CSE 484 / CSE M 584: Computer Security and Privacy

Mobile Platform Security

Fall 2016

Ada (Adam) Lerner
lerner@cs.washington.edu

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

Administrative

- Final project is out!
- An outline of your presentation is due this Friday.
- The final video is due next Friday (the last day of class).

Administrative

- My office hours moved for this week:
  - Moved to Wednesday at 12:30-1:30 pm.

- By appointment is always available.

Administrative

- There will be no lab 3.

Security Mindset Anecdote

- PGP – released in the early 1990s, when encryption with key lengths greater than 40 bits was classified as a "munition" and subject to weapons export laws.

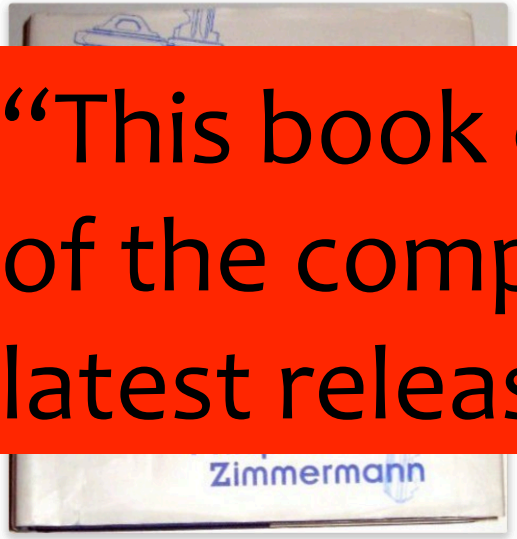
- Its creator, Phil Zimmerman was criminally investigated for "munitions export without a license"

Books › Computers & Technology › Security & Encryption

PGP: Source Code and Internals Hardcover – June 9, 1995

by Philip R. Zimmermann (Author)

★★★★★ 1 customer review



PGP (Pretty Good Privacy) is a computer program for the encryption of data and electronic mail, a powerful "envelope" that allows individuals the same privacy in their communications as enjoyed by governments and large corporations. PGP, which is freely available on the Internet, uses public-key cryptography, specifically the RSA algorithm, which is particularly well suited to the needs of computer-mediated communications. This book contains a formatted vesion of the complete source code for the latest release (2.6.2) of PGP.

See all 2 images

"This book contains a formatted version of the complete source code for the latest release (2.6.2) of PGP."

The First Amendment and Code

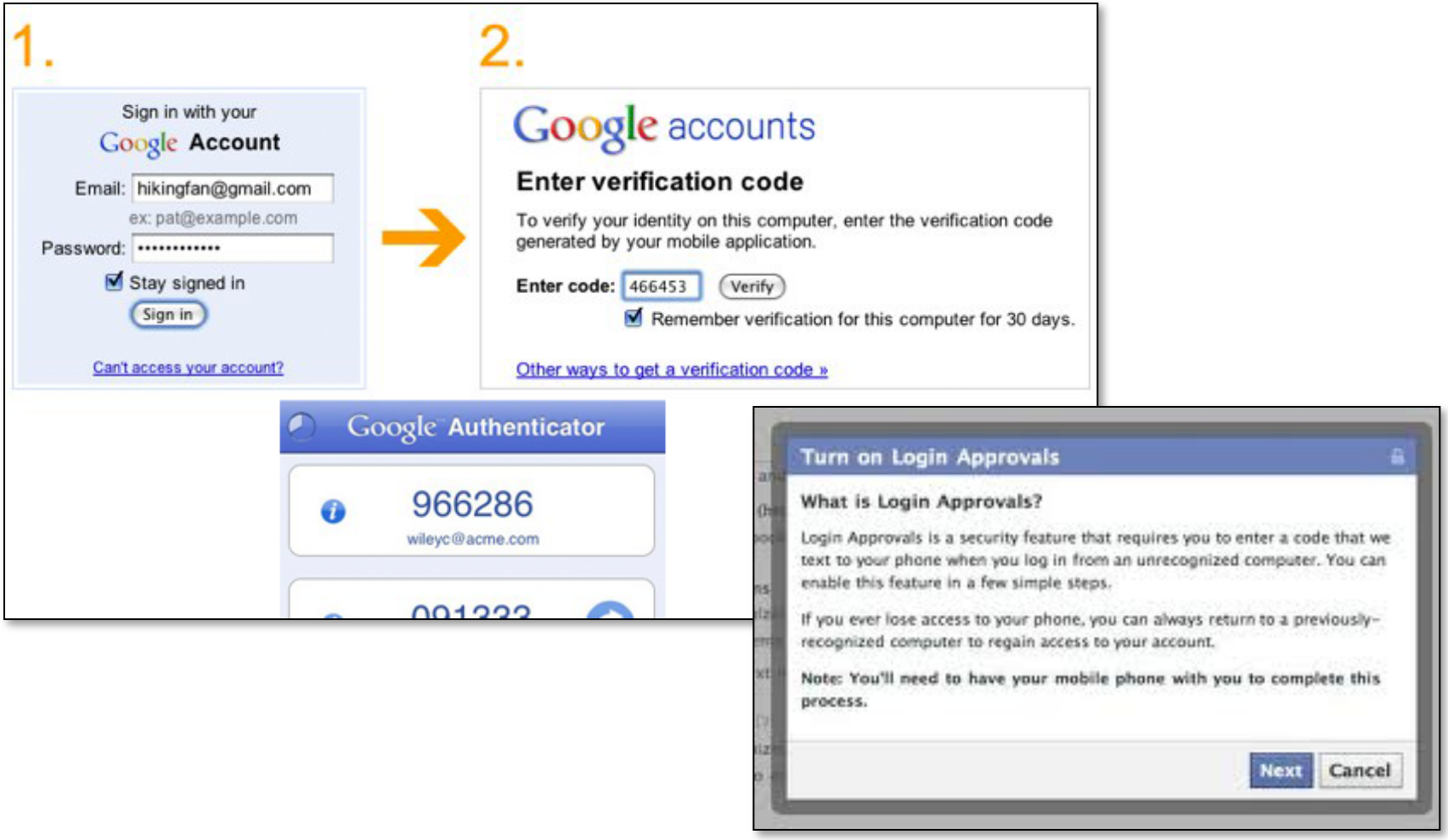
- Federal appears courts have ruled that crypto source code is speech under the First Amendment

- Export restrictions have been loosened (small list of countries are restricted – the same ones to which most US trade is prohibited)

Improving(?) Passwords

- Add biometrics
  - For example, keystroke dynamics or voiceprint
- Graphical passwords
  - Goal: easier to remember? no need to write down?
- Password managers
  - Examples: LastPass, KeePass, built into browsers
- Two-factor authentication
  - Leverage phone (or other device) for authentication

Multi-Factor Authentication



Multi-Factor Authentication

MORE ABOUT YOUR YUBIKEY

YUBIKEY 4

USB; strong crypto and touch-to-sign, plus One-Time-Password, PIV-compatible smart card, and FIDO U2F. *Read more*

YUBIKEY NEO

USB and NFC (for Android mobile); One-Time Password, PIV-compatible smart card, and FIDO U2F. *Read more*

What About Biometrics?

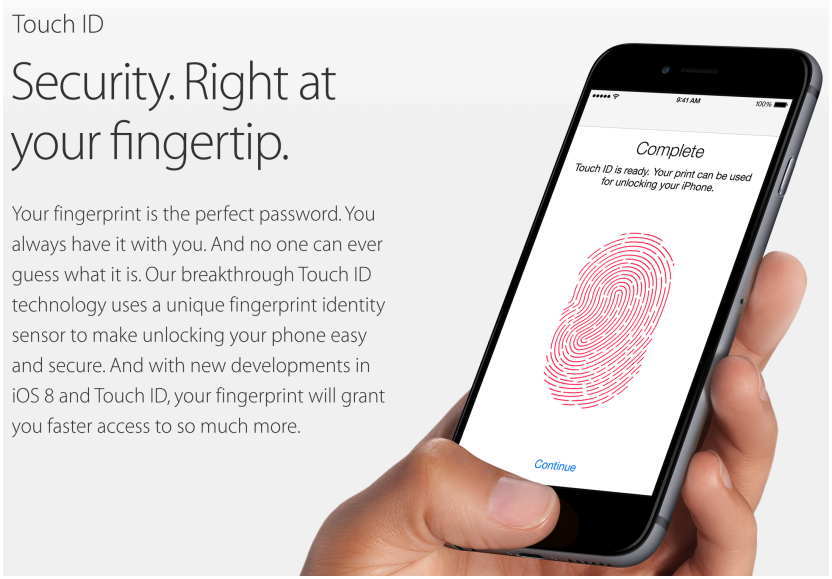
- Authentication: **What you are**
- Unique identifying characteristics to authenticate user or create credentials
  - Biological and physiological: Fingerprints, iris scan
  - Behaviors characteristics - how perform actions: Handwriting, typing, gait
- Advantages:
  - Nothing to remember
  - Passive
  - Can't share (generally)
  - With perfect accuracy, could be fairly unique

Issues with Biometrics

- Private, but not secret
  - Maybe encoded on the back of an ID card?
  - Maybe encoded on your glass, door handle, …
  - Sharing between multiple systems?
- Revocation is difficult (impossible?)
  - Sorry, your iris has been compromised, please create a new one…
- Physically identifying
  - Soda machine to cross-reference fingerprint with DMV?
- Birthday paradox
  - With false accept rate of 1 in a million, probability of false match is above 50% with only 1609 samples

Attacking Biometrics

- An adversary might try to steal biometric info
  - Malicious fingerprint reader
    - Consider when biometric is used to derive a cryptographic key
  - Residual fingerprint on a glass
- Ex: Apple's TouchID

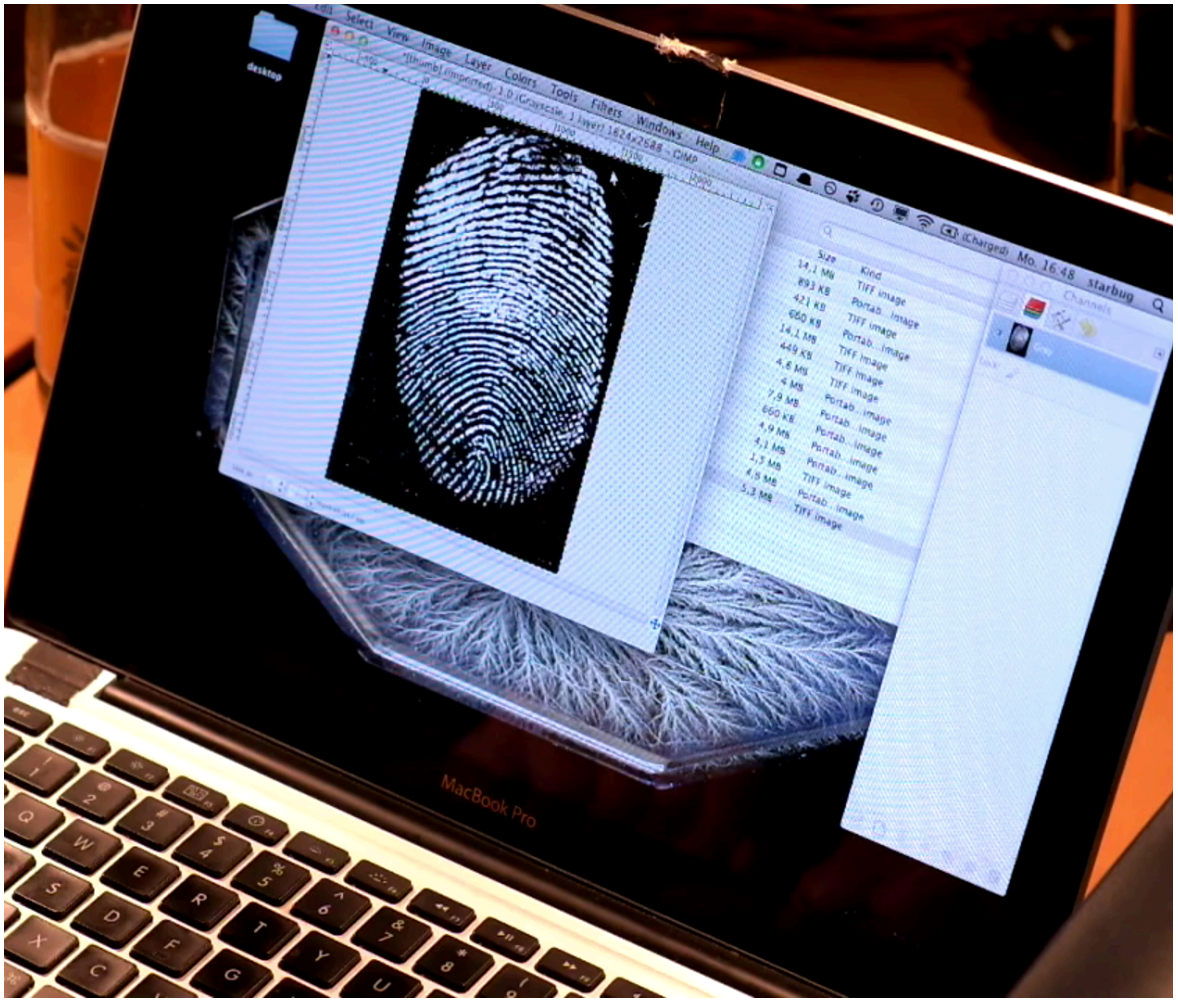


[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics

[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics



[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics

[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics

MOBILE PLATFORM SECURITY

Roadmap

- Mobile malware
- Mobile platforms vs. traditional platforms
- Deep dive into Android

Questions: Mobile Malware

Q1 (bottom third of the room): How might malware authors get malware onto phones?

Q2 (middle third): What are some goals that mobile device malware authors might have? What assets are present on a smartphone?

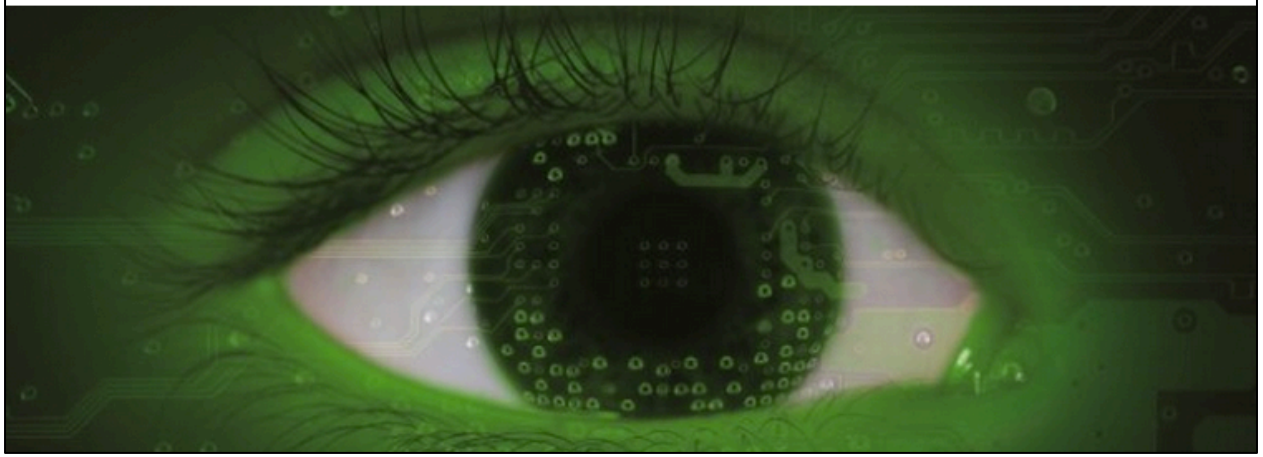
Q3 (top third): What technical things might malware authors do? What are the threats/ vulnerabilities on a smartphone?

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

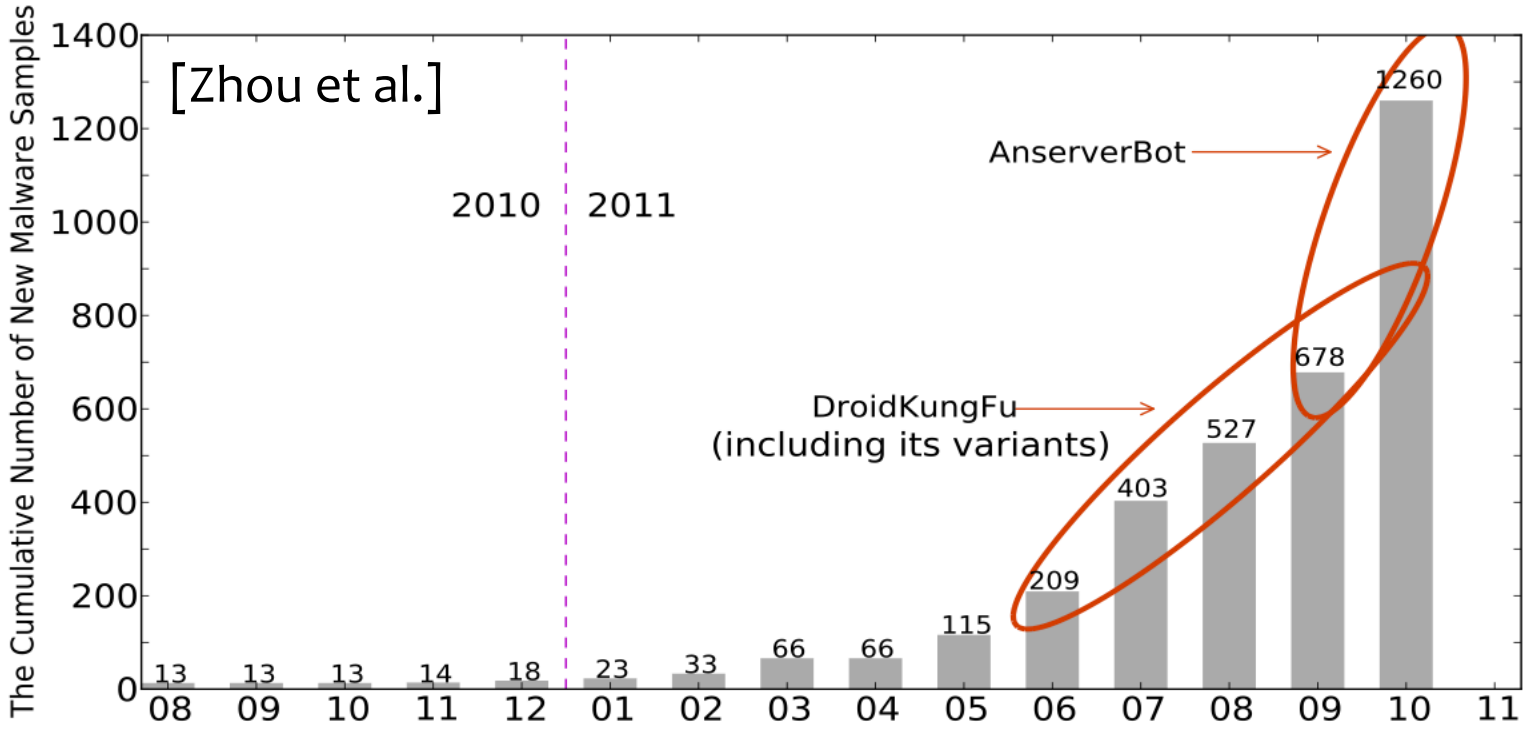


Smartphone (In)Security

Even legitimate applications exhibit questionable behavior.

Malware in the Wild

Android malware is growing.
Today (2016): millions of samples.



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

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 2nd 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"

[Zhou et al.]

(Android) Malware in the Wild

What does it do?

	Root Exploit	Remote Control		Financial Charges			Information Stealing		
		Net	SMS	Phone Call	SMS	Block SMS	SMS	Phone #	User Account
# Families	20	27	1	4	28	17	13	15	3
# Samples	1204	1171	1	256	571	315	138	563	43

Why all these problems with mobile malware?

Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) 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., Linux) 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.

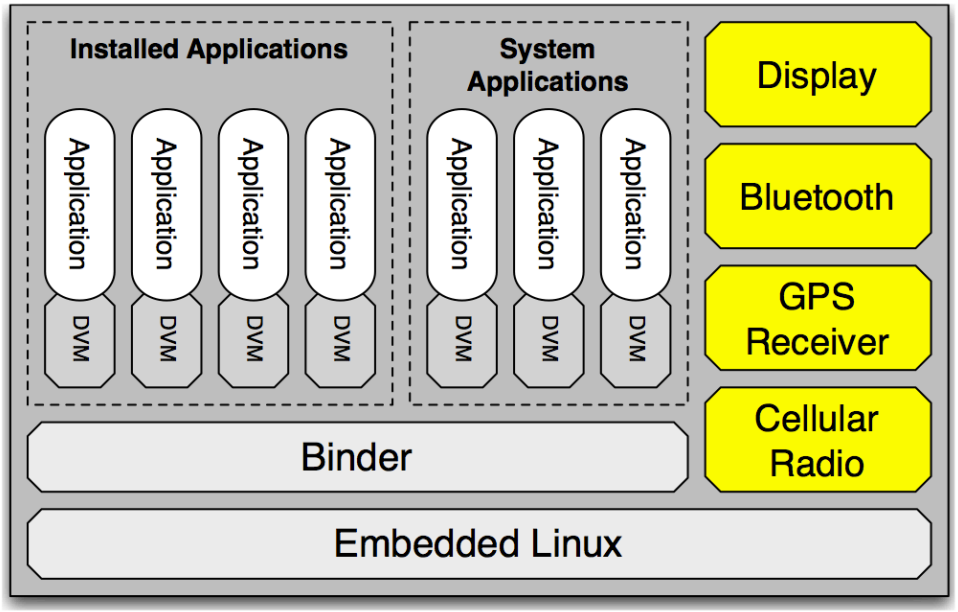
What's Different about Mobile Platforms?

- 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

[Enck et al.]



- 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

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

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.

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

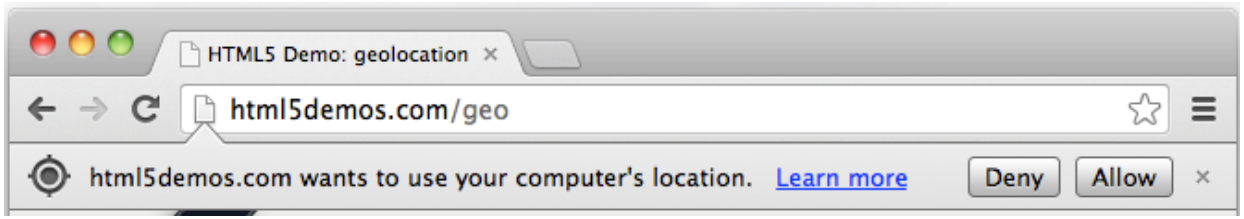
Smartphones (and other modern OSes) try to prevent such attacks by limiting applications' access to:

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

How should operating system grant permissions to applications?

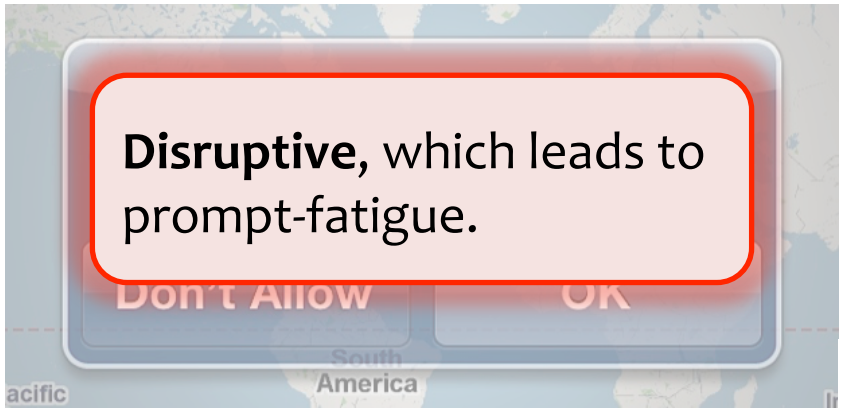
State of the Art

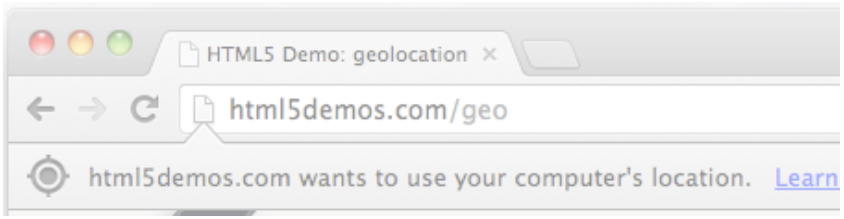
Prompts (time-of-use)



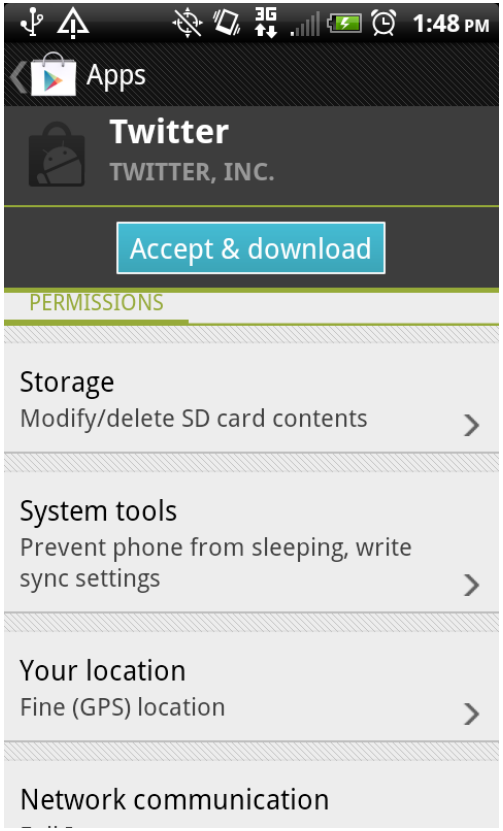
State of the Art

Prompts (time-of-use)



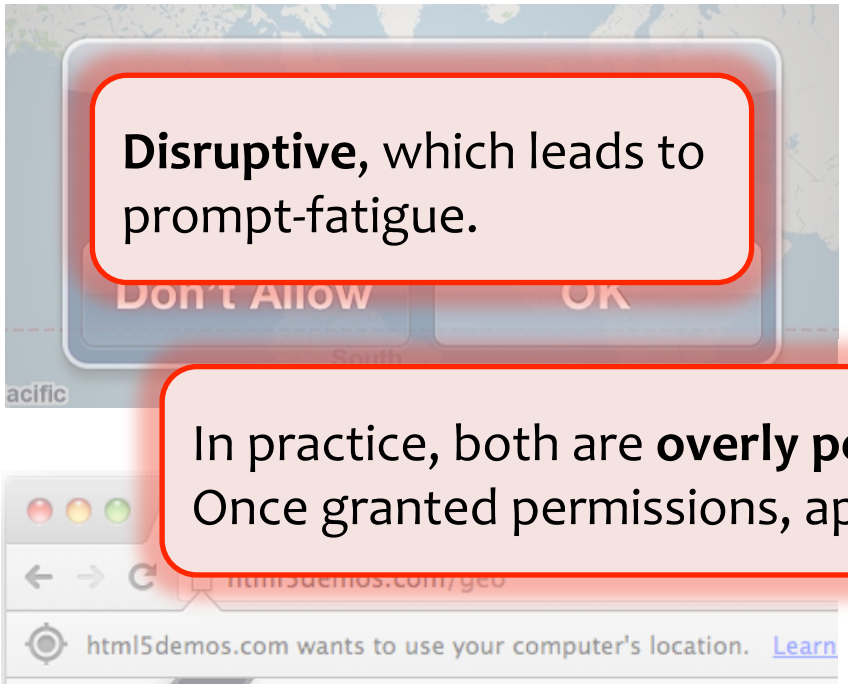


Manifests (install-time)



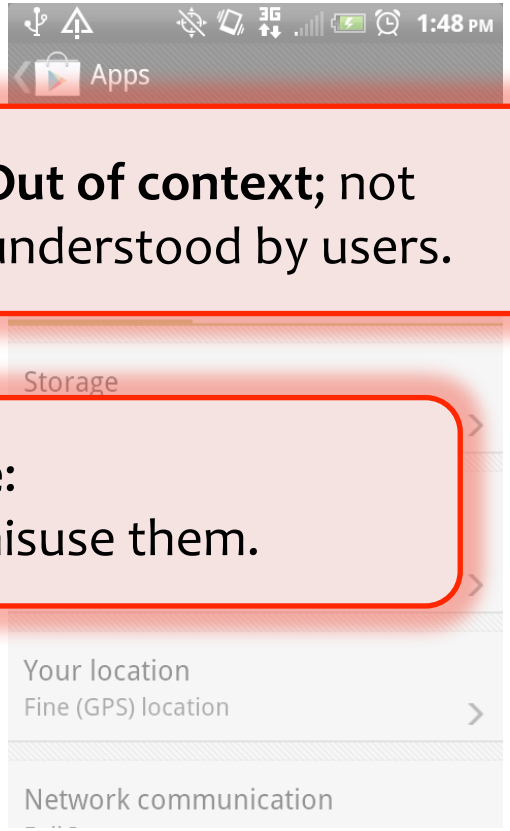
State of the Art

Prompts (time-of-use)



Disruptive, which leads to prompt-fatigue.

Manifests (install-time)



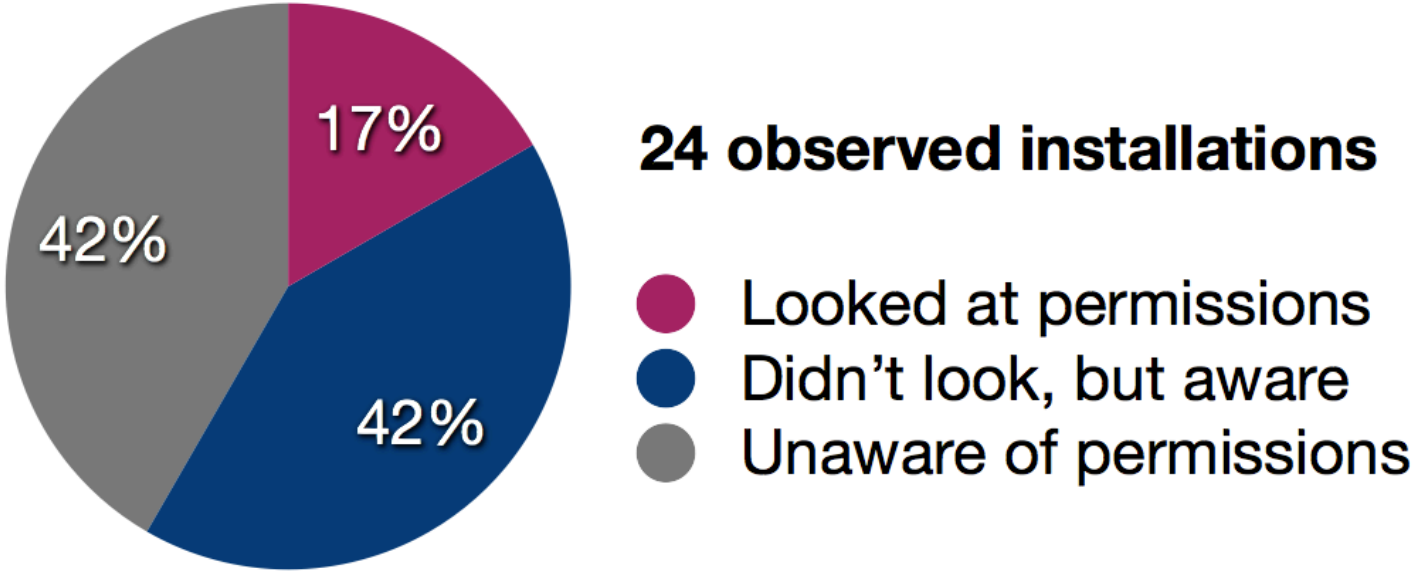
Out of context; not understood by users.

In practice, both are **overly permissive**:
Once granted permissions, apps can misuse them.

[Felt et al.]

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%

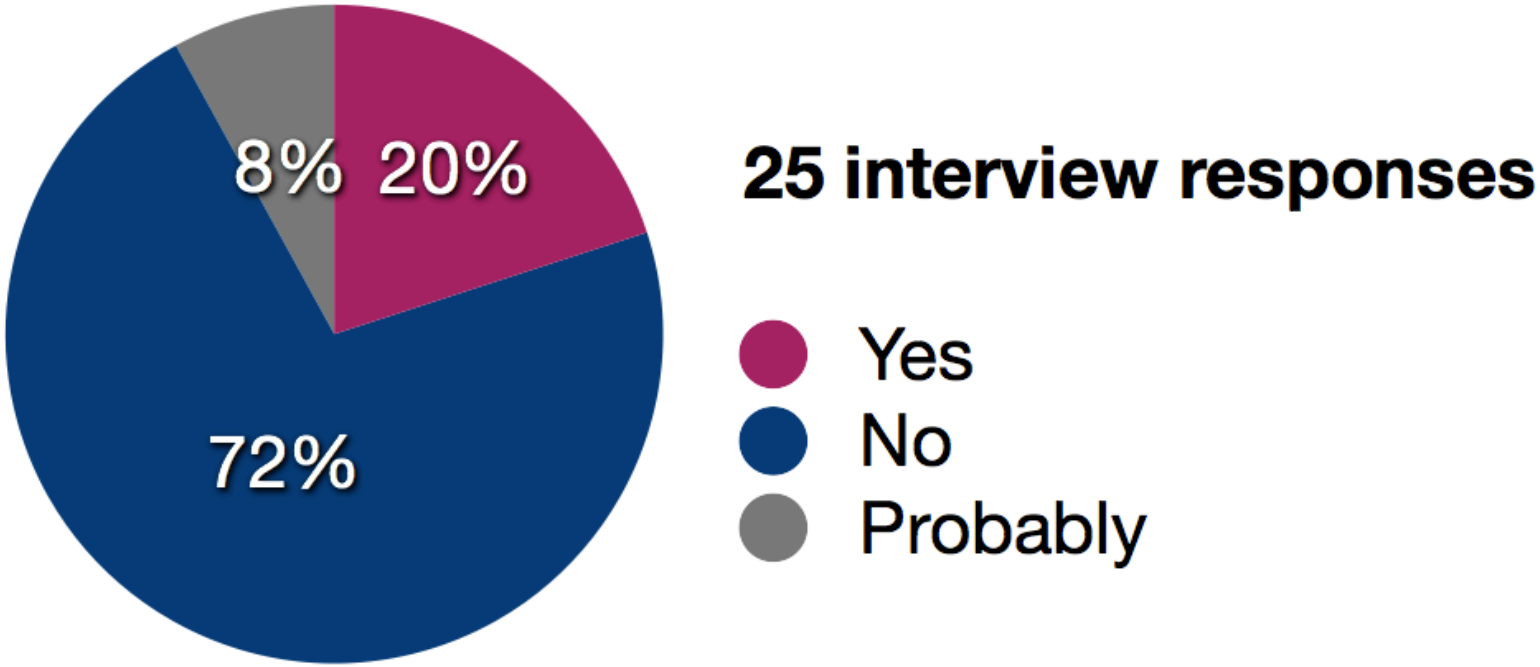
Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. n is the number of respondents. (Internet Survey, $n = 302$)

[Felt et al.]

Are Manifests Usable?

Do users act on permission information?

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

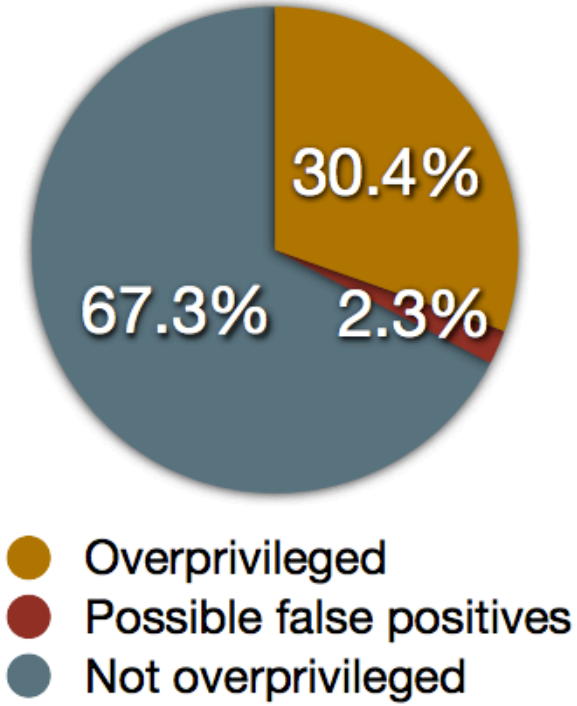


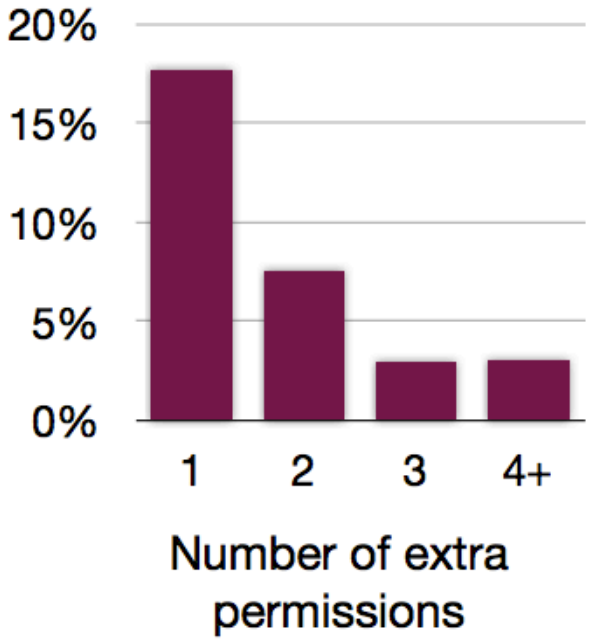
[Felt et al.]

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.)



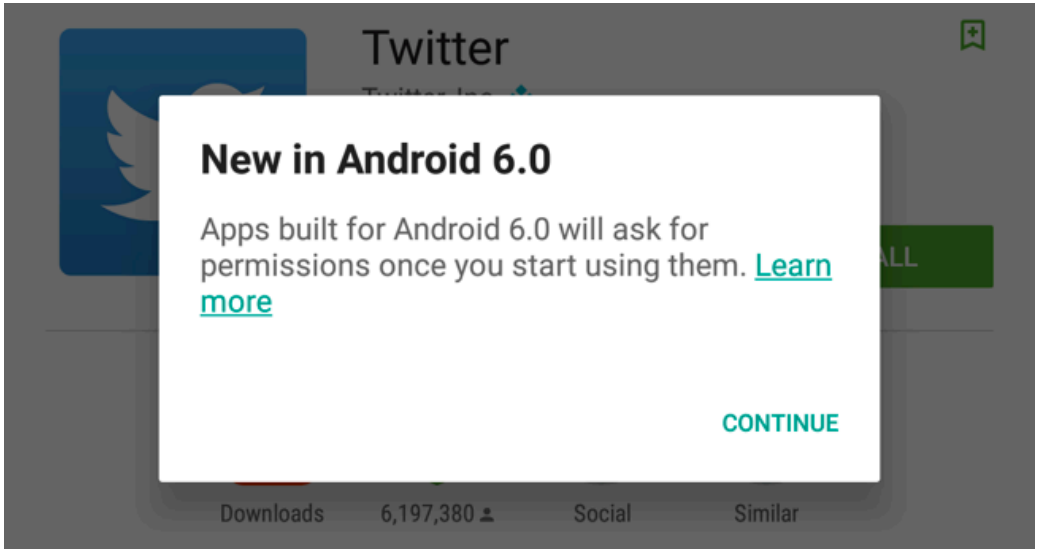


Manifests rely on the user to make good choices at install time

- It's not clear that users know how to make the right choice – or that there IS a right choice.

- I don't want ANY app to access my camera at all times. I just want apps to access my camera when they need to for legitimate purposes!

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.

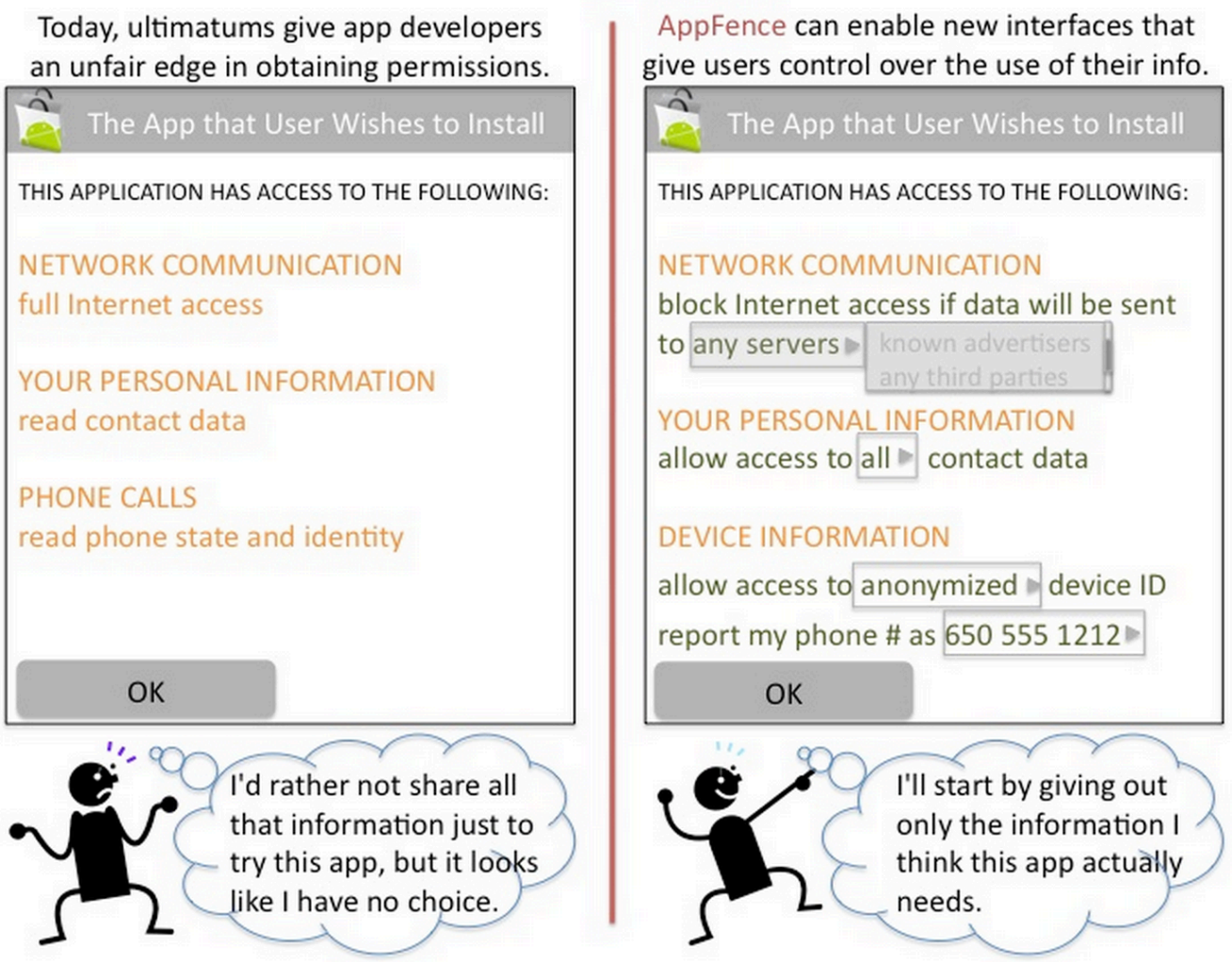
Promps rely on the user to make good choices at use time

- It's not clear that users know how to make the right choice at use time either.

- Still only checks on first use – the app can still use the resource for any reason it wants, at any time now or in the future.

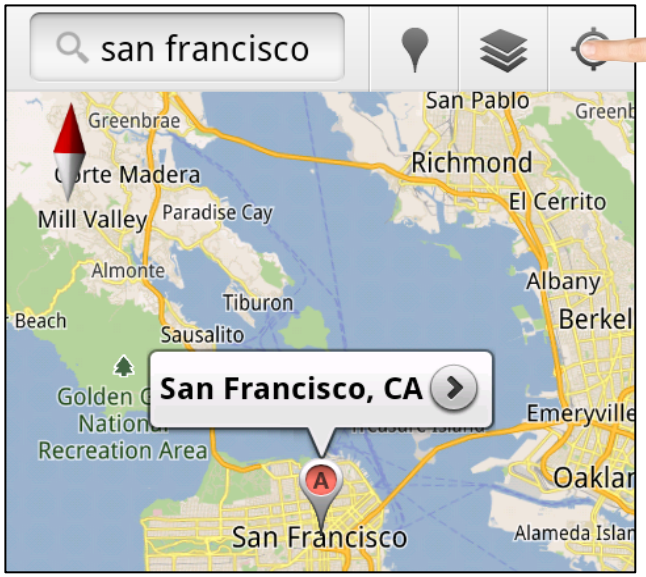
[Hornyack et al.]

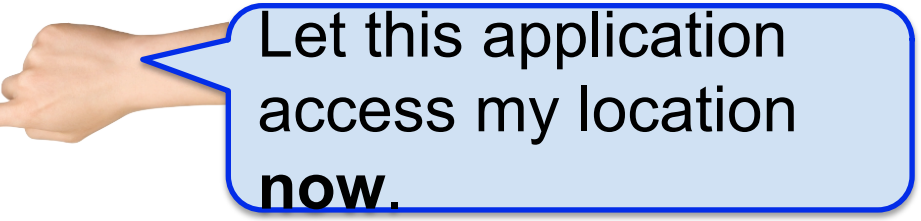
Improving Permissions: AppFence



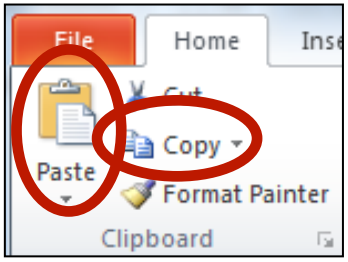
[Roesner et al.]

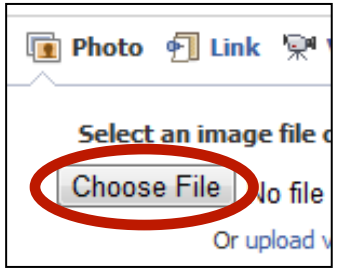
Improving Permissions: User-Driven Access Control





Insight:
A user's natural UI actions within an application implicitly carry permission-granting semantics.





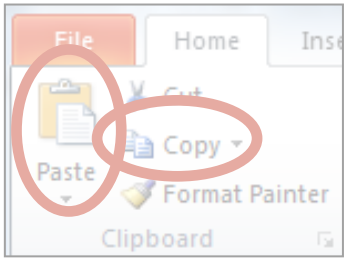
[Roesner et al.]

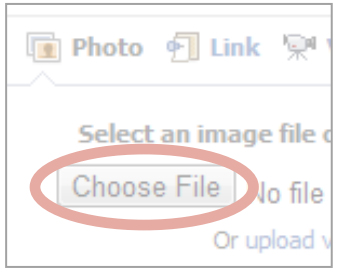
Improving Permissions: User-Driven Access Control

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

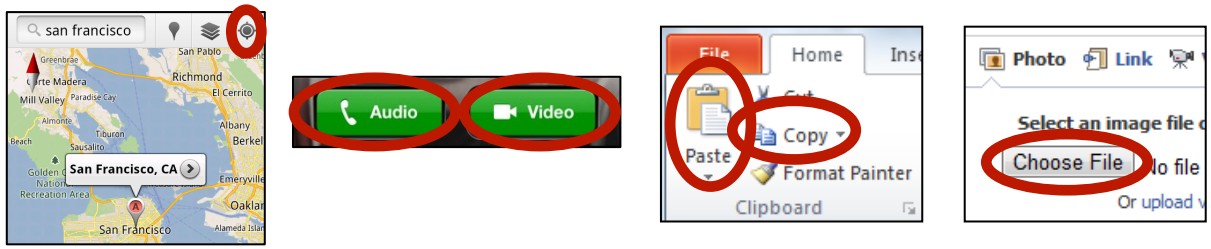
Study shows:
Many users already believe (52% of 186) – and/or desire (68%) – that resource access follows the user-driven access control model.

ions
nplicitly
ting





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.

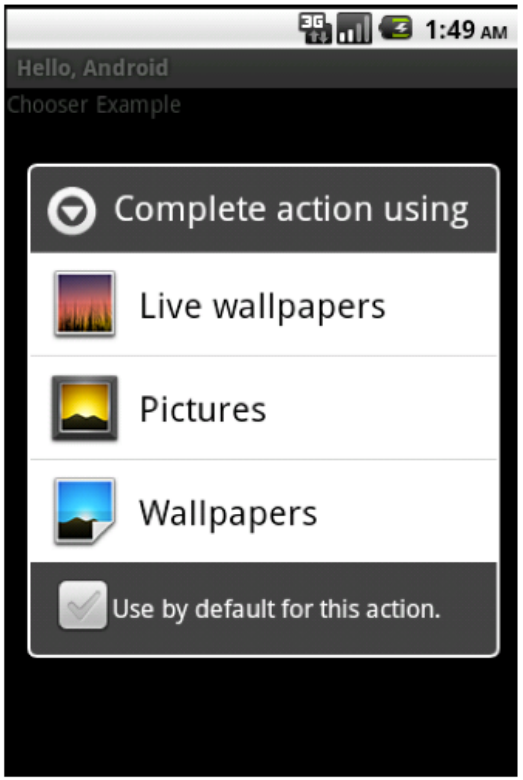
(2) Inter-Process Communication

- 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.]

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!



[Chin et al.]

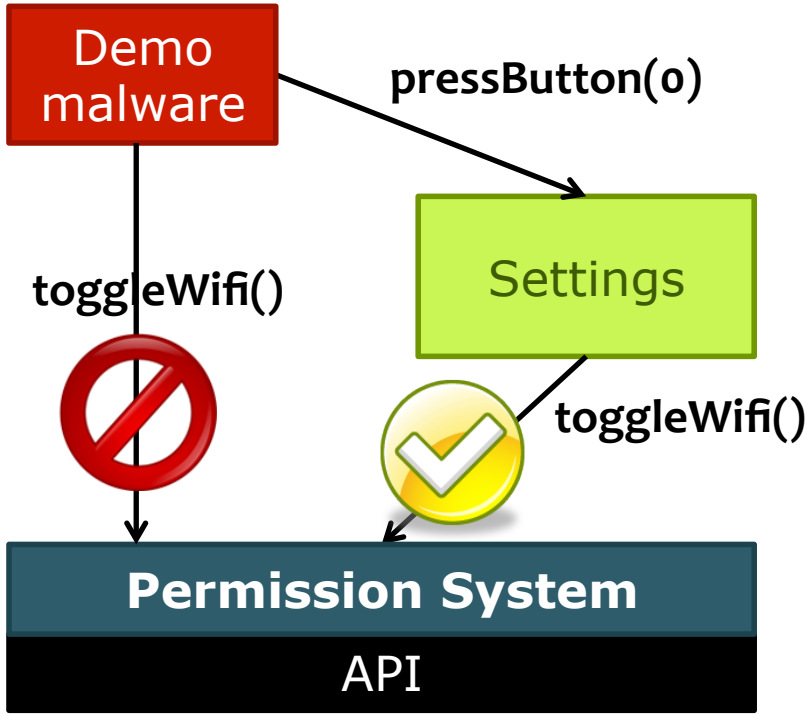
Intent Spoofing

- 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.

[Felt et al.]

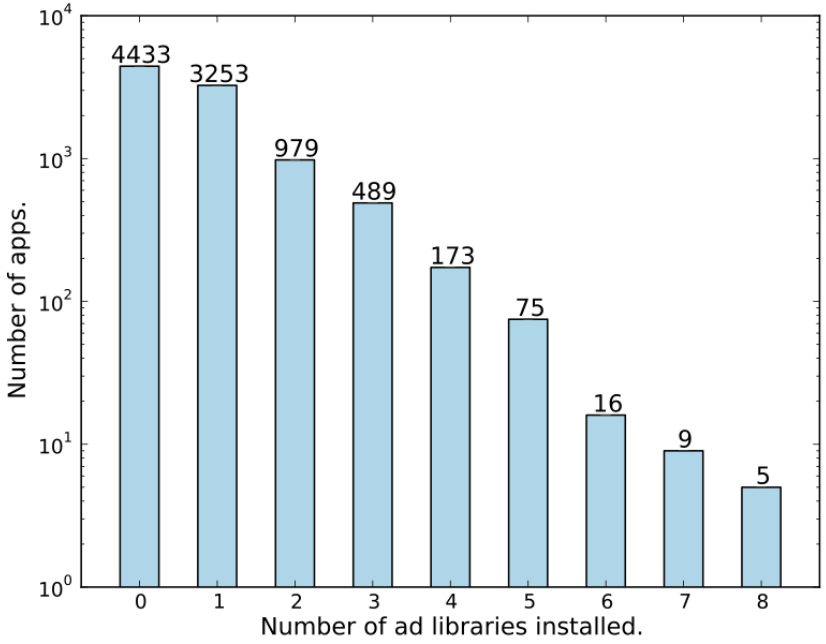
Permission Re-Delegation

- 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.



Aside: Incomplete Isolation

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



[Shekhar et al.]

Ad from ad library

Map from Google library

Social button from Facebook library

More on Android…

Android Application Signing

- Apps are signed
  - Often with self-signed certificates
  - 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

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]

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.0%
4.1.x	Jelly Bean	16	7.2%
4.2.x		17	10.0%
4.3		18	2.9%
4.4	KitKat	19	32.5%
5.0	Lollipop	21	16.2%
5.1		22	19.4%
6.0	Marshmallow	23	7.5%

Data collected during a 7-day period ending on May 2, 2016.
Any versions with less than 0.1% distribution are not shown.