Smartphone Security Overview

Jagdish Prasad Achara, **Vincent Roca**

Inria

8 avril 2016

Outline

1. Security Mechanisms employed in iPhones
2. Security Mechanisms employed in Android-powered Smartphones
3. Comparison between security mechanisms available in iPhone and Android-powered smartphones
4. Security implications of modifying the default software stack of the devices

Outline

1. Security Mechanisms employed in iPhones

2. Security Mechanisms employed in Android-powered Smartphones

3. Comparison between security mechanisms available in iPhone and Android-powered smartphones

4. Security implications of modifying the default software stack of the devices

iPhone

iPhone security reference : `https://www.apple.com/la/iphone/business/docs/iOS_Security_May12.pdf`

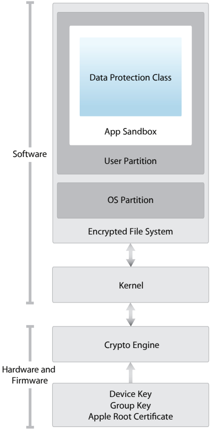
iOS : Software running on iPhone is :

- Immutable code in Boot ROM
- Firmware
- Bootloaders (LLB, iBoot)
- iOS (XNU kernel, system modules, services, apps)
- Third-party Apps downloaded and installed from Apple AppStore

iOS :

- a closed proprietary OS from Apple built on top of XNU kernel
- The majority of iOS runs as non-privileged user ”mobile”
- The entire OS partition is mounted read-only
- Remote login services aren’t included in the system software

iPhone Security Architecture

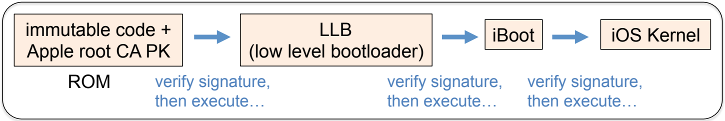


Security architecture diagram of iOS provides a visual overview of the different technologies

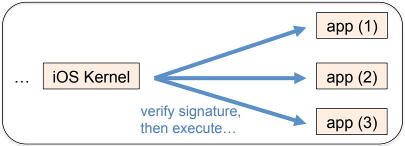
Diagram from Apple iOS Security Document.

iPhone Security features (1)

- Secure Boot Chain
  - Immutable code is laid down during chip fabrication, and is implicitely trusted.

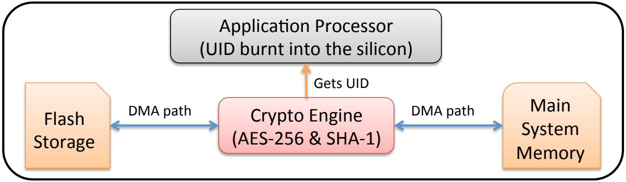

immutable code + Apple root CA PK
ROM
verify signature, then execute…
LLB (low level bootloader)
verify signature, then execute…
iBoot
verify signature, then execute…
iOS Kernel

- Runtime process security by iOS kernel
  - Mandatory code signing extends the concept of chain of trust from the OS to Apps
  - At runtime, code signature checks of all executable memory pages are made as they are loaded



iPhone Security features (2)

- Data protection feature



Depiction of data protection feature on iPhone System on Chip (SoC)

- Four kinds of data protection :
  1. *Complete Protection*
  2. *Protected Unless Open*
  3. *Protected Unless First User Authentication*
  4. *No Protection*

iPhone Security features (3)

- Data protection feature (Contd...)

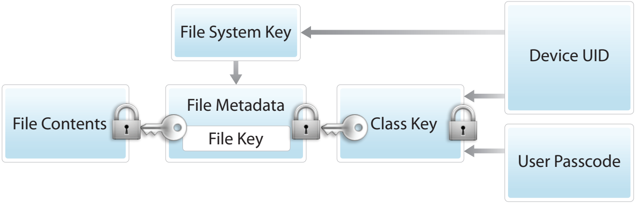
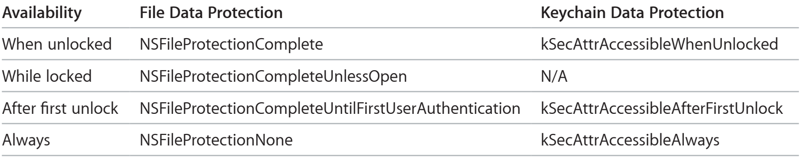


Diagram from Apple iOS Security Document.

- Complete Protection : Class key is protected with user passcode and UID.
- Protected Unless Open : Using asymmetric elliptic curve cryptography.
- Protected Untill First User Authentication : Protects data from attacks that involve a reboot.
- No Protection : Class key is protected only with UID. It is default class.

iPhone Security features (4)

- KeyChain for storing short but sensitive data and optionally, data can be shared with other apps from the same developer
  - Keychain access APIs result in calls to the *securityd* framework.
  - *securityd* determines if a process can access a keychain item or not based on that process's ’”keychain-access-group” and ”application-identifier” entitlement
  - Keychain data protection class structure

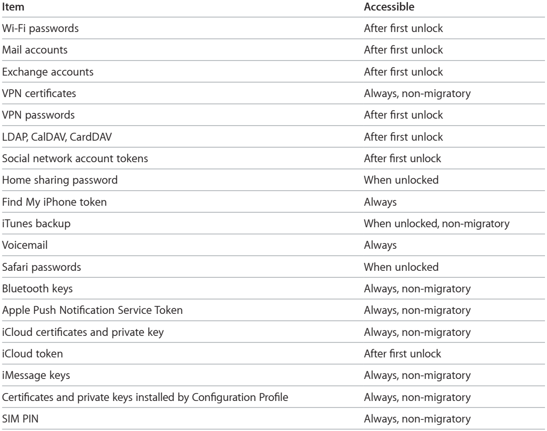


Availability	File Data Protection	Keychain Data Protection
When unlocked	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
While locked	NSFileProtectionCompleteUnlessOpen	N/A
After first unlock	NSFileProtectionCompleteUntilFirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Always	NSFileProtectionNone	kSecAttrAccessibleAlways

Diagram from Apple iOS Security Document.

iPhone Security features (5)

- KeyChain for storing short but sensitive data and optionally, data can be shared with other apps from the same developer (Contd...)
  - Encryption with device UID prevents restoring keychain items at another device (even if it's in "No Protection" class !)

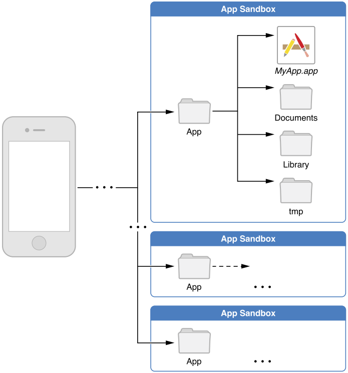


Item	Accessible
Wi-Fi passwords	After first unlock
Mail accounts	After first unlock
Exchange accounts	After first unlock
VPN certificates	Always, non-migratory
VPN passwords	After first unlock
LDAP, CalDAV, CardDAV	After first unlock
Social network account tokens	After first unlock
Home sharing password	When unlocked
Find My iPhone token	Always
iTunes backup	When unlocked, non-migratory
Voicemail	Always
Safari passwords	When unlocked
Bluetooth keys	Always, non-migratory
Apple Push Notification Service Token	Always, non-migratory
iCloud certificates and private key	Always, non-migratory
iCloud token	After first unlock
iMessage keys	Always, non-migratory
Certificates and private keys installed by Configuration Profile	Always, non-migratory
SIM PIN	Always, non-migratory

Diagram from Apple iOS Security Document.

iPhone Security features (6)

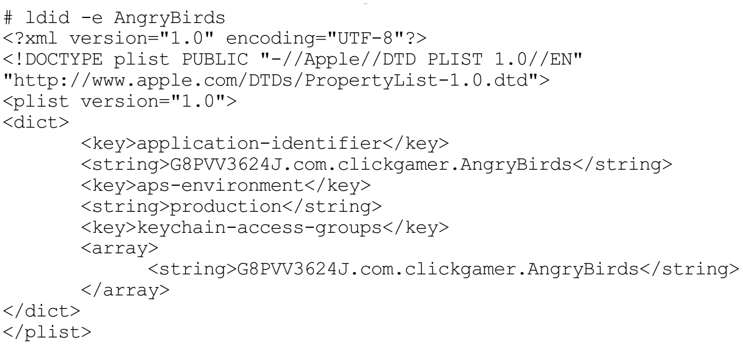
- App Sandboxing
  - System installs each app in its own sandbox directory
  - Sandbox is a set of fine-grained controls that limit access by an app to other apps and system resources.



Above diagram from Apple

iPhone Security features (8)

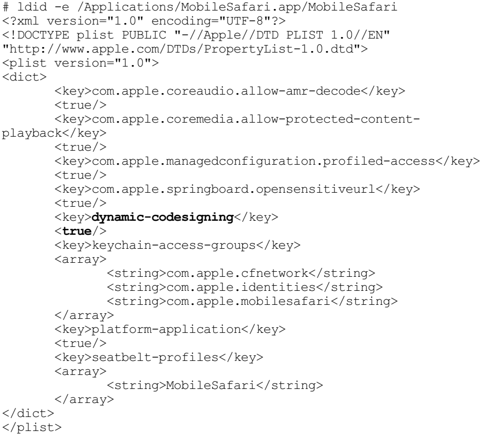
- Use of entitlements for access control
  - Key-value pairs allowing authentication beyond runtime factors like unix user id.
  - Entitlements are digitally signed.
  - Extensively used by System Apps and daemons to perform specific privileged tasks that would otherwise require the process to be run as root.
  - Greatly reduces the potential for privilege escalation by a compromised system app or daemon.



```
# ldid -e AngryBirds
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
       <key>application-identifier</key>
       <string>G8PVV3624J.com.clickgamer.AngryBirds</string>
       <key>aps-environment</key>
       <string>production</string>
       <key>keychain-access-groups</key>
       <array>
             <string>G8PVV3624J.com.clickgamer.AngryBirds</string>
       </array>
</dict>
</plist>
```

iPhone Security features (9)

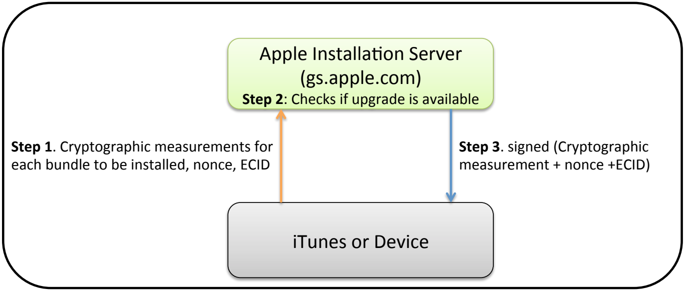
- Protection of memory from the exploitation of memory corruption bugs
  - Since iOS 4.3 use of ASLR (Address Space Layout Randomization)
  - Memory pages marked as both "writable" and "executable" can be used by Apps having Apple-only "dynamic-codesigning" entitlements. Safari uses this entitlements for its JavaScript JIT compiler.



```
# ldid -e /Applications/MobileSafari.app/MobileSafari
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
       <key>com.apple.coreaudio.allow-amr-decode</key>
       <true/>
       <key>com.apple.coremedia.allow-protected-content-
playback</key>
       <true/>
       <key>com.apple.managedconfiguration.profiled-access</key>
       <true/>
       <key>com.apple.springboard.opensensitiveurl</key>
       <true/>
       <key>dynamic-codesigning</key>
       <true/>
       <key>keychain-access-groups</key>
       <array>
              <string>com.apple.cfnetwork</string>
              <string>com.apple.identities</string>
              <string>com.apple.mobilesafari</string>
       </array>
       <key>platform-application</key>
       <true/>
       <key>seatbelt-profiles</key>
       <array>
              <string>MobileSafari</string>
       </array>
</dict>
</plist>
```

iPhone Security features (10)

- System Software Personalization
  - To prevent devices from being downgraded to older versions that lack the latest security features
  - iOS Software Updates can be installed using iTunes or OTA on the device.



Apple Installation Server (gs.apple.com)

Step 2: Checks if upgrade is available

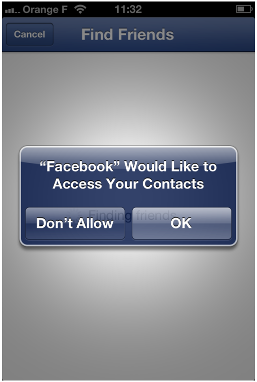
Step 1. Cryptographic measurements for each bundle to be installed, nonce, ECID

Step 3. signed (Cryptographic measurement + nonce +ECID)

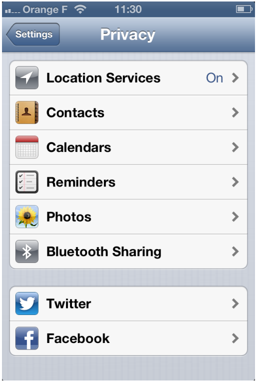
iTunes or Device

iPhone Security features (11)

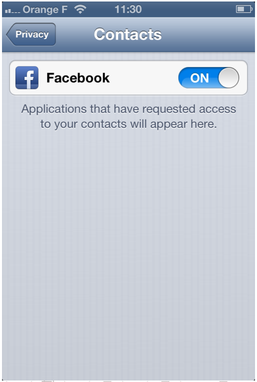
- Application Access to standard iOS APIs
  - Apple claims to verify all submitted Apps for legitimate API access. But with each iOS revision, control is being transferred to the user.
  - Mere access to private data access using APIs prompts a warning to the user and user has the option to allow/deny it.
  - However, there is no mechanism to control the way accesssed information is being used ! RESEARCH TOPIC !



“Facebook” Would Like to Access Your Contacts



Location Services On



Facebook ON

Applications that have requested access to your contacts will appear here.

Outline

1. Security Mechanisms employed in iPhones

2. Security Mechanisms employed in Android-powered Smartphones

3. Comparison between security mechanisms available in iPhone and Android-powered smartphones

4. Security implications of modifying the default software stack of the devices

Android-powered smartphones

Software running on Android-powered smartphones is :

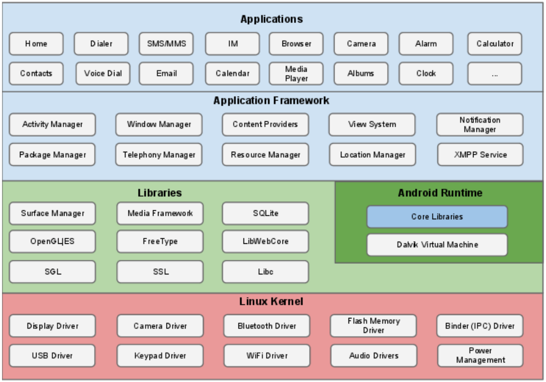
- Immutable code in Boot ROM
- Firmware
- Bootloader
- Android (Linux kernel, system modules, services, apps)
- Third-party Apps (no restriction for the source !)

Android :

- Linux based OS developed by Google in conjuction with Open Handset Alliance
- A small amount of Android OS code runs as root.
- System partition is mounted as read-only and contains Kernel, OS libraries, Application Runtime (DVM), Application framework and System Apps.
- Android apps are most often written in Java and run in the DVM (Dalvik Virtual Machine).

Android Software Stack

All software above the kernel (OS libraries, Android runtime, application framework, system and third party apps) run within the Application Sandbox.



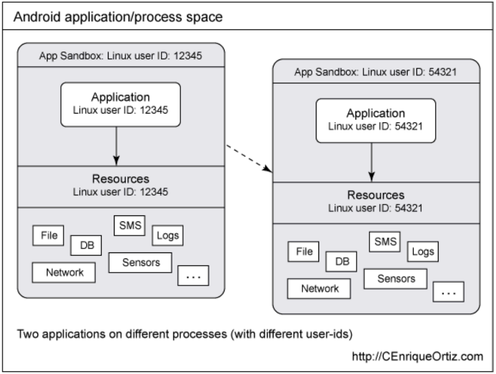
Taken from Android Open Source website.

Android Security features (1)

- Secure Boot Chain
  - Depends on manufacturer and also, on cellular service provider if it's in contract.
  - Sometimes it exists and other times it doesn't.
  - But in any case and unlike iPhones, the secure boot chain **does not extend till Apps !**
- FileSystem Encryption
  - is performed in the kernel using dm-crypt after Android v 3.0.
  - Not on by default.
  - Some custom ROM builders even removed this feature completely !
- Protection of memory from exploitation of memory corruption bugs
  - A memory corruption error will only allow execution of arbitary code in the context of that process.
  - Since Android 4.0 use ASLR (Address Space Layout Randomization)

Android Security features (2)

- Application Sandbox
  - Android System assigns a unique user id to each Android App and runs it as that user in a separate process.
  - Kernel enforces security at the process level through standard Linux facilities.
  - Apps get a dedicated part of the file system which acts as home for that App.



Two applications on different processes (with different user-ids)

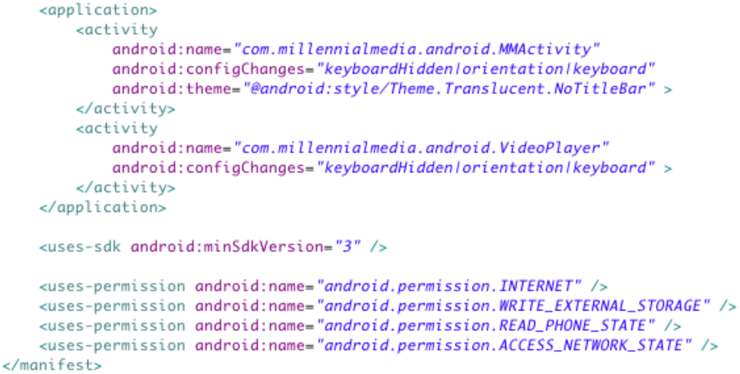
http://CEnriqueOrtiz.com

Android Security features (3)

- Application Signing
  - All installed apps must be signed
  - Helps in application updates
  - Apps coming from same developer can share the same user id (App developer can specify that in the manifest !)
- System Partition and Safe Mode
  - System partition contains Android's kernel, OS libraries, application runtime, application framework and system apps. It is set to read-only.
  - In Safe mode, only System Apps are loaded *i.e.* user can boot the phone in an environment free of third-party software.
- Android Updates
  - OTA or side-loaded updates.
  - With side-loaded updates, downgrade is possible
  - Flashing a new system image always leads to erasing all the data on the device.

Android Security features (4)

- Application Access to standard Android APIs
  - Makes use of Manifest file. All needed-permissions need to be stored in this file.

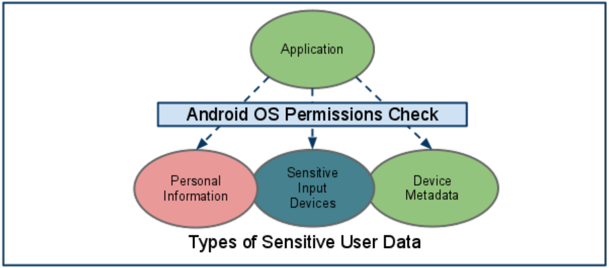


```
        <application>
            <activity
                android:name="com.millennialmedia.android.MMActivity"
                android:configChanges="keyboardHidden|orientation|keyboard"
                android:theme="@android:style/Theme.Translucent.NoTitleBar" >
            </activity>
            <activity
                android:name="com.millennialmedia.android.VideoPlayer"
                android:configChanges="keyboardHidden|orientation|keyboard" >
            </activity>
        </application>

        <uses-sdk android:minSdkVersion="3" />

        <uses-permission android:name="android.permission.INTERNET" />
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
        <uses-permission android:name="android.permission.READ_PHONE_STATE" />
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    </manifest>
```

  - User has to either allow/deny all needed permission for the app at install time.

Android Security features (5)

- Application Access to standard Android APIs (Contd...)
  - User permission is asked for accesssing user private info, internet access, SIM card access and cost-sensitive activities (telephony, SMS, network/data, In-App billing, NFC Access etc.)

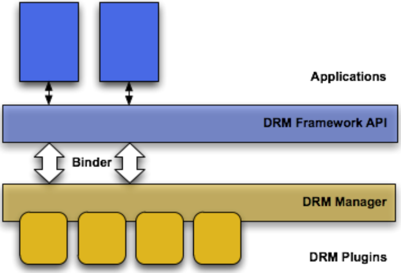


Taken from Android Open Source website.

Android Security features (6)

- Interprocess communication
  - Processes can communicate using any of the traditional UNIX methods e.g. file system, local sockets. Linux permissions still apply !
  - Android's new IPC mechanisms :
    - Binder, Services, Intents and ContentProviders

- Digital Rights Management
  - Provides a DRM framework that lets applications manage rights-protected content according to the license constraints



Outline

1. Security Mechanisms employed in iPhones

2. Security Mechanisms employed in Android-powered Smartphones

3. Comparison between security mechanisms available in iPhone and Android-powered smartphones

4. Security implications of modifying the default software stack of the devices

Comparision between security measures employed in iPhone and Android

- Secure Boot Chain ?
- Data protection features ?
- Runtime security and App Sandboxing ?
- Memory protection ?
- System upgrades and downgrades ?
- What if you lose Android-powered smartphone or iPhone ?
  - If device is passcode locked ?
  - If flash memory is taken out of the device to read the data ?

Outline

1. Security Mechanisms employed in iPhones
2. Security Mechanisms employed in Android-powered Smartphones
3. Comparison between security mechanisms available in iPhone and Android-powered smartphones
4. Security implications of modifying the default software stack of the devices

Modifying the default software stack of iPhone (1)

A very popular term "Jailbreaking" is coined for removing the restrictions put by Apple on its amazing device by modifying the software stack of the device !

- The question is : **Why** one would like to change the software stack ?
  - An open platform for which developers can write software
  - If one would like to have total control over the device
  - To bypass cellular locks and other restrictions put by carrier e.g. WiFi tethering
  - To pirate iPhone Apps
  - To evaluate the security of the device
  - To do some frauds e.g. by changing baseband code or to fake things e.g. by changing network data.

Security implications of Jailbreaking

- If your phone is lost/stolen :
  - Someone can just copy all your data...
  - ... then install some remote-login tools, spyware and rootkit and give you back the phone !
- Apps installed on a Jailbroken phone can get root privileges and have read/write access to the whole filesystem. Everything is possible with right skills !
  - A malicious app can spy all your activities on the phone
  - A malicious app can retrieve and send your personal information to third-parties
- Our opinion : One should use jailbreaked iPhone for personal use only if (s)he knows how to secure the device (implicitely requires knowing all the internals !)

Modifying the default software stack of Android Smartphones

In Android smartphones, software stack is normally modified to get "root" (privileged user) access to the phone and is known as "rooting".

- Why one would like to "root" the phone ?
  - On Android, there is no restriction on the source of apps. The only restriction is the fact that apps can run only as non-privileged user and thereby, people wanting their device without any restriction would go for it.
  - There can be a variety of motivations behind having a device without any restricitons, like removing cellular restrictions, evaluating the security or performing malicious activities.
- How do you "root" Android smartphones ?
  - If your device have an unlocked bootloader !
  - Certain manufacturers don't actually set ro.secure to 1.
  - Hack one of system process running in privileged mode e.g. z4root, gingerbreak...to execute arbitrary code (Well, the arbitrary code normally mounts /system in read-write mode and installs su command.)

Security implications of "Rooting"

- If a "rooted" Android smartphone is lost/stolen, ALL user data on the device is at risk if adb access is enabled. Even if Android encyption feature is ON !
- "Rooting" normally involves flashing custom ROM and certain custom ROM builders removed the Encryption option from the ROM ! It means data is stored in the flash as plain-text !
- Malicious apps can of course spy the activities on the device and steal personal information.