CSE 484: Computer Security and Privacy

Mobile Devices

Spring 2024

David Kohlbrenner

dkohlbre@cs

Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, David Kohlbrenner, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Logistics

- FP Part A due today
  - We'll get feedback to you quickly on FP parts
  - Prioritizing these over other things

- We've seen a couple of different heap bugs people are running in to
  - Some of these are _existing_ bugs in tinyserv, you don't need to fix these
  - Buggy code tends to be buggy :/

Mobile devices

What is the difference?

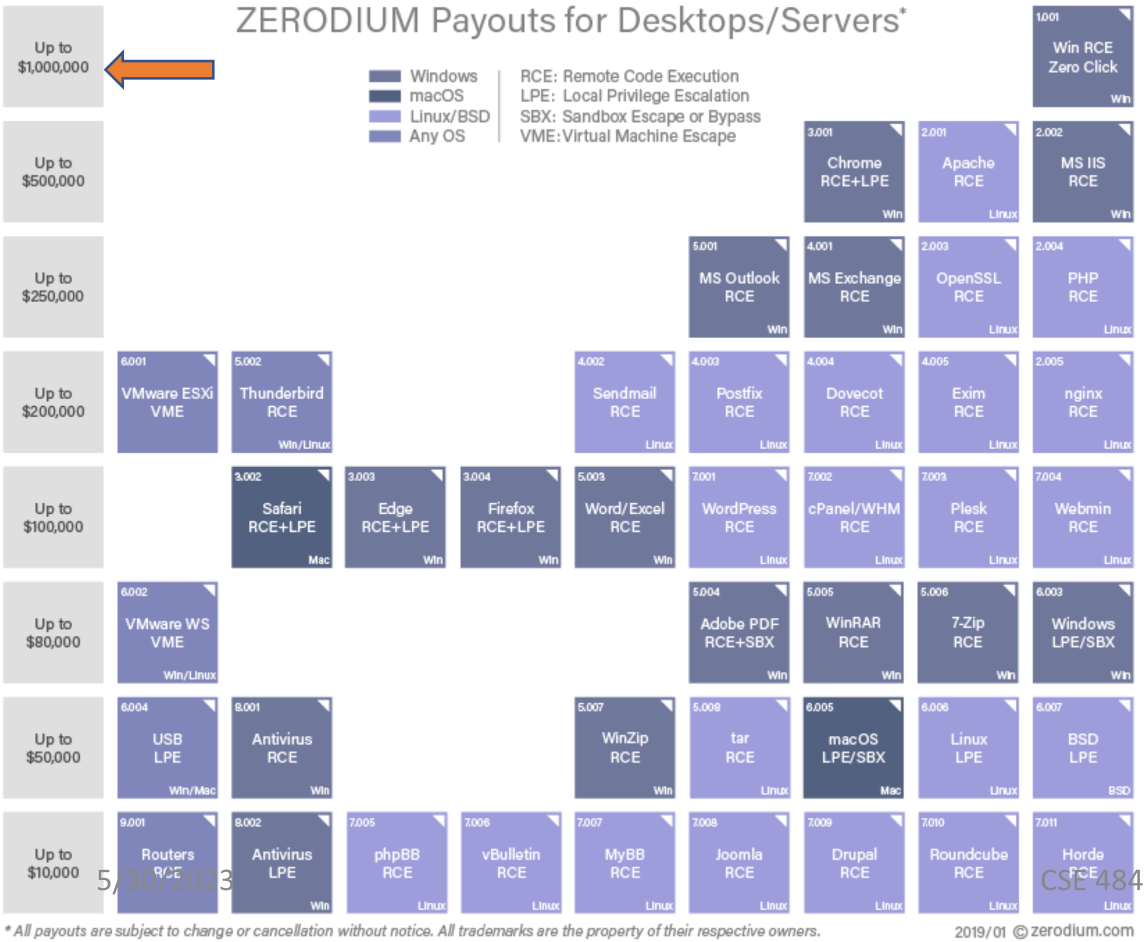
- Mobile devices (smartphones)
- Tablets
- Laptops
- Desktops
- Servers

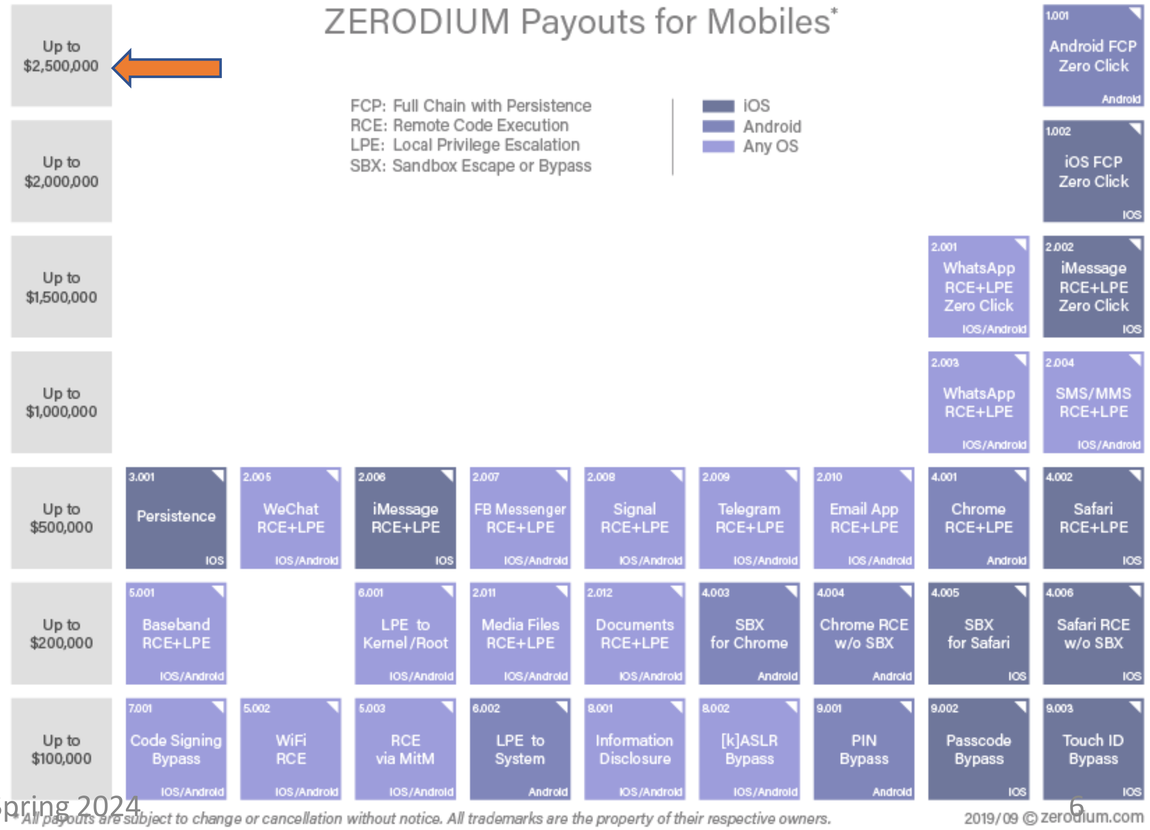
A surprising difference

Mobile security is *really really good*

A surprising difference

Mobile security is *really really good*





Why?

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.

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

Why isolate on mobile devices and not PCs?

- Application isolation is *great!*

- Phones drew lessons from desktops
- Desktops draw lessons from phones
- Browsers learning too

- App Isolation sometimes available for PCs
  - Windows 10 Sandbox (May 2019)

- Browsers: Site Isolation

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

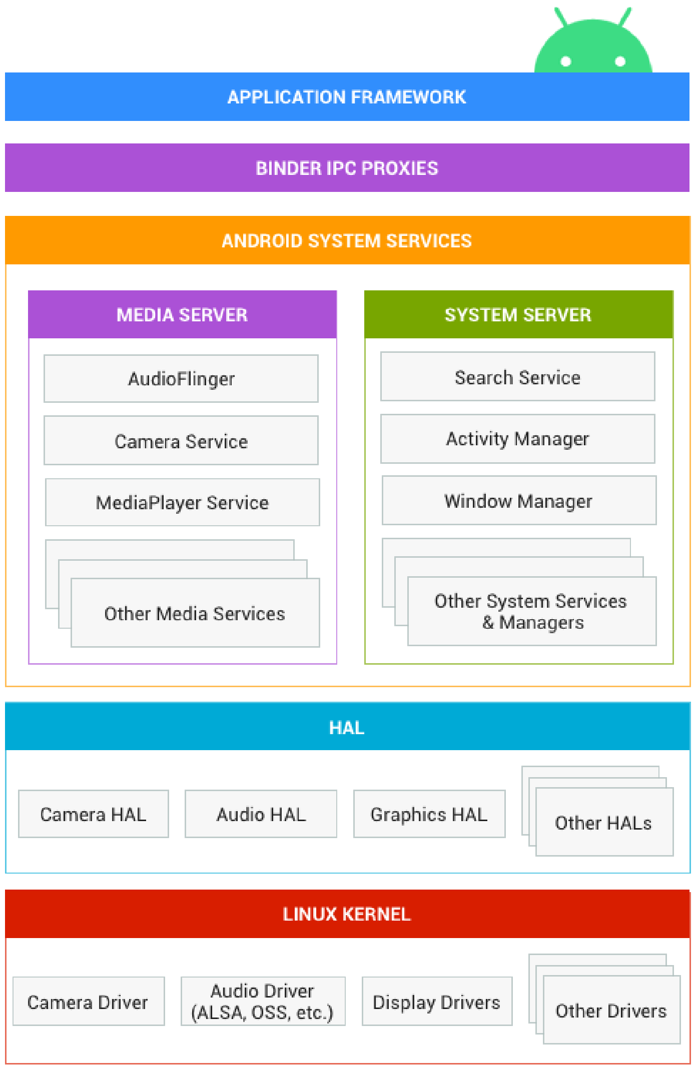


Figure 1. Android system architecture

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?

What can go wrong?

“Threat Model” 1: Malicious applications

What can go wrong?

Threat Model 1: Malicious applications

Example attacks:

- Premium SMS messages
- Track location
- Record phone calls
- Log SMS
- Steal data
- Phishing

Some of these are unique to phones (SMS, rich sensor data)

What can go wrong?

Threat Model 2: Vulnerable applications

Example concerns:

- User data is leaked or stolen
  - (on phone, on network, on server)
- Application is hijacked by an attacker

(1) Permission Granting Problem

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

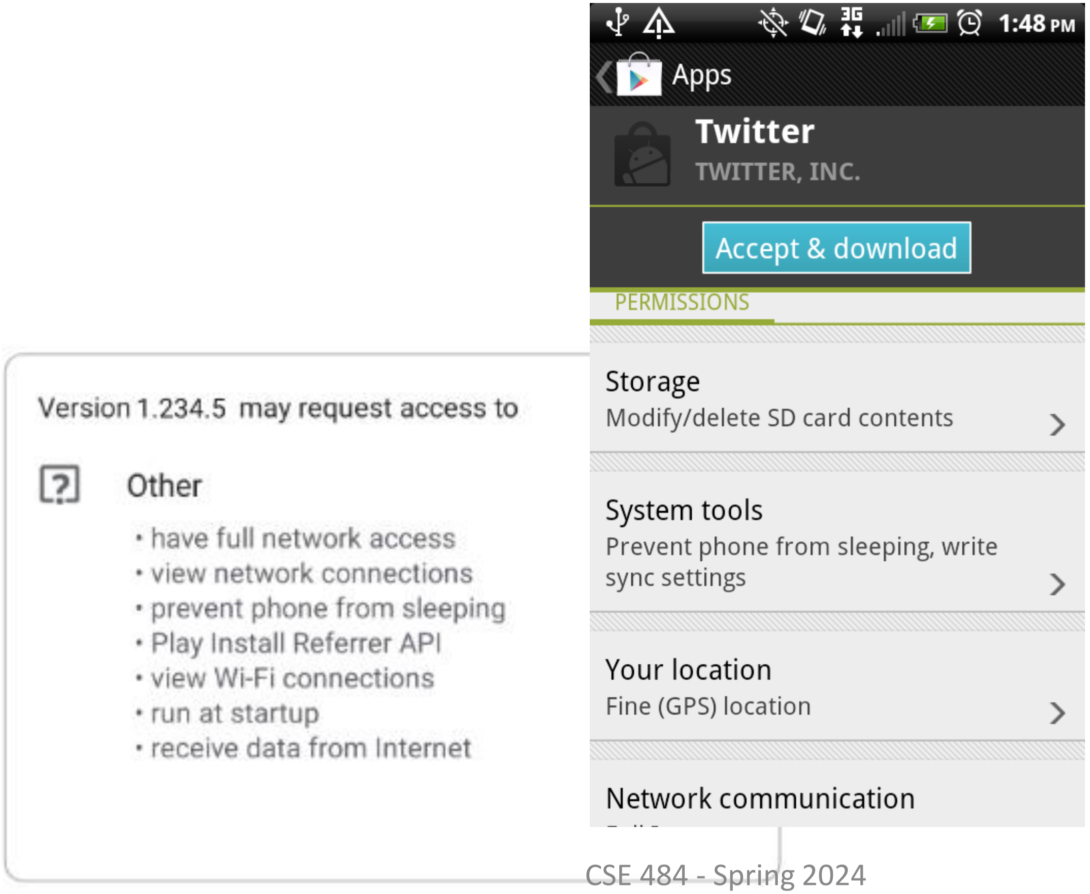
- System Resources (clipboard, file system).
- Devices (camera, GPS, phone, …).

How should operating system grant permissions to applications?

Standard approach: Ask the user.

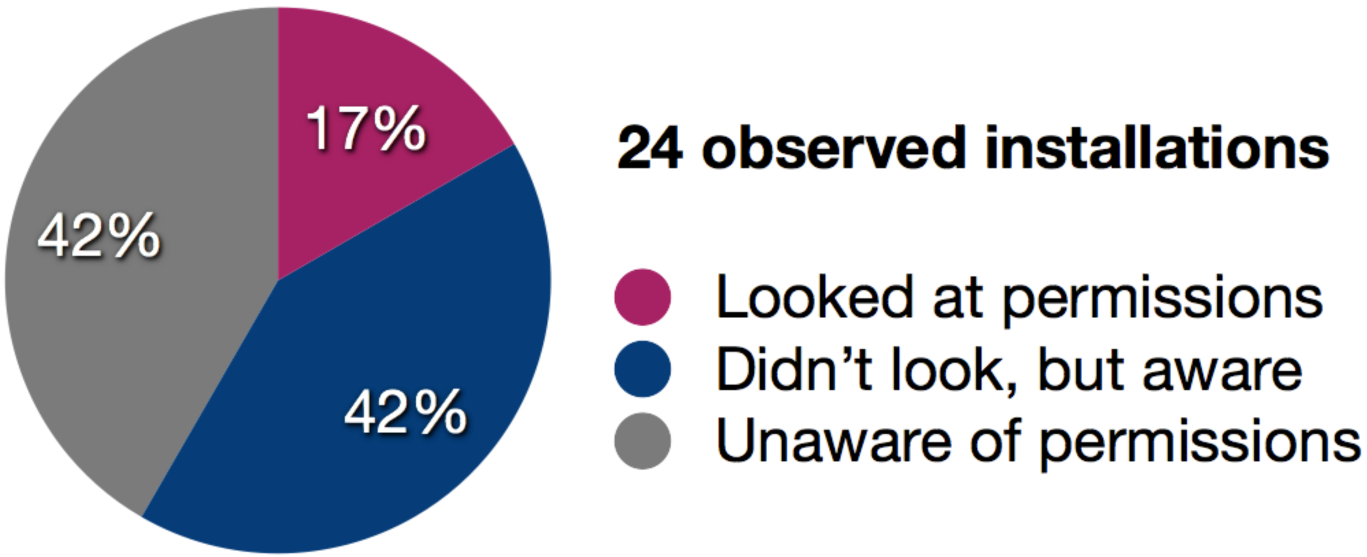
Android’s old approach: Manifests

- Big list of things the app wants at install time



Are Manifests Usable?

Do users pay attention to permissions?



… but 88% of users looked at reviews.

[Felt et al.]

Are Manifests Usable?

Do users understand the warnings?

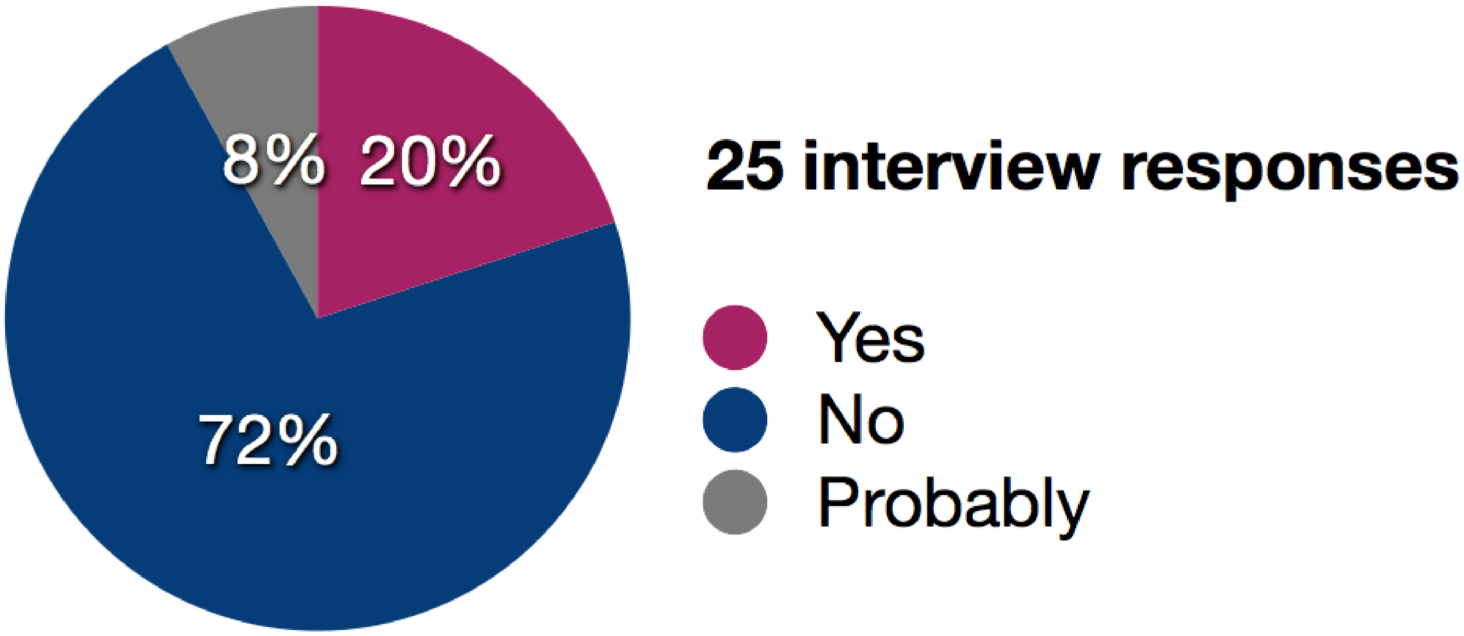
	Permission	n	Correct Answers	
1 Choice	READ_CALENDAR	101	46	45.5%
	CHANGE_NETWORK_STATE	66	26	39.4%
	$READ\_SMS_1$	77	24	31.2%
	CALL_PHONE	83	16	19.3%
2 Choices	WAKE_LOCK	81	27	33.3%
	WRITE_EXTERNAL_STORAGE	92	14	15.2%
	READ_CONTACTS	86	11	12.8%
	INTERNET	109	12	11.0%
	READ_PHONE_STATE	85	4	4.7%
	$READ\_SMS_2$	54	12	22.2%
4	CAMERA	72	7	9.7%

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?

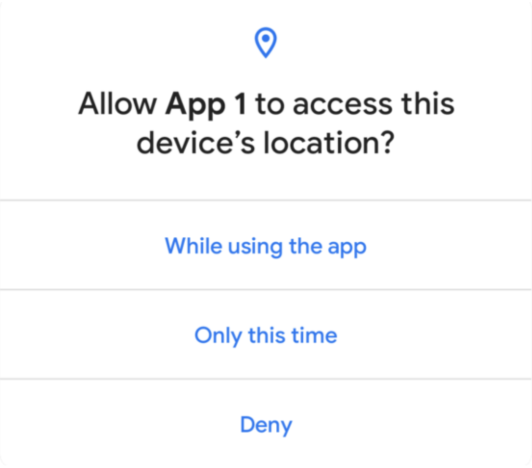
Do users act on permission information?

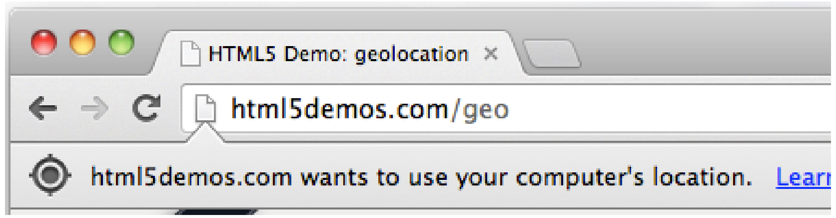
"Have you ever not installed an app because of permissions?"



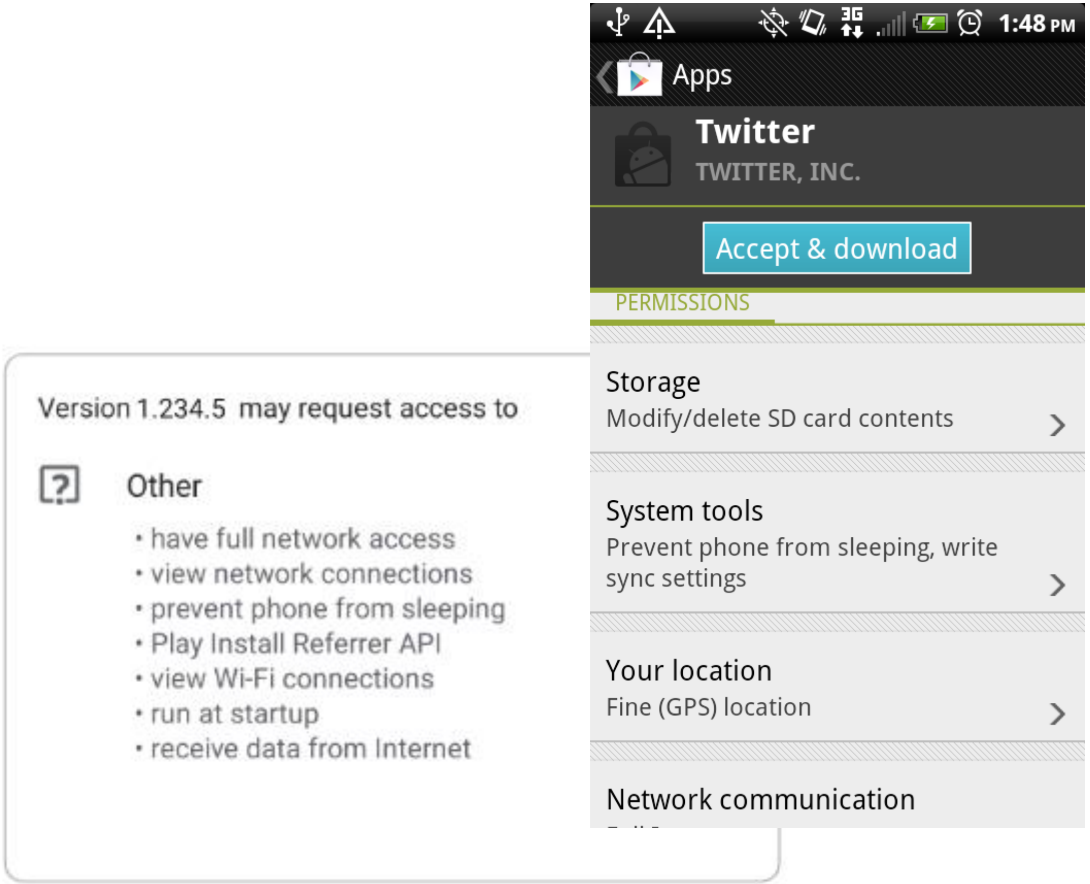
State of the Art

Prompts (time-of-use)

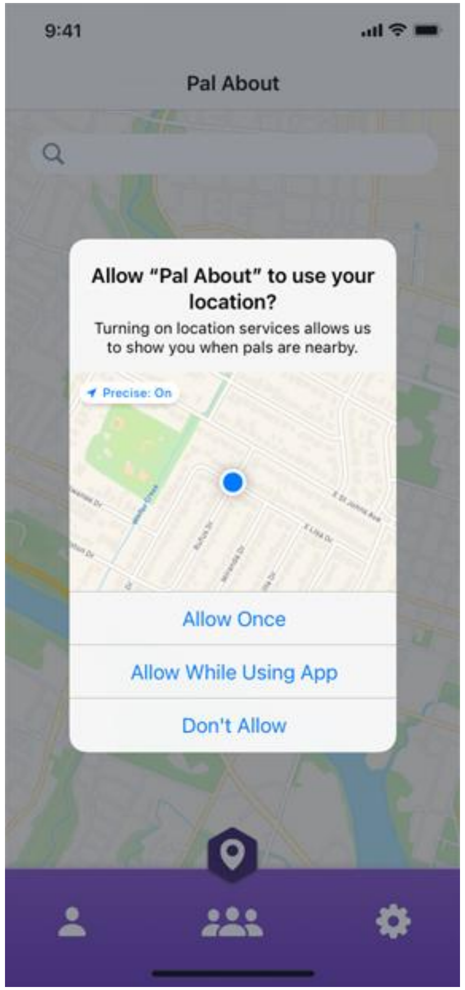


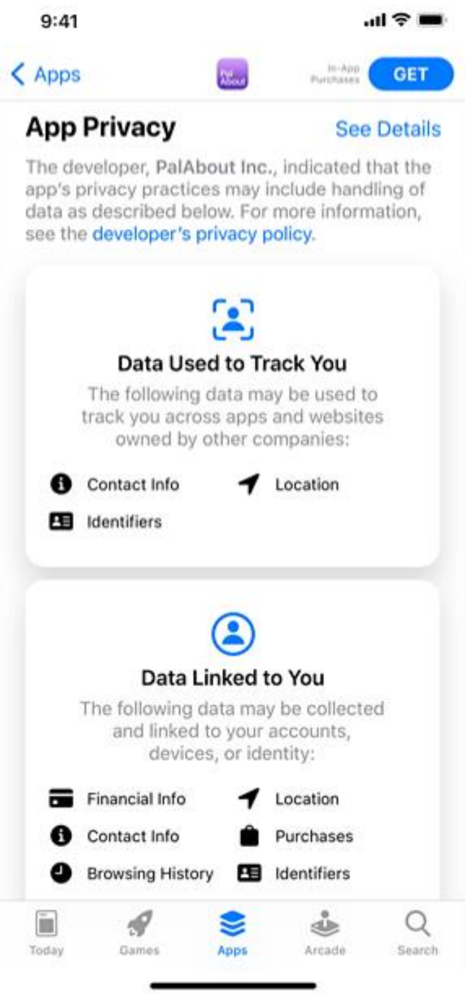


Manifests (install-time, old model)



State of the Art (iOS)





https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/accessing-user-data/

(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.

Eavesdropping and Spoofing

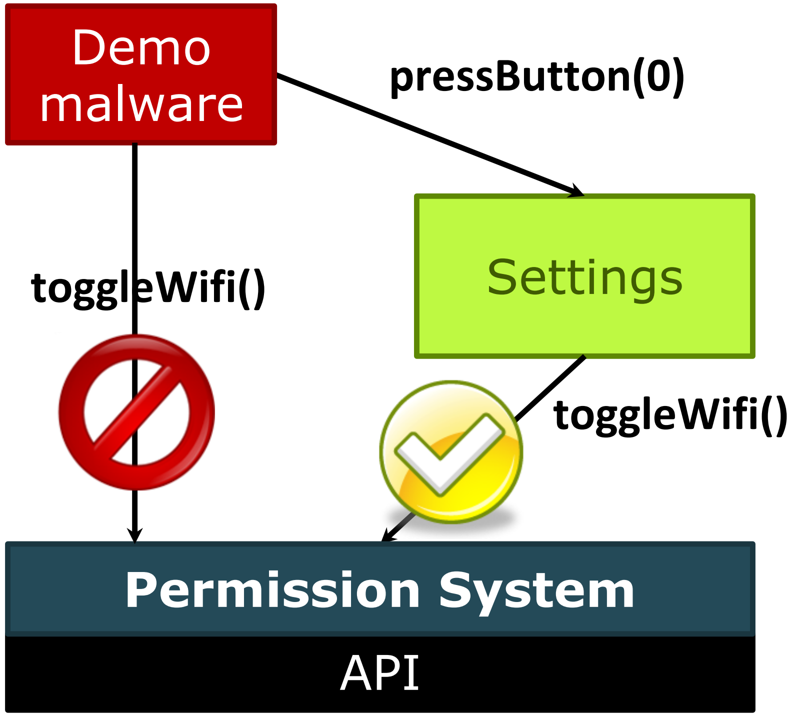
- Buggy apps might accidentally:
  - Expose their component-to-component messages publicly → eavesdropping
  - Act on unauthorized messages they receive → spoofing

[Felt et al.]

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.



Other Android Security Features

- Secure hardware
- Full disk encryption
- Modern memory protections (e.g., ASLR, non-executable stack)
- Application signing
- App store review

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

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
  - Many changes made in past few years (e.g. Project Treble)

Android Platform Version (API Level)	Distribution (as of April 10, 2020)
Android 4.0 "Ice Cream Sandwich" (15)	0.2%
Android 4.1 "Jelly Bean" (16)	0.6%
Android 4.2 "Jelly Bean" (17)	0.8%
Android 4.3 "Jelly Bean" (18)	0.3%
Android 4.4 "KitKat" (19)	4%
Android 5.0 "Lollipop" (21)	1.8%
Android 5.1 "Lollipop" (22)	7.4%
Android 6.0 "Marshmallow" (23)	11.2%
Android 7.0 "Nougat" (24)	7.5%
Android 7.1 "Nougat" (25)	5.4%
Android 8.0 "Oreo" (26)	7.3%
Android 8.1 "Oreo" (27)	14%
Android 9 "Pie" (28)	31.3%
Android 10 (29)	8.2%

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

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.

What about iOS?

- Apps are sandboxed
- Encrypted user data
  - Often in the news…
- App Store review process is (was? 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

iOS model vs Android

- Monolithic vs fragmented
- Closed vs open
- Single distributor vs many

Lessons Being Learned from Other Spaces

- Mobile phone platforms built on lessons learned from desktops
- Desktops and Browsers learning from Mobile phones
- Overall, trying to increase security for all platforms