Mobile OS Security

CS463/ECE424

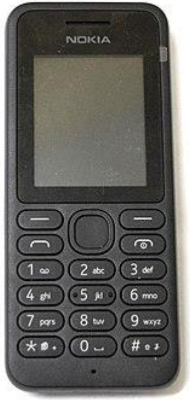
University of Illinois

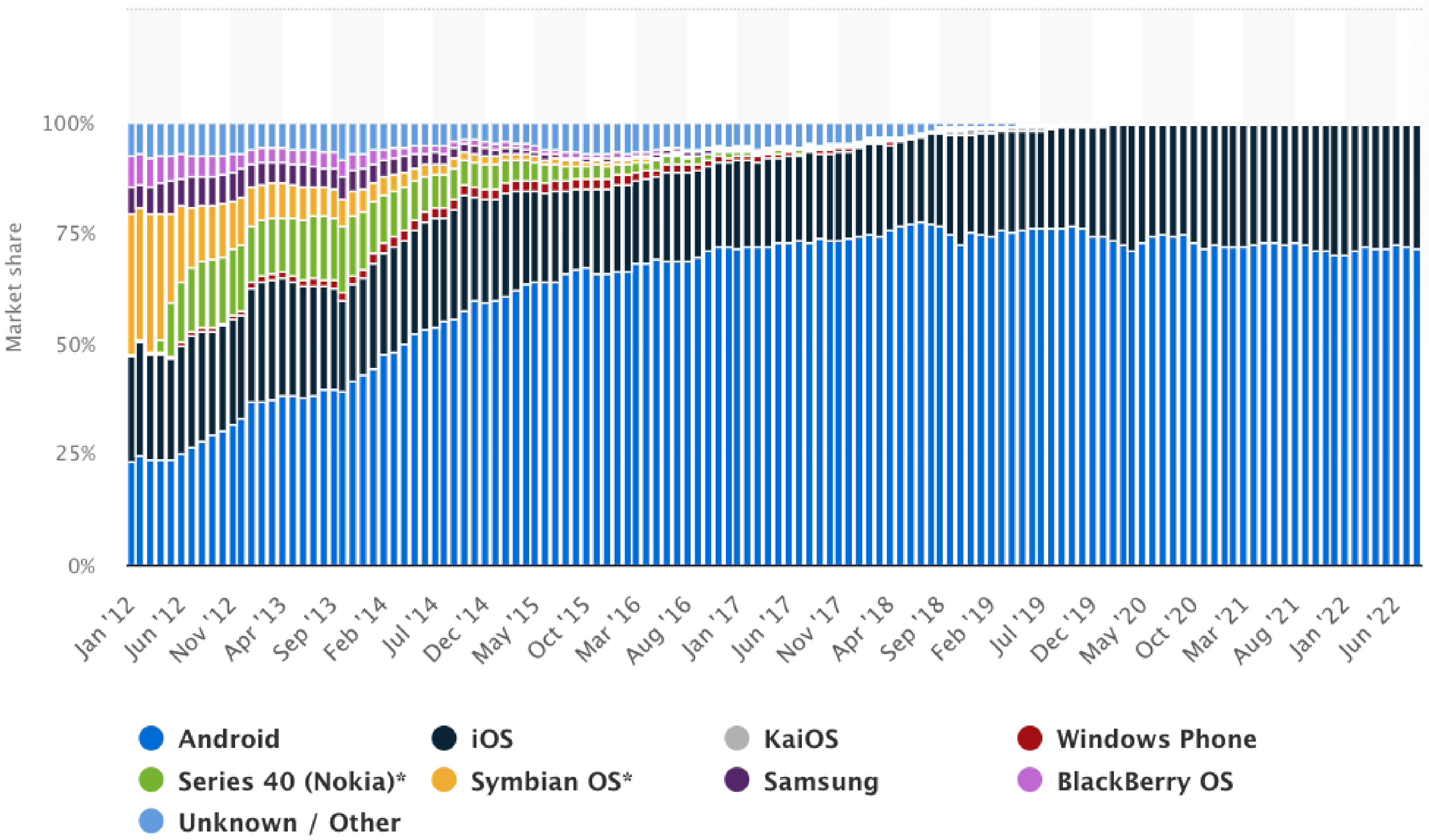
Mobile OS Security Overview (Today)
Attacks on Android (Next Class)

Mobile Phone Evolution

- Basic Phone
  - Phone call + SMS
- Feature Phones
  - Extra features on the phone firmware itself
  - Typically provided by the phone manufacturer
- Smart Phones
  - API available that enables third-party apps

My phone when
I was in college





https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

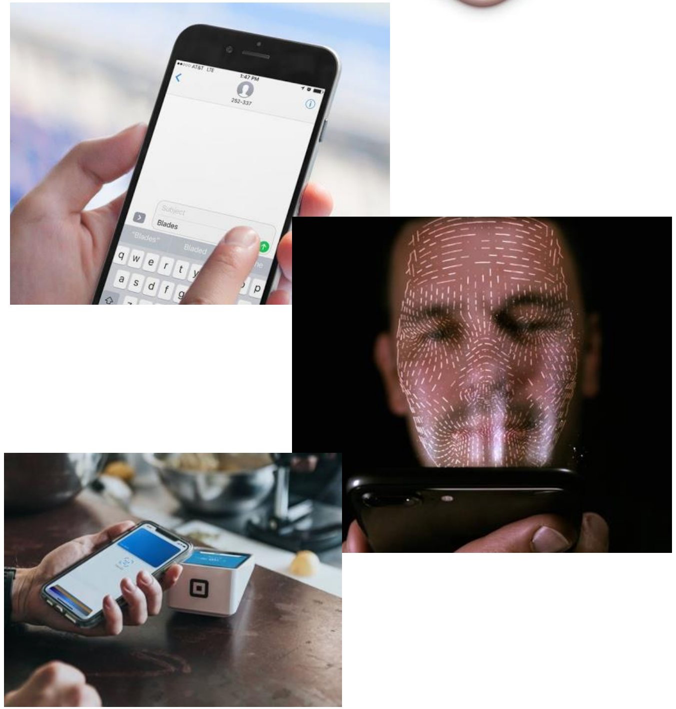
PC vs. Smartphones

- Why worry specifically about mobile OS security?
  - Why not use the same security principles we developed for PC?

- PC vs. smart phones
  - Users: root privileges typically not given to user
  - Persistent personal data, persistent login within apps
  - Battery performance is an issue
    - Implementing some security features may drain battery
  - Network usage can be expensive

PC vs. Smart Phones

- Unique features in Smart Phones
  - Location Data
    - GPS and Wifi-based tracking
  - Premium SMS Messages (expensive)
  - Placing and recording phone calls
  - Different authentication mechanisms
    - Fingerprint reader (available across platform)
    - Face Unlock (Android 5.0)
    - Trusted Places, Devices, Voice (Android 5.0)
  - Mobile payments
  - Specific third-party app markets



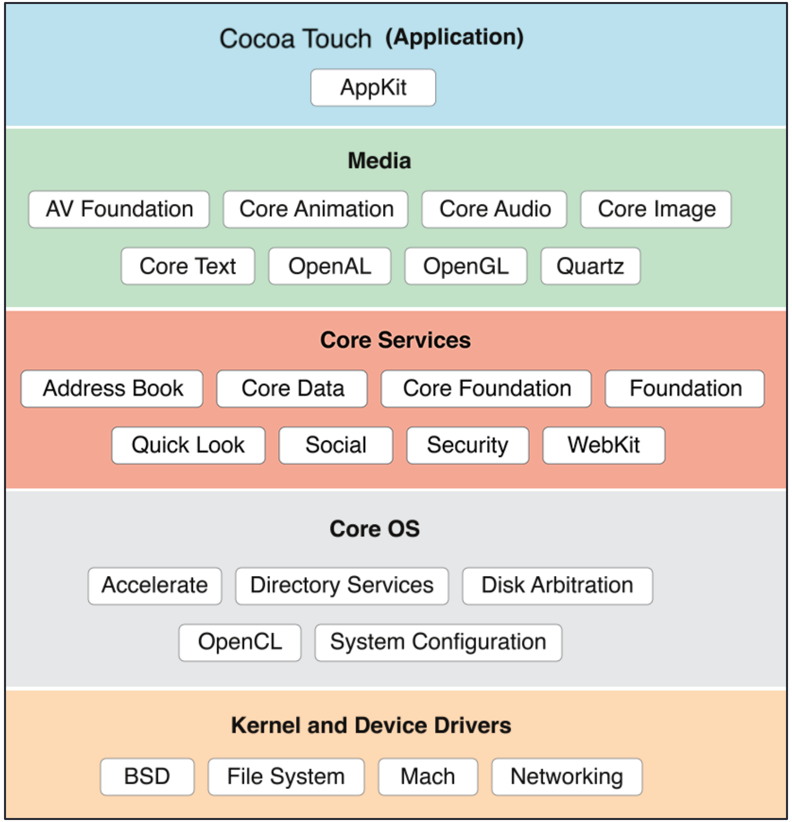
Android Security Model

Apple iOS Security (Briefly)

Mobile OS Security Frameworks

iOS Platform

- Kernel, Core OS, and Core services
  - Kernel: based on Mach kernel (like Mac OS X)
  - APIs: files, network, SQLite, POSIX threads, UNIX sockets, etc.
- Media layer
  - Supports 2D and 3D drawing, audio, video
- Cocoa Touch
  - Application framework, file management, network operations, UIKit
- Implemented in C, Objective-C, or Swift



Cocoa Touch **(Application)**

AppKit

Media

AV Foundation | Core Animation | Core Audio | Core Image

Core Text | OpenAL | OpenGL | Quartz

Core Services

Address Book | Core Data | Core Foundation | Foundation

Quick Look | Social | Security | WebKit

Core OS

Accelerate | Directory Services | Disk Arbitration

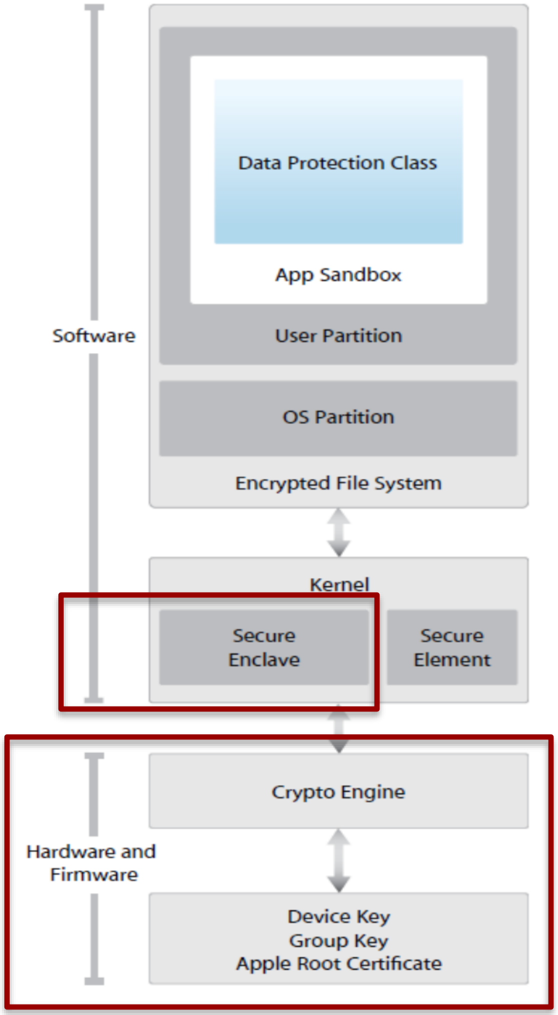
OpenCL | System Configuration

Kernel and Device Drivers

BSD | File System | Mach | Networking

iOS Security

- **System security**: integrated software/hardware
  - Cyrpto engines built in hardware (apple's TPM)
  - Support secure enclave
- **Encryption and data protection**: protects user data even when a device is lost
  - E.g., the file system is encrypted
- **App security**: secure platform foundation
  - E.g., app sandboxing
- **Device controls**: prevent unauthorized use of the device and enable remote device management



FBI–Apple Encryption Dispute

- Why passcode matters?
  - The passcode is also used together with hardware ID for generating encryption keys
- File system is encrypted even when powered off
  - The file system key is encrypted (wrapped) with an ephemeral key (never stored on disk)
  - Ephemeral key is stored in secure enclave (RAM) when powered on, thrown away once powered off
  - Ephemeral key is re-created when the device is turned on combining user passcode and hardware UID

iOS System Security

- Secure boot chain (application processor)
  - All startup process components are crypto-signed by Apple
  - Ensure integrity and proceed only after verifying the chain of trust
  - BootROM -> {Low-level bootloader ->} iBoot -> kernel
- Secure enclave coprocessor (Apple’s interpretation of TPM)
  - Secure crypto-processor (secure boot; encrypted memory)
  - Provides primitive cryptographic functions
  - Provides secure storage of cryptographic keys
  - Responsible of processing fingerprint/face data.
- Touch ID/Face ID

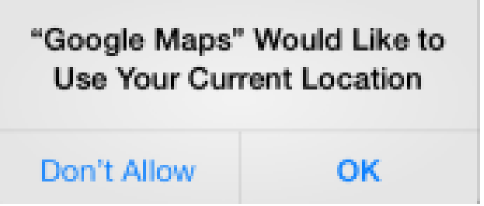
A strong passcode forms the foundation of iOS device’s cryptographic protection

iOS App Security

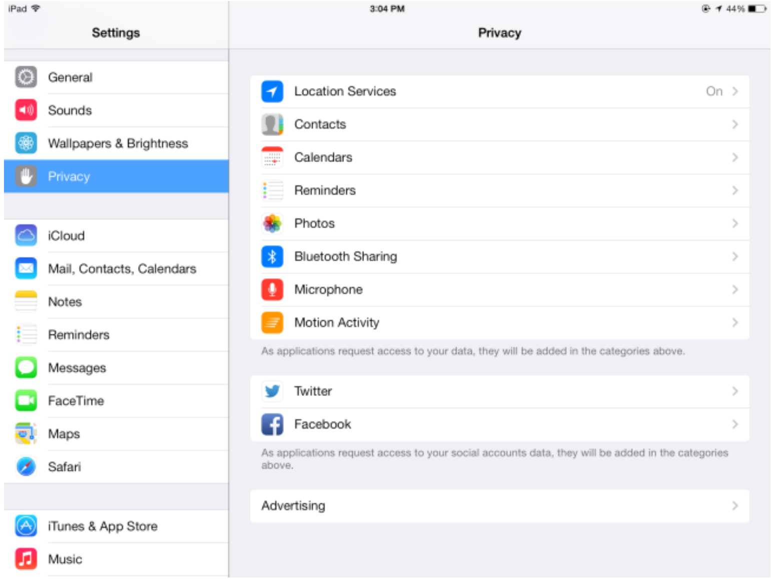
- Mandatory code signing
  - All apps must be signed using an apple-issued certificate
- Runtime protection
  - App "sandbox" prevents access to other app's data
  - System resources, kernel shielded from user apps: third-party apps and majority of iOS run under a non-privileged user-id: "mobile"
  - Inter-app communication and background tasks only through iOS APIs
  - Access to user information (or iCloud) by third-party must be declared
- Application data protection
  - Apps can take advantage of built-in hardware encryption

iOS Permissions

- iOS apps all have common default permissions (e.g. Internet)
- iOS 6, Sep 2012 onwards: certain permissions need to be enabled dynamically (e.g., location)



"Google Maps" Would Like to Use Your Current Location

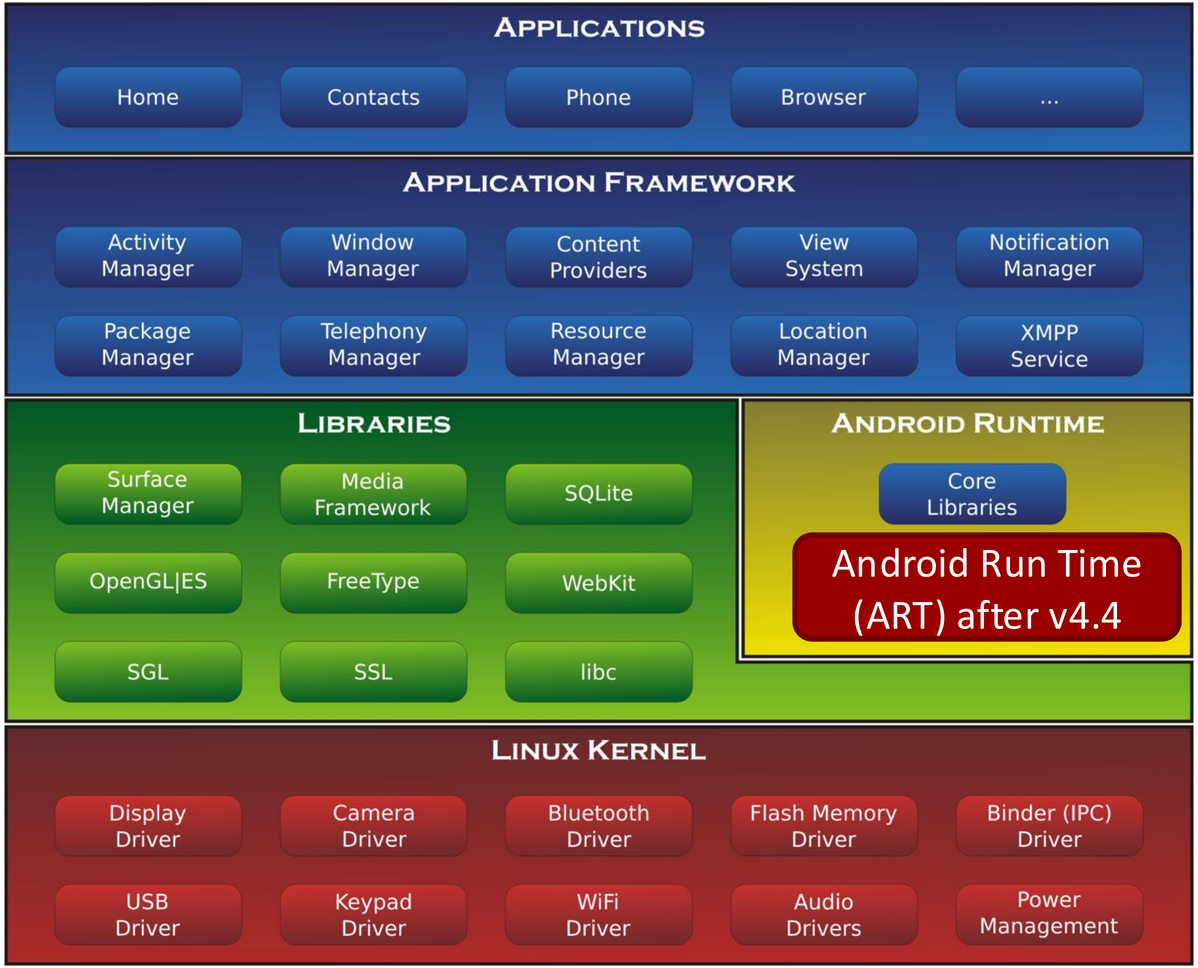


As applications request access to your data, they will be added in the categories above.

As applications request access to your social accounts data, they will be added in the categories above.

Android

- Platform outline:
  - Linux kernel
  - Embedded Web Browser
  - SQL-lite database
  - Software for secure network communication
    - Open SSL, Bouncy Castle crypto API and Java library
  - Java platform for running applications
  - C language infrastructure
  - Video APIs, Bluetooth, vibrate phone, etc.



APPLICATIONS

Home, Contacts, Phone, Browser, ...

APPLICATION FRAMEWORK

Activity Manager, Window Manager, Content Providers, View System, Notification Manager

Package Manager, Telephony Manager, Resource Manager, Location Manager, XMPP Service

LIBRARIES

Surface Manager, Media Framework, SQLite

OpenGL|ES, FreeType, WebKit

SGL, SSL, libc

ANDROID RUNTIME

Core Libraries

Android Run Time (ART) after v4.4

LINUX KERNEL

Display Driver, Camera Driver, Bluetooth Driver, Flash Memory Driver, Binder (IPC) Driver

USB Driver, Keypad Driver, WiFi Driver, Audio Drivers, Power Management

Android Market

- Open market
  - Less rigorously reviewed (*the situation is improved recently*)
  - Bad applications may show up on market
  - Malware writers can get code onto platform: self-signed applications

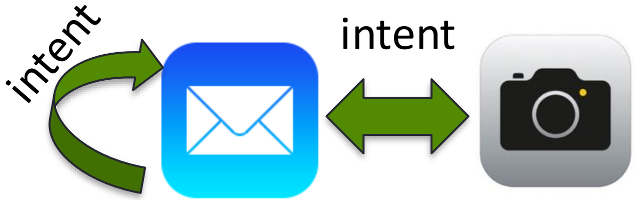
- App permissions granted on user installation Android < 6.0, at runtime for Android >= 6.0

Android Application Structure

- Four main components
  - Activity – one-user task
    - E.g., scroll through your inbox
  - Service – Java daemon that runs in background
    - E.g., application that streams music
  - Broadcast receiver
    - “mailboxes” for messages from other applications
  - Content provider
    - Store and share data using a relational database interface
- Activating components
  - Using “Intents” (a form of IPC)

Android Intents

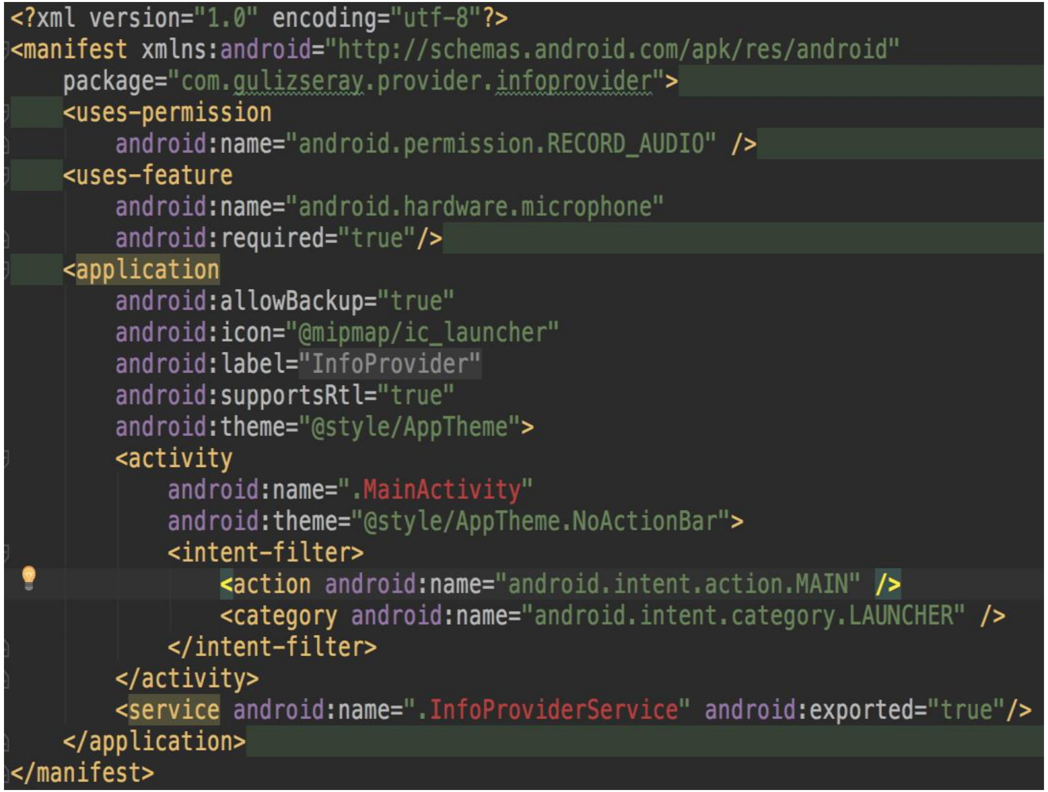
- Message between components in same or different apps
- Intent is a bundle of information
  - **action** to be taken
  - **data** to act on
  - **category** of component to handle the intent
  - instructions on how to launch a target *activity*



- Routing can be
  - **Explicit**: delivered only to a specific receiver
  - **Implicit**: all components that have registered to receive that action will get the message (more “dangerous”)

Android Manifest File

- Declarations
  - Components
  - Component capabilities
    - Intent filters
    - Permissions etc.
  - App requirements
    - Permissions
    - Sensors etc.



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.gulizseray.provider.infoprovider">
    <uses-permission
        android:name="android.permission.RECORD_AUDIO" />
    <uses-feature
        android:name="android.hardware.microphone"
        android:required="true"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="InfoProvider"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".InfoProviderService" android:exported="true"/>
    </application>
</manifest>
```

Android Permissions

- Example of permissions provided by Android
  - “android.permission.INTERNET”
  - “android.permission.READ_EXTERNAL_STORAGE”
- Protection levels
  - Dangerous

    Granted in runtime
  - Normal

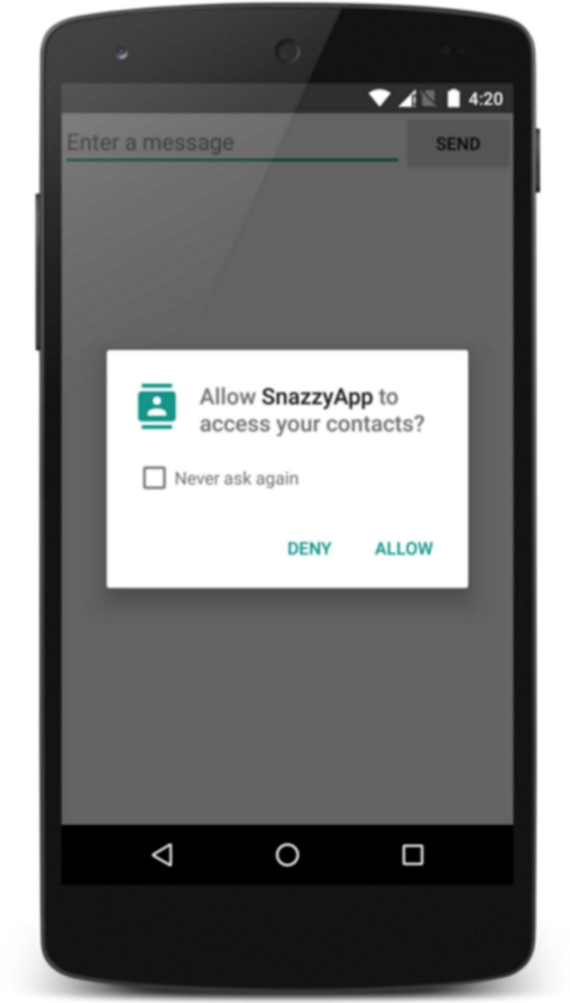
    Granted during app installation
  - Signature

    Custom permission (granted at installation)
    Allow/disallow other apps to use your features
  - SignatureOrSystem

    Permission to call system components

Android Runtime Permissions

- Dangerous permissions granted at runtime
- Normal and signature permissions still granted at installation
- Only valid for apps API >= 23 (Android 6.0)
- Permissions granted based on a permission group basis (changes in Oreo v8)

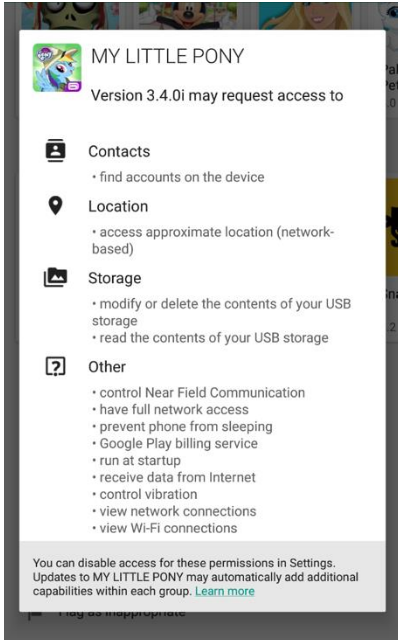


Allow **SnazzyApp** to access your contacts?

□ Never ask again

Android Permission Groups

- All dangerous permissions in a group will be granted!



MY LITTLE PONY

Version 3.4.0i may request access to

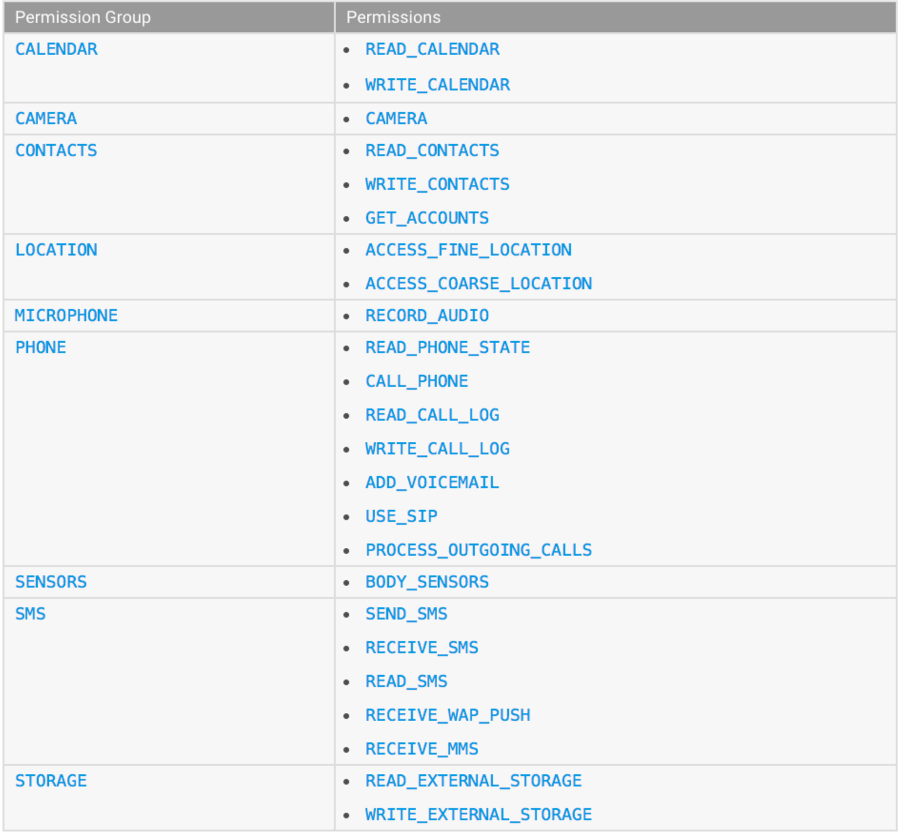
Contacts
- find accounts on the device

Location
- access approximate location (network-based)

Storage
- modify or delete the contents of your USB storage
- read the contents of your USB storage

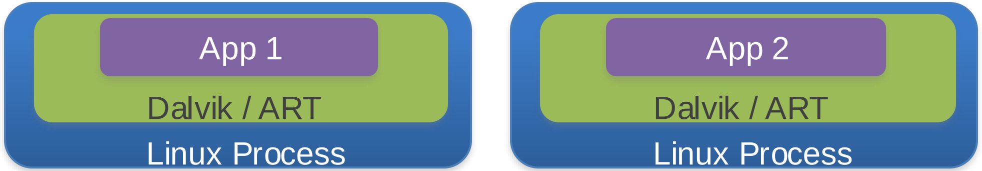
Other
- control Near Field Communication
- have full network access
- prevent phone from sleeping
- Google Play billing service
- run at startup
- receive data from Internet
- control vibration
- view network connections
- view Wi-Fi connections

You can disable access for these permissions in Settings. Updates to MY LITTLE PONY may automatically add additional capabilities within each group. Learn more



Permission Group	Permissions
CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP • PROCESS_OUTGOING_CALLS
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

Isolation

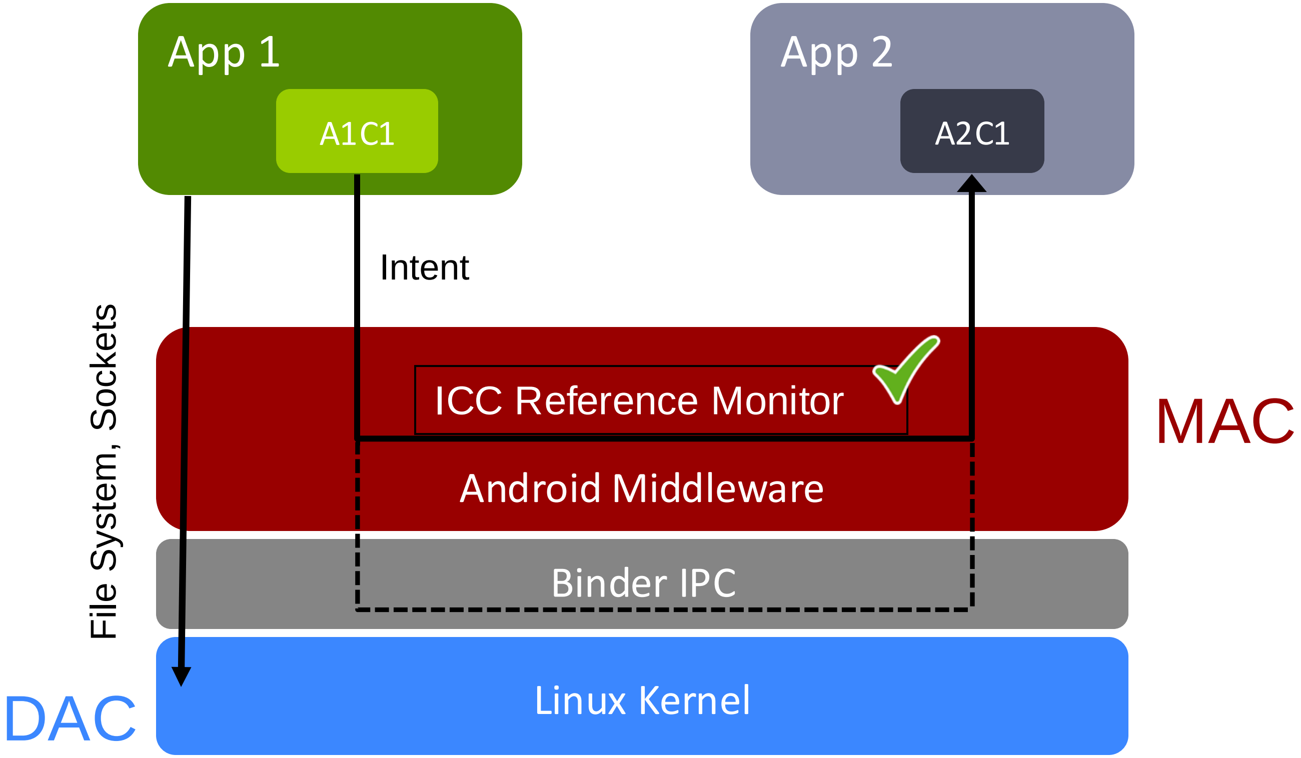


- Multi-user Linux operating system

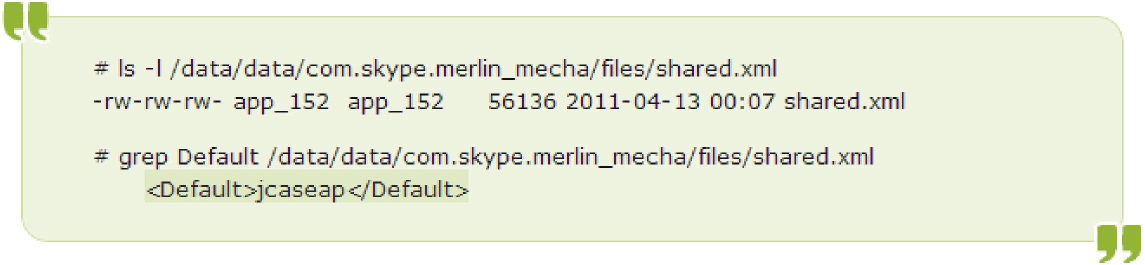
- Each app normally runs as “a different user”
  - Each app has its own VM
  - Traditional linux-based permissions (DAC: Discretionary Access Control)

- Applications announce permission requirements
  - Create an allowlist – user grants access
  - Inter-component communication (ICC) reference monitor checks permissions (MAC: Mandatory Access Control)

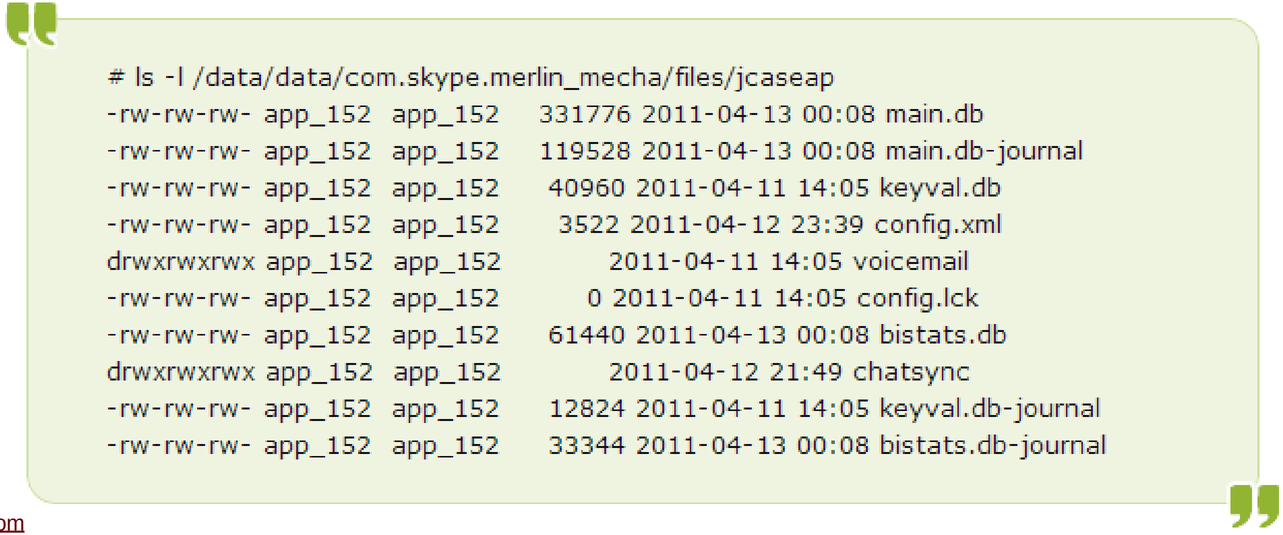
Binder IPC & Permission Enforcement



Skype Privacy Leak



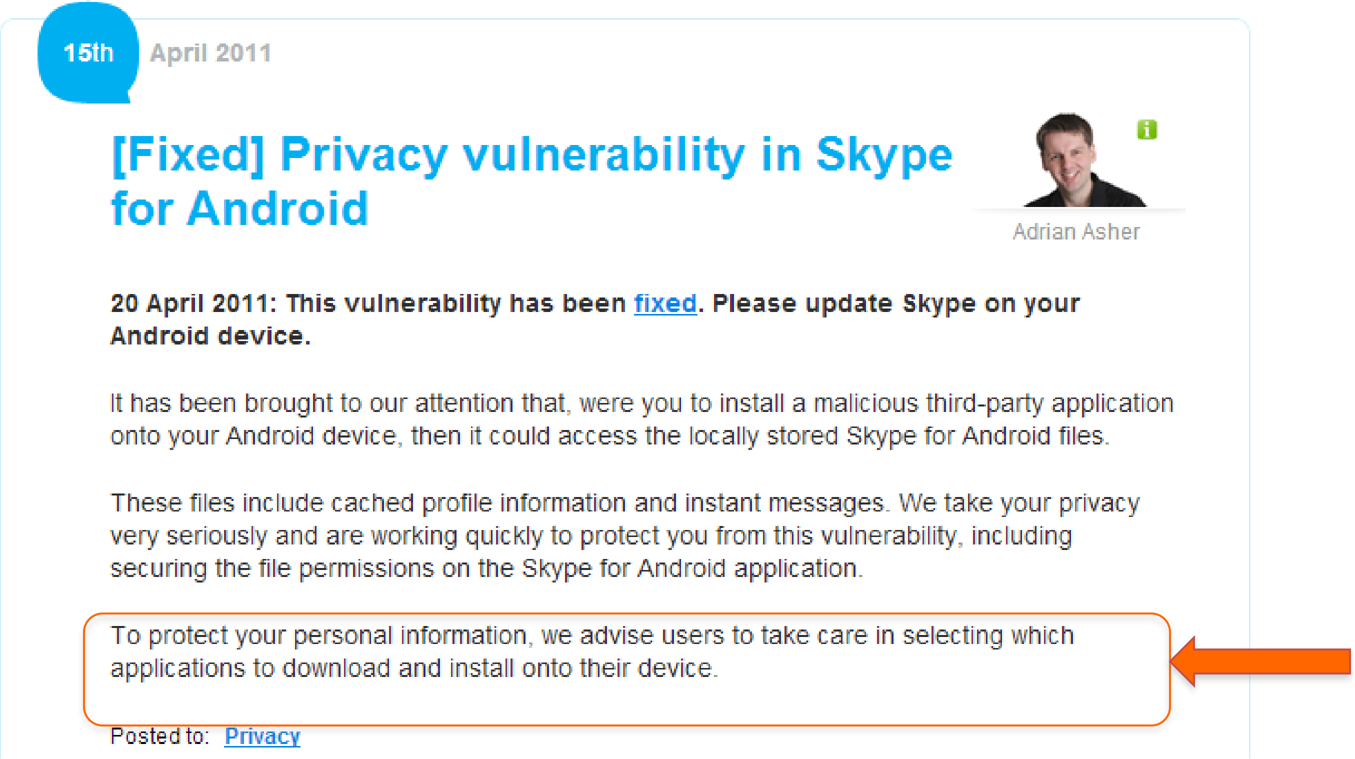
```
# ls -l /data/data/com.skype.merlin_mecha/files/shared.xml
-rw-rw-rw- app_152  app_152     56136 2011-04-13 00:07 shared.xml

# grep Default /data/data/com.skype.merlin_mecha/files/shared.xml
     <Default>jcaseap</Default>
```



```
# ls -l /data/data/com.skype.merlin_mecha/files/jcaseap
-rw-rw-rw- app_152  app_152    331776 2011-04-13 00:08 main.db
-rw-rw-rw- app_152  app_152    119528 2011-04-13 00:08 main.db-journal
-rw-rw-rw- app_152  app_152     40960 2011-04-11 14:05 keyval.db
-rw-rw-rw- app_152  app_152      3522 2011-04-12 23:39 config.xml
drwxrwxrwx app_152  app_152           2011-04-11 14:05 voicemail
-rw-rw-rw- app_152  app_152         0 2011-04-11 14:05 config.lck
-rw-rw-rw- app_152  app_152     61440 2011-04-13 00:08 bistats.db
drwxrwxrwx app_152  app_152           2011-04-12 21:49 chatsync
-rw-rw-rw- app_152  app_152     12824 2011-04-11 14:05 keyval.db-journal
-rw-rw-rw- app_152  app_152     33344 2011-04-13 00:08 bistats.db-journal
```

http://www.androidpolice.com

Skype Privacy Leak



15th April 2011

[Fixed] Privacy vulnerability in Skype for Android

Adrian Asher

20 April 2011: This vulnerability has been fixed. Please update Skype on your Android device.

It has been brought to our attention that, were you to install a malicious third-party application onto your Android device, then it could access the locally stored Skype for Android files.

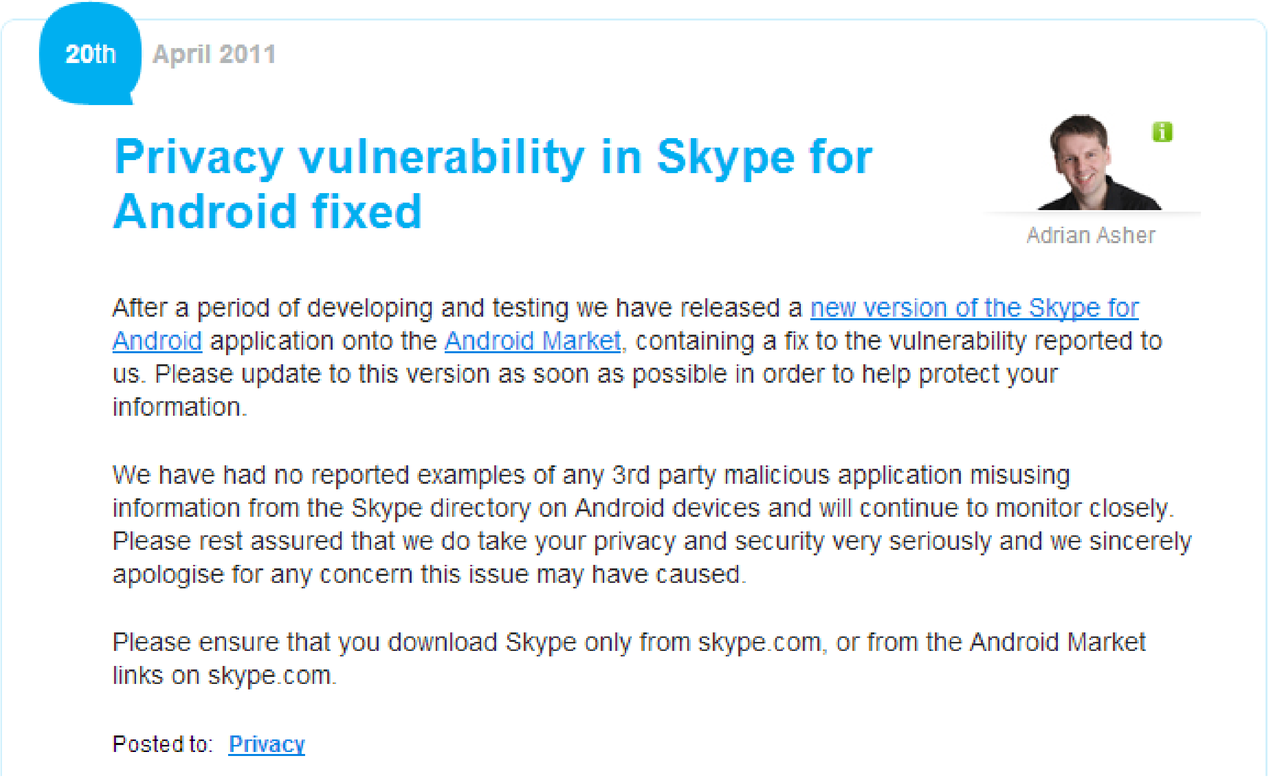
These files include cached profile information and instant messages. We take your privacy very seriously and are working quickly to protect you from this vulnerability, including securing the file permissions on the Skype for Android application.

To protect your personal information, we advise users to take care in selecting which applications to download and install onto their device.

Posted to: Privacy

http://blogs.skype.com/security/2011/04/privacy_vulnerability_in_skype.html

Skype Privacy Leak



20th April 2011

Privacy vulnerability in Skype for Android fixed

Adrian Asher

After a period of developing and testing we have released a new version of the Skype for Android application onto the Android Market, containing a fix to the vulnerability reported to us. Please update to this version as soon as possible in order to help protect your information.

We have had no reported examples of any 3rd party malicious application misusing information from the Skype directory on Android devices and will continue to monitor closely. Please rest assured that we do take your privacy and security very seriously and we sincerely apologise for any concern this issue may have caused.

Please ensure that you download Skype only from skype.com, or from the Android Market links on skype.com.

Posted to: Privacy

http://blogs.skype.com/security/2011/04/privacy_vulnerability_in_skype.html

Comparison: iOS vs Android

- App approval process
  - Android apps from “open” app store
  - iOS vendor-controlled store of vetted apps
- Application permissions
  - Android permission: install-time manifest (< 6.0), + ask-on-first-use (>= 6.0)
  - All iOS apps have same set of “sandbox” privileges, modified from iOS6 onwards
- App programming language
  - Android apps written in Java; no buffer overflow
  - iOS apps written in Objective-C (now Swift)
- TPM on smartphones!

Countermeasures

- App-store based model
  - Rely on manual audit and accountability
  - Rely on automated analysis
- Hardened Platform Security
  - Security Enhancements for Android (SEAndroid)
    - SELinux (Security-Enhanced Linux) equivalent for Android

Google Bouncer

- Perform Static and Dynamic Analysis
  - Exact details not known
- Static Analysis
  - Look at information flow from sources to sinks
  - Impractical to do it for all possible cases
    - Choose some sensitive sinks and sources
- Dynamic Analysis
  - Run app for 5 minutes (emulator)
  - Look for hidden behavior
    - Unknown heuristics
- If flagged → manual analysis → suspension

Discussion Questions

- Which permission model do you prefer: Installation-Time vs Ask-On-First-Use vs something else?

- We’ve seen that most of mobile malware target Android phones. Why do you think this is happening?

- What could a malware do on a mobile device vs a desktop machine?

Reading

- *https://www.apple.com/business/docs/iOS_Security_Guide.pdf*
- *https://source.android.com/security/*
- [EnckIEEES&P09] Understanding Android Security. Enck, William, Machigar Ongtang, and Patrick McDaniel. "Understanding android security." *IEEE security & privacy.* 2009
- [WijesekeraUsenix15]: *Primal Wijesekera, University of British Columbia; Arjun Baokar, Ashkan Hosseini, Serge Egelman, and David Wagner, University of California, Berkeley; Konstantin Beznosov, University of British Columbia*
- [SmalleyNDSS13]: *Smalley, Stephen, and Robert Craig. "Security Enhanced (SE) Android: Bringing Flexible MAC to Android." NDSS. Vol. 310. 2013.*