CSE 484: Computer Security and Privacy

Physical Security + Stepping Back + Mobile

Spring 2021

Tadayoshi Kohno

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 ...

Last Time: Physical Security + Lockpicking

- If you're interested in the subject of lockpicking, see:
  - Blaze, "Cryptology and Physical Security: Rights Amplification in Master-Keyed Mechanical Locks"
  - Blaze, "Safecracking for the Computer Scientist"
  - Tool, "Guide to Lock Picking"
  - Tobias, "Opening Locks by Bumping in Five Seconds or Less"

Adversarial Goals: Connecting Physical Security and Computer Security

- **Confidentiality** … adversary should not be able to enter and steal information (e.g., see the spy movies, or think about bank computer screens facing windows)
- **Integrity** … adversary should not be able to enter property and remove items, or damage items, or place new items (e.g., installing spy device)
- **Availability** … adversary should no be able to deny legitimate entry (denial of service) into an environment (e.g., put superglue in a lock, or gum, or break a wrong key in lock)

Threat Modeling (Security Reviews)

- Benefits/Harms: What are benefits and harms of technology?
- Adversaries: Who might try to attack, and why?
- Vulnerabilities: How might the system be weak?
- Threats: What actions might an adversary take to exploit vulnerabilities?
- Risk: How important are assets? How likely is exploit?
- Possible Defenses

- E.g., Different defenses and considerations might be appropriate in different situations (e.g., gym locker, bank, nuclear weapons silos)
- E.g., Different adversaries (insiders, like former tenants or ex-employees, or outsiders)

Approaches to Security

- Prevention
  - Stop an attack
  - E.g., door locks and fences and bars on windows in physical world environment
- Detection
  - Detect an ongoing or past attack
  - E.g., video camera in physical world environment
- Response
  - Respond to attacks
  - E.g.., home alarm system that calls police when entry is detected

Whole System is Critical

- Securing a system involves a whole-system view
  - Cryptography
  - Implementation
  - People
  - Physical security
  - Everything in between

- This is because "security is only as strong as the weakest link," and security can fail in many places
  - No reason to attack the strongest part of a system if you can walk right around it.

Overlapping Defensive Ideas

- Defense in Depth
  - Layers, e.g., cardkey access then physical keys
- Deterrents (which can also be layers)
  - Home alarm systems
  - Cameras
- Least Privilege
  - At UW:
    - Grad keys can open certain doors
    - Faculty keys can open all those doors and more doors
    - Custodial keys can open even more doors
    - (see previously cited document from Matt Blaze to understand how this works)

Saltzer and Schroeder (1975 paper)

- See the paper: http://web.mit.edu/Saltzer/www/publications/protection/
- Wikipedia's summary of principles on next slide (since Wikipedia summary is shorter): https://en.wikipedia.org/wiki/Saltzer_and_Schroeder%27s_design_principles
  - Connections and insights can be made by thinking about these principles in the context of physical security

Saltzer and Schroeder (1975 paper)

- **Economy of mechanism:** Keep the design as simple and small as possible.
- **Fail-safe defaults:** Base access decisions on permission rather than exclusion.
- **Complete mediation:** Every access to every object must be checked for authority.
- **Open design:** The design should not be secret.
- **Separation of privilege:** Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key.
- **Least privilege:** Every program and every user of the system should operate using the least set of privileges necessary to complete the job.
- **Least common mechanism:** Minimize the amount of mechanism common to more than one user and depended on by all users.
- **Psychological acceptability:** It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly.
- **Work factor:** Compare the cost of circumventing the mechanism with the resources of a potential attacker.
- **Compromise recording:** It is sometimes suggested that mechanisms that reliably record that a compromise of information has occurred can be used in place of more elaborate mechanisms that completely prevent loss.

What’s Wrong With This Picture?

What's Wrong With This Picture?

Think About the Whole System

Usability

- Usability is so important, that the importance of usability has permeated much of this course
- But let's now take a few moments to consider usability specifically

- And I encourage everyone to consider taking an HCI course!
- And to always think about *all* the stakeholders that might be impacted by a system
  - Direct stakeholders
  - Indirect stakeholders
- Developers are users too ☺ (i.e., consider making it easy/usable to develop secure solutions)

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)

Question

- What does usable security mean?
- What does it mean for a system to have usable security?

How to Improve?

- These are all **concepts** that people have discussed (not that everyone agrees):
  - Security education and training
  - Help users build accurate mental models
  - Find ways to make systems better match people's natural mental models
  - Make security invisible
  - Make security the least-resistance path
- **On your own:** Think about usability challenges that you have encoutered, with respect to security, and what would have made those systems more usable
- **Big recommendation:** Think proactively about all stakeholders (not just people similar to the system designers)

Social Engineering

- Art or science of skillfully maneuvering human beings to take action in some aspect of their lives
  - From Social Engineering: The Art of Human Hacking by Christopher Hadnagy
  - (Also see: The Art of Deception: Controlling the Human Element of Security by Kevin Mitnick and William Simon)
- Used by
  - Hackers
  - Penetration testers
  - Spies
  - Identity thieves
  - Disgruntled employees
  - Scam artists
  - Executive recruiters
  - Salespeople
  - Governments

Example

- Hello?
- Hello?
- Hello?
- You called me?
- You called me?
- There’s something wrong with this phone – what kind of phone do you have?
- (From DEFCON social engineering competition winner)

Example

- Take this survey, win and iPhone
- Call “victims”, to explain that they were victims of a phishing training, which they failed, and now need to clear up their computer
- Have them download and install clean up software
- Yes, okay to bypass “unknown source” warning for the software install
- Okay, great, now next, I need you to now change your password on this main system…
- Good, good, you are clearly a responsible employee. Thank you for taking this so seriously. Now I need you to download a new certificate for your directory server, let me tell you how…
- (Inspired by a talk by Chris Hadnagy, though I might have exact words wrong)

Example from Mark Seiden

- Every time he pen tests a company, he carries with him a printed document that says
  - “This person is doing a pen test of security, authorized by the CEO”
  - “If you have any questions, call this number <number>”
  - Signed by the CEO
- 50% of times that he is stopped by a security guard, he shows them the paper and they say “oh, okay, that makes sense”, and then lets him proceed
- 50% of the remaining 50% of the times: the security guard calls the phone number *on the paper…*

Next: Mobile Platform Security

Roadmap

- Mobile malware
- Mobile platforms vs. traditional platforms
- Dive into (evolution of) Android

Mobile Malware: Threat Modeling

Q1: How might malware authors get malware onto phones?

Q2: What are some goals that mobile device malware authors might have, or technical attacks they might attempt? How might this differ from desktop settings?

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

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)
  - Prerequisites
    - Windows 10 May 2019 update version 1903 installed
    - Hardware virtualization enabled
    - Windows 10 Pro or Enterprise
- 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

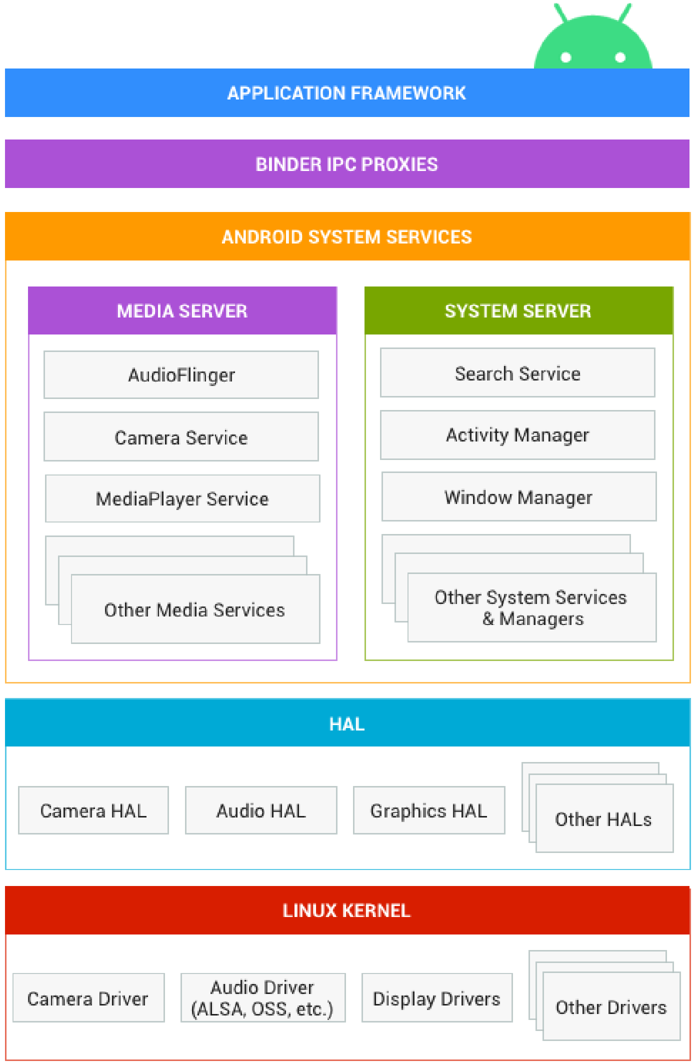


Figure 1. Android system architecture

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?

Next Slides

- Not presenting next slides; previous slides cover main content
- Following slides available, for those interested in learning more

(1) Permission Granting Problem

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

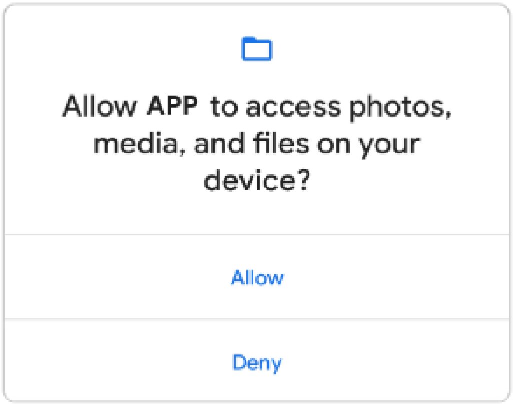
- System Resources (clipboard, file system).
- Devices (camera, GPS, phone, …).

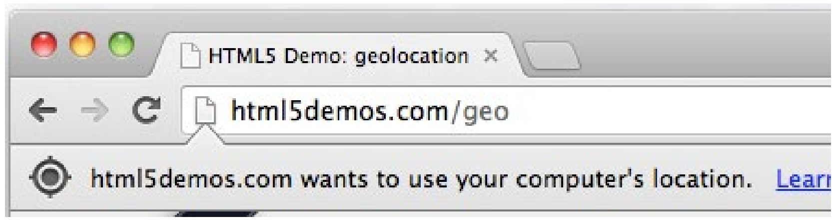
How should operating system grant permissions to applications?

Standard approach: Ask the user.

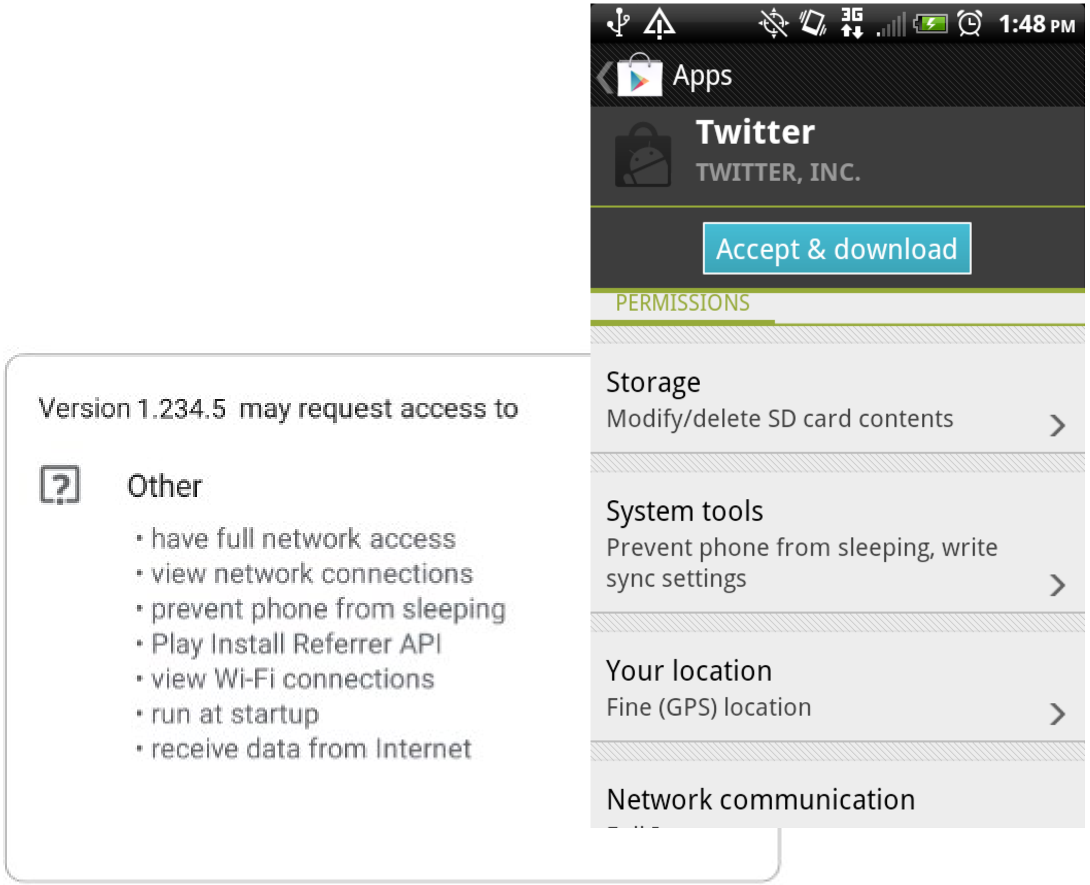
State of the Art

Prompts (time-of-use)





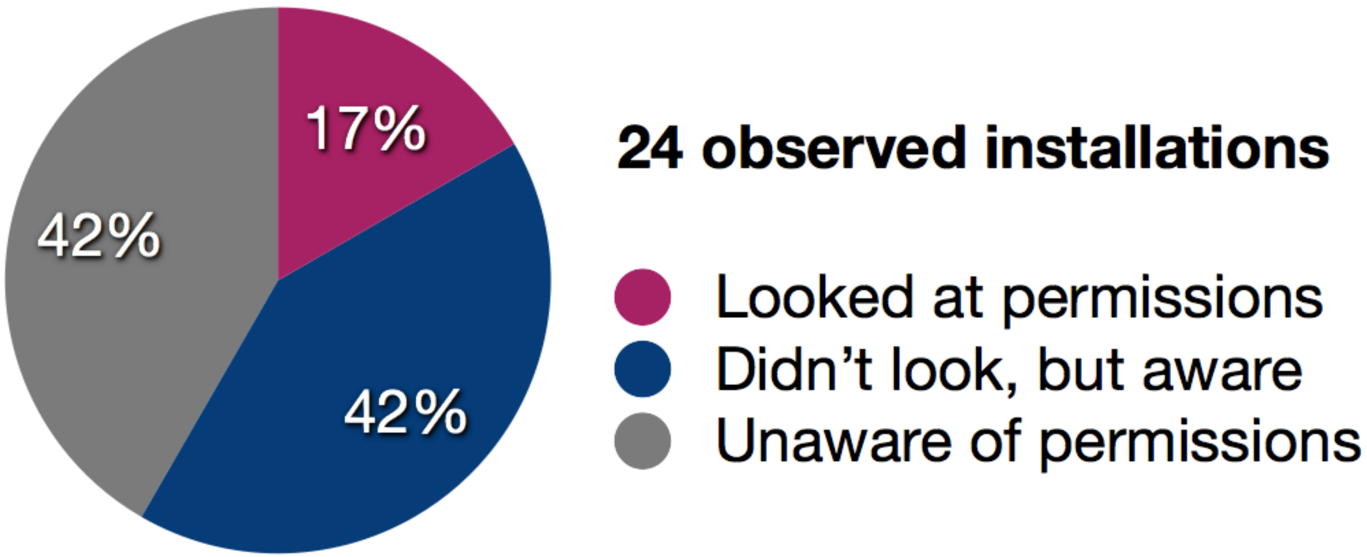
Manifests (install-time)



[Felt et al.]

Are Manifests Usable?

Do users pay attention to permissions?



… but 88% of users looked at reviews.

Are Manifests Usable?

Do users understand the warnings?

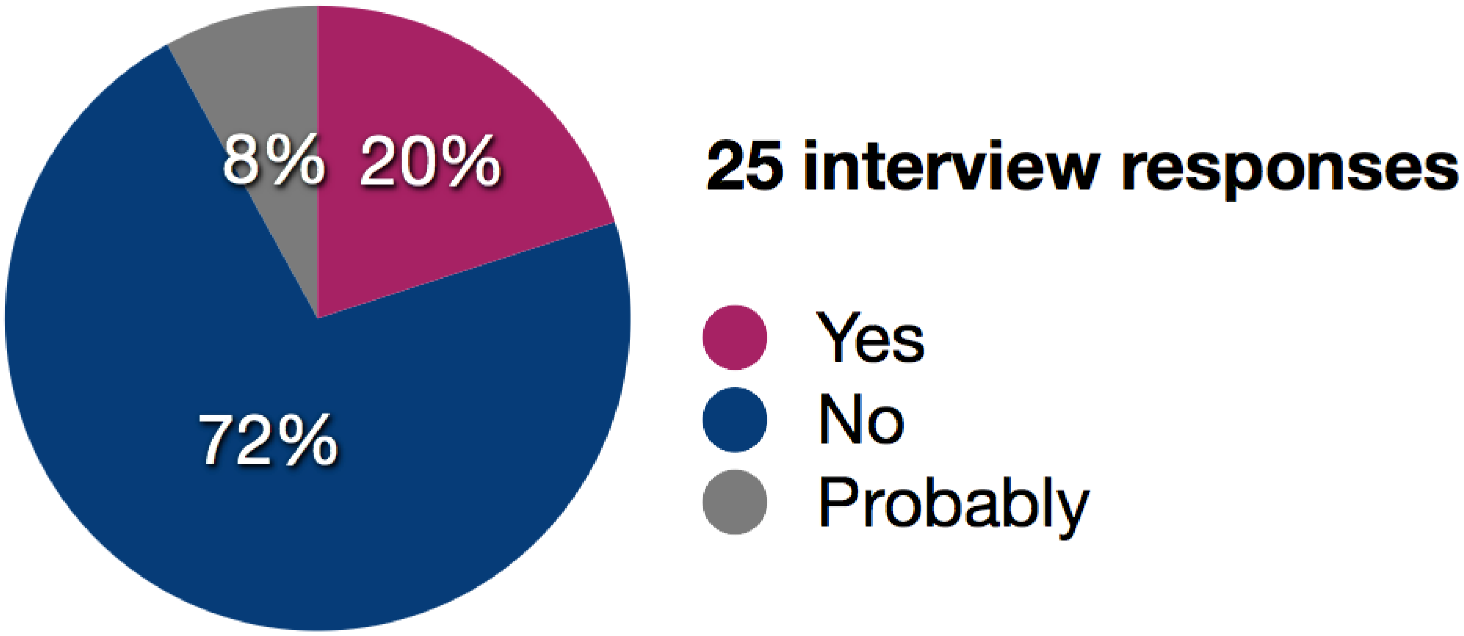
	Permission	n	Correct Answers	
1 Choice	READ_CALENDAR	101	46	45.5%
	CHANGE_NETWORK_STATE	66	26	39.4%
	READ_SMS1	77	24	31.2%
	CALL_PHONE	83	16	19.3%
2 Choices	WAKE_LOCK	81	27	33.3%
	WRITE_EXTERNAL_STORAGE	92	14	15.2%
	READ_CONTACTS	86	11	12.8%
	INTERNET	109	12	11.0%
	READ_PHONE_STATE	85	4	4.7%
	READ_SMS2	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, $n = 302$)

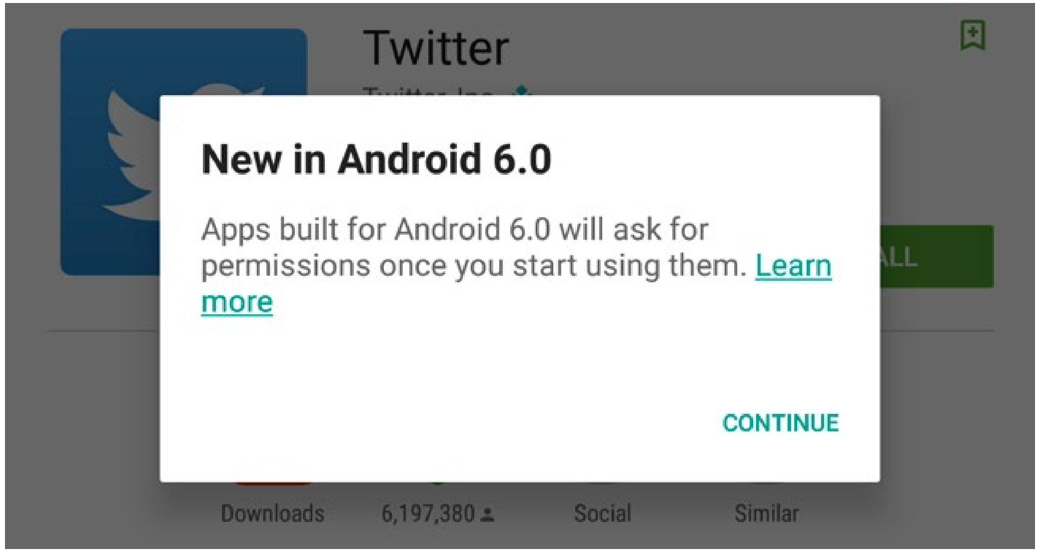
Are Manifests Usable?

Do users act on permission information?

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



Android 6.0: Prompts!



- First-use prompts for sensitive permission (like iOS).
- Big change! Now app developers needed to check for permissions or catch exceptions.

(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.

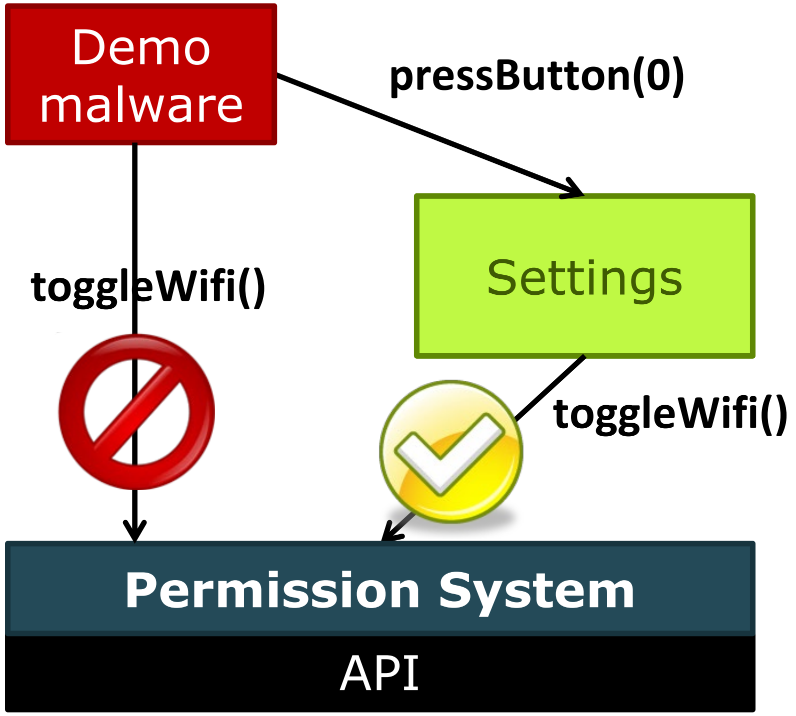
[Chin et al.]

Eavesdropping and Spoofing

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

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