Admin

• **Today/Friday:** mobile platform security

• **Wednesday:**
  – Guest lecture: Christoph Kern, Google (web security)

• **Assignments:**
  – Sign up for HW3 fuzzing access asap
  – Project Checkpoint #2 due Friday
Roadmap

• Mobile malware
• Mobile platforms vs. traditional platforms
• Deep dive into Android
  – Continued Friday
Questions: Mobile Malware

Q1: How might malware authors get malware onto phones?

Q2: What are some goals that mobile device malware authors might have?

Q3: What technical things might malware authors do?
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.

Android flashlight app tracks users via GPS, FTC says hold on
Mobile Malware Attack Vectors

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

- Similar to desktop/PCs:
  - Connects to botmasters
  - Steal data
  - Phishing
  - Malvertising
Malware in the Wild

Android malware grew quickly!
Today: millions of samples.

[Zhou et al.]
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., Unix) 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., Unix) 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., Unix) 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

[Enck et al.]
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.
Android Applications

- **Activities** provide user interfaces.
- **Services** run in the background.
- **BroadcastReceivers** receive messages sent to multiple applications (e.g., BOOT_COMPLETED).
- **ContentProviders** are databases addressable by their application-defined URIs.

- **AndroidManifest.xml**
  - Specifies application components
  - Specifies required permissions
Challenges with Isolated Apps

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

1. **Permissions**: How can applications access sensitive resources?
2. **Communication**: How can applications communicate with each other?
(1) 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)

- "WhereIsMyCar" Would Like to Use Your Current Location
  - Don’t Allow
  - OK

- html5demos.com wants to use your computer's location. Learn

**Manifests** (install-time)

- Twitter
  - Accept & download

- Permissions:
  - Storage: Modify/delete SD card contents
  - System tools: Prevent phone from sleeping, write sync settings
  - Your location: Fine (GPS) location
  - Network communication
State of the Art

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

*Disruptive, which leads to prompt-fatigue.*

**Manifests** (install-time)

![Manifest Permissions Example](image)
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?

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

... but 88% of users looked at reviews.

[Felt et al.]
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>READCALENDAR</td>
<td>101</td>
<td>46</td>
</tr>
<tr>
<td>CHANGENETWORKSTATE</td>
<td>66</td>
<td>26</td>
</tr>
<tr>
<td>READSMS1</td>
<td>77</td>
<td>24</td>
</tr>
<tr>
<td>CALLPHONE</td>
<td>83</td>
<td>16</td>
</tr>
<tr>
<td>WAKE_LOCK</td>
<td>81</td>
<td>27</td>
</tr>
<tr>
<td>WRITEEXTERNALSTORAGE</td>
<td>92</td>
<td>14</td>
</tr>
<tr>
<td>READCONTACTS</td>
<td>86</td>
<td>11</td>
</tr>
<tr>
<td>INTERNET</td>
<td>109</td>
<td>12</td>
</tr>
<tr>
<td>READPHONESTATE</td>
<td>85</td>
<td>4</td>
</tr>
<tr>
<td>READSMS2</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: 8%
- No: 20%
- Probably: 72%