CSE 484 / CSE M 584

Computer Security: Android Security

TA: Franzi Roesner

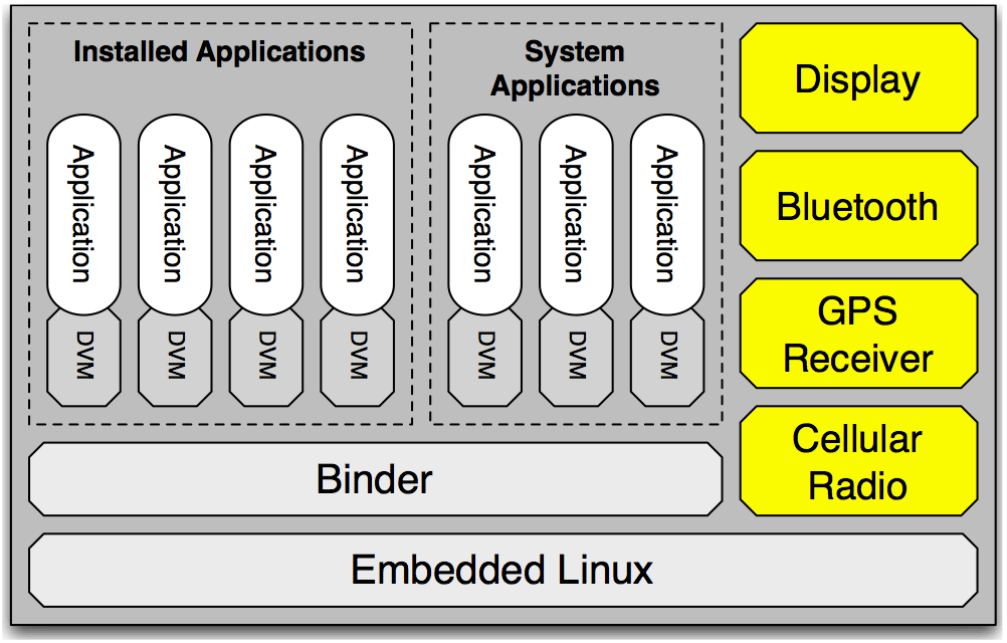
franzi@cs.washington.edu

Logistics

- **Homework #3** due tomorrow, 5pm.
- **Lab #3** due Friday, March 15, 11pm.
  - Email Ian or me for usernames!
- Lab #2 grades up, contact us with questions.
- Next week (3/14): Last section, final review.
- **Final exam:** Tuesday, 3/19, 2:30-4:30pm

Android Application Isolation

- Each app runs with its own user ID.
  - Android isolates them from each other.
  - Different from desktops!



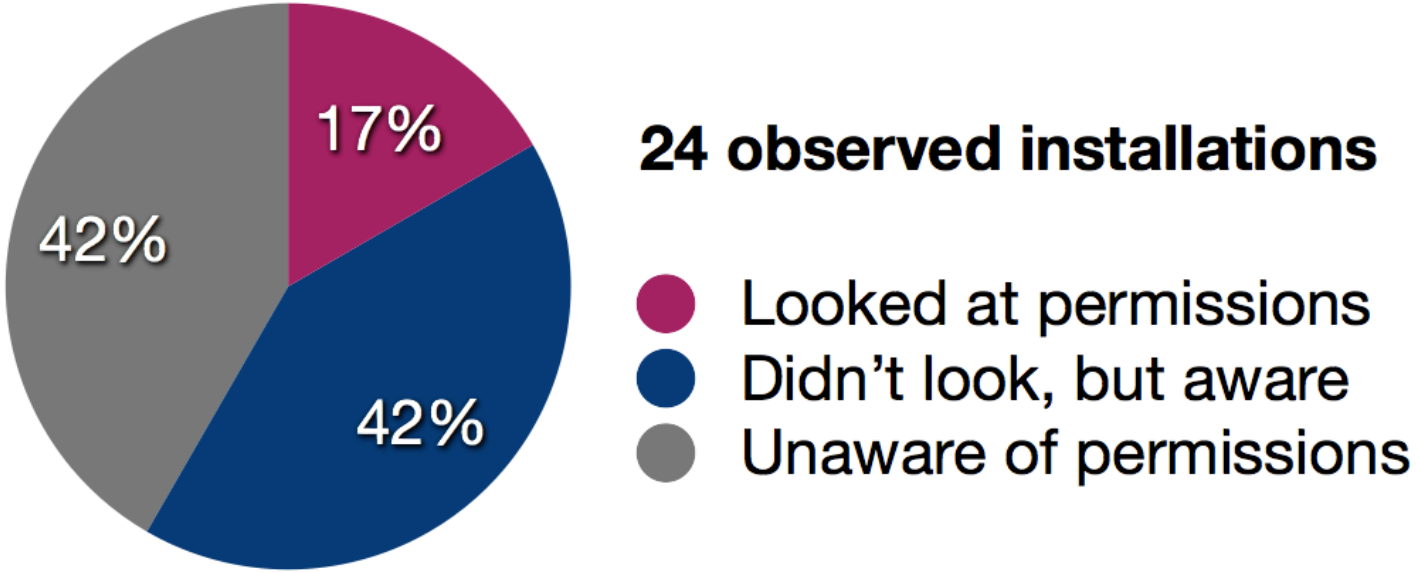
[From Enck et al., "A Study of Android Application Security", USENIX Security 2011.]

Application Permissions

- Apps must request permissions to access sensitive resources.
  - INTERNET, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, CAMERA, CALL_PHONE, READ_CALENDAR, READ_PHONE_STATE, SEND_SMS, REBOOT, and many more.
- Permissions requested from users at install-time: not optional!

Are Manifests Usable?

Do users pay attention to permissions?



… but 88% of users looked at reviews.

[From Felt et al., "Android Permissions: User Attention, Comprehension, and Behavior", SOUPS 2012.]

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%

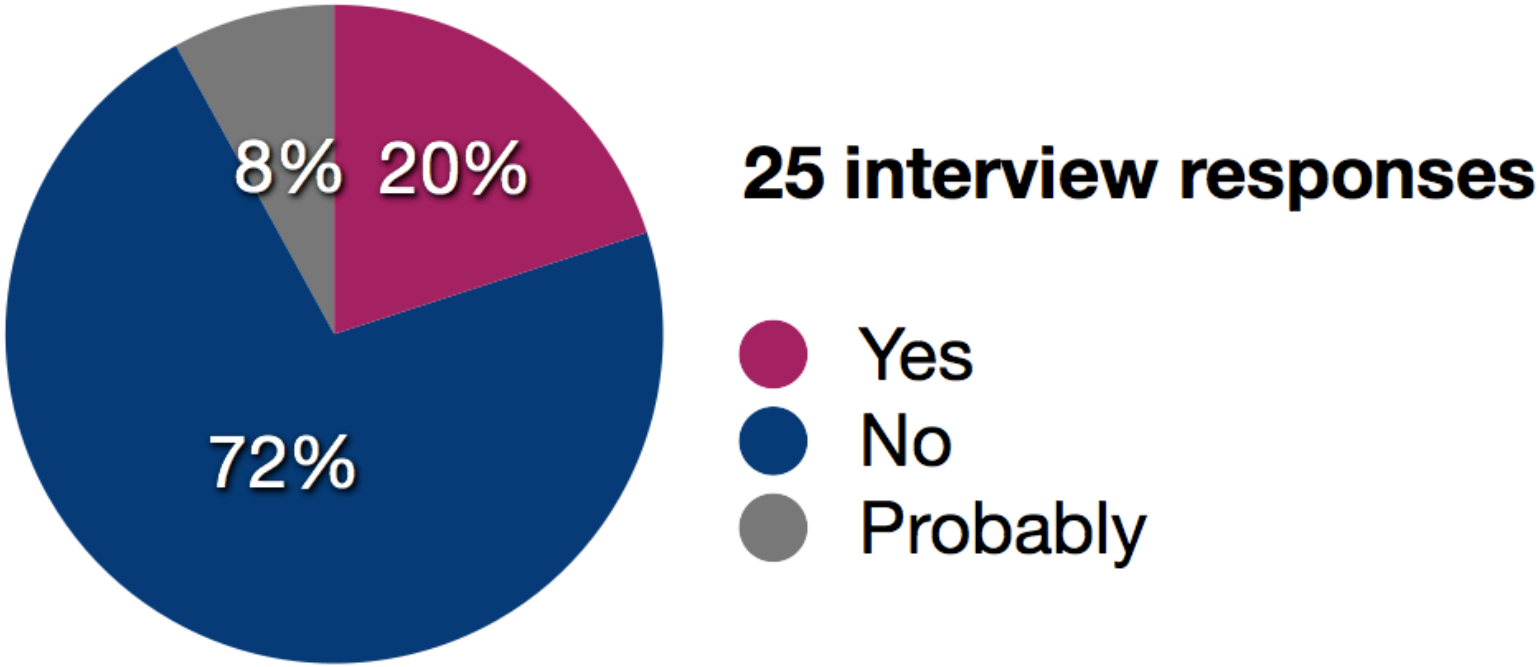
Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. n is the number of respondents. (Internet Survey, $n = 302$)

[From Felt et al., "Android Permissions: User Attention, Comprehension, and Behavior", SOUPS 2012.]

Are Manifests Usable?

Do users act on permission information?

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

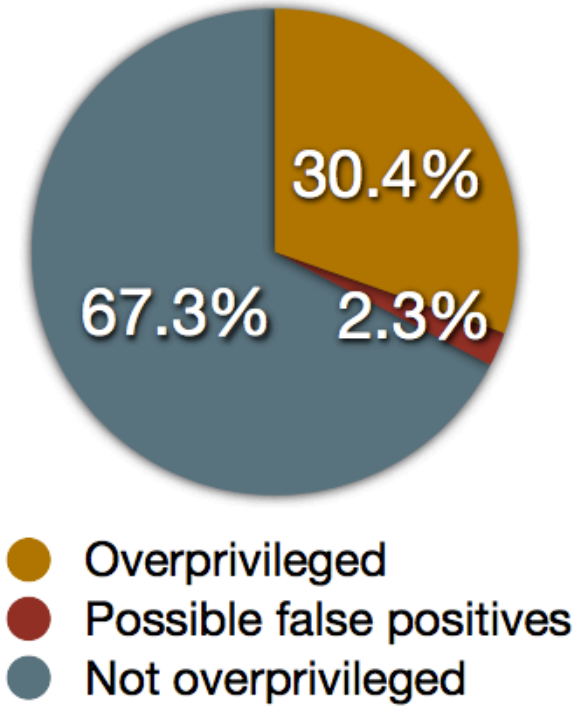


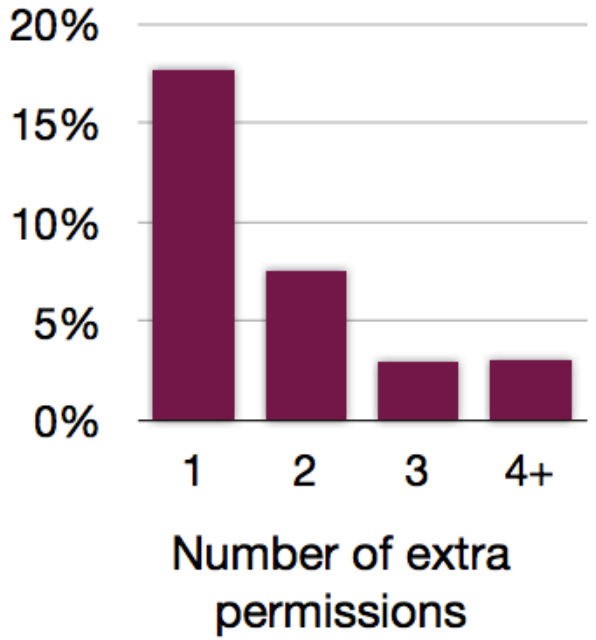
[From Felt et al., "Android Permissions: User Attention, Comprehension, and Behavior", SOUPS 2012.]

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.)

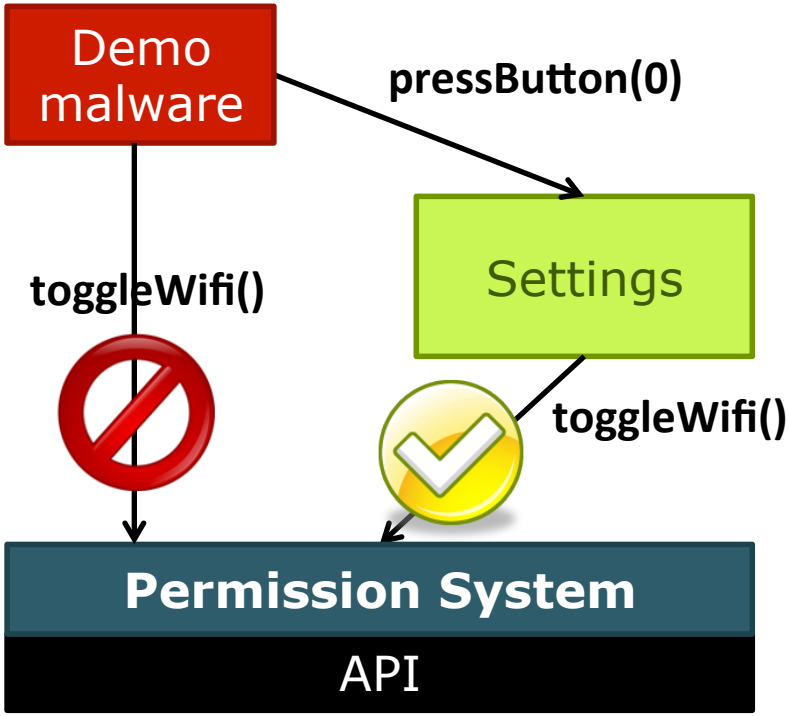




[From Felt et al., "Android Permissions Demystified", CCS 2011.]

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.



[From Felt et al., "Permission Re-Delegation: Attacks and Defenses", USENIX Security 2011.]

Android Application Components

- 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.

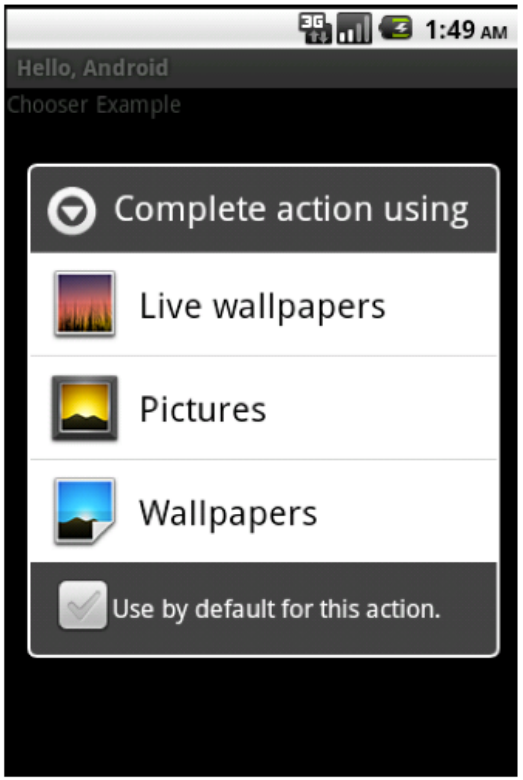
- Specified in each app's AndroidManifest.xml.

Inter-Process Communication

- Primary mechanisms: 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!



[From Chin et al., "Analyzing Inter-Application Communication in Android", MobiSys 2011.]

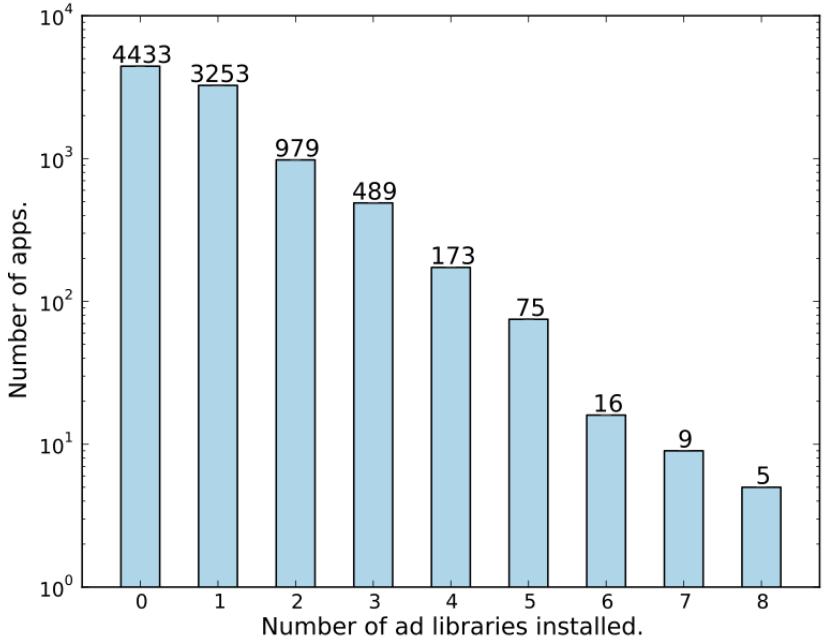
Intent Spoofing

- 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.

[From Chin et al., "Analyzing Inter-Application Communication in Android", MobiSys 2011.]

Information Leaks and Tracking

- Many apps include advertising or analytics libraries.
  - Unlike on the web (where we have iframes), these libraries always run with the host application's permissions.



[From Shekhar et al., "AdSplit: Separating smartphone advertising from applications", USENIX Security 2012.]

Resource		Demanded	Sent to Anywhere		Sent to A&A	
phone_state	IMEI	83	31	37%	14	17%
	Phone#	83	5	6%	0	0%
location		73	45	62%	30	41%
contacts		29	7	24%	0	0%
camera		12	1	8%	0	0%
account		11	4	36%	0	0%
logs		10	0	0%	0	0%
microphone		10	1	10%	0	0%
SMS/MMS messages		10	0	0%	0	0%
history&bookmarks		10	0	0%	0	0%
calendar		8	0	0%	0	0%
subscribed_feeds		1	0	0%	0	0%

These libraries may leak data.

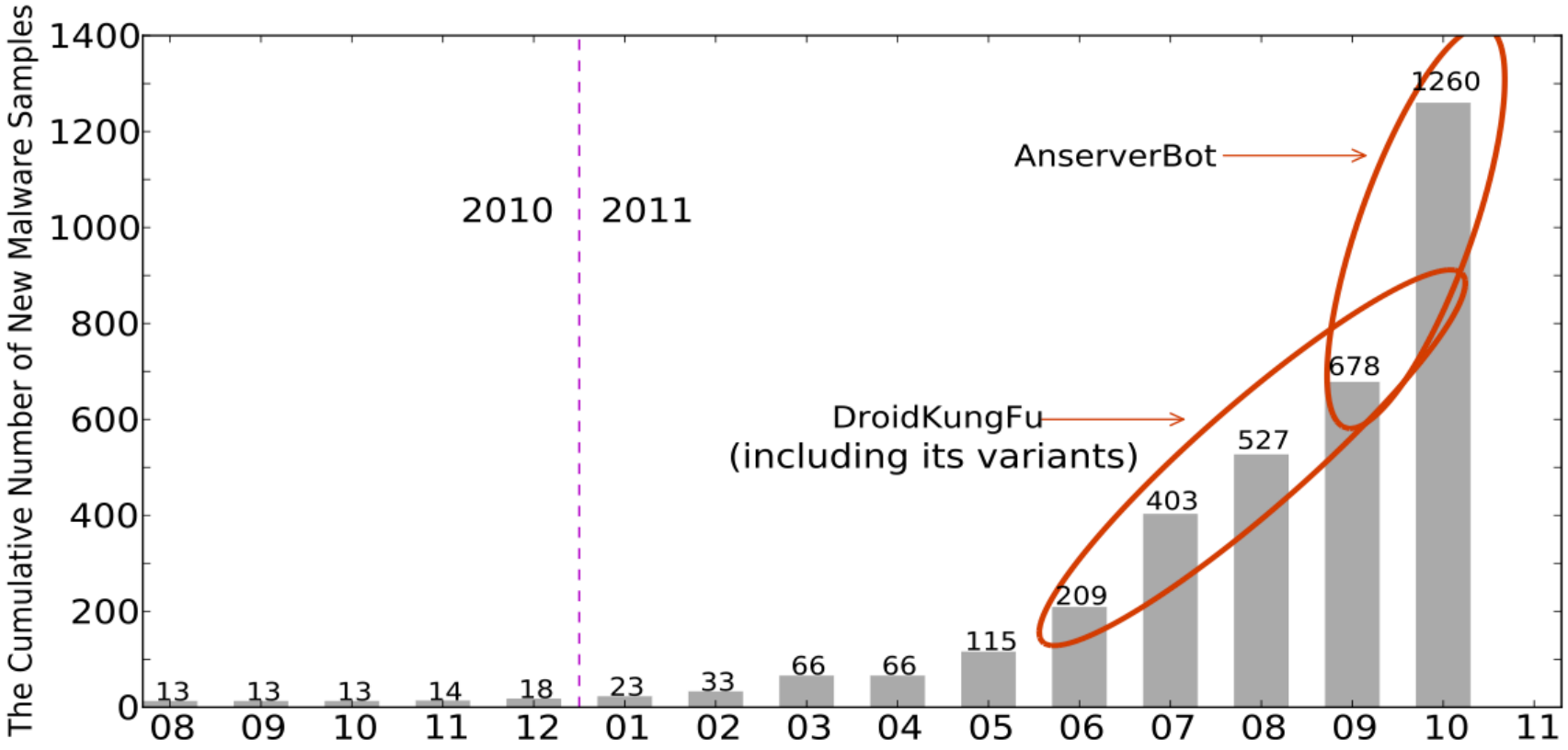
[From Hornyack et al., "These Aren't the Droids You're Looking For: Retrofitting Android to Protect Data from Imperious Applications", CSS 2011.]

Rooting

- Allows user to run applications with root privileges.
  - e.g., modify/delete system files, app management, CPU management, network management
- Done by exploiting vulnerability in firmware to install `su` binary.

Malware in the Wild

Android malware is growing.



[From Zhou et al., "Dissecting Android Malware: Characterization and Evolution", Oakland 2012]

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

[From Zhou et al., “Dissecting Android Malware: Characterization and Evolution”, Oakland 2012]

Defensive Research for Android

- Separating ads from apps
  - AdDroid (Felt et al.), AdSplit (Shekhar et al.)
- User-driven access control (Roesner et al.)
- Dynamic information flow tracking
  - e.g., TaintDroid (http://appanalysis.org/), AppFence (http://appfence.org/)
- Static analysis for malware detection
  - e.g., SPARTA (http://www.cs.washington.edu/sparta/)
- Many more!