CSE 484 / CSE M 584
Computer Security:
Android Security

TA: Franzi Roesner
franzi@cs.washington.edu
Logistics

- **Homework #3** due tomorrow, 5pm.
- **Lab #3** due Friday, March 15, 11pm.
  - Email Ian or me for usernames!
- Lab #2 grades up, contact us with questions.
- Next week (3/14): Last section, final review.
- **Final exam:** Tuesday, 3/19, 2:30-4:30pm
Android Application Isolation

• Each app runs with its own user ID.
  – Android isolates them from each other.
  – Different from desktops!

Application Permissions

- Apps must request permissions to access sensitive resources.
  - INTERNET,
    ACCESS_COARSE_LOCATION,
    ACCESS_FINE_LOCATION, CAMERA,
    CALL_PHONE, READ CALENDAR,
    READ_PHONE_STATE, SEND_SMS,
    REBOOT, and many more.

- Permissions requested from users at install-time: not optional!
Are Manifests Usable?

Do users pay attention to permissions?

[Diagram showing 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><strong>1 Choice</strong></td>
<td></td>
<td></td>
</tr>
<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><strong>2 Choices</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAKELOCK</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><strong>4 Choices</strong></td>
<td></td>
<td></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?”

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

25 interview responses

Over-Permissioning

- Android permissions are badly documented.
- Researchers have mapped APIs → permissions.

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

![Percentage Pie Chart]

- Overprivileged: 30.4%
- Possible false positives: 67.3%
- Not overprivileged: 2.3%

![Bar Chart]

Number of extra permissions:
- 1: 20%
- 2: 15%
- 3: 5%
- 4+: 20%

[From Felt et al., “Android Permissions Demystified”, CCS 2011.]
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 Application Components

- **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.

- Specified in each app’s AndroidManifest.xml.
Inter-Process Communication

• Primary mechanisms: intents
  – Sent between application components
    • e.g., with `startActivity(intent)`
  – Explicit: specify component name
    – e.g., `com.example.testApp.MainActivity`
  – Implicit: specify action (e.g., `ACTION_VIEW`) and/or data (URI and MIME type)
  – Apps specify Intent Filters for their components.
Unauthorized Intent Receipt

• Attack #1: Eavesdropping / Broadcast Thefts
  – Implicit intents make intra-app messages public.
• Attack #2: Activity Hijacking
  – May not always work:
• Attack #3: Service Hijacking
  – Android picks one at random upon conflict!

[From Chin et al., “Analyzing Inter-Application Communication in Android”, MobiSys 2011.]
Intent Spoofing

• Attack #1: General intent spoofing
  – Receiving implicit intents makes component public.
  – Allows data injection.

• Attack #2: System intent spoofing
  – Can’t directly spoof, but victim apps often don’t check specific “action” in intent.

[From Chin et al., “Analyzing Inter-Application Communication in Android”, MobiSys 2011.]
Information Leaks and Tracking

- Many apps include advertising or analytics libraries.
  - Unlike on the web (where we have iframes), these libraries always run with the host application’s permissions.

These libraries may leak data.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Demanded</th>
<th>Anywhere</th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>phone_state</td>
<td>83</td>
<td>31</td>
<td>14 17%</td>
</tr>
<tr>
<td>Phone#</td>
<td>83</td>
<td>5</td>
<td>0 0%</td>
</tr>
<tr>
<td>location</td>
<td>73</td>
<td>45</td>
<td>30 41%</td>
</tr>
<tr>
<td>contacts</td>
<td>29</td>
<td>7</td>
<td>0 0%</td>
</tr>
<tr>
<td>camera</td>
<td>12</td>
<td>1</td>
<td>0 0%</td>
</tr>
<tr>
<td>account</td>
<td>11</td>
<td>4</td>
<td>0 0%</td>
</tr>
<tr>
<td>logs</td>
<td>10</td>
<td>0</td>
<td>0 0%</td>
</tr>
<tr>
<td>microphone</td>
<td>10</td>
<td>1</td>
<td>0 0%</td>
</tr>
<tr>
<td>SMS/MMS messages</td>
<td>10</td>
<td>0</td>
<td>0 0%</td>
</tr>
<tr>
<td>history&amp;bookmarks</td>
<td>10</td>
<td>0</td>
<td>0 0%</td>
</tr>
<tr>
<td>calendar</td>
<td>8</td>
<td>0</td>
<td>0 0%</td>
</tr>
<tr>
<td>subscribed_feeds</td>
<td>1</td>
<td>0</td>
<td>0 0%</td>
</tr>
</tbody>
</table>


[From Hornyack et al., “These Aren’t the Droids You’re Looking For: Retrofitting Android to Protect Data from Imperious Applications”, CSS 2011.]
Rooting

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

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

Android malware is growing.

Malware in the Wild

What does it do?

<table>
<thead>
<tr>
<th># Families</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>20</td>
<td>20</td>
<td>27</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1204</td>
<td>1204</td>
<td>1171</td>
<td>1</td>
<td>256</td>
</tr>
</tbody>
</table>

Defensive Research for Android

- Separating ads from apps
  - AdDroid (Felt et al.), AdSplit (Shekhar et al.)
- User-driven access control (Roesner et al.)
- Dynamic information flow tracking
  - e.g., TaintDroid (http://appanalysis.org/), AppFence (http://appfence.org/)
- Static analysis for malware detection
  - e.g., SPARTA (http://www.cs.washington.edu/sparta/)
- Many more!