Cellular Networks and Mobile Computing
COMS 6998-8, Spring 2012

Instructor: Li Erran Li
(lel2139@columbia.edu)

http://www.cs.columbia.edu/~coms6998-8/

Lecture 13: Mobile Privacy

Mobile Privacy

Data privacy

- Detecting and preventing privacy leaks
  - PiOS for iOS
  - TaintDroid for Android

- Stealthy information leaks through covert channels and prevention
  - Soundcomber

- Auditing to determine which files accessed after device loss
  - Keypad

Location privacy [Presented by Sameer Choudhary]

- Quantifying location privacy

PiOS: Detecting Privacy Leaks in iOS Applications

Manuel EGELE, Christopher KRUEGEL, Engin KIRDA, Giovanni VIGNA
{maeg,chris,vigna}@cs.ucsb.edu, ek@ccs.neu.edu
Int. Secure Systems Lab, UCSB & TU Vienna & Northeastern University
SAP Security Info Session, Wed. April 20th 2011

Motivation

- App Store: 300k apps available, 10 billion apps downloaded
- iOS apps are created by third party developers
- “iPhone Developer License Agreement“
- States guidelines (e.g., user's privacy)
- Submitted binaries are scrutinized by Apple through a secret vetting process
- Apps passing the vetting process → App Store
- Cydia: repository for jailbroken devices

Courtesy: Egele et. al

Motivation (cont.)

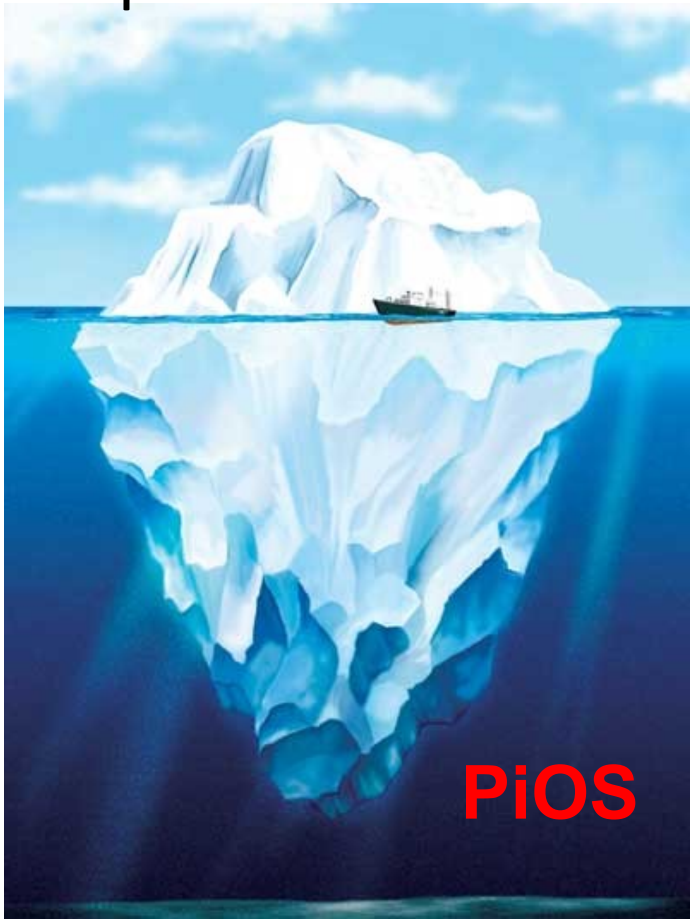
- Vetting process is not flawless
- ”Bad apps” make it to the App Store
- Apple removed several apps after they were available e.g.,
  - Flashlight (enables tethering w/o the network operator's consent)
  - Storm8 games harvested device phone numbers
  - MogoRoad collect phone numbers of free app users

  → Telemarketers called and offered the paid full version

Courtesy: *Egele et. al*

Motivation (cont.)

- How big is the problem?



Overview

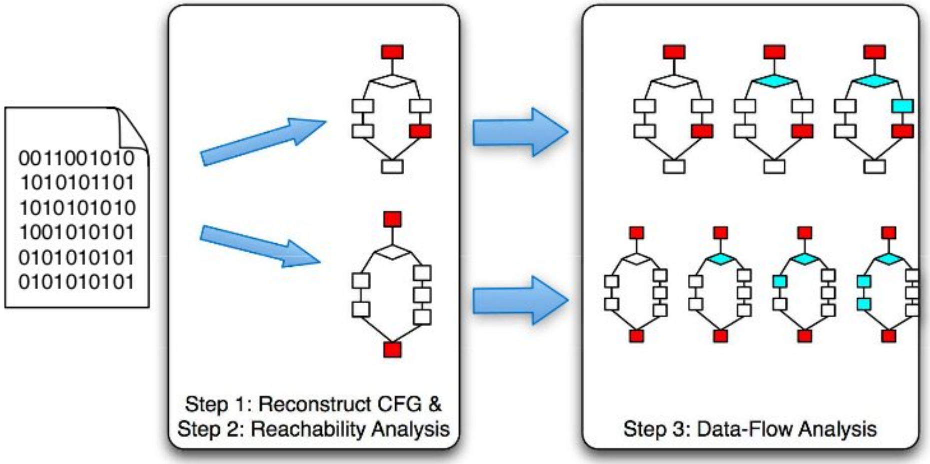
- Motivation
- Goals & Challenges
- PiOS Analysis
  - Extract CFG
  - Reach ability analysis
  - Data Flow analysis
- Evaluation
- Summary

Goals & Challenges

- Goals
  - Identify Apps that access privacy sensitive information and transmit this information over the Internet without user intervention or consent
  - Perform this analysis on a large body of Apps
  - Gain insight in how Apps handle privacy sensitive data
- Challenges
  - Apps are only available as binary executable
  - App Store Apps are encrypted

Courtesy: *Egele et. al*

Approach



1. Extract control flow graph (CFG)
2. Identify sources of sensitive information and network communication sinks
   – Perform reachability analysis between sources and sinks
3. Data flow analysis on detected paths

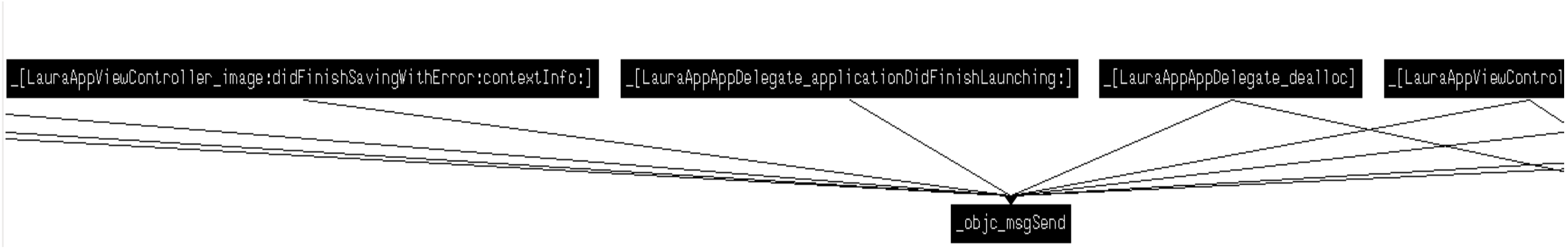
Courtesy: *Egele et. al*

Background (iOS & DRM)

- App Store apps are encrypted and digitally signed by Apple
- Loader verifies signature and performs decryption in memory
- Decrypting App Store apps:
  - Attach with debugger while app is running
  - Dump decrypted memory regions
  - Reassemble binary, toggle encrypted flag
- Cydia Apps are not encrypted

Analysis (CFG)

- IDA Pro generated CFG for "Bomberman"



objc_msgSend

Courtesy: *Egele et. al*

Analysis (CFG)

- Most iOS apps are written in Objective-C
- Cornerstone: objc_msgSend dispatch function
- Task: Resolve type of receiver and value of selector for objc_msgSend calls
  - Backwards slicing
  - Forward propagation of constants and types
- Result: Inter and intra procedural CFG is constructed from successfully resolved objc_msgSend calls

Courtesy: *Egele et. al*

Background (objc_msgSend)

- objc_msgSend dynamic dispatch function
- Arguments:
  - Receiver (Object)
  - Selector (Name of method, string)
  - Arguments (vararg)
- Method look-up:
  - Dynamically traverses class hierarchy
  - Calls the method denoted by selector

Non-trivial to do statically

Courtesy: *Egele et. al*

Analysis (CFG)

- Most iOS apps are written in Objective-C
- Cornerstone: objc_msgSend dispatch function
- Task: Resolve type of receiver and value of selector for objc_msgSend calls
  - Backwards slicing
  - Forward propagation of constants and types
- Result: Inter and intra procedural CFG is constructed from successfully resolved objc_msgSend calls

Example ObjC to ASM

- 1 LDR R0, =off_24C58 → **UIDevice**
- 2 LDR R1, =off_247F4 → **currentDevice**
- 3 LDR R0, [R0]
- 4 LDR R1, [R1]
- 5 BLX _objc_msgSend **r0? UIDevice** **r1? ::currentDevice**
- 6 LDR R1, =off_247F0
- 7 LDR R1, [R1] → **uniqueIdentifier**
- 8 BLX _objc_msgSend **UIDevice** **r1? ::uniqueIdentifier**
- …

9 STR R0, [SP,#0x60+var_34]
10 LDR R3, [SP,#0x60+var_34]
…
11 BLX _objc_msgSend **NSString** ::initWithFormat:(fmt: "**uniqueid=%@&username=%@&country=%@&email=%@**")
…
12 BLX _objc_msgSend **POSTScore** ::startPostingData:toURL: (0x1b478)

Finding Privacy Leaks

- Inter and intra procedural Control Flow Graph
- Reachability Analysis (find paths)
  - From interesting *sources*
  - To network *sinks*
- Implicit interruption of CFG for user-input (e.g., dialog boxes, etc.)
  - Touch events are generated by the OS not in the developer's code

Courtesy: *Egele et. al*

Sources and Sinks

- Sources:
  - Address book
  - GPS coordinates
  - Unique device ID
  - Photos
  - Email account settings
  - WiFi connection information
  - Phone information (phone #, call lists, etc.)
  - YouTube application Settings
  - MobileSafari settings and history
  - Keyboard Cache (every word typed w/o passwords)

Sinks:

- NSUrlConnection, NSString::initWithContentsOfURL, etc.

Data Flow Analysis

- For each source/sink pair perform reachability analysis
  - Is there a path in the CFG that connects the source to the sink?
- Along paths that result from reachability analysis
  - Taint flow analysis
  - Conservatively taint results of methods without implementation if at least one input parameter is tainted

Courtesy: *Egele et. al*

Evaluation

- 1,407 Applications (825 from App Store, 582 from Cydia)
  - Resolving calls to objc_msgSend
  - 4,156,612 calls
  - 3,408,421 identified (82%)

    i.e., class and selector exist and match
- Pervasive ad and statistic libraries:
  - 772 Apps (55%) contain at least one such library
  - Leak UDIDs, GPS coordinates, etc.

Ad and Statistic Libraries

- 82% use AdMob (Google)
- Transmit UDID and AppID on start-up and ad request
- Ad company can build detailed usage profiles
  - Gets info from all Apps using the ad library
- UDIDs cannot be linked to a person directly
- Problem: Location based Apps
  - Access to GPS is granted per App libraries linked into location based apps have access to GPS too

Is Leaking UDIDs a Problem?

- UDIDs cannot be linked to a person directly
- But: Combine UDID with additional information e.g.,
  - Google App can link UDID to a Google account
  - Social networking app get user's profile (often name)
- Linking ICC-ID with UDID is trivial
  - 114,000 iPad 3G users

Evaluation Data Flow Analysis

- Reachability analysis: 205 apps
- Enumerate all paths from source to sink with length < 100 basic blocks

- Perform data flow analysis along these paths
- PiOS detected flows of sensitive data for 172 apps (TP)
- 6 true negatives, 27 false negatives
  - FN e.g., aliased pointers, format string from config file, JSON library (i.e., invoking JSONRepresentation on each object in a dictionary, PiOS does not track types in aggregates)

Evaluation: Leaked Data

Source	#App Store 825	#Cydia 582	Total 1407
DeviceID	170 (21%)	25(4%)	195(14%)
Location	35(4%)	1(0.2%)	36(3%)
Address book	4(0.5%)	1(0.2%)	5(0.4%)
Phone number	1(0.1%)	0(0%)	1(0.1%)
Safari history	0(0%)	1(0.2%)	1(0.1%)
Photos	0(0%)	1(0.2%)	1(0.1%)

Courtesy: *Egele et. al*

Evaluation: Case Studies (1)

- Address book contents:
  - Apps have unrestricted access to the address book
  - Facebook and Gowalla transmit the complete AB
  - Facebook: detailed warning that data will be sent
  - Gowalla (Social networking app):
    - User can send Invitations to contacts
    - Complete AB is sent on load (i.e., before the user chooses a contact)

    → "We couldn't find any friends from your Address Book who use Gowalla. Why don't you invite some below?"

Courtesy: *Egele et. al*

Evaluation: Case Studies (2)

- Phone number
- Nov. 2009 Apple removed all Storm8 titles (social games) from App Store
  - because apps transmitted phone numbers (SBFormattedPhoneNumber)
  - New versions don't have that code anymore
- Old version of “Vampires“ PiOS detected the privacy leak

→ Improvement over Apple vetting process

Summary

- PiOS is able to create a CFG from ObjC binaries
- 82% of the calls to objc_msgSend could be resolved
- Data flow analysis is used to identify privacy leaks
- PiOS showed how pervasive ad and statistics libraries are used in apps
- PiOS identified unknown and known privacy leaks that lead to App Store removal in the past

DETECTING PRIVACY LEAKS IN SMARTPHONE APPLICATIONS AT RUN TIME

Byung-Gon Chun
Yahoo! Research

Joint work with William Enck, Peter Gilbert,
Landon P. Cox, Jaeyeon Jung,
Patrick McDaniel, Anmol N. Sheth

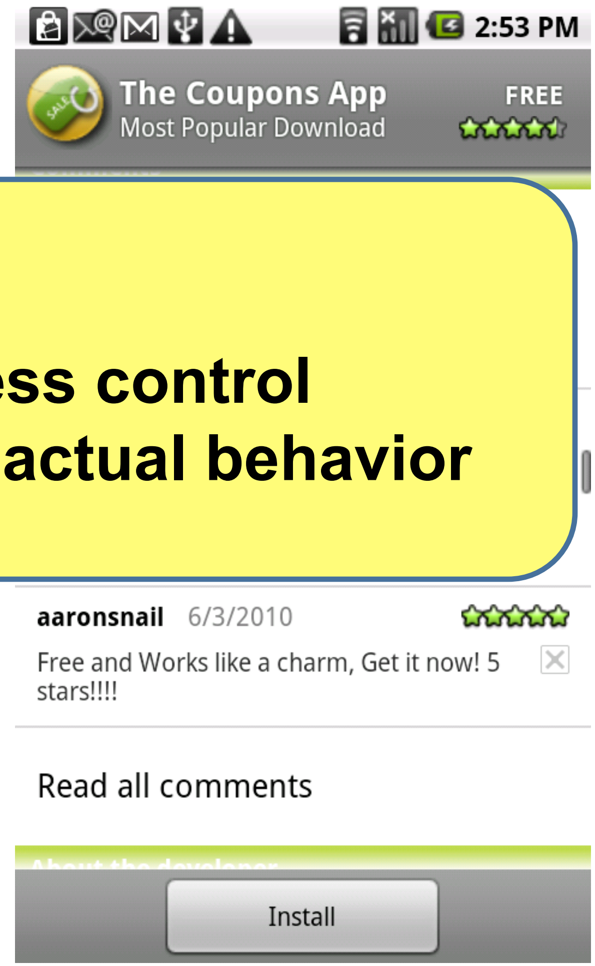
Roadmap

- Approach
- TaintDroid design
- Performance study
- Application study

TaintDroid Goal

Monitor app behavior to determine when privacy sensitive information leaves the phone in real time

Current “Best” Practice



- **Trust-or-cancel**
- **Coarse-grained access control**
- **No visibility into the actual behavior**

Courtesy: Byung-Gon et. al

TaintDroid Approach

- Look inside of applications to watch how they use privacy sensitive data
- Trust-or-cancel → Trust-but-verify

Challenges

- Smartphones are resource constrained
- Third-party applications are entrusted with several types of privacy sensitive information
- Context-based privacy information is dynamic and can be difficult to identify when sent
- Applications can share information

Dynamic Taint Analysis

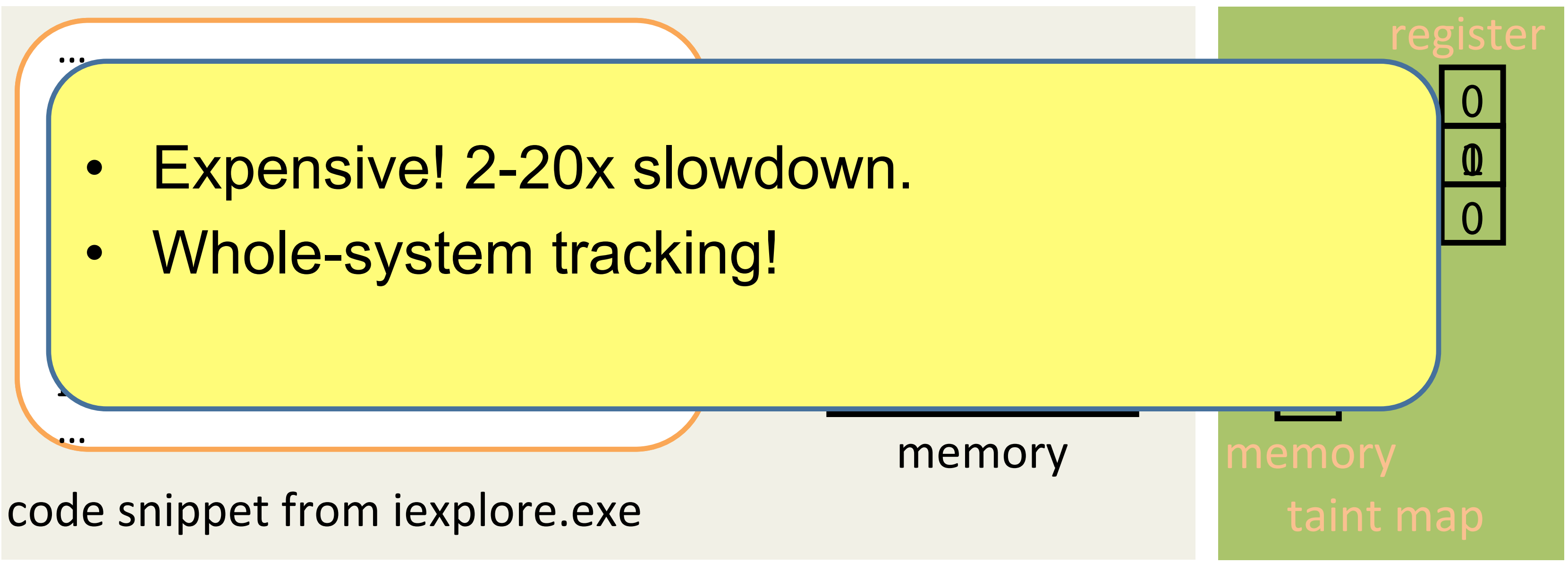
- A technique that tracks information dependencies from an origin
- Taint
  - Source
  - Propagation
  - Sink

```
C = Taint_source()
…
A = B + C
…
Network_send(A)
```


Courtesy: Byung-Gon et. al

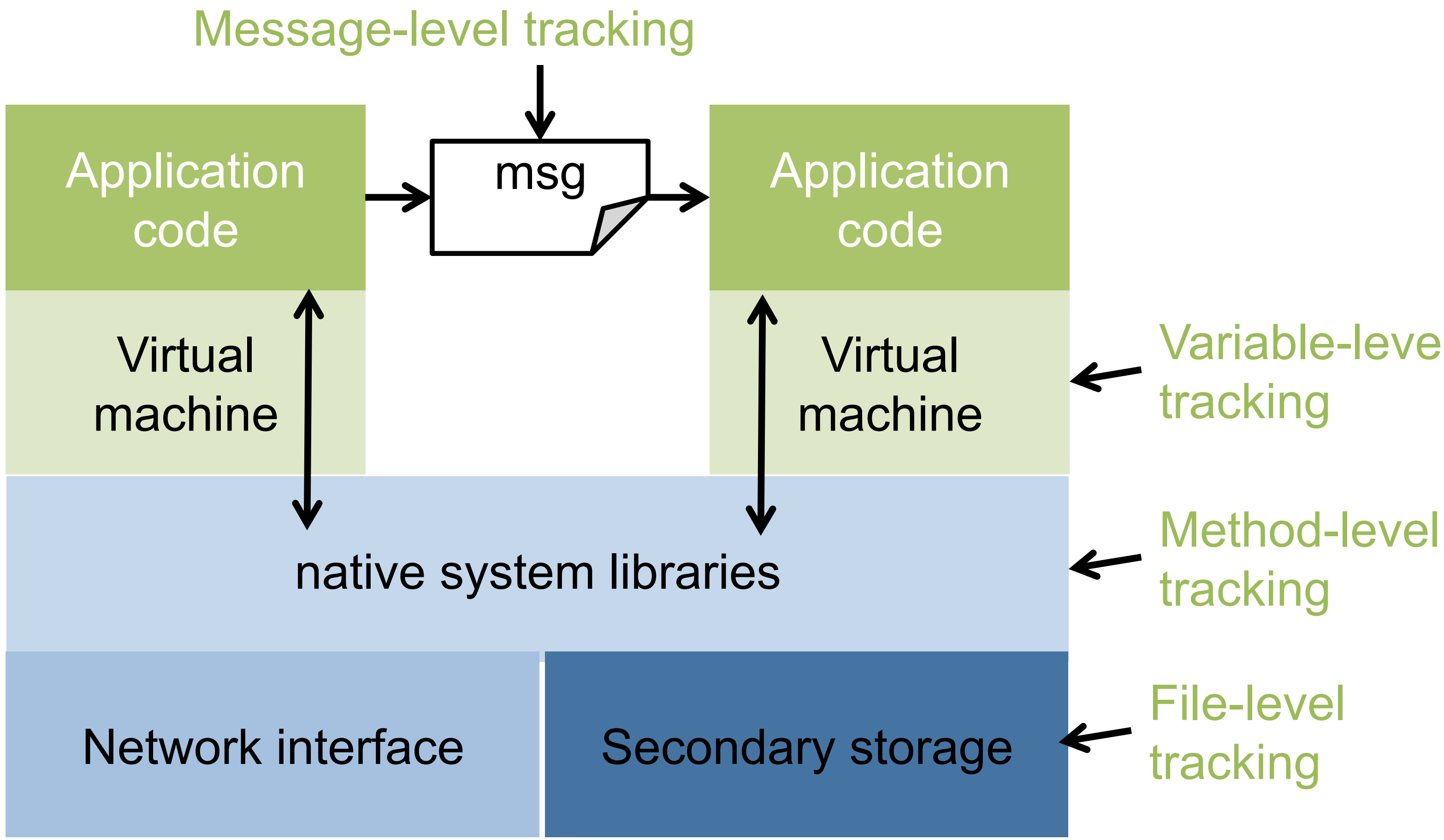
Dynamic Taint Analysis in Action

- Expensive! 2-20x slowdown.
- Whole-system tracking!

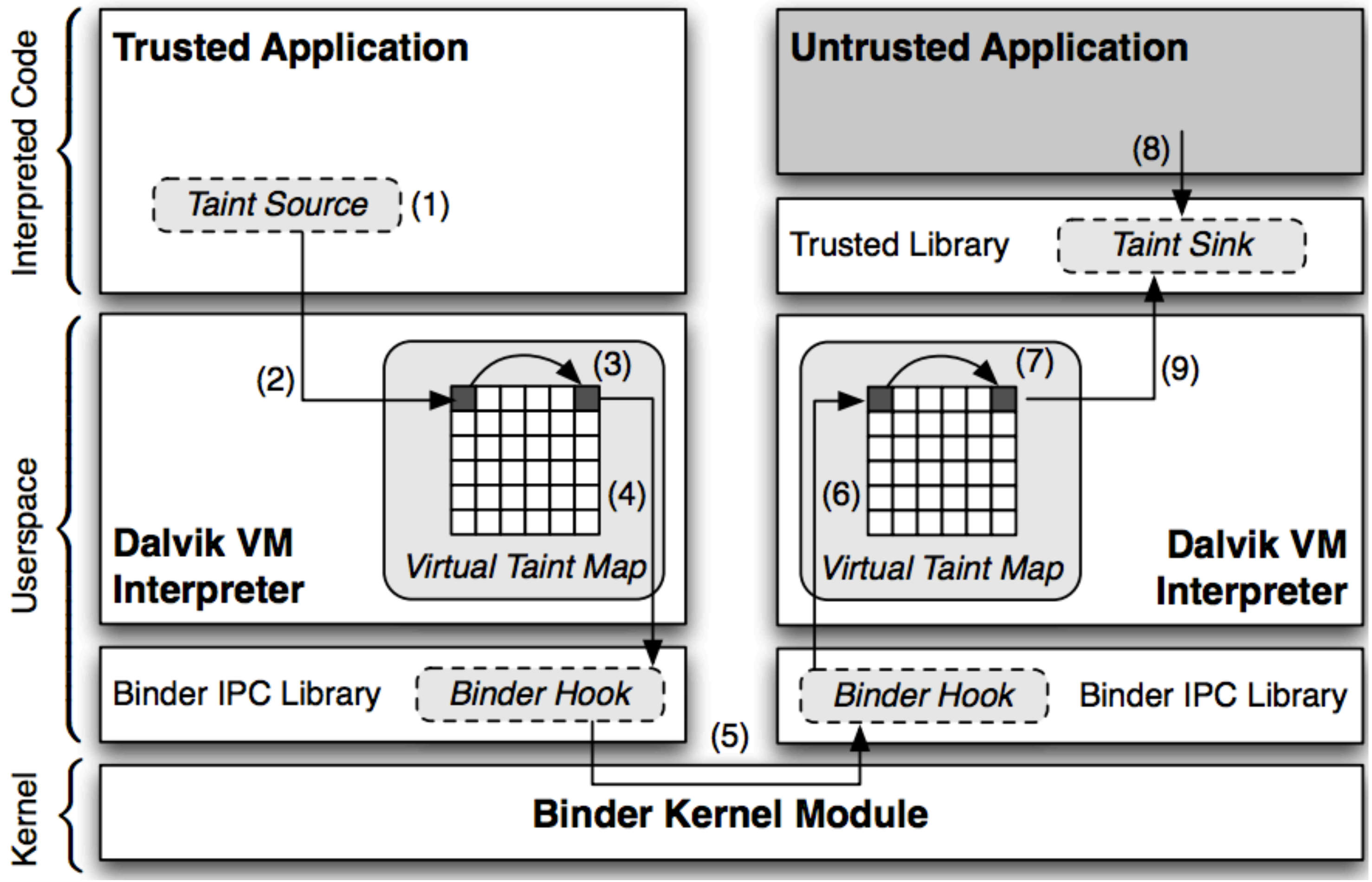


TaintDroid

Leverage Android Platform Virtualization



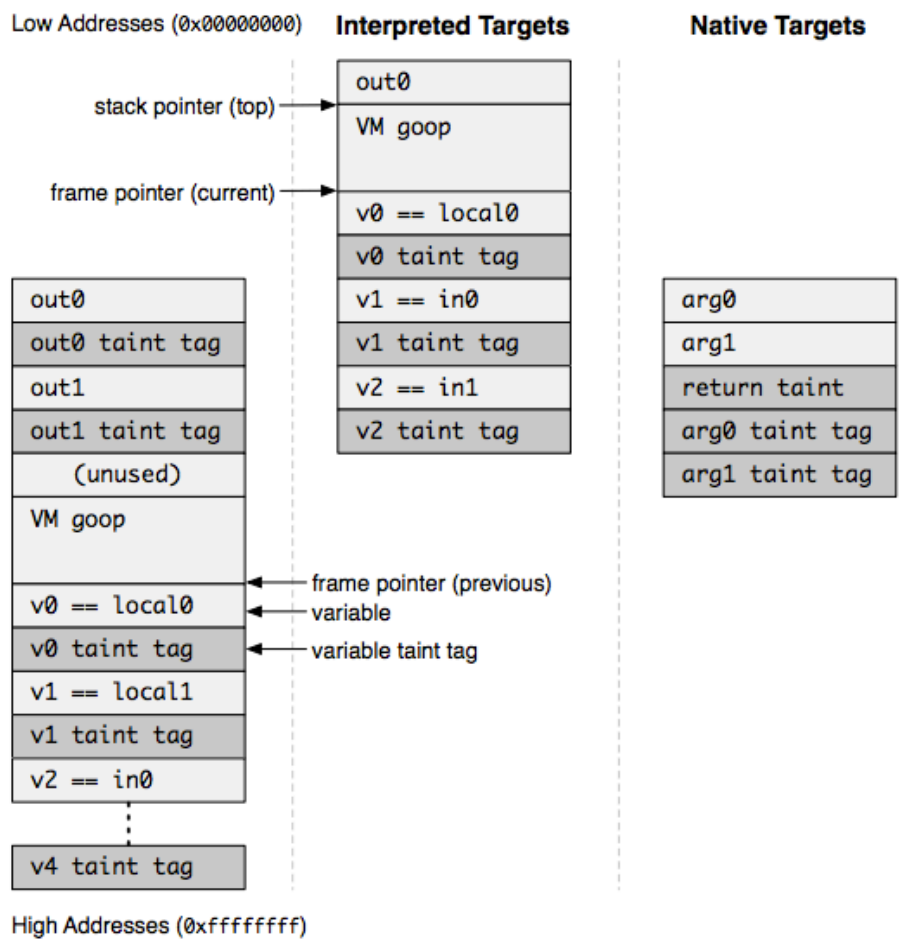
TaintDroid Android Architecture in Detail



VM Variable-level Tracking

- Modified the Dalvik VM interpreter to store and propagate taint tags (a taint bitvector) on variables
  - Local variables and method args: taint tags stored adjacent to variables on the internal execution stack.
  - Class fields: similar to locals, but inside static field heap objects
  - Arrays: one taint tag per array to minimize overhead

Modified Stack Format Example



DEX Taint Propagation Logic

Op Format	Op Semantics	Taint Propagation	Description
const-op vA C	vA ← C	T(vA) ← 0	Clear vA taint
move-op vA vB	vA ← vB	T(vA) ← T(vB)	Set vA taint to vB taint
move-op-R vA	vA ← R	T(vA) ← T(R)	Set vA taint to return taint
return-op vA	R ← vA	T(R) ← T(vA)	Set return taint (0 if void)
move-op-E vA	vA ← E	T(vA) ← T(E)	Set vA taint to exception taint
throw-op vA	E ← vA	T(E) ← T(vA)	Set exception taint
unary-op vA vB	vA ← op vB	T(vA) ← T(vB)	Set vA taint to vB taint
binary-op vA vB vC	vA ← vB op vC	T(vA) ← T(vB) U T(vC)	
binary-op vA vB	vA ← vA op vB	T(vA) ← T(vA)UT(vB)	Set vA taint to vA taint U vB taint
binary-op vA vB C	vA ← vB op C	T(vA) ← T(vB)	Set vA taint to vB taint
aput-op vA vB vC	vB[vC] ← vA	T(vB[]) ← T(vB[]) UT(vA)	Update array vB taint with vA taint
…			

Native Methods

- Applications execute native methods through the Java Native Interface (JNI)
- TaintDroid uses a combination of heuristics and method profiles to patch VM tracking state

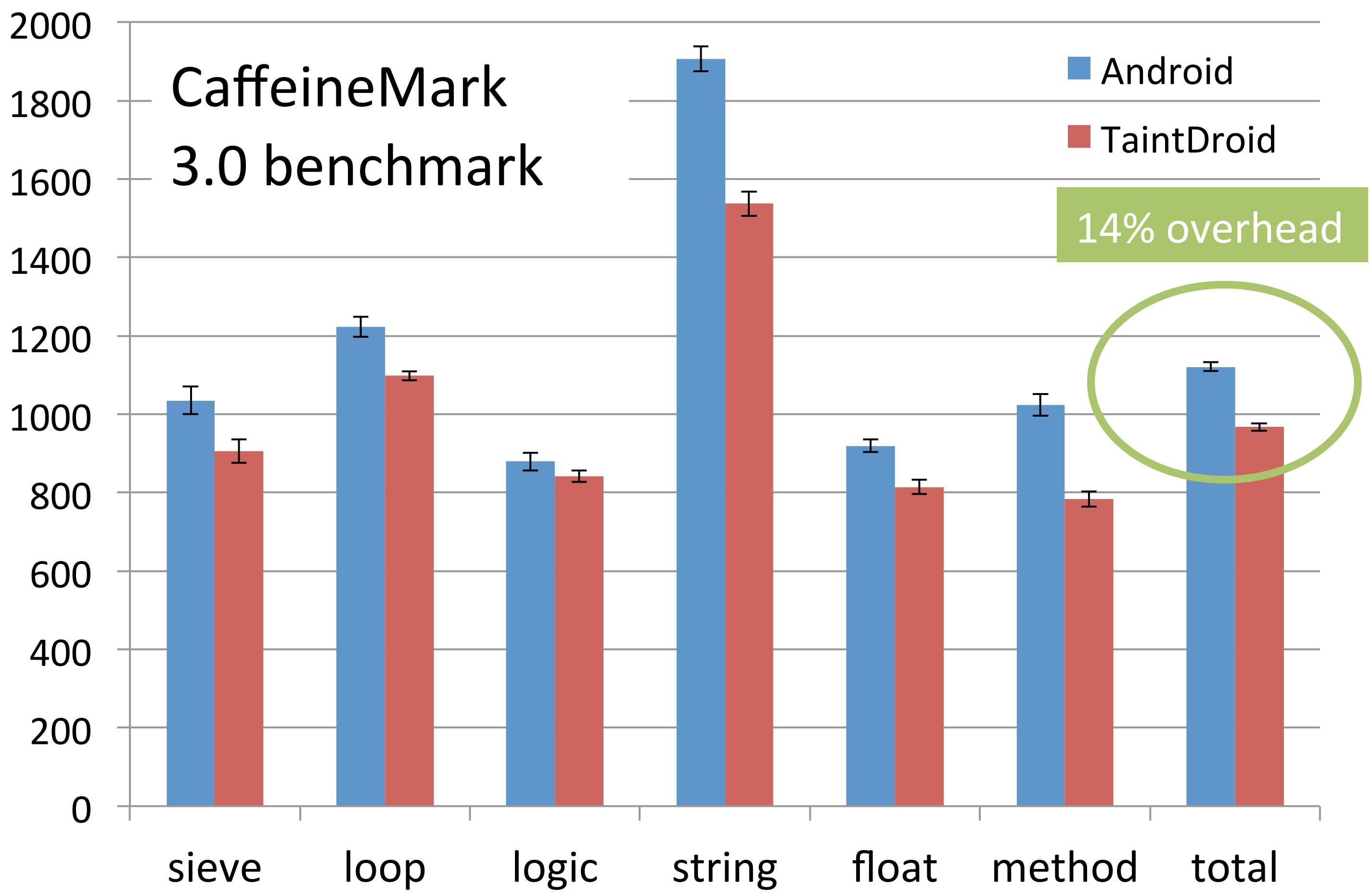
IPC and File Taint Propagation

- Message-level tracking for IPC
  - Marshall data items
  - Unmarshall data items
- Persistent storage tracked at the file level
  - Single taint tag stored in the file system XATTR

Roadmap

- Approach
- TaintDroid design
- Performance study
- Application study

Performance Study: Microbenchmark



Performance Study

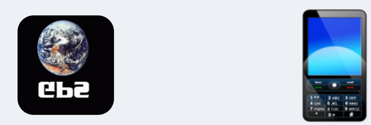
- Memory overhead: 4.4%
- IPC overhead: 27%
- Macro-benchmark
  - App load: 3% (2ms)
  - Address book: (<20ms) 5.5% create, 18% read
  - Phone call: 10% (10ms)
  - Take picture: 29% (0.5s)

Courtesy: Byung-Gon et. al

Taint Adaptors

- Sources
  - Low-bandwidth sensors: location, accelerometer
  - High-bandwidth sensors: microphone, camera
  - Information databases: address book, SMS storage
  - Device identifiers: IMEI, IMSI, ICC-ID, Phone number
- Sink: network

Application Study

Applications (with the Internet permission)	#	Permissions
The Weather Channel, Cetos, Solitarie, Movies, Babble, Manga Browser	6	
Bump, Wertago, Antivirus, ABC --- Animals, Traffic Jam, Hearts, Blackjack, Horoscope, 3001 Wisdom Quotes Lite, Yellow Pages, Datelefonbuch, Astrid, BBC News Live Stream, Ringtones	14	
Layar, Knocking, Coupons, Trapster, Spongebot Slide, ProBasketBall	6	
MySpace, ixMAT, Barcode Scanner	3	
Evernote	1	

Findings: Location

- 15 of the 30 apps shared physical location with at least an ad server (admob.com, ad.qwapi.com, ads.mobclix.com, data.flurry.com)

```
e.g., received data with tag 0x411 data=
  [GET /servernameA1?
  hello=1&time=1&bumpid=354957030504982&
  locale=en_US&gpslong=-122.316&gpslat=4
  7.662&gpsaccuracy=32.000&timezone=0…
```

- In no case was sharing obvious to user or in EULA
  - In some cases, periodic and occurred without app use

Findings: Phone Identifiers

- 7 apps sent IMEI and 2 apps sent phone #, IMSI, ICC-ID to remote servers without informing the user
- Frequency was app-specific, e.g., one app sent phone information every time the phone booted

Courtesy: Byung-Gon et. al

Demo

- http://appanalysis.org/demo/index.html

Conclusion

- Efficient, system-wide, dynamic taint tracking for mobile platforms.
  - 14% overhead for computing-intensive work
- Private data leak is prevalent
  - 20 of the 30 studied applications share information in a way that was not expected

www.appanalysis.org

Soundcomber

A Stealthy and Context-Aware Sound Trojan for Smartphones

- Roman Schlegel
- City University of Hong Kong

- Kehuan Zhang, Xiaoyong Zhou, Mehool Intwala,
- Apu Kapadia, XiaoFeng Wang
- Indiana University Bloomington

The smartphone in your pocket is really a computer

- 1 GHz Processor
- 512MB / 16GB
- Android OS (Linux)

Courtesy: Roman et. al

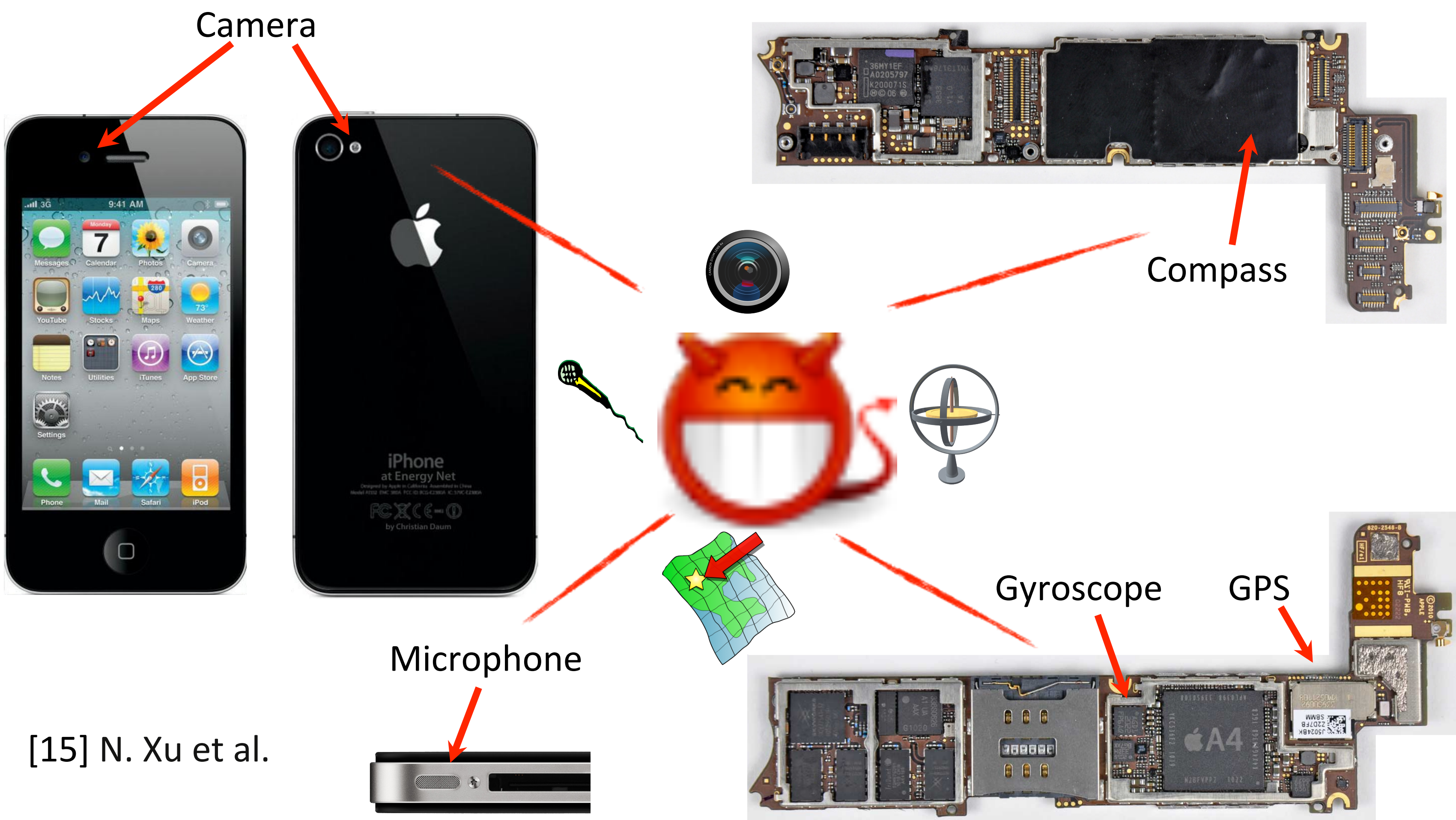
No surprise malware targets smartphones

- Android malware steals info from 1'000'000 users[1]
- Trojan sends premium-rate text messages[2]
- Security experts release Android root-kit[3]

1. http://nakedsecurity.sophos.com/2010/07/29/android-malware-steals-info-million-phone-owners/
2. http://news.cnet.com/8301-27080_3-20013222-245.html
3. http://www.reuters.com/article/idUSTRE66T52O20100730

Courtesy: Roman et. al

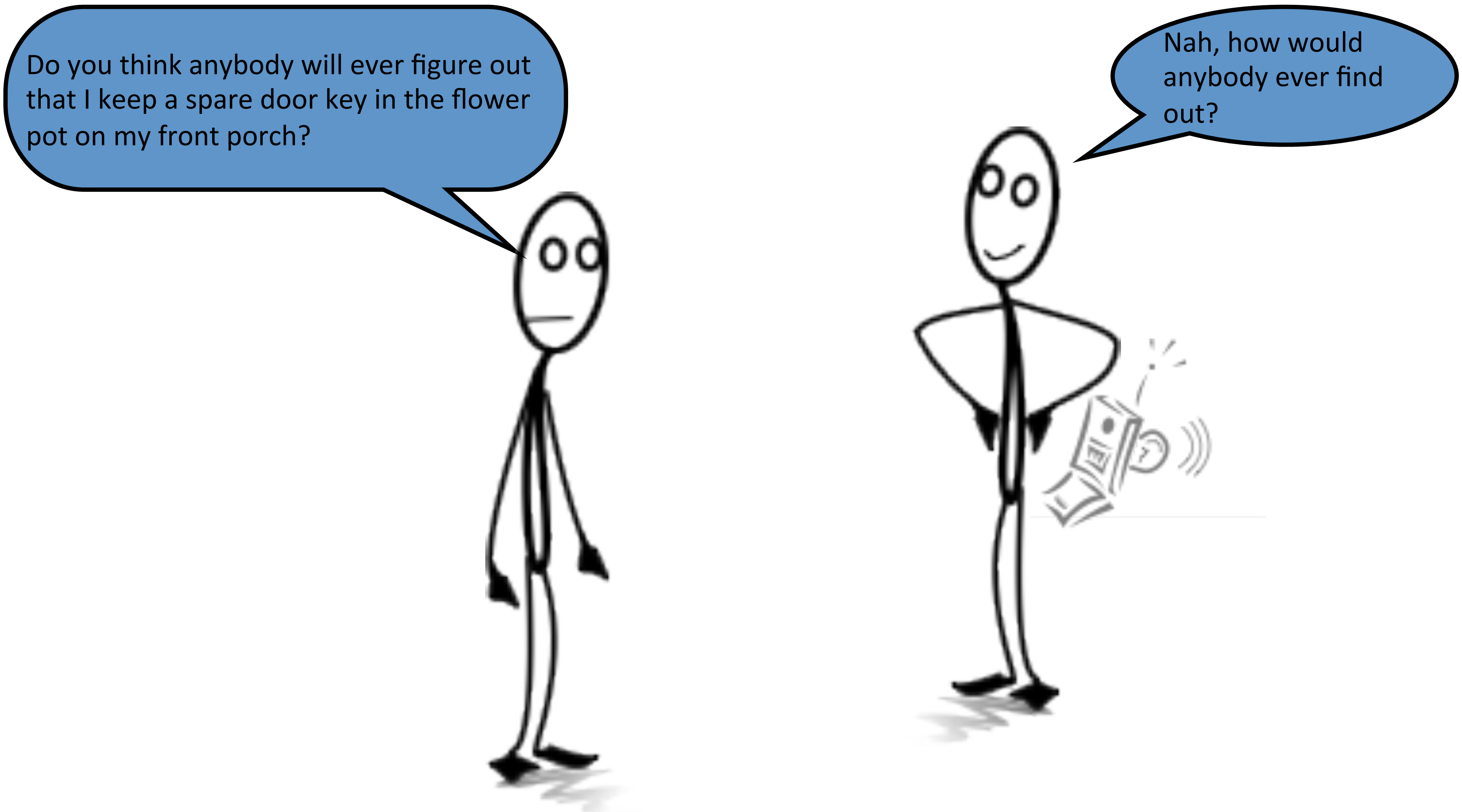
But "sensory malware" can do much more



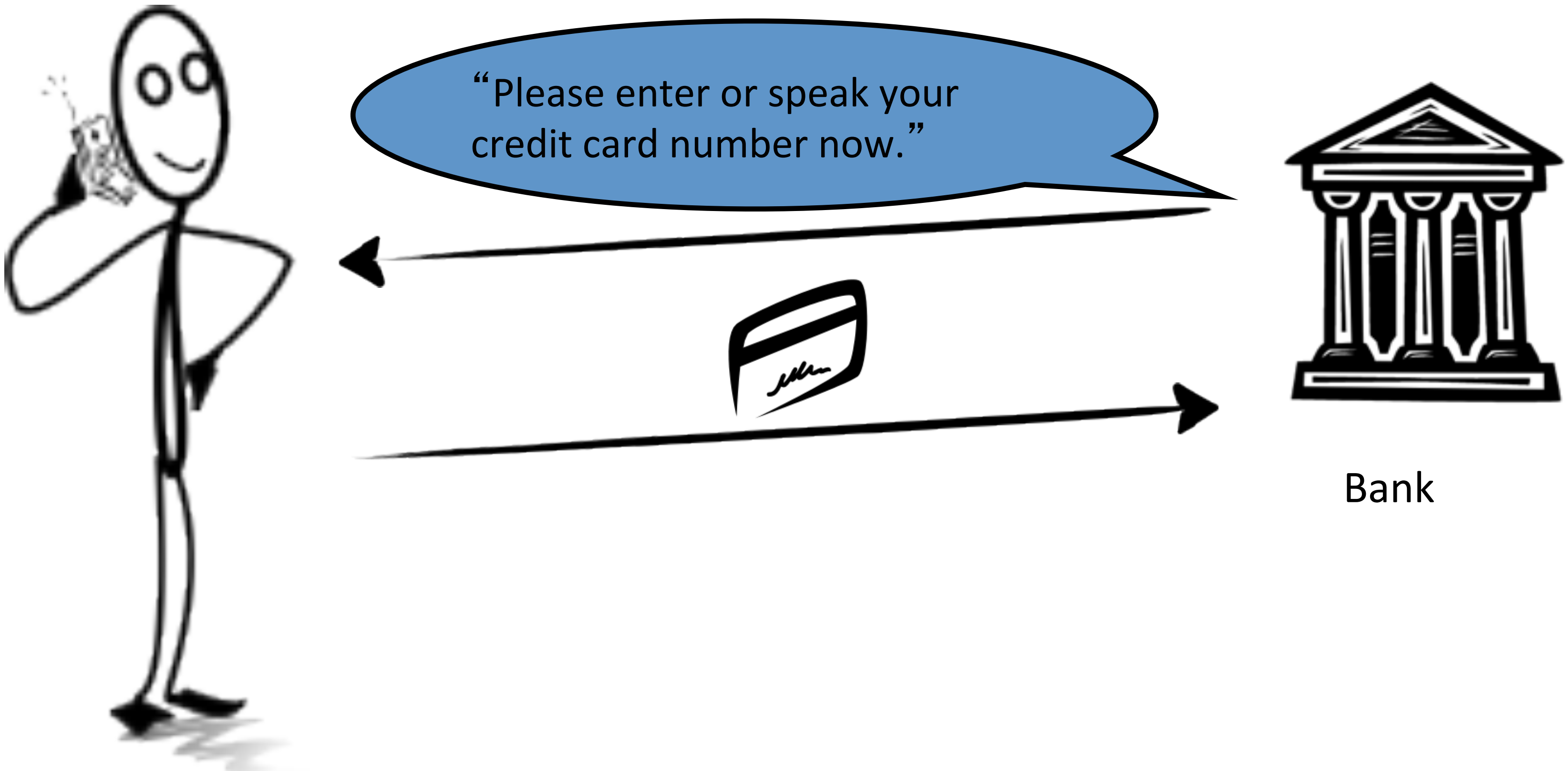
[15] N. Xu et al.

Courtesy: Roman et. al

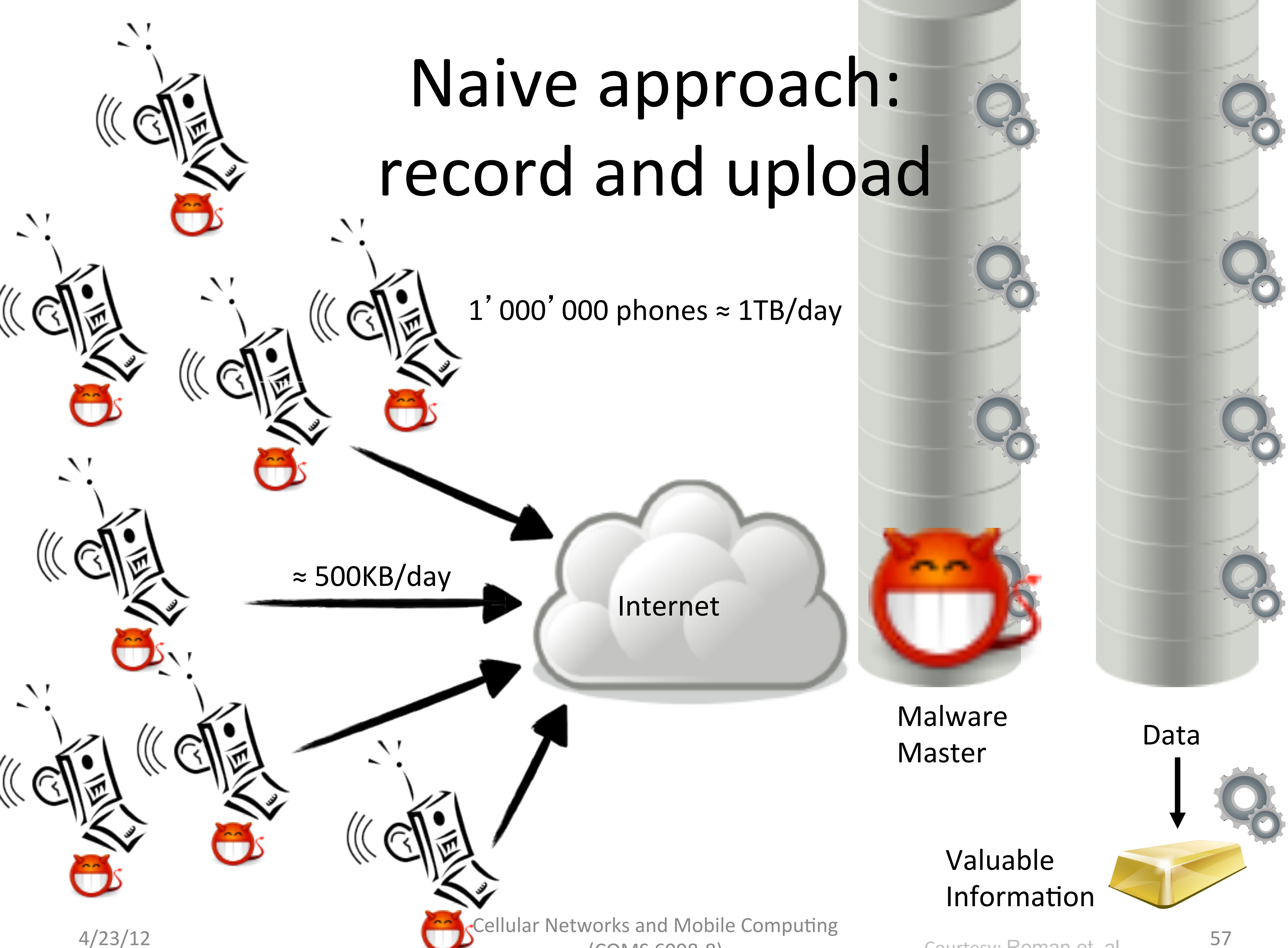
What can malware overhear?



Some situations are easy to recognize

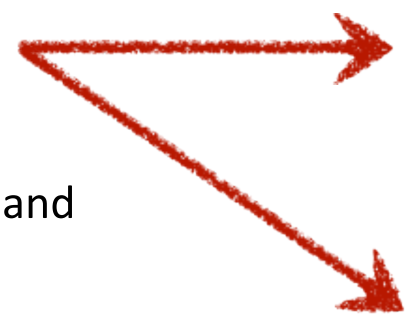


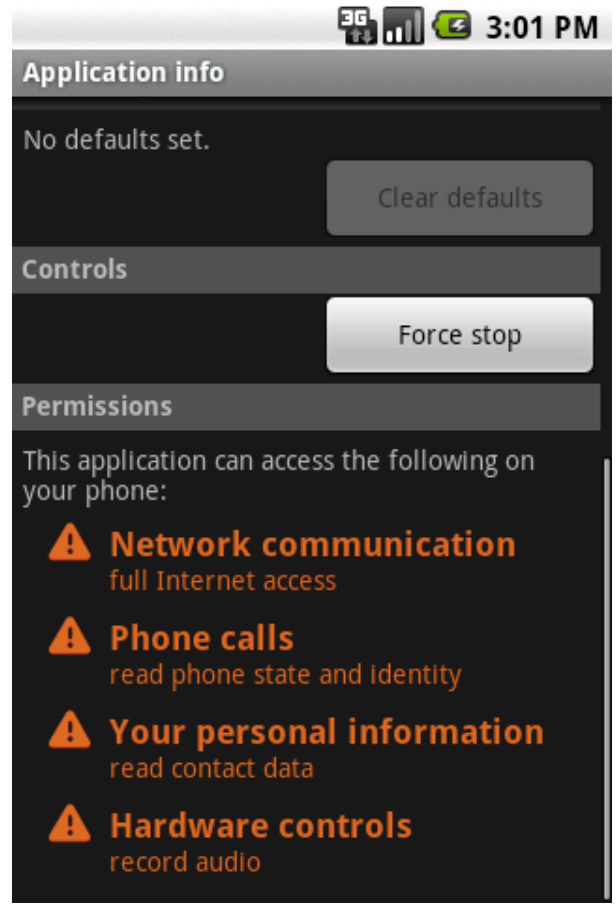
Naive approach: record and upload



Courtesy: Roman et. al

Certain combinations of permissions are suspicious



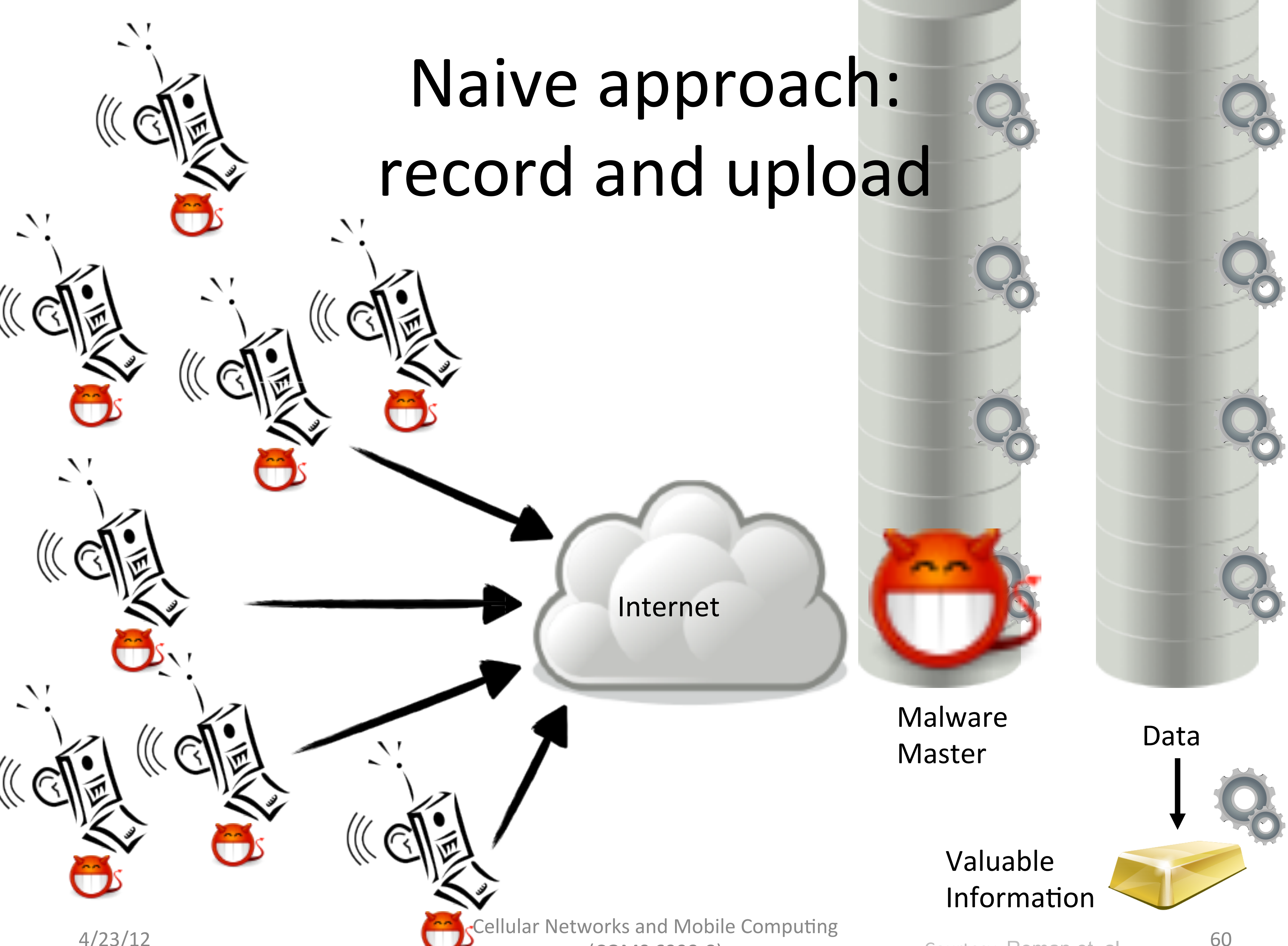


Can easily be recognized and disallowed
([5] W. Enck et al.)

Our contributions over the naive approach

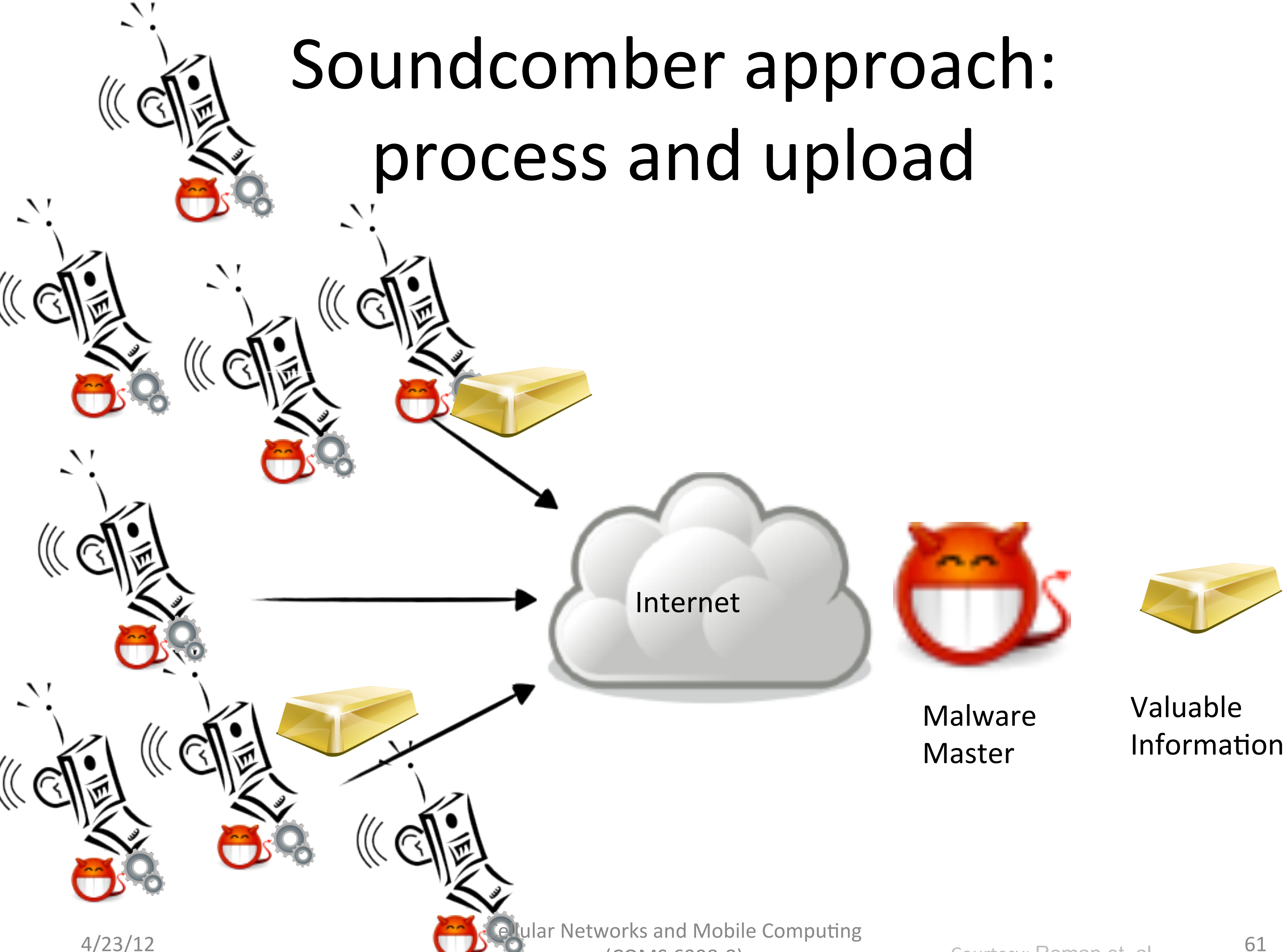
- – *targeted* and *local* extraction of valuable data
- inconspicuous permissions
- stealthiness

Naive approach: record and upload



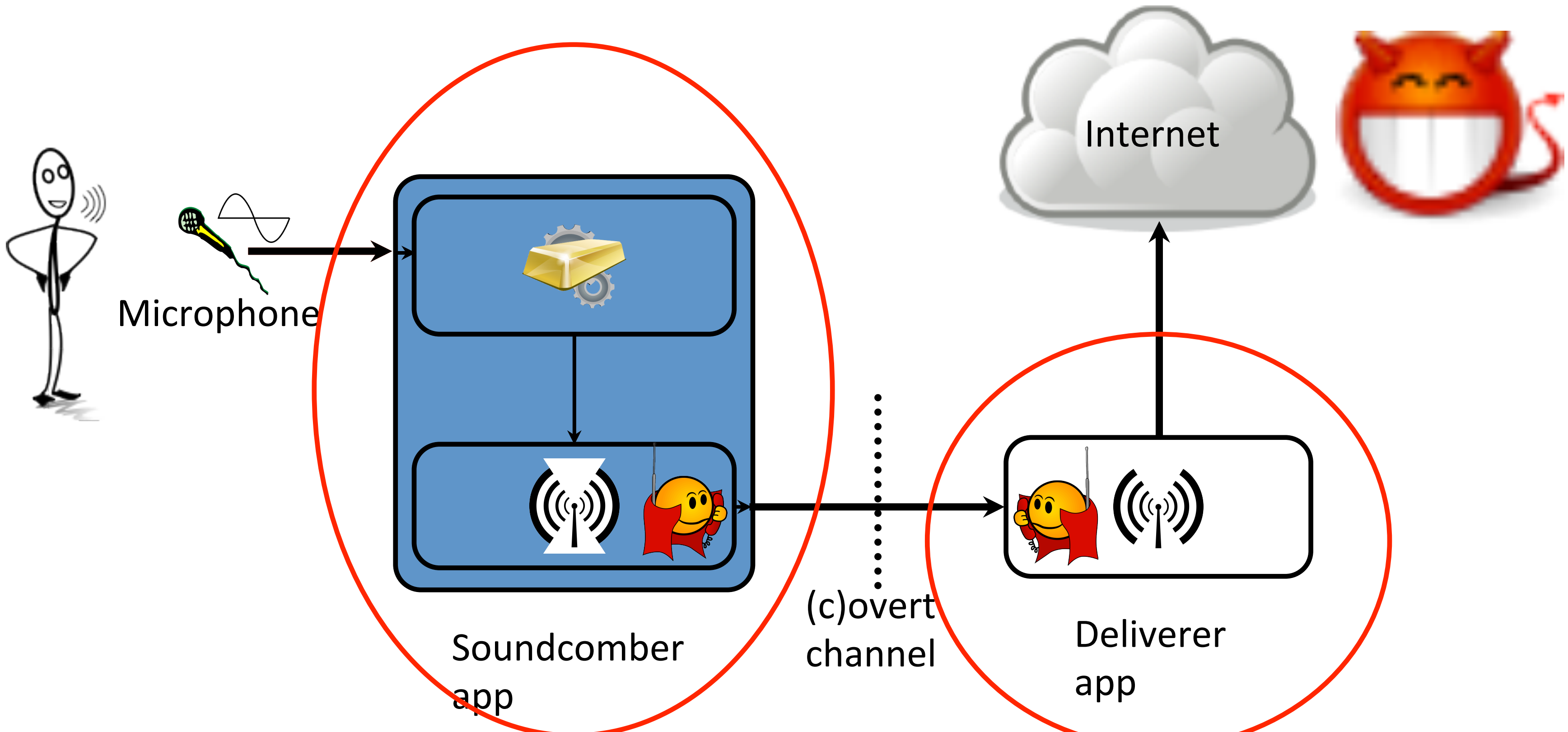
Courtesy: Roman et. al

Soundcomber approach: process and upload

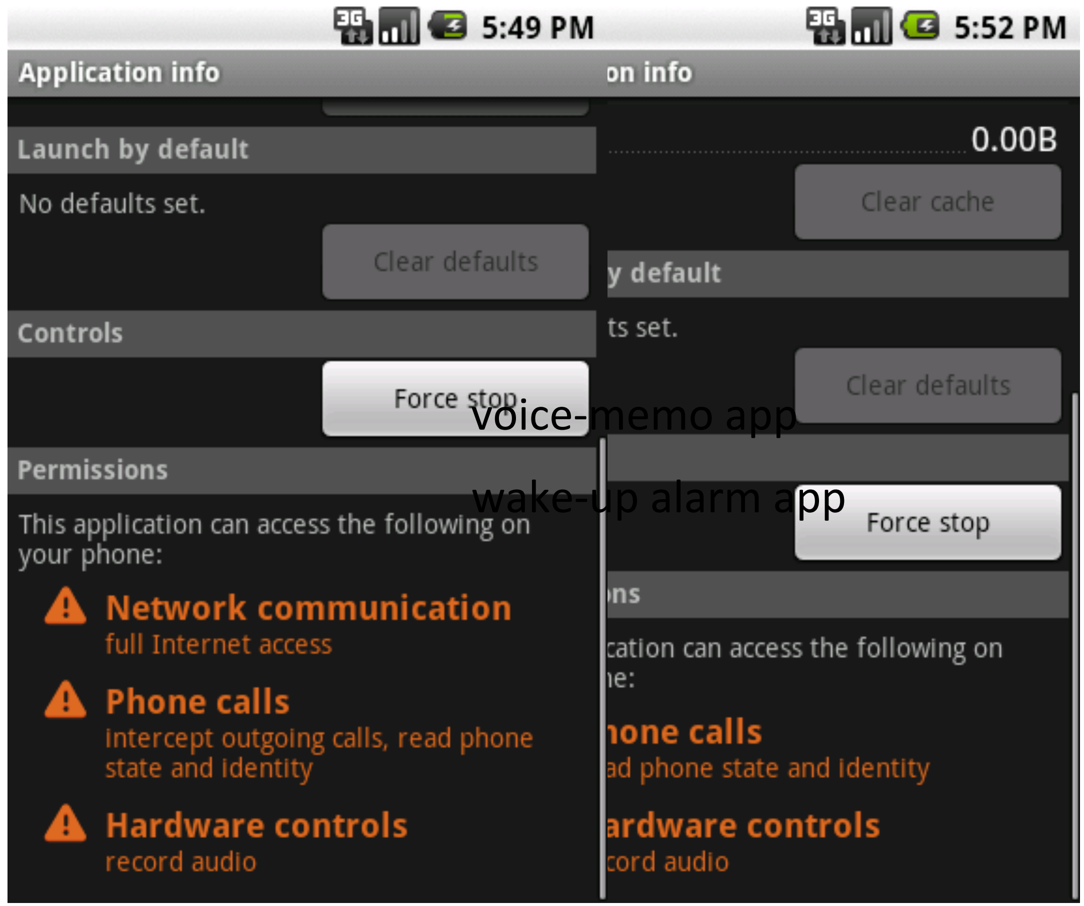


Courtesy: Roman et. al

Two trojans are stealthier than one



Soundcomber minimizes the necessary permissions

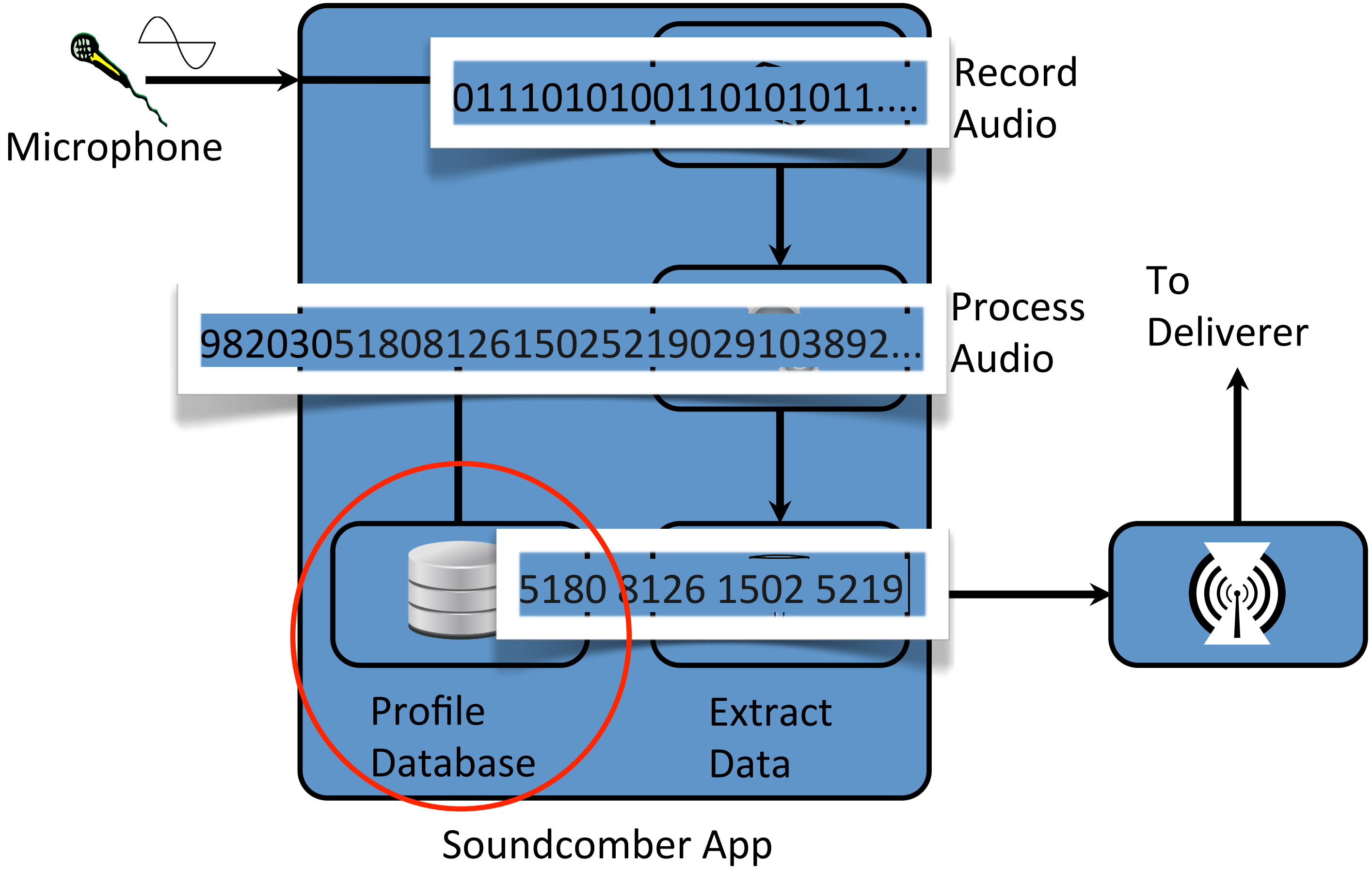


Hotline greetings can be fingerprinted easily

Tricking the user into installing two apps

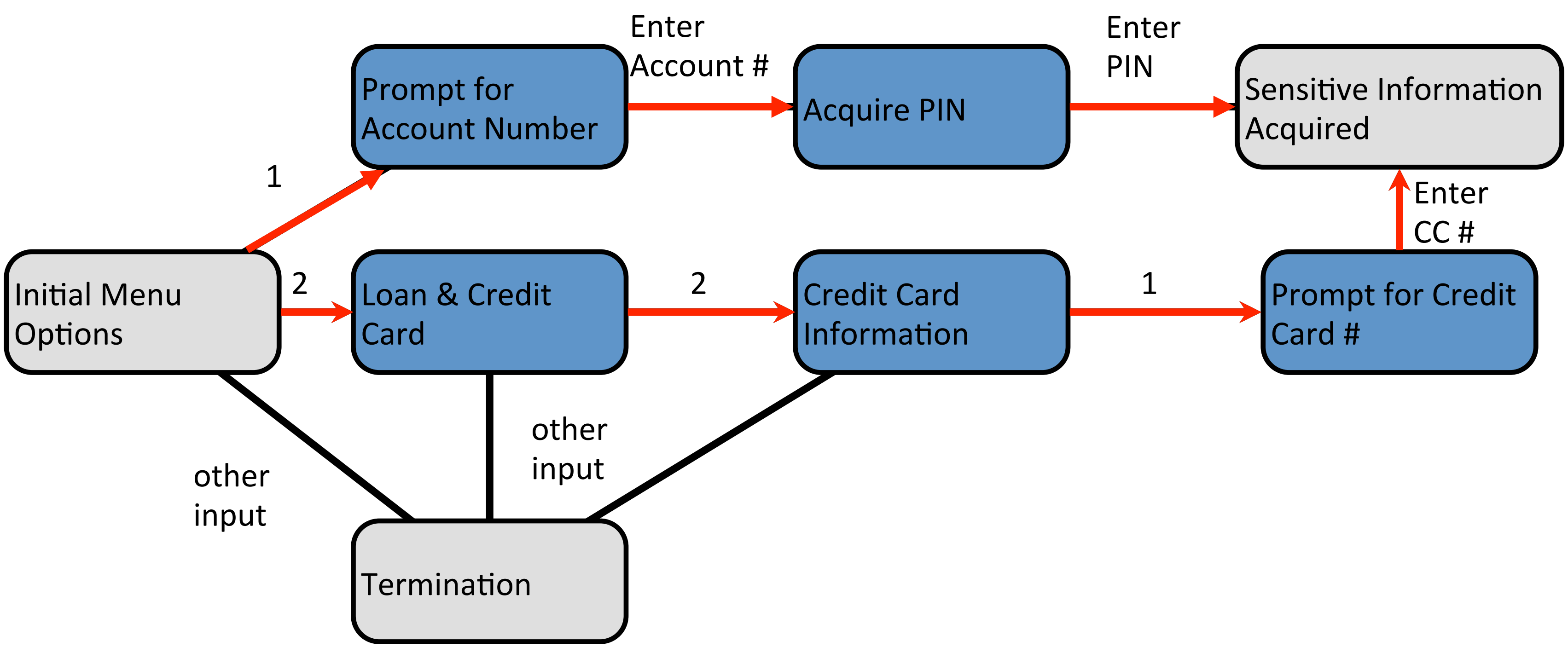
- pop-up ad
- packaged app

Soundcomber extracts sensitive information locally



Courtesy: Roman et. al

Profiles allow for context aware extraction



Courtesy: Roman et. al

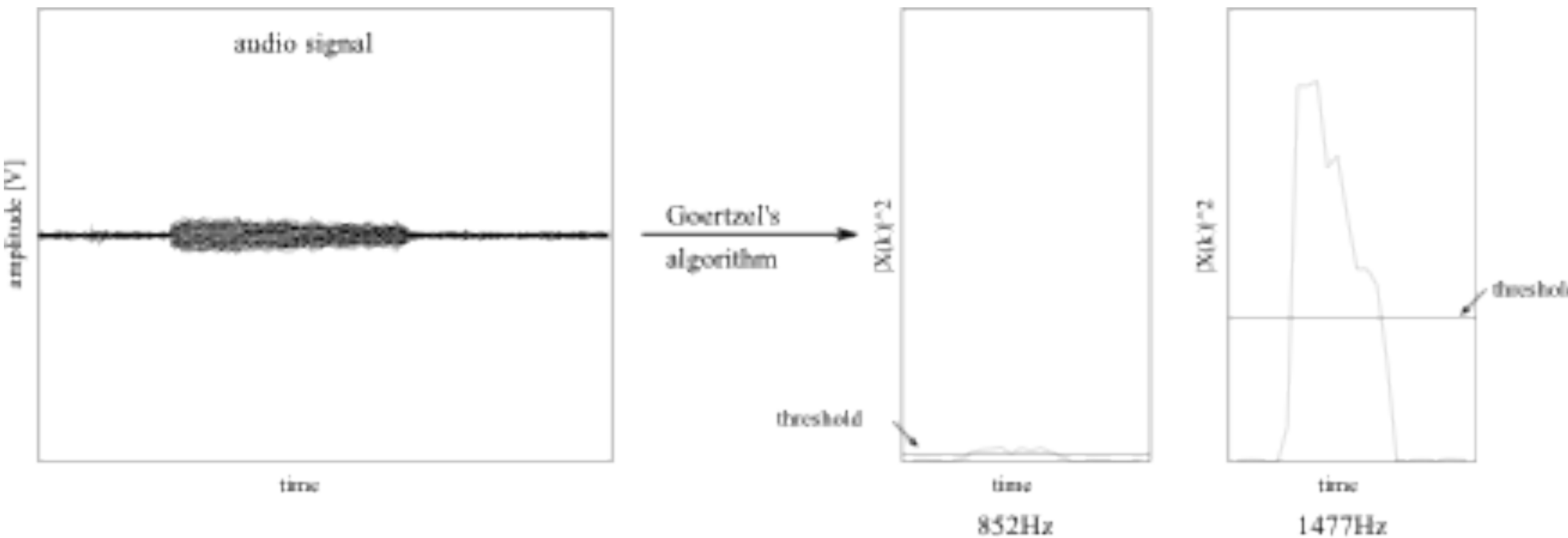
DTMF tones are "dual tones"

- 8 frequencies
- 2 simultaneous frequencies for each digit
- used to navigate hotline menus

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

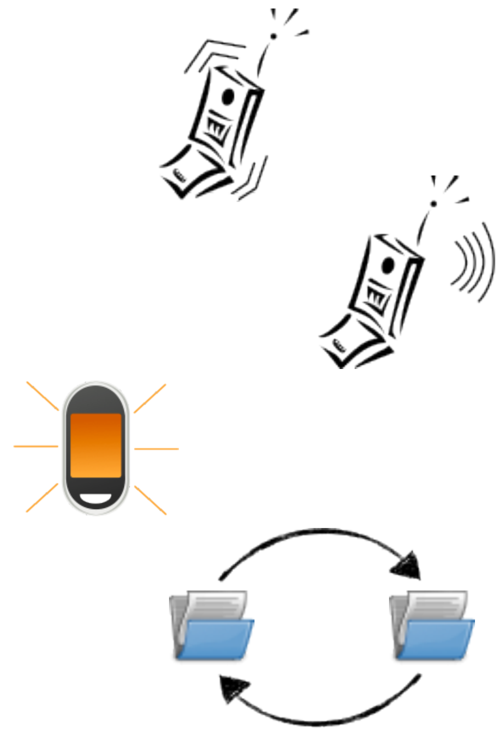
Courtesy: Roman et. al

Soundcomber dynamically adjusts thresholds to detect faint tones



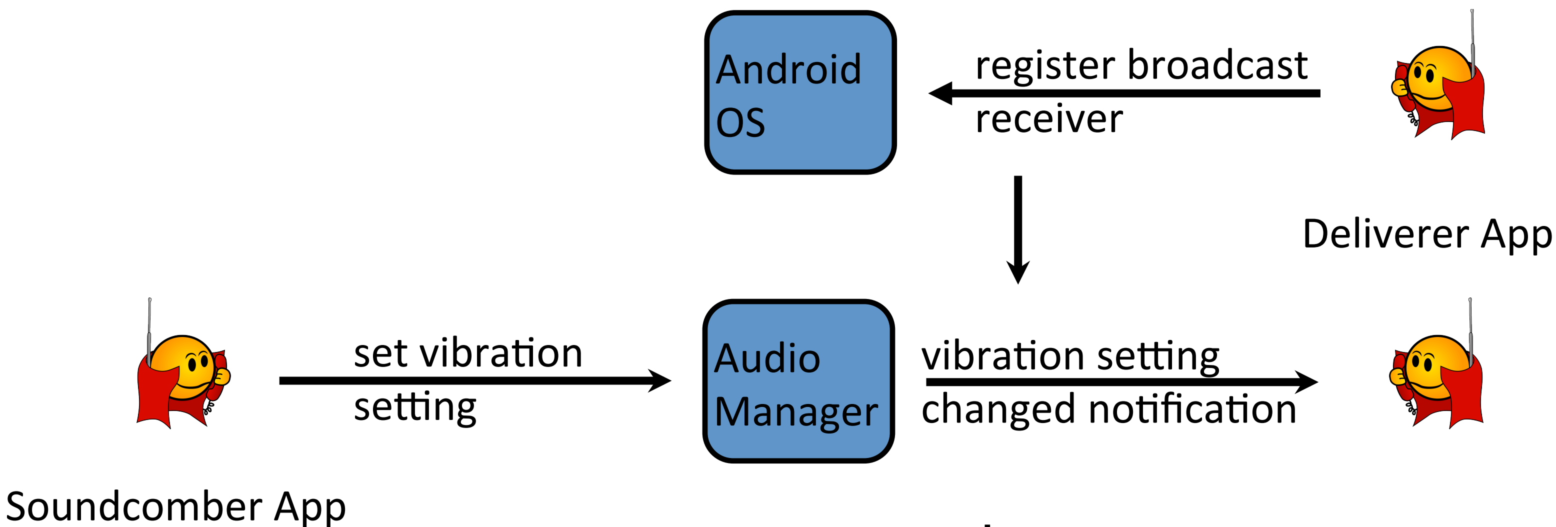
Android introduces new covert channels

- vibration settings (87 bps)
- volume settings (150 bps)
- screen (5.3 bps)
- file locks (685 bps)



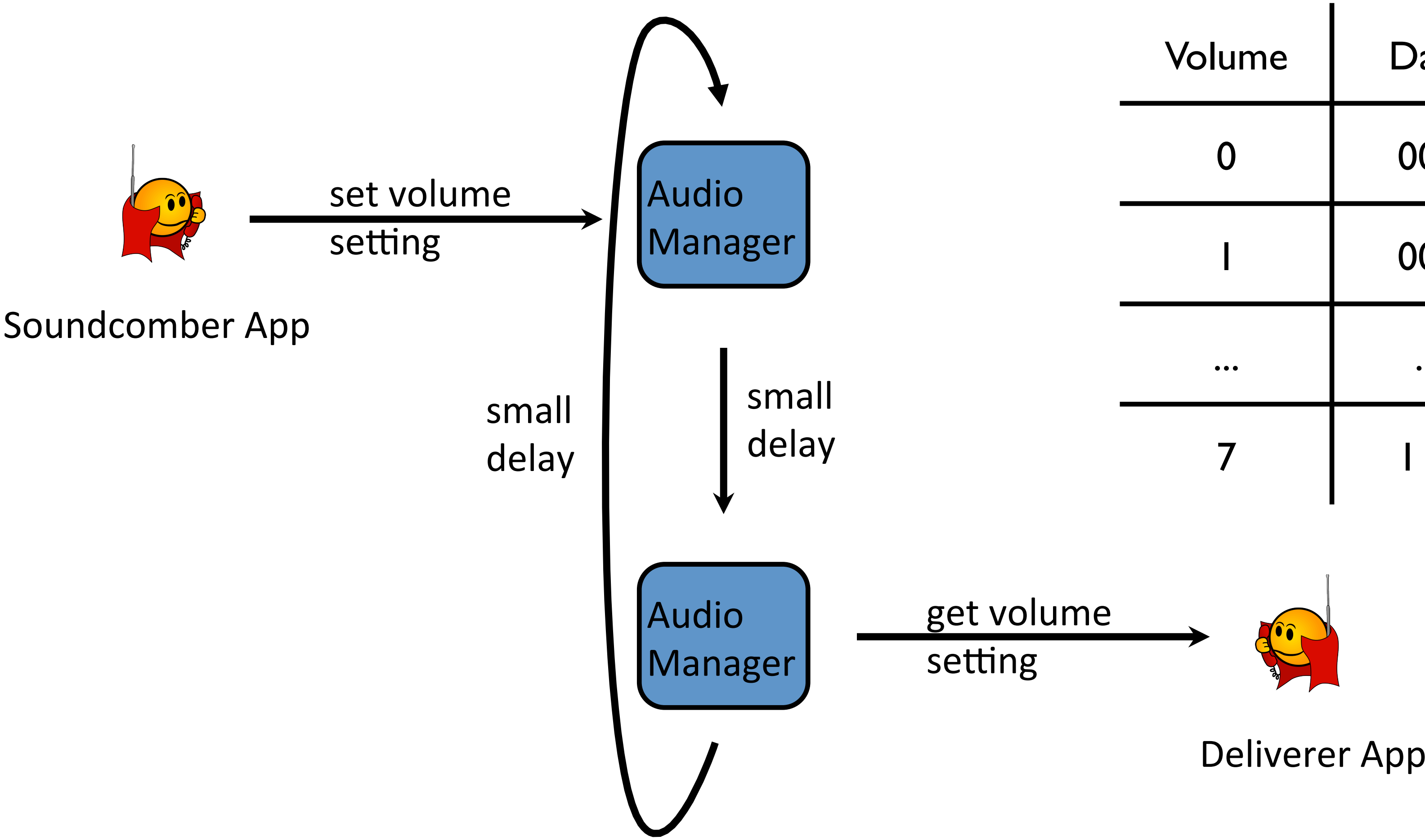
Courtesy: Roman et. al

Vibration settings are broadcast to interested apps



Setting	Bit
VIBRATE_SETTING_ON	0
VIBRATE_SETTING_ ONLY_SILENT	1

Volume settings can be modified and accessed by any app



Volume	Data
0	000
1	001
…	…
7	111

Soundcomber is fast and accurate

	No Error	1 Error	>= 2 Errors	1 missing	>= 2 missing
Speech	55 %	12.5 %	15 %	7.5 %	10 %
Tone	85 %	5 %	0	10 %	0

	Recording Length	Processing Time
Speech	20 s	7 s
Tone	45 s	8 s

Hotlines can be fingerprinted with reasonable accuracy

- 20 recorded samples of 5 different hotlines (4 each)
- 20 samples of normal conversation

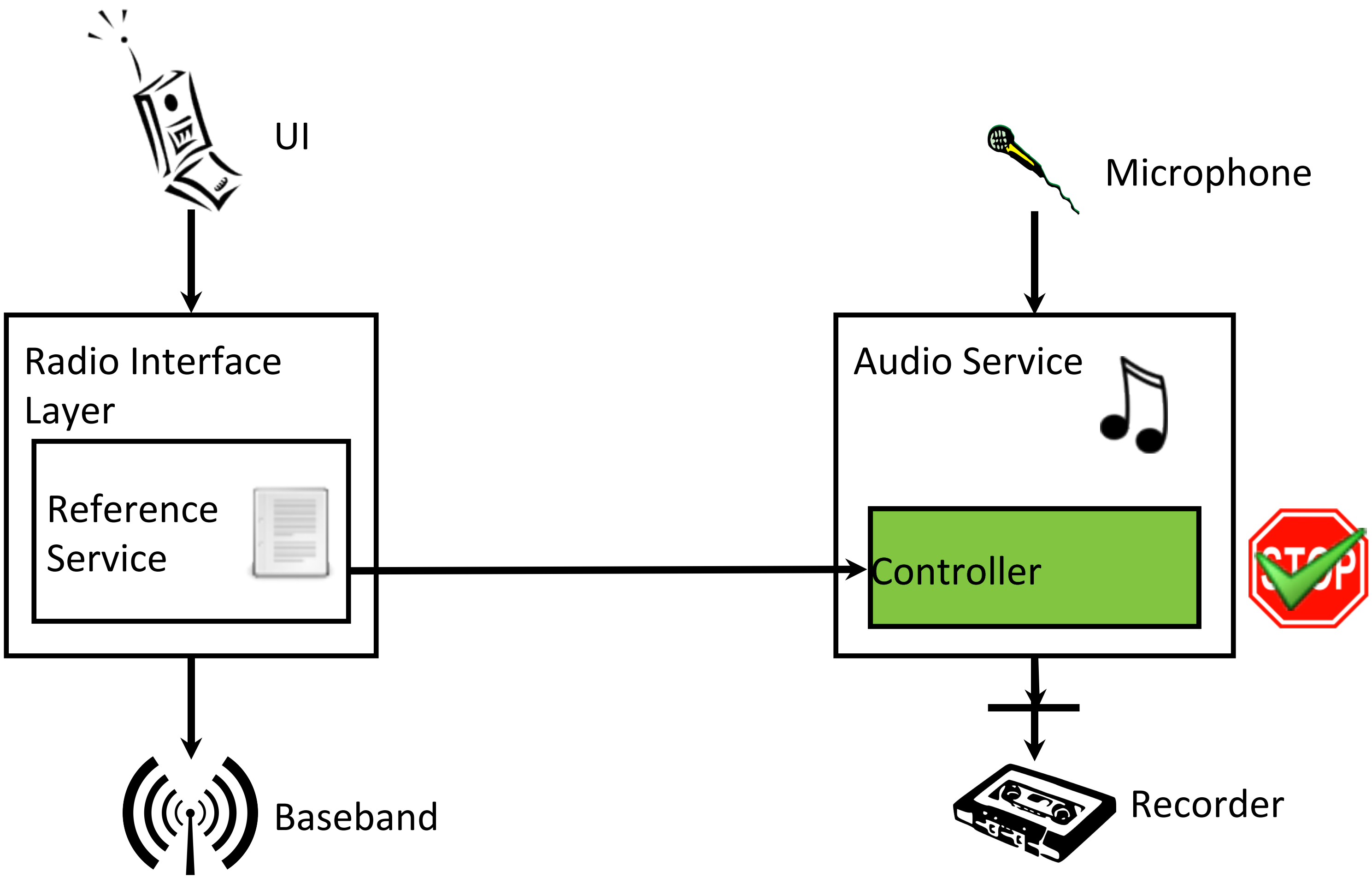
	Correct	Missed	Wrong
Hotline	55 %	40 %	5 %

	Correct
Conversation	100 %

Keeping Soundcomber hidden and undetectable

- defer/throttle processing
- track user presence
- performance enhancements

Defense: disable recording when a sensitive number is called



Courtesy: Roman et. al

Demo

- Demo video
  - http://www.youtube.com/watch?v=Z8ASb-tQVpU

Conclusion

- stealthy, sensory malware is a real threat
- need to explore other such threats
- develop generalized defenses to such attacks

Keypad: Auditing Encrypted File system for Theft-prone Devices

Roxana Geambasu
John P. John
Steve Gribble
Yoshi Kohno
Hank Levy

University of Washington

Slides and Video Presentation

- Slides
  - http://www.cs.columbia.edu/~roxana/research/projects/keypad/eurosys2011keypad_talk.ppt
- Video presentation
  - http://www1.cs.columbia.edu/streaming/common/player.php?file=/streaming/2011-Spr/geambasu/geambasu.flv