, ultimatums give app developers an unfair edge in obtaining permissions.

AppFence can enable new interfaces that give users control over the use of their info.

<table>
<thead>
<tr>
<th>The App that User Wishes to Install</th>
</tr>
</thead>
<tbody>
<tr>
<td>THIS APPLICATION HAS ACCESS TO THE FOLLOWING:</td>
</tr>
<tr>
<td>NETWORK COMMUNICATION  full Internet access</td>
</tr>
<tr>
<td>YOUR PERSONAL INFORMATION  read contact data</td>
</tr>
<tr>
<td>PHONE CALLS  read phone state and identity</td>
</tr>
</tbody>
</table>

I’d rather not share all that information just to try this app, but it looks like I have no choice.

I’ll start by giving out only the information I think this app actually needs.
Improving Permissions: User-Driven Access Control

Insight:
A user’s natural UI actions within an application implicitly carry permission-granting semantics.
Improving Permissions: User-Driven Access Control

Our study shows: Many users already believe (52% of 186) – and/or desire (68%) – that resource access follows the user-driven access control model.
New OS Primitive: Access Control Gadgets (ACGs)

Approach: Make resource-related UI elements first-class operating system objects (access control gadgets).

• To receive resource access, applications must embed a system-provided ACG.

• ACGs allow the OS to capture the user’s permission granting intent in application-agnostic way.
Android Fragmentation

• Many different variants of Android (unlike iOS)
  – Motorola, HTC, Samsung, ...

• Less secure ecosystem
  – Inconsistent or incorrect implementations
  – Slow to propagate kernel updates and new versions

[https://developer.android.com/about/dashboards/index.html]
What about iOS?

• Apps are sandboxed
• Encrypted user data
  – See recent news...
• App Store review process is (maybe) stricter
  – But not infallible: e.g., see Wang et al. “Jekyll on iOS: When Benign Apps Become Evil” (USENIX Security 2013)
• No “sideloading” apps
  – Unless you jailbreak