Mobile OS Security

CSE 451 – November 30, 2015

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Today's Goals

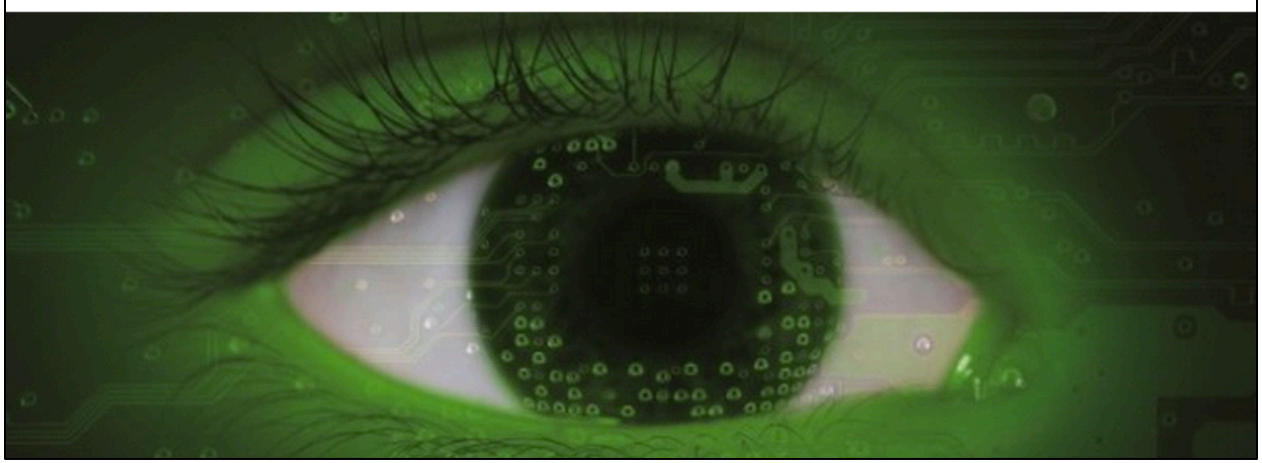
- Introduce some OS security concepts through a case study of mobile OSes, particularly Android.
- Along the way, highlight that it matters how these systems interface with people (users & devs).

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.

Top Mobile Apps Overwhelmingly Leak Private Data: Study

By Robert Lemos | Posted 2013-07-31 Email Print

paid apps
pplication-

isk more often
more likely to
applications,

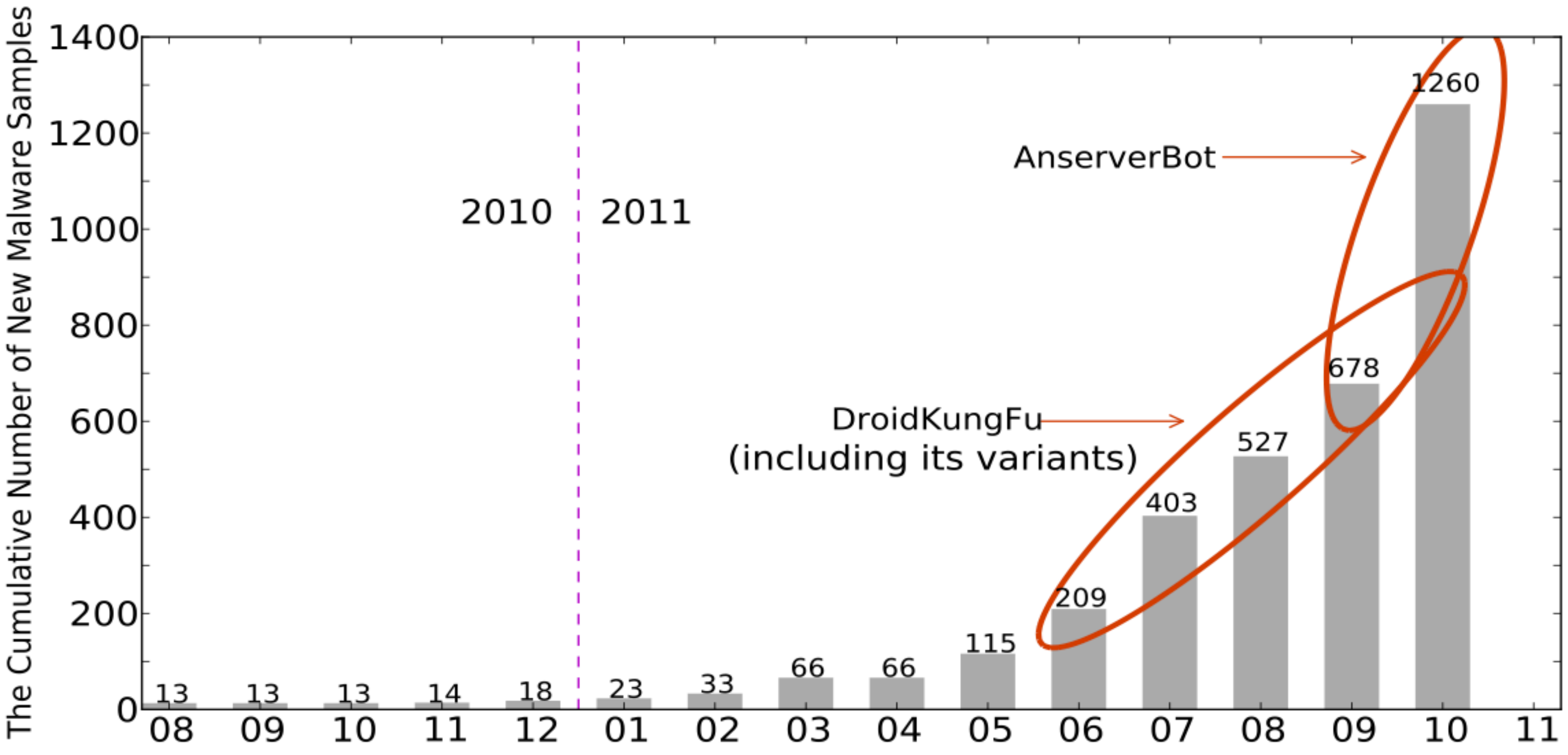
Hornyack et al.: 43 of 110 Android applications sent location or phone ID to third-party advertising/analytics servers.

Android flashlight app tracks users via GPS, FTC says hold on

By Michael Kassner in IT Security, December 11, 2013, 9:49 PM PST

Malware in the Wild

Android malware is growing.



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.

```
FranziBook:Desktop franzi$ whoami
franzi

FranziBook:Desktop franzi$ id
uid=501(franzi) gid=20(staff) groups=20(staff),401(com.apple.sharepoint.group.1),5
02(access_bpf),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_apps
erveradm),98(_lpadmin),33(_appstore),100(_lpoperator),204(_developer),395(com.appl
e.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh)

FranziBook:Desktop franzi$ ls -l hello.txt
-rw-r--r--  1 franzi  staff  0 Nov 29 10:08 hello.txt

FranziBook:Desktop franzi$ chmod 700 hello.txt
FranziBook:Desktop franzi$ ls -l hello.txt
-rwx------  1 franzi  staff  0 Nov 29 10:08 hello.txt
```

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.

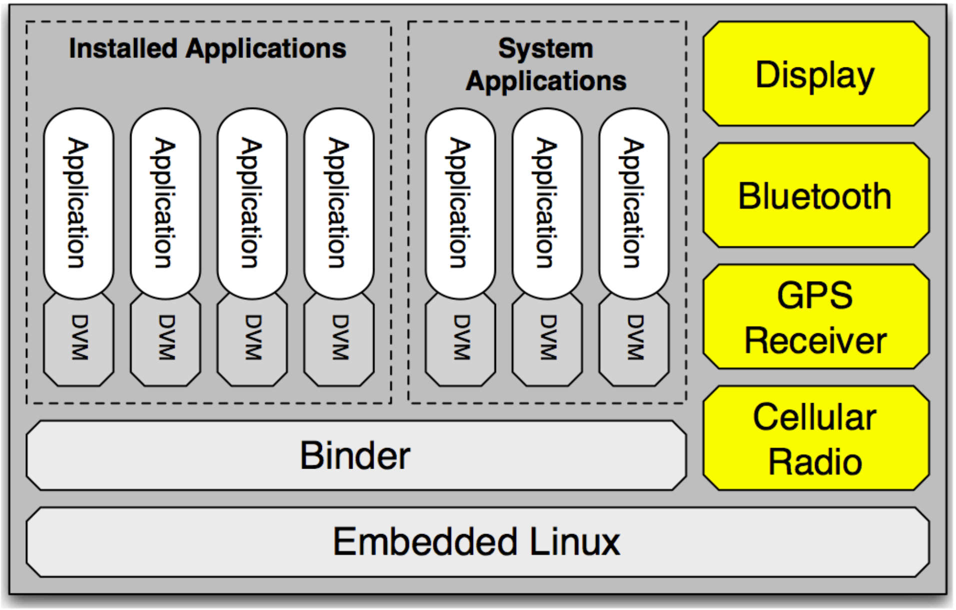
What's Different about Mobile Platforms?

- 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

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?

   → the rest of today's lecture

2. Communication: How can applications communicate with each other?

   → specific communication APIs (there may be vulnerabilities in how apps use them)

(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, …).

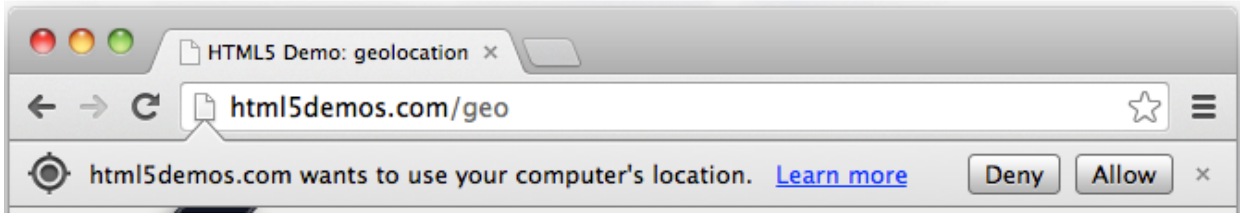


How should operating system grant permissions to applications?

Standard approach: Ask the user.

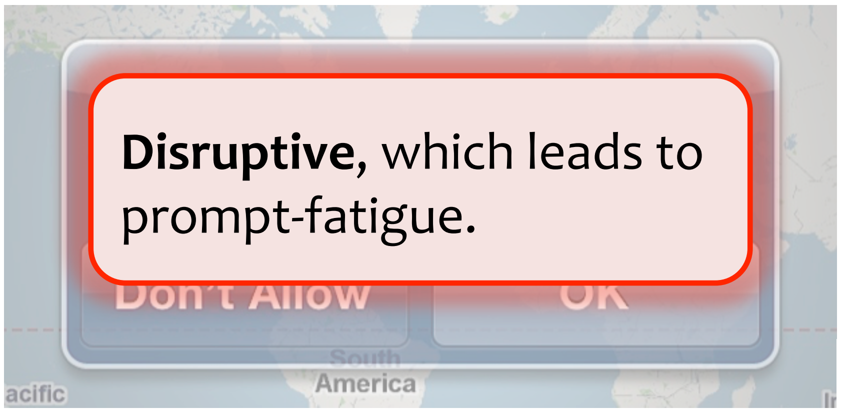
State of the Art

Prompts (time-of-use)

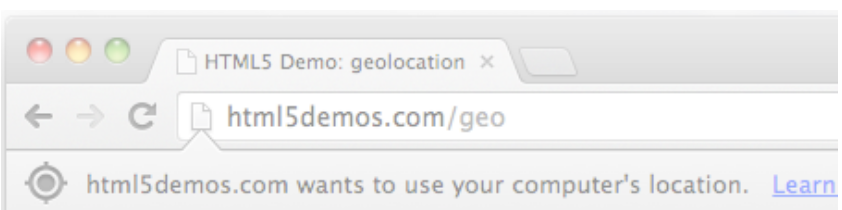


State of the Art

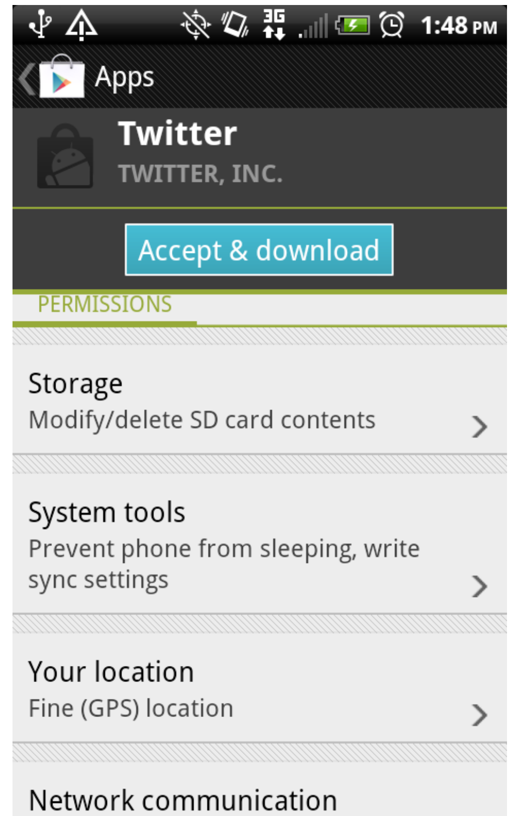
Prompts (time-of-use)



Disruptive, which leads to prompt-fatigue.

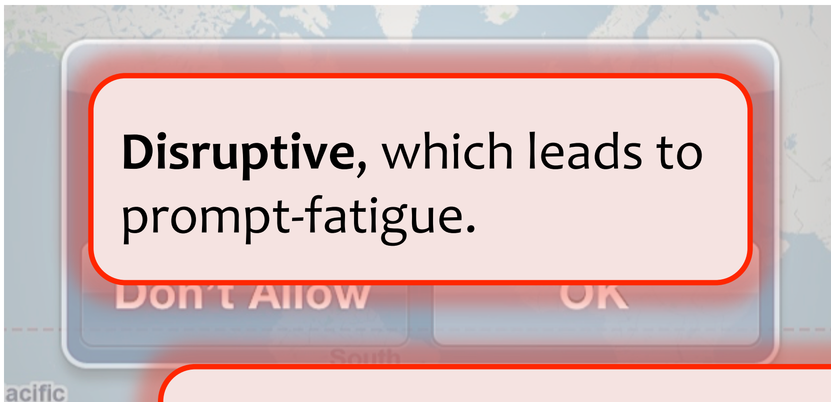


Manifests (install-time)

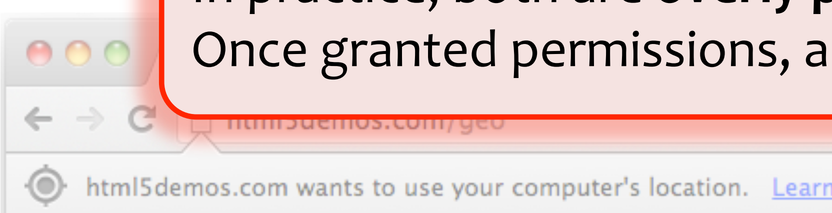


State of the Art

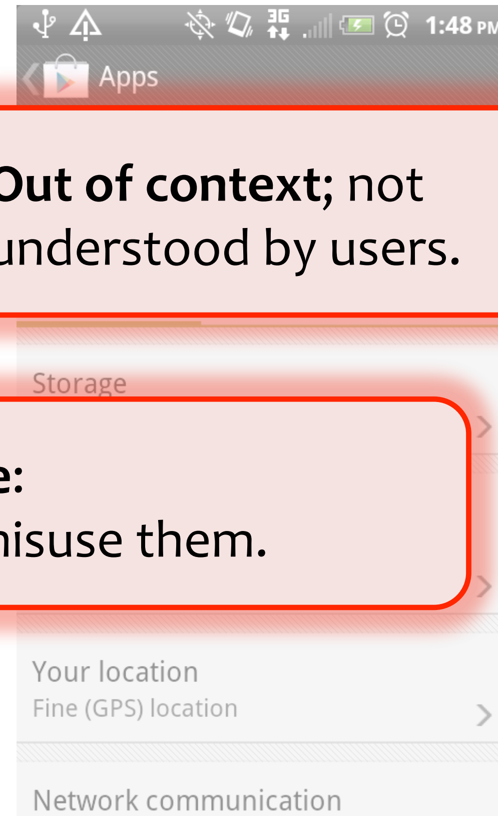
Prompts (time-of-use)



Disruptive, which leads to prompt-fatigue.



Manifests (install-time)

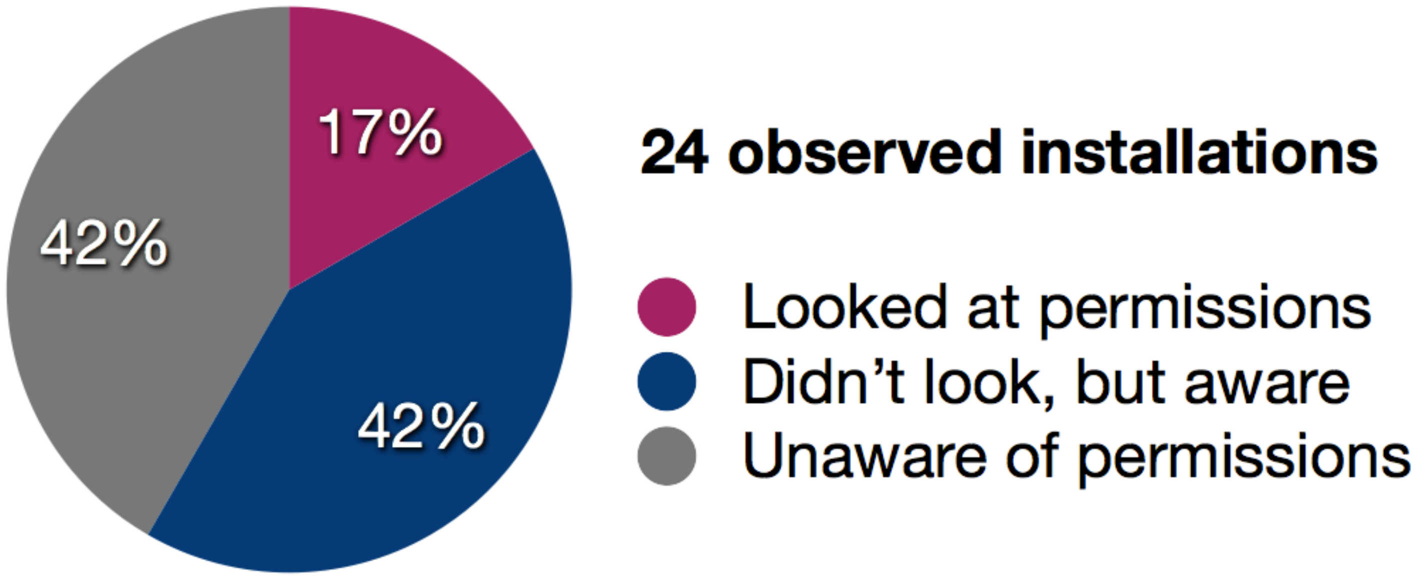


Out of context; not understood by users.

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

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%

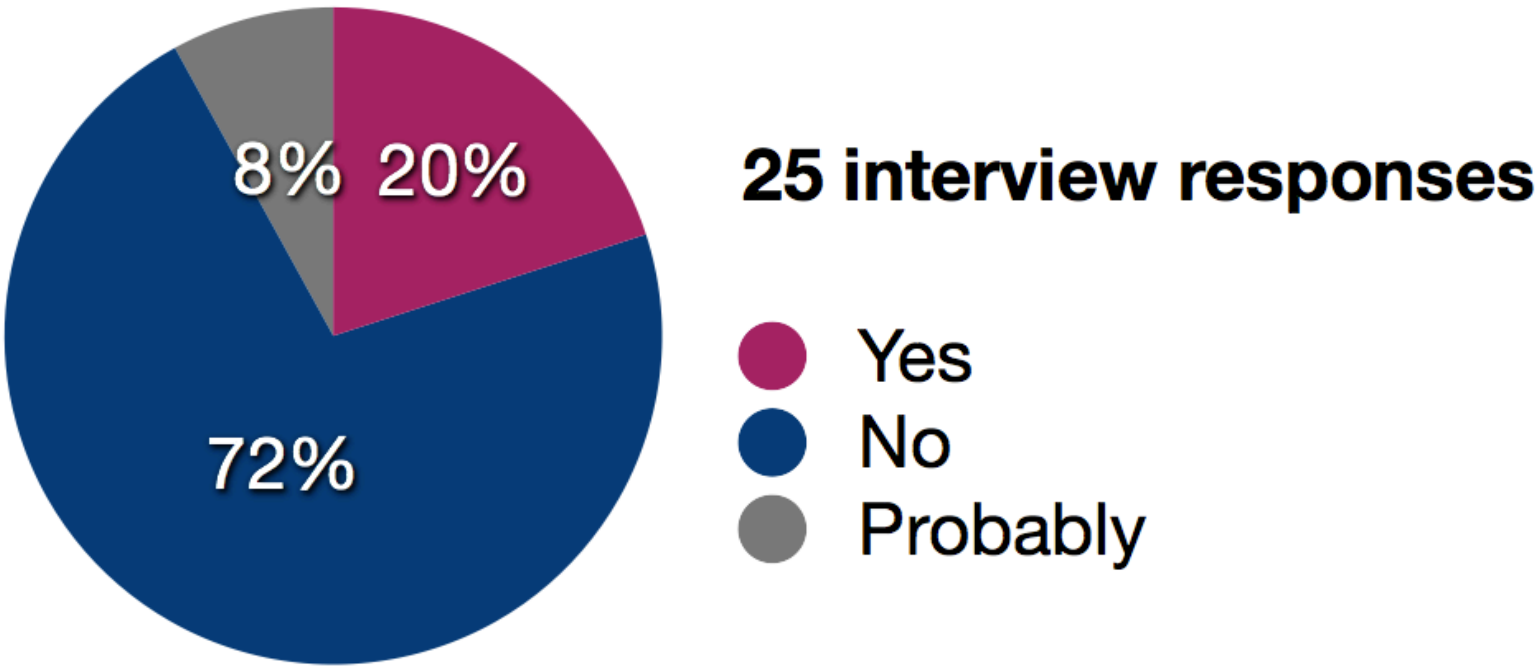
Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. n is the number of respondents. (Internet Survey, $n = 302$)

[Felt et al.]

Are Manifests Usable?

Do users act on permission information?

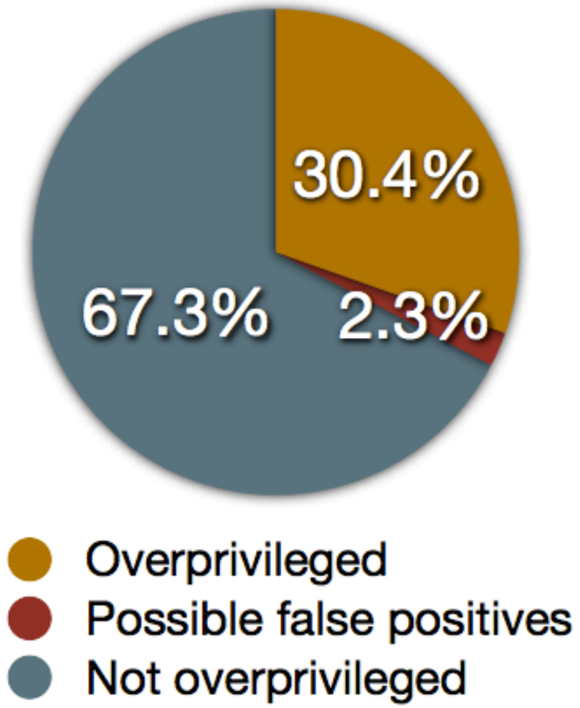
"Have you ever not installed an app because of permissions?"

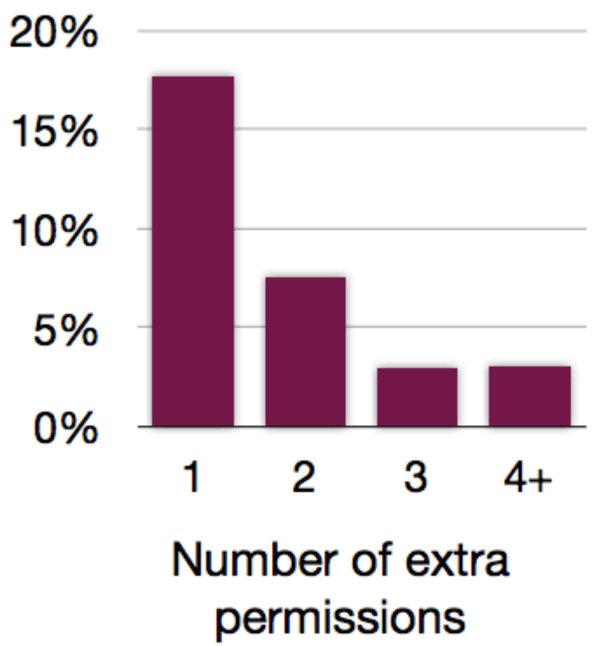


Over-Permissioning

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

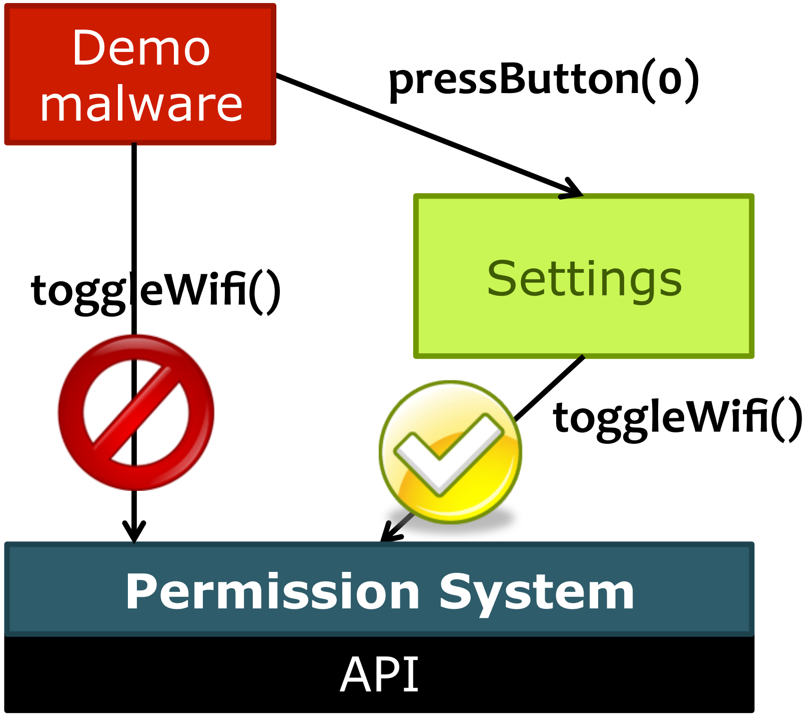
www.android-permissions.org (Felt et al.), http://pscout.csl.toronto.edu (Au et al.)





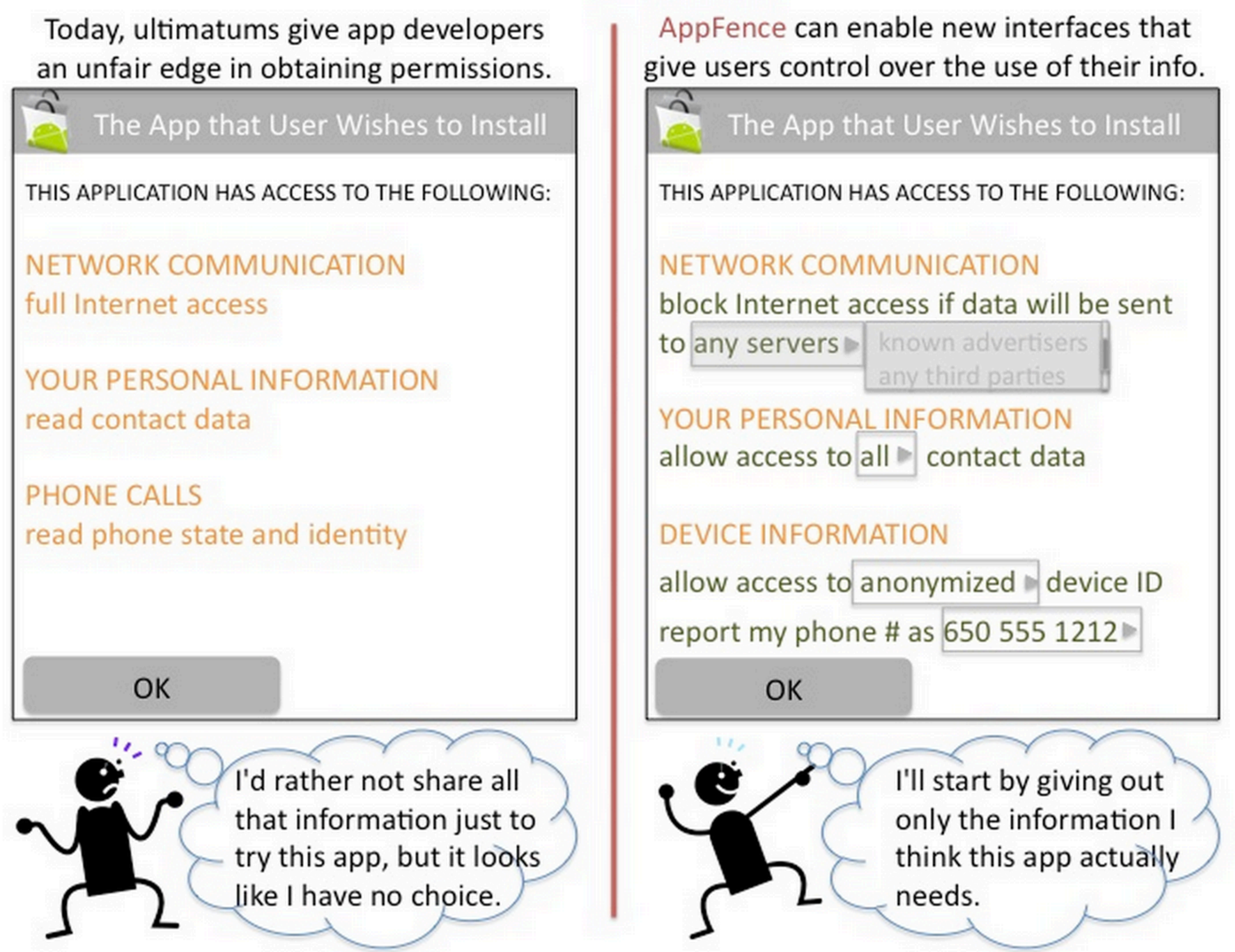
Permission Re-Delegation

- 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.

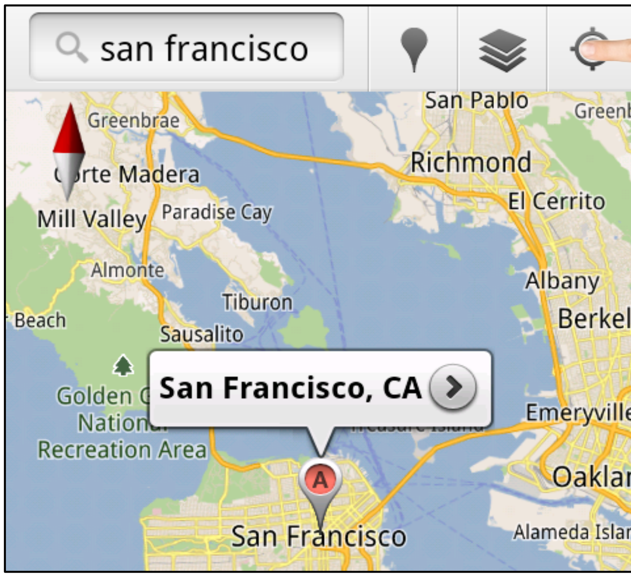


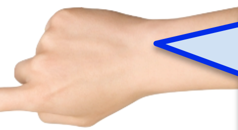
[Hornyack et al.]

Improving Permissions: AppFence



Improving Permissions: User-Driven Access Control

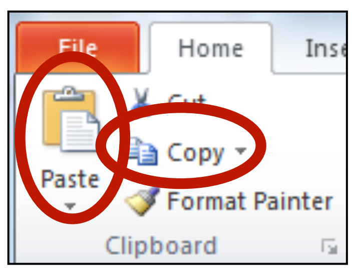


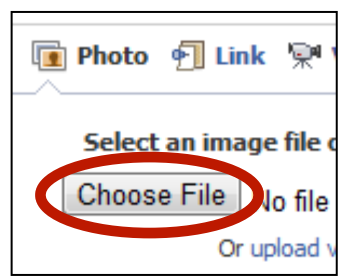


Let this application access my location **now**.

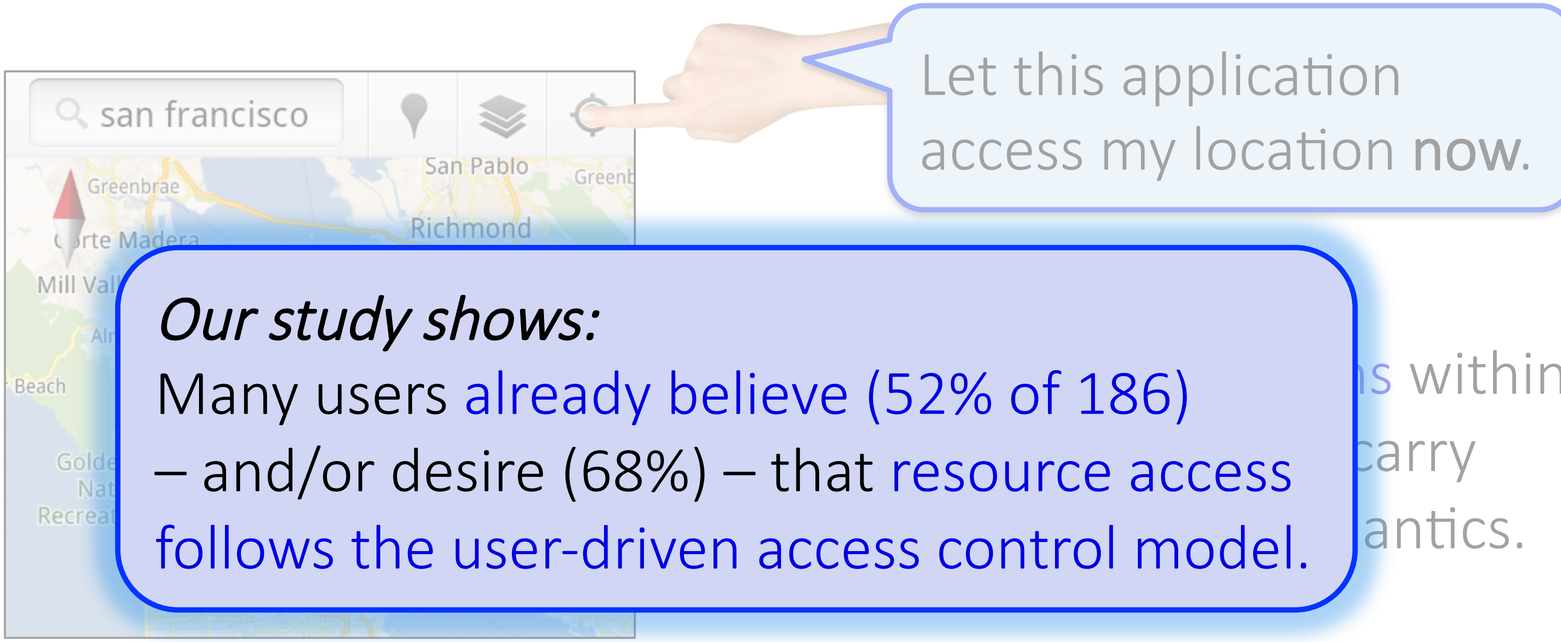
Insight:

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



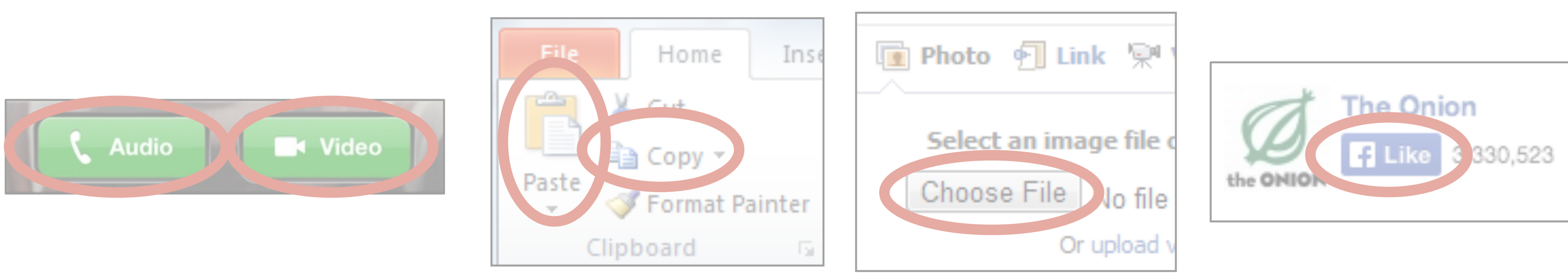


Improving Permissions: User-Driven Access Control

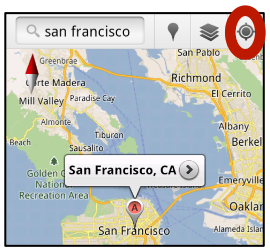


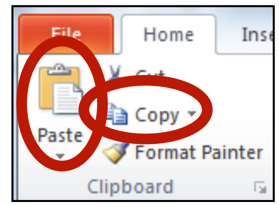
Our study shows:
Many users already believe (52% of 186) – and/or desire (68%) – that resource access follows the user-driven access control model.

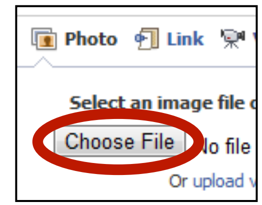
s within
carry
antics.



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.