CSE 484 / CSE M 584: Computer Security and Privacy

Mobile Usability

Autumn 2018

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Ada Lerner, John Manferdelli, John Mitchell, Franziska Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials...
Admin

• HW 3 due Nov 30
• Lab 3 out today (this afternoon), due Dec 7 (Quiz Section on Nov 29)

• Wednesday evening lecture (tonight): *Extra credit* in-class assignment

• Next Monday: Guest Lecturer: Emily McReynolds, Microsoft
• Next Wednesday: Ivan Evtimov, Adversarial Machine Learning
• Next Friday: No lecture – extra time to work on your projects and labs
  – But there is an *extra credit* in-class assignment, if you would like (2 more Enigma talks)
Admin

- Final Project Proposals: Looked great!
- Final Project Checkpoint: Nov 30 – preliminary outline and references
- Final Project Presentation: Dec 10 – 12-15-minute video – **must** be on time
- Explore something of interest to you, that could hopefully benefit you or your career in some way – technical topics, current events, etc
Review: 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?
Review: Two Ways to Ask the User

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.
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.
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.)
Let this application access my location now.

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

Study:
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.
Permission Re-Delegation

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

• Settings application is deputy: has permissions, and accidentally exposes APIs that use those permissions.
Aside: Incomplete Isolation

Embedded UIs and libraries always run with the host application’s permissions! (No same-origin policy here…)

[Shekhar et al.]

[chart showing number of ad libraries installed]

Ad from ad library
Map from Google library
Social button from Facebook library
Android Application Signing

• Apps are signed
  – Signed application certificate defines which user ID is associated with which applications
  – Different apps run under different UIDs

• Shared UID feature
  – Shared Application Sandbox possible, where two or more apps signed with same developer key can declare a shared UID in their manifest
Shared UIDs

• App 1: Requests GPS / camera access
• App 2: Requests Network capabilities

• Generally:
  – First app can’t exfiltrate information
  – Second app can’t exfiltrate anything interesting
• With Shared UIDs (signed with same private key)
  – Permissions are a superset of permissions for each app
  – App 1 can now exfiltrate; App 2 can now access GPS / camera
File Permissions

• Files written by one application cannot be read by other applications
  – Previously, this wasn’t true for files stored on the SD card (world readable!) – Android cracked down on this

• It is possible to do full file system encryption
  – Key = Password/PIN combined with salt, hashed
Android Permission Recommendations

• Only use the permissions necessary for your app to work
• Pay attention to permissions required by libraries
• Be transparent
• Make system accesses explicit. Providing continuous indications when you access sensitive capabilities (for example, the camera or microphone) ...

https://developer.android.com/training/permissions/usage-notes
(2) Inter-Process Communication

• Primary mechanism in Android: **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!
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.
Memory Management

- Address Space Layout Randomization to randomize addresses on stack
- Hardware-based No eXecute (NX) to prevent code execution on stack/heap
- Stack guard derivative
- Some defenses against double free bugs (based on OpenBSD’s dmalloc() function)
- etc.

[See http://source.android.com/tech/security/index.html]
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]

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.3 - 2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>1.0%</td>
</tr>
<tr>
<td>4.0.3 - 4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>0.8%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>3.2%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>4.6%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>1.3%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>18.8%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>8.7%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>23.3%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>31.2%</td>
</tr>
<tr>
<td>7.0</td>
<td>Nougat</td>
<td>24</td>
<td>6.6%</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>25</td>
<td>0.5%</td>
</tr>
</tbody>
</table>

Data collected during a 7-day period ending on May 2, 2017. Any versions with less than 0.1% distribution are not shown.
What about iOS?

- Apps are sandboxed
- Encrypted user data
- 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)
What’s Next?

• This about these issues for the next computing platform
  – Augmented Reality?
  – Cars?
  – Smarthomes?
Usability
On Usability

• Why is usability important?
  – People are the critical element of any computer system
    • People are the real reason computers exist in the first place
  – Even if it is possible for a system to protect against an adversary, people may use the system in other, less secure ways
  – Usability errors can lead people to think that they are using a secure setting when in fact they are not (e.g., certain password managers)
Root Causes?

• Computer systems are complex; users lack intuition
• Users in charge of managing own devices
  – Unlike other complex systems, like healthcare or cars.
• Hard to gauge risks
  – “It won’t happen to me!”
• Annoying, awkward, difficult
• Social issues
  – Send encrypted emails about lunch?...
Question

• What does usable security mean?
• What does it mean for a system to have usable security?
How to Improve?

• Security education and training
• Help users build accurate mental models
• Make security invisible
• Make security the least-resistance path
• ...?