Brief Tour about Android Security

Jim Huang (黃敬群) <**jserv**@0xlab.org>

Developer, 0xlab

Oct 4, 2012

Rights to copy

http://0xlab.org/

contact@0xlab.org

Corrections, suggestions, contributions and translations are welcome!

Latest update: Oct 4, 2012

creative commons

COMMONS DEED

Attribution – ShareAlike 3.0

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

BY: **Attribution**. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/3.0/legalcode

About Myself

- Contributor, Android Open Source Project
  50+ contributions in AOSP
- Consultant, MediaTek, Inc.
- Maintainer of Chewing (新酷音) Input Method
- Background:
  - Consumer electronics (smatphone, digital TV, feature phone, GPS navigator, medical devices), RTOS/Microkernel designs, Compiler optimizations, embedded systems

Agenda

(1) Security in Action

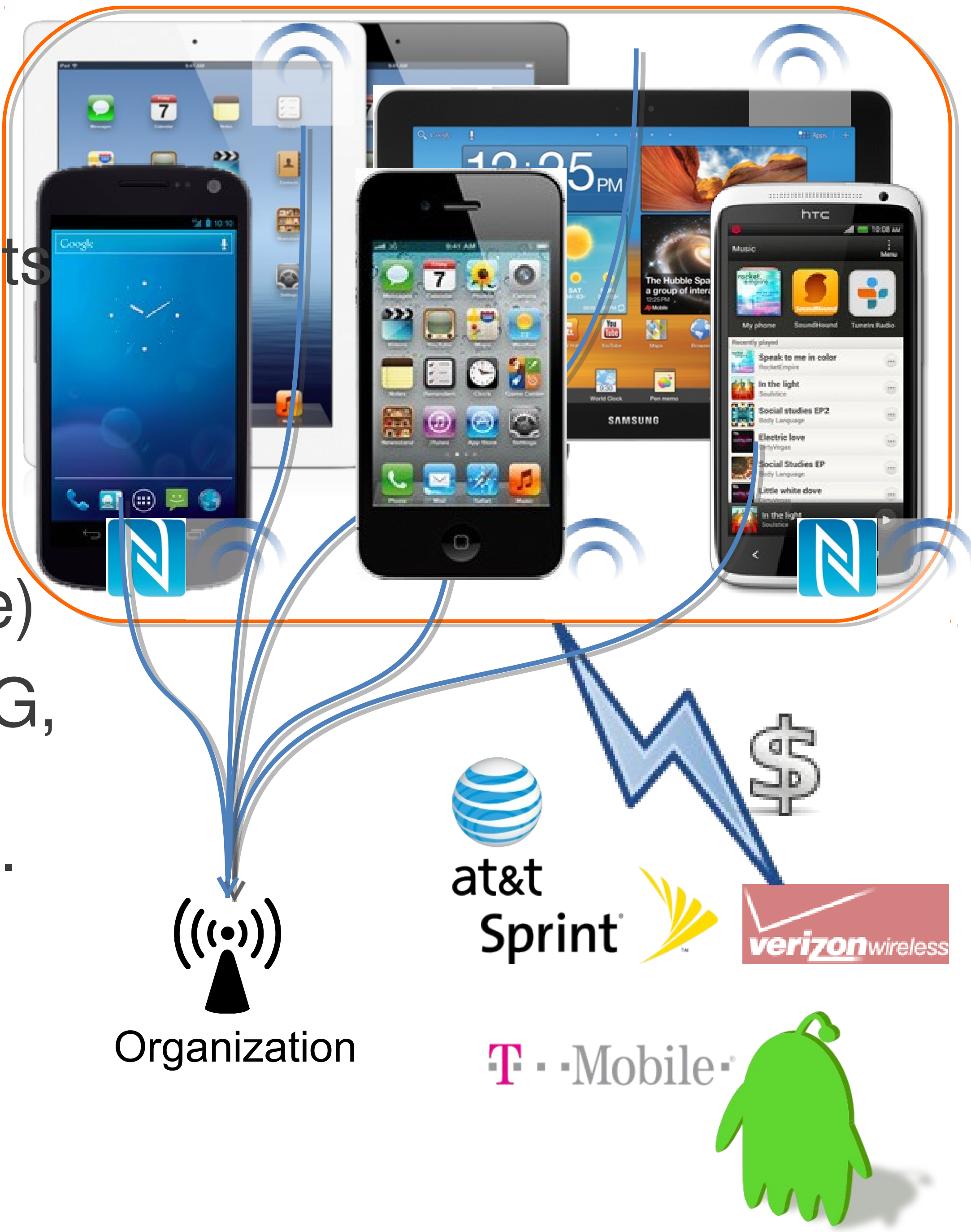
(2) Android Security Architecture

(3) Protection & Prevention

Security in Action

Mobile Devices

- Mobile *computers*:
  - Mainly smartphones, tablets
  - Sensors: GPS, camera, accelerometer, etc.
  - Computation: powerful CPUs (≥ 1 GHz, multi-core)
  - Communication: cellular/4G, Wi-Fi, near field communication (NFC), etc.
- Many connect to cellular networks: *billing system*

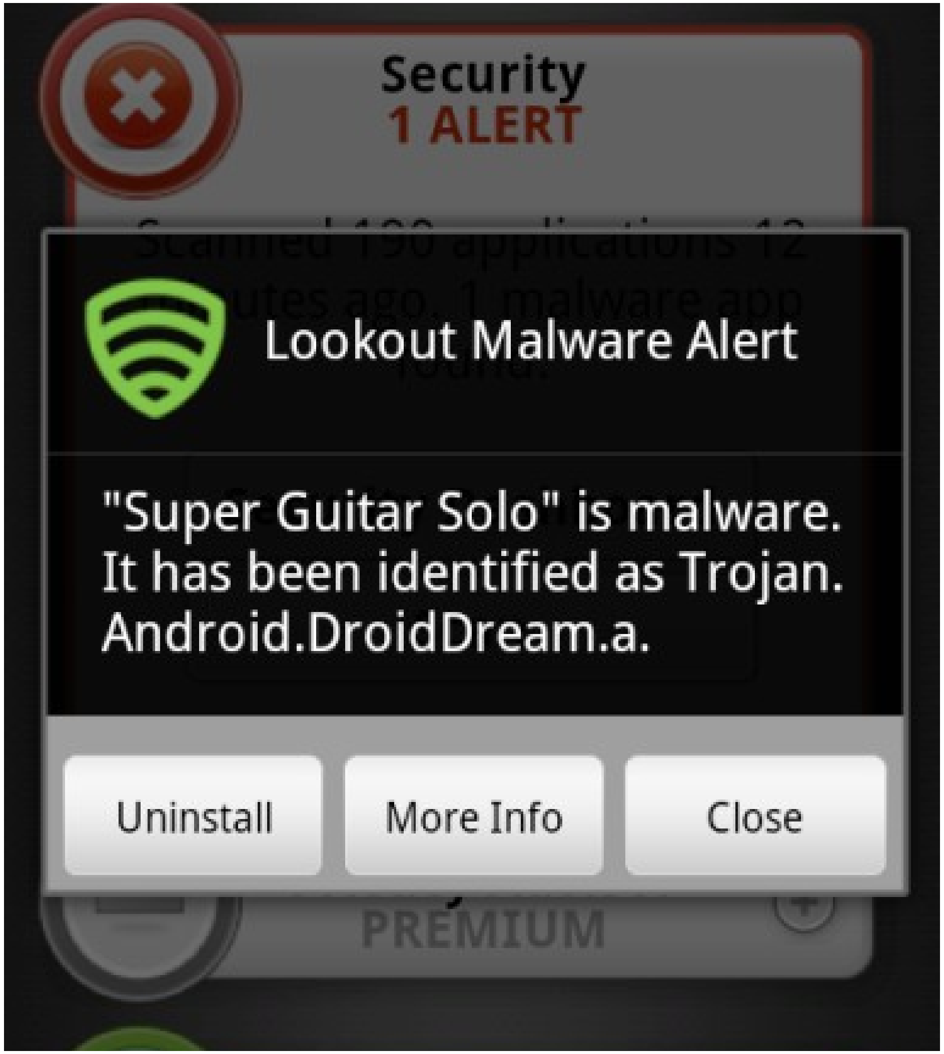


Mobile Threats and Attacks

- Mobile devices make attractive targets:
  - People store much personal info on them: email, calendars, contacts, pictures, etc.
  - Sensitive organizational info too…
  - Can fit in pockets, easily lost/stolen
  - Built-in billing system: SMS/MMS (mobile operator), in-app purchases (credit card), etc.
    - Many new devices have near field communications (NFC), used for contactless payments, etc.
    - Your device becomes your credit card
- Much Android malware, much less for iOS
- NFC-based billing system vulnerabilities

Android: DroidDream Malware

- Infected 58 apps on Android Market, March 2011
- 260,000 downloads in 4 days
- How it worked:
  - Rooted phone via Android Debug Bridge (adb) vulnerability
  - Sent premium-rate SMS messages at night ($$$)
- Google removed apps 4 days after release, banned 3 developers from Market
- More malware found since

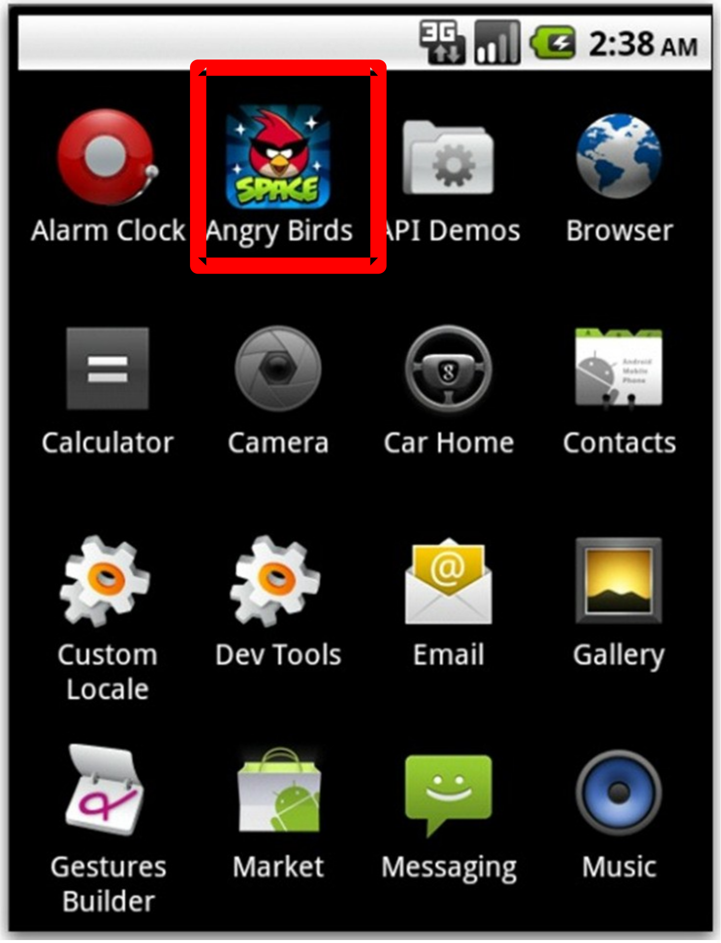


Lookout Malware Alert

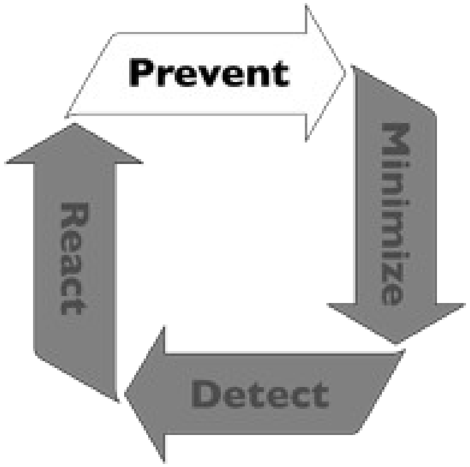
"Super Guitar Solo" is malware. It has been identified as Trojan. Android.DroidDream.a.

Android: Fake Angry Birds Space

- Bot, Trojan
- Masquerades as game
- Roots Android 2.3 devices using “Gingerbreak” exploit
- Device joins botnet



Security Philosophy



- Finite time and resources
- Humans are hard to understand risk
- Safer to assume that
  - Most developers do not understand security
  - Most users do not understand security
- Security philosophy cornerstones need to...
  - prevent security breaches from occurring
  - minimize the impact of a security breach
  - detect vulnerabilities and security breaches
  - react to vulnerabilities and security breaches swiftly

Prevent

- 5 million new lines of code
- Uses almost 100 open source libraries
- Android is open source ⇒ can't rely on obscurity
- Concentrated on high risk areas
  - Remote attacks
  - Media codecs
  - New/custom security features
- Low-effort/high-benefit features
  - ProPolice stack overflow protection
  - Heap protection in dlmalloc

Minimize

- We cannot rely on prevention alone
  - Vulnerabilities happen
- Users will install malware
- Code will be buggy
- How can we minimize the impact of a security issue?
- My webmail cannot access my banking web app
  - Same origin policy
- Why can malware access my browser? my banking info?
- Extend the web security model to the OS

Detect

- A lesser-impact security issue is still a security issue
- Internal detection processes
  - Developer education
  - Code audits
  - Fuzzing
  - Honeypot
- Everyone wants security ⇒ allow everyone to detect issues
  - Users
  - Developers
  - Security Researchers

React

- Autoupdaters are the best security tool since Diffie-Hellman
- Every modern operating system should be responsible for:
  - Automatically updating itself
  - Providing a central update system for third-party applications
- Android's Over-The-Air update system (OTA)
  - User interaction is optional
  - No additional computer or cable is required
  - Very high update rate

Android Security Architecture

Android Platform Security Architecture

- Android re-purposes traditional operating system security controls to
  - Protect data
  - Protect system resources (including network)
  - Provide Application isolation
- Mandatory application sandbox
- Secure interprocess communication
- Application signing
- Application-defined and user-granted permissions

Linux Security

- Linux is used in millions of security-sensitive environments.
  - constantly being researched, attacked, and fixed by thousands of developers,
  - Linux has become trusted by many
- A user-ID-based permissions model
- Process isolation
- Extensible mechanism for secure IPC
- The ability to remove unnecessary and potentially insecure parts of the kernel

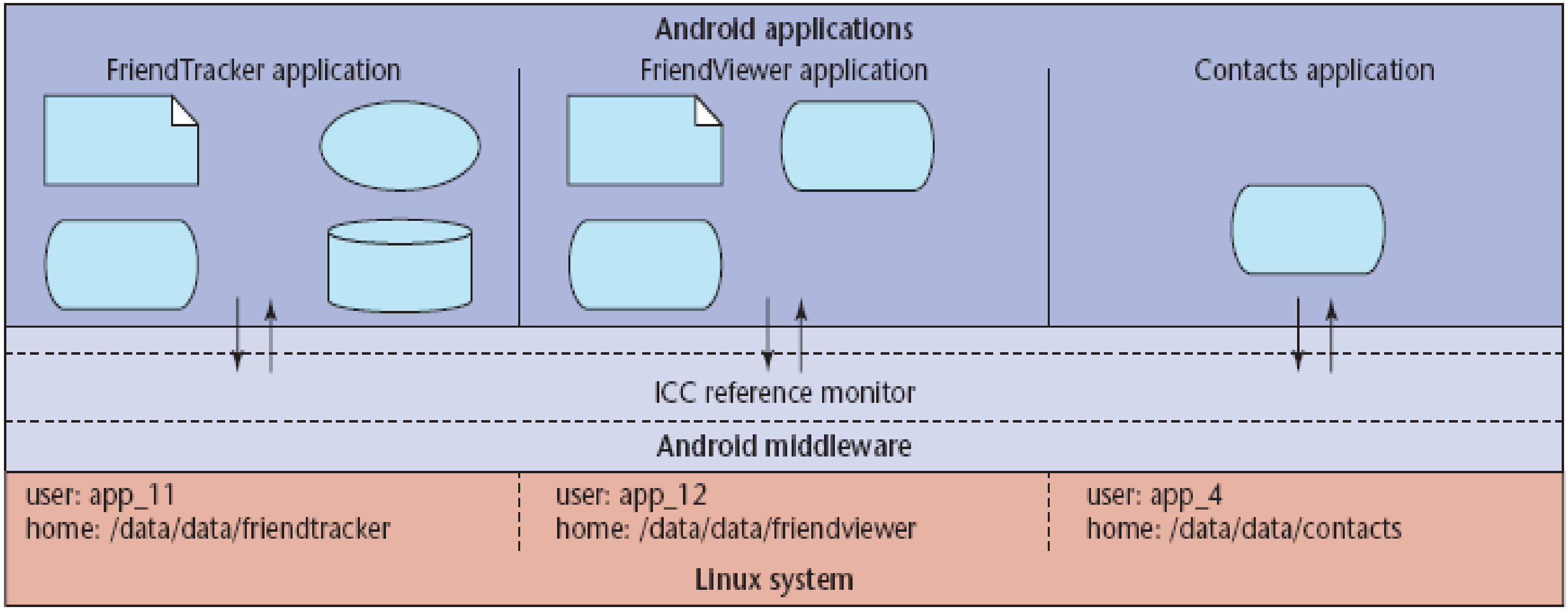
Android Security Bascis

- Applications, by default, have no permissions
- Permissions list: Manifest.permission
- Applications statically declare the permissions they require
  - Android system prompts the user for consent at the time the application is installed
  - no mechanism for granting permissions dynamically (at run-time)
  - in AndroidManifest.xml, add one or more <uses-permission> tags

```
<uses-permission android:name=
"android.permission.RECEIVE_SMS" />
```

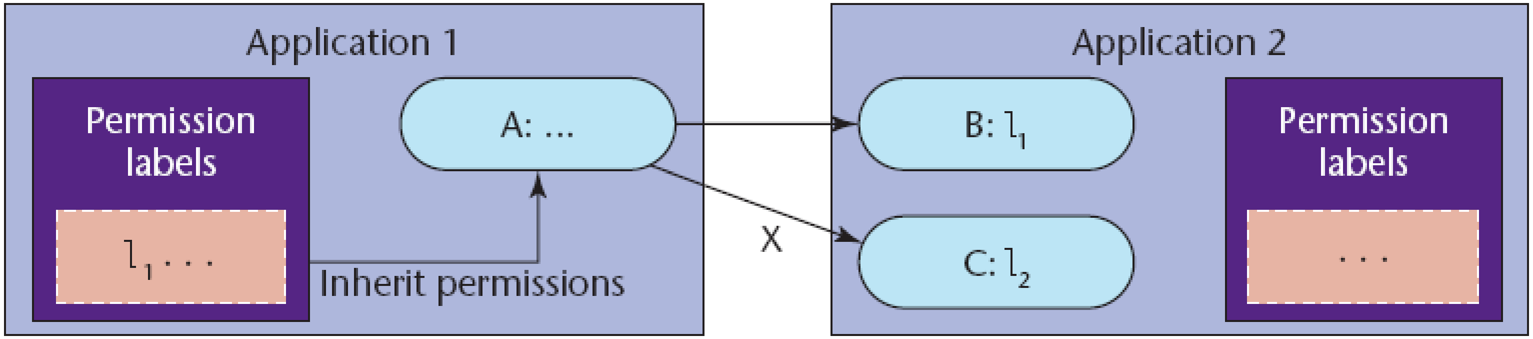
Security Enforcement

- Android protect application at system level and at the Inter-component communication (ICC) level. This article focus on the ICC level enforcement.
- Each application runs as a unique user identity, which lets Android limit the potential damage of programming flaws.



Security Enforcement

- Core idea : labels assignment to applications and components
- A reference monitor provides mandatory access control (MAC) enforcement of how applications access components.
- Access to each component is restricted by assigning it an access permission label; applications are assigned collections of permission labels.

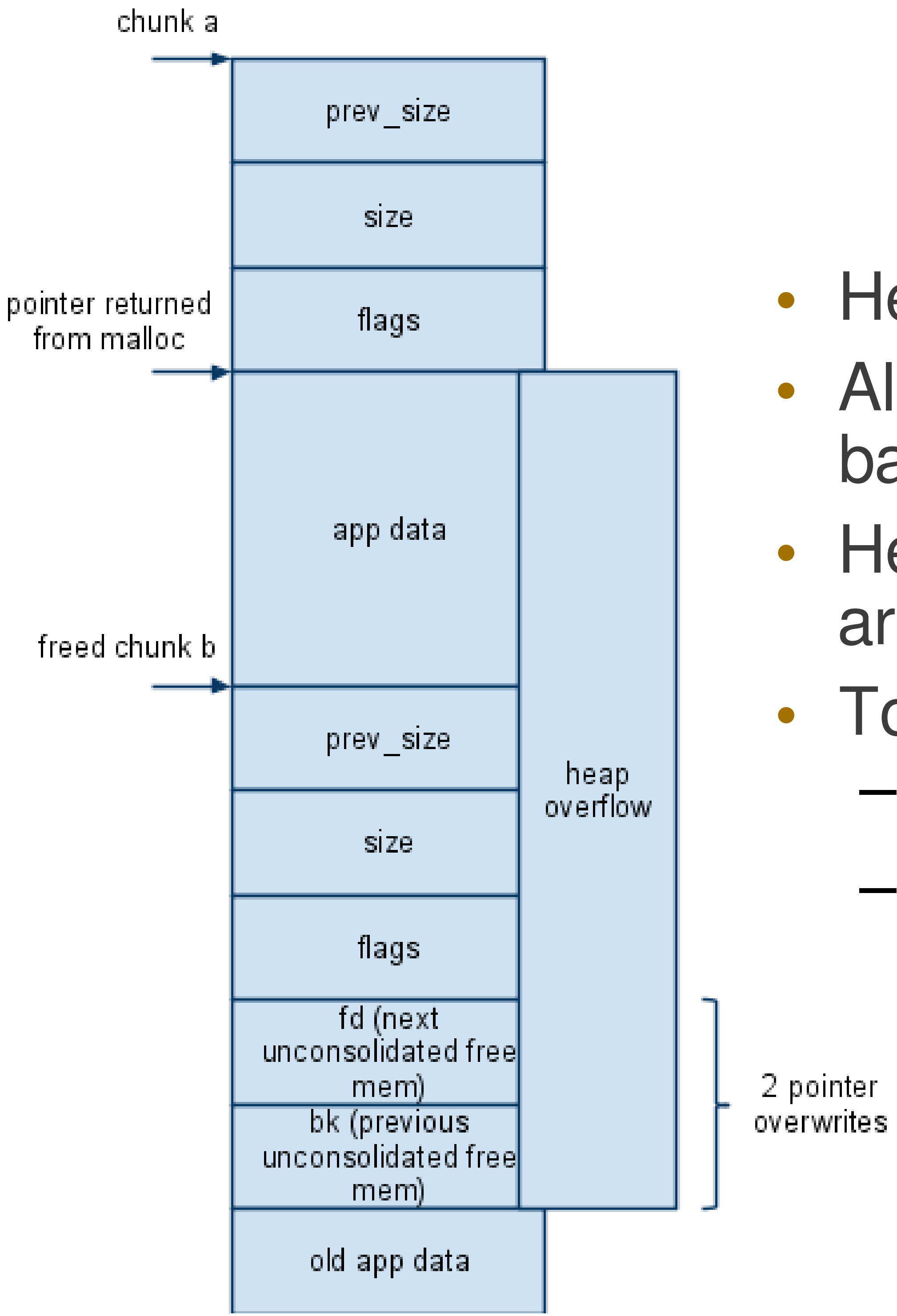


Android Security Extra

- Hardware-based No eXecute (NX) to prevent code execution on the stack and heap
- ProPolice canaries to prevent stack buffer overruns
- safe-iop safe integer op lib for C
- Extensions to dlmalloc to prevent double free() vulnerabilities and to prevent heap exploits
- OpenBSD calloc to prevent integer overflows during memory allocation
- Linux mmap_min_addr() to mitigate null pointer dereference privilege escalation

dlmalloc

(written by Doug Lea)



- Heap consolidation attack
- Allocation meta-data is stored in band
- Heap overflow can perform 2 arbitrary pointer overwrites
- To fix, check:
  - b->fd->bk == b
  - b->bk->fd == b

System Files

- The system partition
  - Android's kernel as well as the OS libraries, application runtime, application framework, and applications.
  - set to read-only
- When a user boots the device into Safe Mode
  - only core Android applications are available.
  - free of third-party software.

OS Protected APIs

- Cost-Sensitive APIs
  - Telephony
  - SMS/MMS
  - Network/Data connections
  - In-App Billing
  - NFC Access
- Sensitive Data Input Devices
  - Location data (GPS)
  - Camera functions
  - microphone
- Bluetooth functions
- Personal Information

IPC

- Standard IPC
  - file system, local sockets, or signals.
  - Linux permissions still apply.
- new IPC mechanisms:
- Binder: RPC mechanism for in-process and cross-process calls. Via a custom Linux driver.
- Services: interfaces directly accessible using binder.
- Intents: A message object that represents an "intention" to do something.
- ContentProviders: A data storehouse

Application Signing

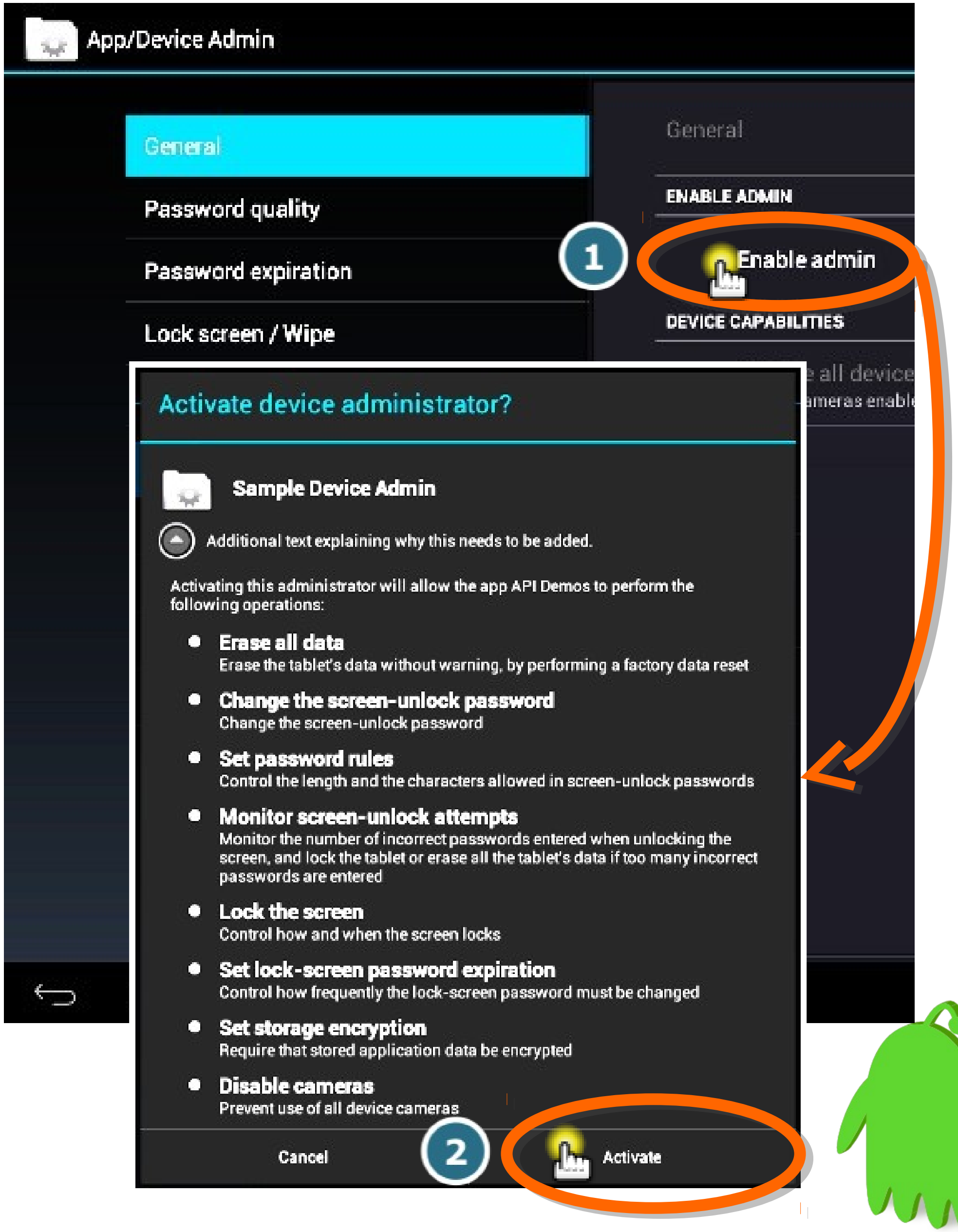
- Why self signing?
  - Market ties identity to developer account
  - CAs have had major problems with fidelity in the past
  - No applications are trusted. No "magic key"
- What does signing determine?
  - Shared UID for shared keys
  - Self-updates

Application Signing

- All .apk files must be signed with a certificate
  - identifies the author of the application.
  - does not need to be signed by a certificate authority
- allows the system to grant or deny applications
  - access to signature-level permissions
  - request to be given the same Linux identity as another application.
- If the public key matches the key used to sign any other APK, the new APK may request to share a UID with the other APK.

Device Administration

- Since Android 3.0
- Remote wipe
- Require strong password
- Full device encryption
- Disable camera



App/Device Admin

General
Password quality
Password expiration
Lock screen / Wipe

General
ENABLE ADMIN
1
Enable admin
DEVICE CAPABILITIES
e all device
ameras enable

Activate device administrator?

Sample Device Admin

Additional text explaining why this needs to be added.

Activating this administrator will allow the app API Demos to perform the following operations:

- **Erase all data**
  Erase the tablet's data without warning, by performing a factory data reset
- **Change the screen-unlock password**
  Change the screen-unlock password
- **Set password rules**
  Control the length and the characters allowed in screen-unlock passwords
- **Monitor screen-unlock attempts**
  Monitor the number of incorrect passwords entered when unlocking the screen, and lock the tablet or erase all the tablet's data if too many incorrect passwords are entered
- **Lock the screen**
  Control how and when the screen locks
- **Set lock-screen password expiration**
  Control how frequently the lock-screen password must be changed
- **Set storage encryption**
  Require that stored application data be encrypted
- **Disable cameras**
  Prevent use of all device cameras

Cancel
2
Activate

Permissions

- Whitelist model
  - Allow minimal access by default
  - User accepted access
- Ask users fewer questions
- Make questions more understandable
- 194 permissions

PERMISSION_GRANTED or PERMISSION_DENIED Context.checkCallingPermission() Arbitrarily fine-grained permissions

Context.checkPermission(String, pid, uid)

Android Sandbox

- The sandbox is based on separation of
  - Processes
  - file permissions
  - Authenticated IPC
- Each application
  - is a different “user”; its own UID
  - runs in its own Linux process
  - its own Dalvik VM
- Sandboxes native code and sys applications

Android Sandbox

- Place access controls close to the resource, not in the VM
  - Smaller perimeter ⇒ easier to protect
- Default Linux applications have too much power
- Lock down user access for a "default" application
- Fully locked down applications limit innovation
- Relying on users making correct security decisions is tricky

File-system Encryption

- full file system encryption
- Android 3.0 and later
- AES128
- Password + random salt

Other Protections

- Mechanisms:
  - Android 1.5+: stack buffer, integer overflow protection; double free, chunk consolidation attack prevention
  - Android 2.3+: format string protection, NX, null pointer dereference mitigation
  - Android 4.0+: ASLR implemented
  - Android 4.1+: ASLR strengthened, plug kernel leaks

Rooting of Android Devices

- root
  - uid == 0 as in Linux
  - has full access to all
  - applications and all application data
  - System
  - the kernel and a few core applications
- Boot Loaders
  - embedded system boot techniques
  - “Locked”: Check a signature of the OS files being booted, or installed.

SIM Card Access

- Low level access to the SIM card is not available to third-party apps.
- The OS handles all communications with the SIM card including access to personal information (contacts) on the SIM card memory.
- Applications also cannot access AT commands, as these are managed exclusively by the Radio Interface Layer (RIL). The RIL provides no high level APIs for these commands.

GSM Vulnerabilities

- GSM
  - Largest Mobile network in the world
  - 3.8 billion phones on network
- David Hulton and Steve Muller developed method to quickly crack GSM encryption
  - Can crack encryption in under 30 seconds
  - Allows for undetectable evesdropping
- Similar exploits available for CDMA phones

SMS Vulnerabilities

- Short Messaging System
  - Very commonly used protocol
  - Used to send "Text Messages"
- GSM uses 2 signal bands, 1 for "control", the other for "data".
  - SMS operates entirely on the "control" band.
- High volume text messaging can disable the "control" band, which also disables voice calls.
- Can render entire city 911 services unresponsive.

MMS Vulnerabilities

- Unsecure data protocol for GSM
- Extends SMS, allows for WAP connectivity
- Exploit of MMS can drain battery 22x faster
- Multiple UDP requests are sent concurrently, draining the battery as it responds to request
- Does not expose data
- Does make phone useless

Case Study: Android SMS worm

- Worm spreads to all contacts via social engineering, sideloading, etc.
- Logger stored/forwarded all received SMS messages
- Only needed SEND_SMS, RECEIVE_SMS, READ_SMS permissions
- Can send 100 SMS messages/hour
- One group put SMS logger on Google Play

Bluetooth Vulnerabilities

- Short range wireless communication protocol
- Used in many personal electronic devices
- Requires no authentication
- An attack, if close enough, could take over Bluetooth device.
- Attack would have access to all data on the Bluetooth enabled device
- Practice known as bluesnarfing

Case Study: Google Wallet

- Google Wallet enables smartphone payments
  - Uses NFC technology
- credit card info stored securely in secure element
  - Separate chip, SD card, SIM card
  - Unfortunately, other data are not stored as securely

Case Study: Google Wallet

- Some information can be recovered from databases on phone:
  - Name on credit card
  - Expiration date
  - Recent transactions
- Google Analytics tracking can reveal customer behavior from non-SSL HTTP GET requests
- NFC alone does not guarantee security
  - Radio eavesdropping, data modification possible
  - Relay attacks, spoofing possible with libnfc

Sophisticated NFC Attack in Android

- Charlie Miller's Black Hat 2012 presentation: Android phones can be hijacked via NFC
  - NFC/Android Beam on by default on Android 2.3+, Android 4.0+
  - Place phone 3–4 cm away from NFC tag, other NFC-enabled phone
  - Attacker-controlled phone sends data to tag/device, can crash NFC daemon, Android OS
  - For Android 4.0–4.0.1, can remotely open device browser to attacker-controlled webpage

Information Misuse by Apps

- phone identifiers: phone number, IMEI (device identifier), IMSI (subscriber identifier), and ICC-ID (SIM card serial number).
- Phone identifiers are frequently leaked through plaintext requests.
- Phone identifiers are used as device fingerprints.
- Phone identifiers, specifically the IMEI, are used to track individual users.
- Not all phone identifier use leads to exfiltration.
- Phone identifiers are sent to advertisement and analytics servers.

Protections and Prevention

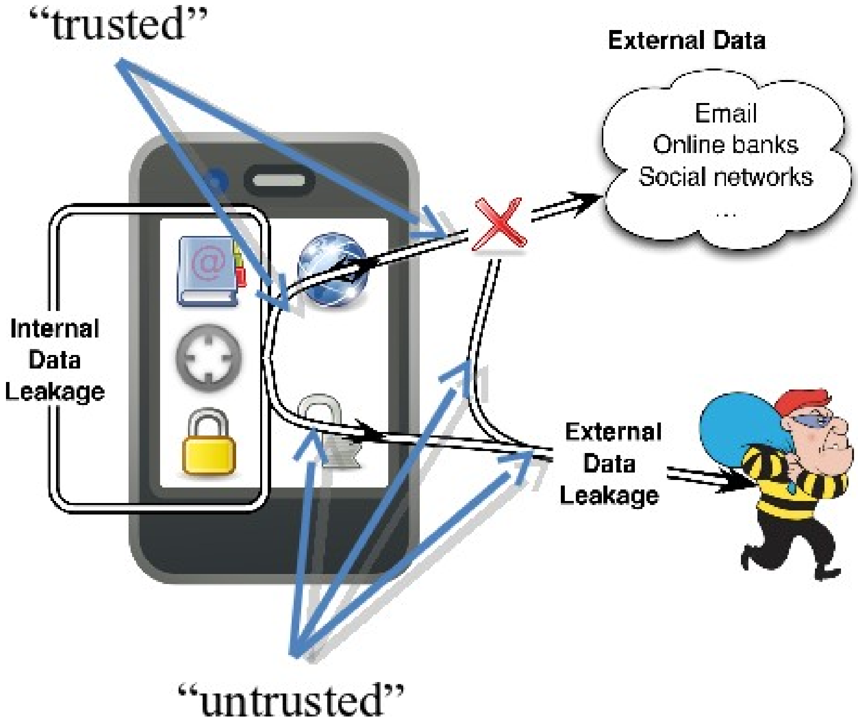
Information Leaking in Mobile Device

- Types of mobile device information sources:
  - Internal to device (e.g., GPS location, IMEI, etc.)
  - External sources (e.g., CNN, Chase Bank, etc.)
- Third-party mobile apps can leak info to external sources
  - Send out device ID (IMEI/EID), contacts, location, etc.
  - Apps ask permissions to access such info; users can ignore!
  - Apps can intercept info sent to a source, send to different destination!
- Motives:
  - Monitor employees' activity using accelerometers
  - Ads, market research (user location, behavior, etc.)

Information Tracking Flow (ITF)

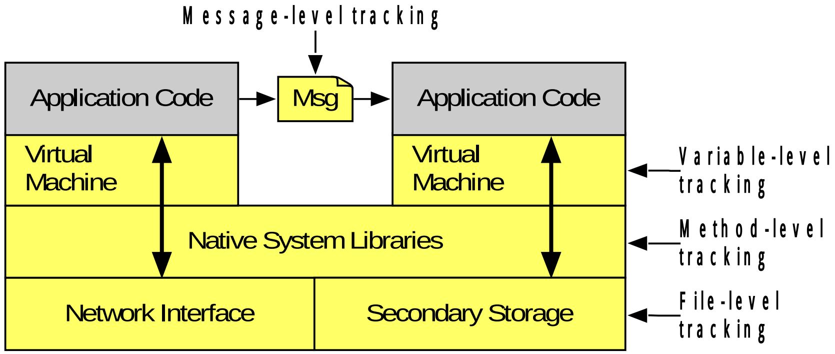
- IFT tracks each information flow among internal, external sources
  - Each flow is *tagged*, e.g., “untrusted”
  - Tag propagated as information flows among internal, external sources
  - Sound alarm if data sent to third party
- Challenges
  - Reasonable runtime,

    space overhead
  - Many information sources

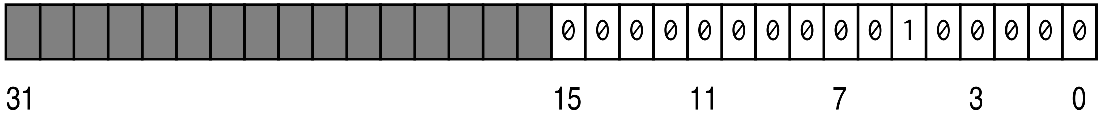


TaintDroid

- IFT system on Android 2.1
- System firmware (*not* app)
- Modifies Android's Dalvik VM, tracks info flows across methods, classes, files
- Tracks the following info:
  - Sensors: GPS, camera, accelerometer, microphone
  - Internal info: contacts, phone #, IMEI, IMSI, Google acct
  - External info: network, SMS
- Notifies user of info leakage



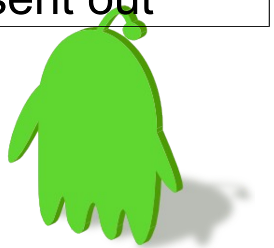
TaintDroid



- Use a 32-bit tag structure
- Set bit indicates an information flow (or sensor in use)
- Tested 30 popular Android apps (Internet permission)
- 37/105 flagged network connections were legitimate
- 15/30 apps leaked data to ad/market research firms, (admob.com, flurry.com, etc.); *not* obvious to user

applications	#	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	
Layer, Knocking, Coupons, Trapster, Spongebot Slide, ProBasketBall	6	
MySpace, Barcode Scanner, ixMAT	3	
Evernote	1	

Bit #	Tracks
31–16	Unused
15	History sent out
14	Google account sent out
13	Device serial # sent out
12	ICCID (SIM card ID) sent out
11	IMSI (subscriber ID) sent out
10	IMEI (device ID) sent out
9	SMS sent out
8	Accelerometer in use
7	Camera in use
6	“Last” location sent out
5	Data sent out over network
4	GPS location sent out
3	Phone # sent out
2	Microphone in use
1	Contacts sent out
0	Location sent out



Realtime Protection

- Apps developed to monitor other applications
  - Lookout Security & Antivirus
  - Also monitors for privacy leaks
- Have the ability to monitor the inter process communication
- Monitor for malicious activity

Pre-installation Detection

- Kirin security tool
- Analyze security configuration from the package manifest before app installation
- Every application has a security configuration which tells the OS what inter-process communication (IPC) are going to be used

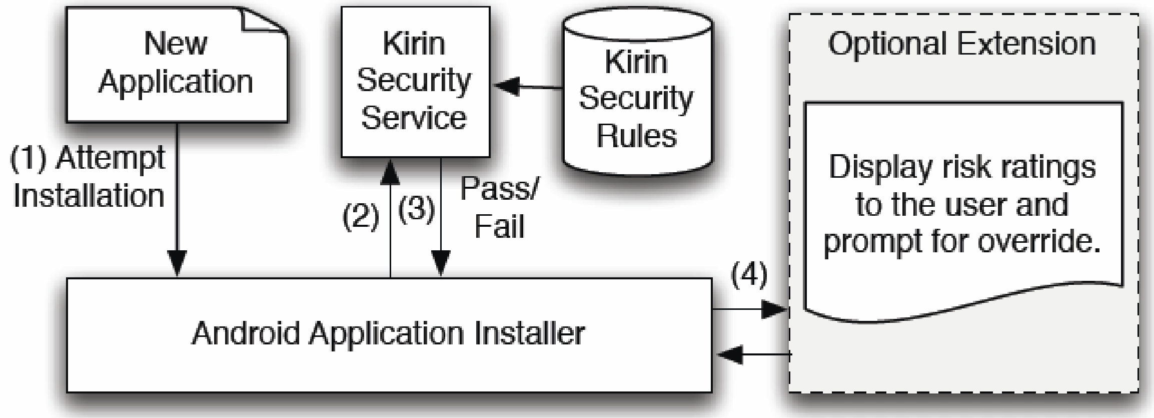


Figure 1: Kirin based software installer

Reference

Reference

- Android Security Overview, source.android. com/tech/security/
- Nils, “Building Android Sandcastles in Android’s Sandbox,” Oct 2010, BlackHat
- William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri, “A Study of Android Application Security”, 20th USENIX Security, Aug 2011

0xlab

http://0xlab.org