NDSS 2012, February 7th 2012, San Diego

NC STATE UNIVERSITY
Department of Computer Science

Systematic Detection of Capability Leaks in Stock Android Smartphones

Michael Grace Yajin Zhou Zhi Wang Xuxian Jiang

North Carolina State University

Phones and Computers

"I have always wished that my computer would be as easy to use as my telephone. My wish has come true. I no longer know how to use my telephone."

– Bjarne Stroustrup (designer of C++)

Phones: the PCs of the Future?

- Smartphone shipments increased 42% between 3Q 2010 and 3Q 2011 (Gartner, 11/15/2011)

- More smartphones shipping than personal computers (IDC, 2/7/2011)
  - New markets: first computer = smartphone

Smartphone ≠ Handheld PC

- Unique abilities specific to the form factor
  - Many sensors: “context-aware”
  - Dialup ➔ always on ➔ always with you
- Resource constrained
- Different vendor relationships and primacy

Related Work

- Problems with Permissions
  - e.g., **Kirin** [Enck *et al.*, CCS '09], **Soundcomber** [Schlegel *et al.*, NDSS '11], **Stowaway** [Felt *et al.*, CCS '11], **Guess Who's Texting You** [Schrittwieser *et al.*, NDSS '12]…
- Information Leak Detection
  - e.g., **PiOS** [Egele *et al.*, NDSS '11], **TaintDroid** [Enck *et al.*, OSDI '10]…
- Phone Defenses
  - e.g., **MockDroid** [Beresford *et al.*, HotMobile '11], **TISSA** [Zhou *et al.*, TRUST '11], **AppFence** [Hornyack *et al.*, CCS '11], **Permission Re-Delegation** [Felt *et al.*, USENIX Security '11], **QUIRE** [Dietz *et al.*, USENIX Security '11], **XManDroid** [Bugiel *et al.*, NDSS '12], **MoCFI** [Davi *et al.*, NDSS '12]…
- Market Issues
  - e.g., **DroidMOSS** [Zhou *et al.*, CODASPY '12], **DroidRanger** [Zhou *et al.*, NDSS '12]…

Firmware and Fragmentation

- A conspicuous gap in the body of work!
- Not like on desktops, or other smartphone platforms
- Research Goal: Determine the impact firmware customizations have on security and privacy

Android Capabilities

- Platform defines some APIs
- APIs *may* require capabilities (called permissions)
- Applications can define APIs the same way
- What happens when an application defines a new API based on a restricted old one?
  - That's up to the author!

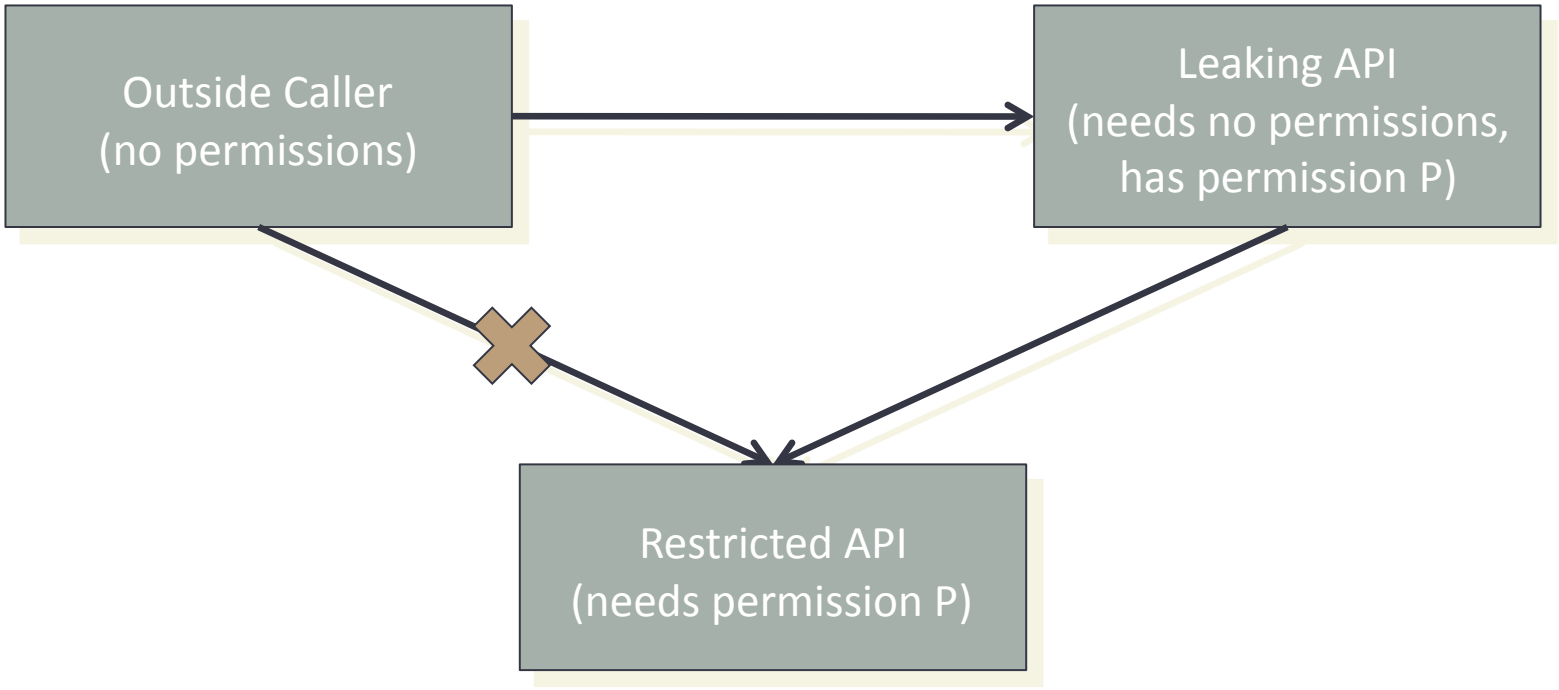
Capability Leaks

- Capability Leak: A situation where an app can gain access to a restricted API without requesting proper permission

- Explicit Capability Leak: Broadening access to a restricted API by exposing it via another API

- Implicit Capability Leak: Inheriting permissions from other applications

Explicit Capability Leaks



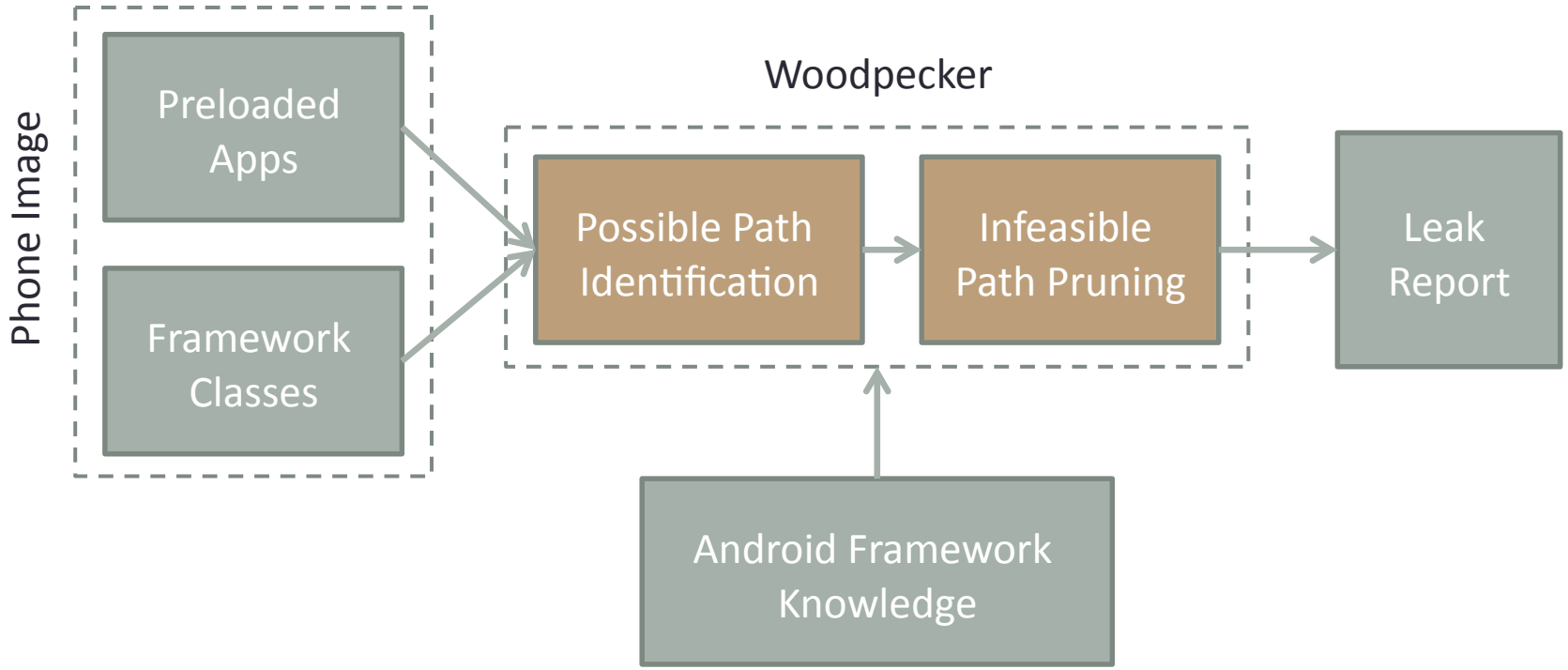
Detecting Capability Leaks

- Android SDK gives us no tools!

- Function composition
  - Capability leak: g(x) = f(x) + some other stuff

- Intuitive algorithm:
  1. Find interesting (dangerous) APIs (f(x))
  2. Find new API definitions (g(x))
  3. Link them!

System Overview



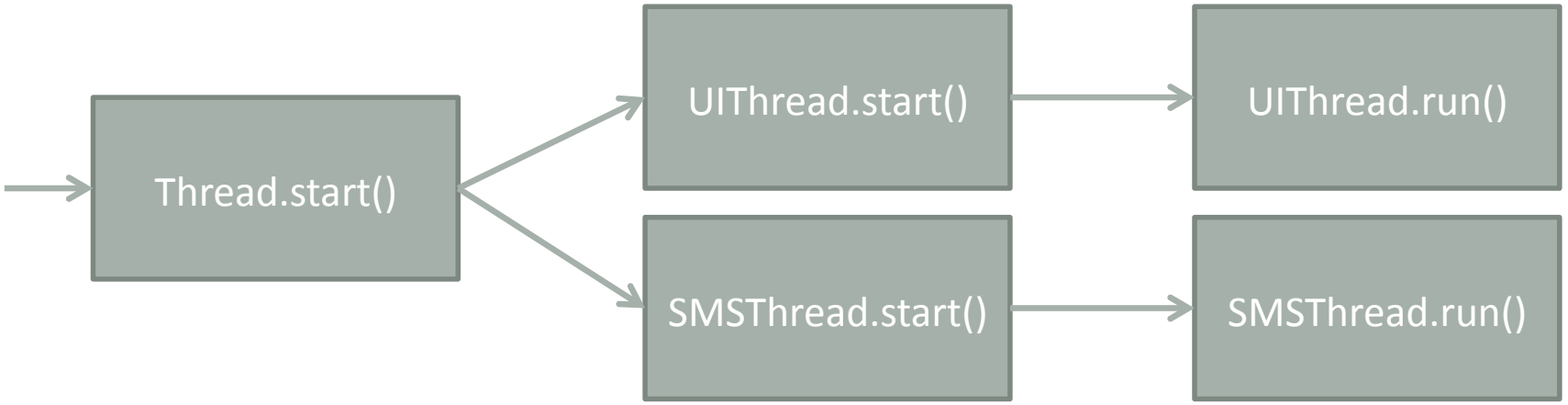
Possible Path Identification

1. Construct a control-flow graph

2. Find all paths from an IPC entry point to an API of interest

Possible Path Identification: Challenges

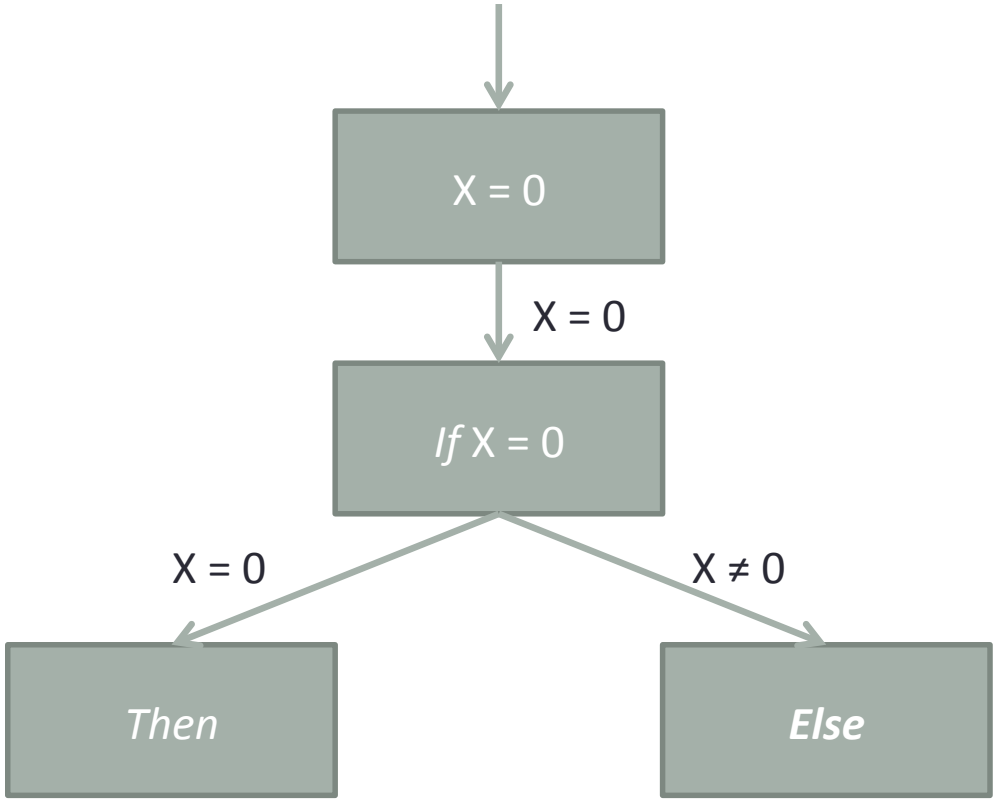
- Object references
  - Class hierarchy used to conservatively resolve references
- Extensive use of callbacks
  - Use framework knowledge to stitch together callbacks



Infeasible Path Pruning

- *Many* potential paths exist
  - Most are either impossible or uninteresting
- Must prune these uninteresting paths
  - Branch conditions need an understanding of program data-flow
  - Explicit permission checks are “infeasible paths”
- Our approach: *Symbolic Path Simulation*

Symbolic Path Simulation



Implementation

- Based on the baksmali decompiler (1.2.6)

- Covers 13 permissions, controlling:
  - Phone information
  - Location API
  - Phone dialing
  - Sending text messages
  - Camera/microphone
  - Rebooting/shutting down the device
  - Installing/removing apps
  - Factory reset

Evaluation

htc quietly brilliant™

MOTOROLA

SAMSUNG

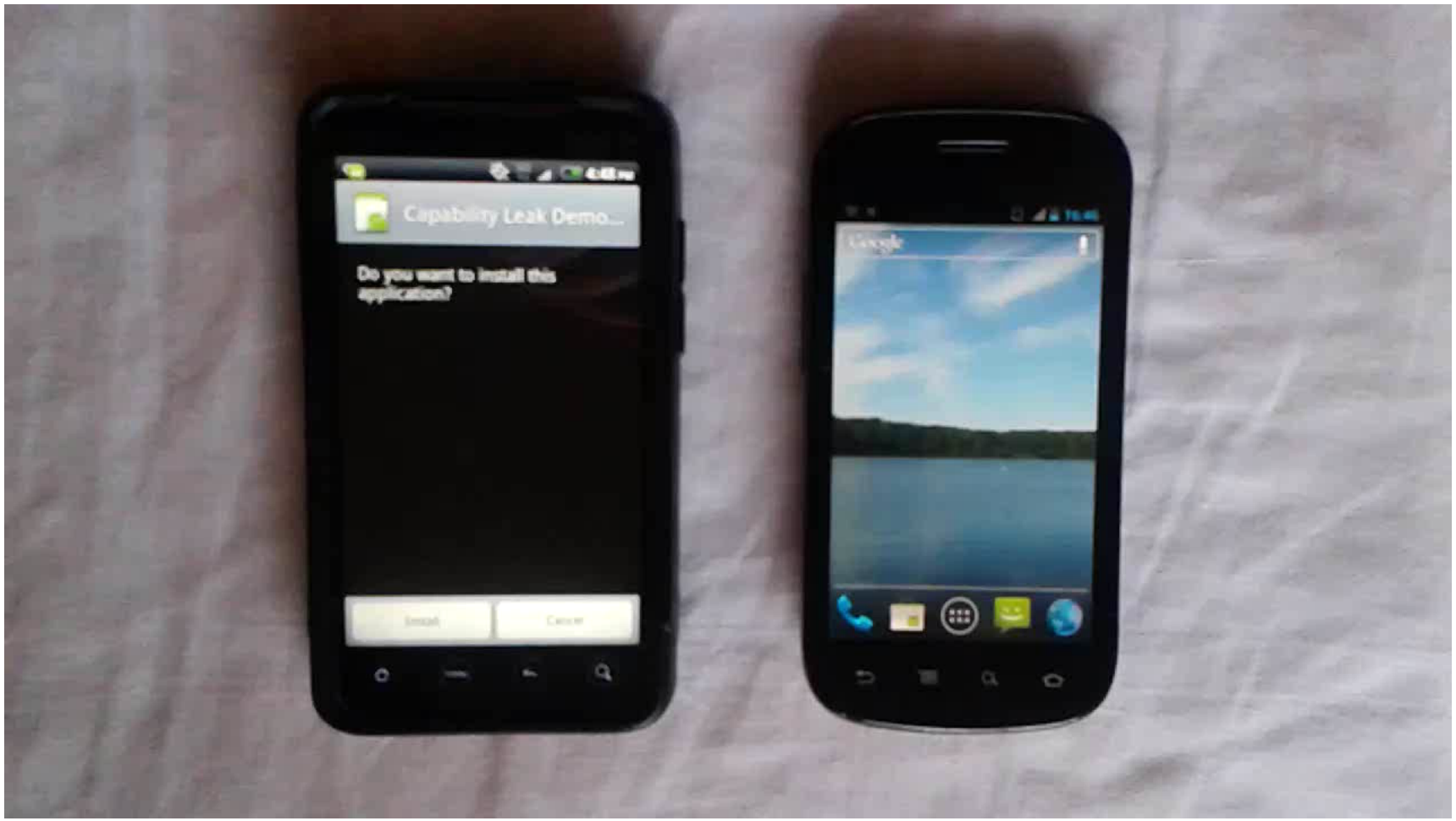
Google™

Explicit Capability Leaks Found

NC STATE UNIVERSITY

	HTC			Motorola		Samsung	Google	
Permission	Legend	EVO 4G	Wildfire S	DROID	DROID X	Epic 4G	Nexus One	Nexus S
Coarse Location	✓	✓			✓			
Fine Location	✓	✓			✓			
Call Phone						✓		
Call Privileged								
Camera	✓	✓	✓					
Delete Packages	✓	✓	✓	✓	✓	✓	✓	✓
Install Packages								
Master Clear					✓			
Read Phone State				✓				
Reboot		✓						
Record Audio	✓	✓	✓					
Send SMS	✓	✓	✓					
Shutdown		✓						

Demo


Capability Leak Demo...
Do you want to install this application?

Implicit Capability Leaks Found

NC STATE UNIVERSITY

	HTC			Motorola		Samsung	Google	
Permission	Legend	EVO 4G	Wildfire S	DROID	DROID X	Epic 4G	Nexus One	Nexus S
Coarse Location	✓	✓	✓					
Fine Location			✓					
Call Phone						✓		
Call Privileged			✓					
Camera								
Delete Packages								
Install Packages								
Master Clear								
Read Phone State	✓	✓	✓			✓		
Reboot								
Record Audio								
Send SMS								
Shutdown								

Performance Measurement

Vendor	Model	Time	# Apps
HTC	Legend	3366.63s	125
	EVO 4G	4175.03s	160
	Wildfire S	3894.37s	144
Motorola	DROID	2138.38s	76
	DROID X	3311.94s	161
Samsung	Epic 4G	3732.56s	138
Google	Nexus One	2059.47s	76
	Nexus S	1815.71s	72

Discussion

- Accuracy
  - False negatives: native code, undocumented extensions
  - False positives: conservative analysis
- Threads and Time
  - Instruction interleaving, shared state
  - Example: callback handling

Conclusions

- Capability leaks present a tangible threat to security and privacy on existing Android smartphones

- We present a system, Woodpecker, to detect these capability leaks

Thank you!

Implicit Capability Leaks

- Applications don't have permissions, *user identifiers* (UIDs) do.