Smartphone Security Overview

Jagdish Prasad Achara, Claude Castelluccia

INRIA Rhone-Alpes

5 décembre 2012

Outline

Outline

Smartphones and Security (1)

- What is a Smartphone ?
- Smartphone vs Feature phone ?

Above images from Google

- Smartphone categorization :
  1. Based on hardware present
     - However, we'll look hardware only from security point of view
  2. Based on software running
     - Two different paradigms of OSes : *Android* and *iOS*

Smartphones and Security (2)

Smartphone security features

1. Security against physical hardware attack
2. Protection against malware
3. A mechanism to avoid illegal access to the smartphone and its data

Above image from Google

Smartphones and Security (3)

Smartphone vs other computing platforms

vs

Above images from Google

1. Smartphones provide mobile services, a computing platform, Internet access, GPS navigation unit, Digital Camera etc. into a single device
2. Extreme mobile nature of smartphones
3. They are very personal to the users

Smartphones and Security (4)

Smartphone security importance

Above Images from Google

1. It stores a plethora of personal information !
2. Facebook, Twitter, Banking, Hotel Reservation Apps caches data
3. An illegitimate access to baseband hardware may result in big blunders

Outline

iPhone

iPhone Security Reference :
`http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf`

Software running on iPhone is :

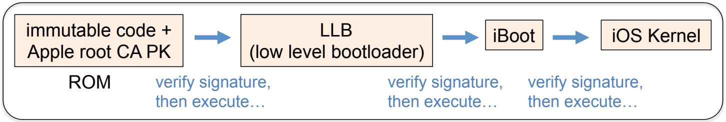
- Immutable code in Boot ROM
- Firmware
- Bootloaders (LLB, iBoot)
- iOS (XNU kernel, system modules, services, apps)
- Third-party Apps downloaded and installed from Apple AppStore

iOS :

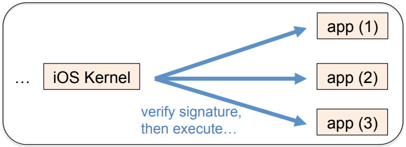
- a closed proprietary OS from Apple built on top of XNU kernel
- The majority of iOS runs as non-privileged user "mobile"
- The entire OS partition is mounted read-only
- Remote login services aren't included in the system software

iPhone Security features (1)

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

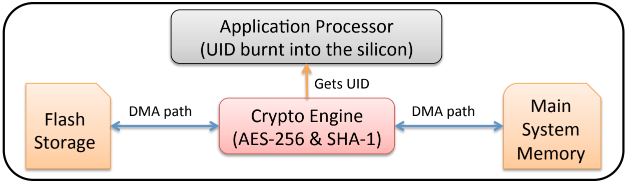

immutable code + Apple root CA PK
ROM
verify signature, then execute…
LLB (low level bootloader)
verify signature, then execute…
iBoot
verify signature, then execute…
iOS Kernel

- 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...)
  - Class key protects file key.

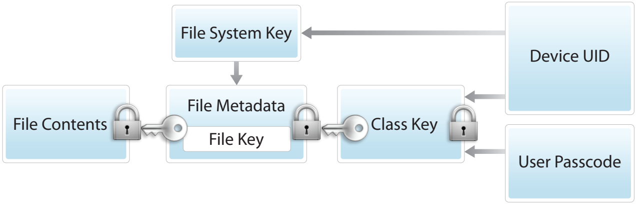
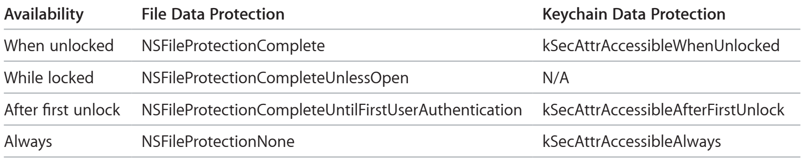


Diagram from Apple iOS Security Document.

1. *Complete Protection :* Class key is protected with user passcode and UID.
2. *Protected Unless Open :* Using asymmetric elliptic curve cryptography to generate back the per-file private key from per-file public key and private class key.
3. *Protected Untill First User Authentication :* Protects data from attacks that involve a reboot.
4. *No Protection :* Class key is protected only with UID. It is default class and prevents reading the data directly from flash storage.

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

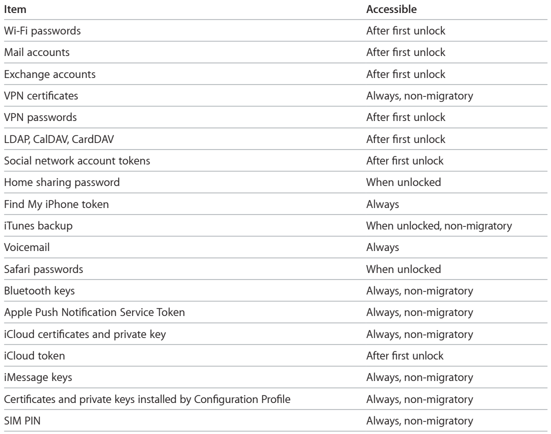


Availability	File Data Protection	Keychain Data Protection
When unlocked	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
While locked	NSFileProtectionCompleteUnlessOpen	N/A
After first unlock	NSFileProtectionCompleteUntilFirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Always	NSFileProtectionNone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

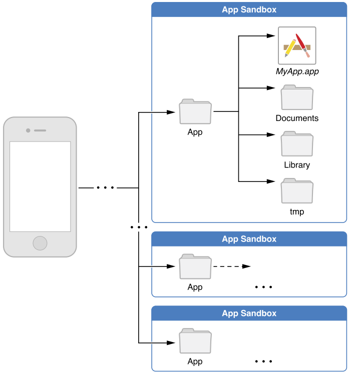


Item	Accessible
Wi-Fi passwords	After first unlock
Mail accounts	After first unlock
Exchange accounts	After first unlock
VPN certificates	Always, non-migratory
VPN passwords	After first unlock
LDAP, CalDAV, CardDAV	After first unlock
Social network account tokens	After first unlock
Home sharing password	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.

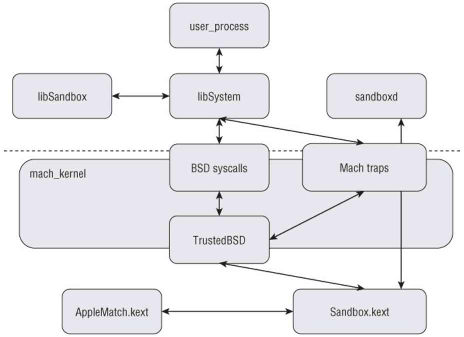


Above diagram from Apple

iPhone Security features (7)

- App Sandboxing Contd... (How it is implemented ?)
  - As a policy module for the TrustedBSD mandatory access control framework

Components of the iOS sandbox

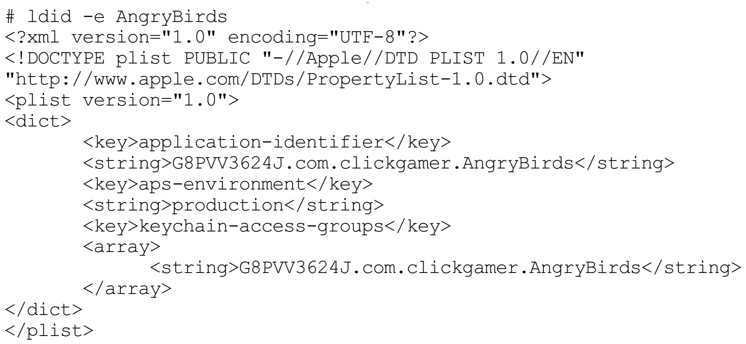


Above diagram from iOS Hacker's Handbook

  - sandboxd begins with calling sandbox_init in libSystem —> Uses libsandbox.dylib to turn human-readable policy "Only allow access to /priavte/var/mobile/Apps/AppHome" in binary format —> Passes to mac_syscall system call handled by TrustedBSD and eventually to Sandbox.kext for processing —> kext installs sandbox profile for that process

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>
```

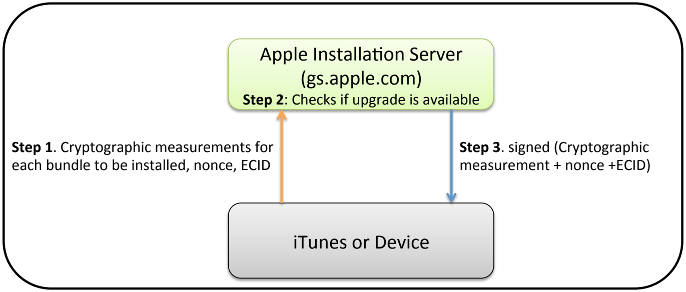
iPhone Security features (9)

- Protection of memory from the exploitation of memory corruption bugs
  - 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>
```

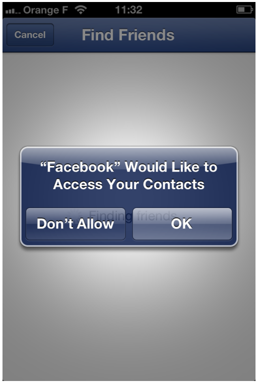
iPhone Security features (10)

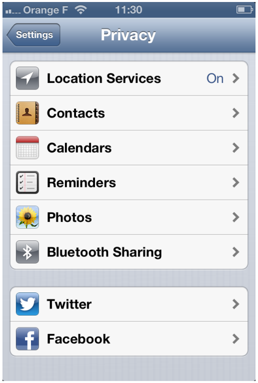
- System Software Personalization
  - To prevent devices from being downgraded to older versions that lack the latest security features
  - iOS Software Updates can be installed using iTunes or OTA on the device.

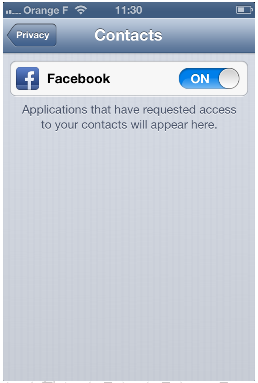

Apple Installation Server (gs.apple.com)
Step 2: Checks if upgrade is available
Step 1. Cryptographic measurements for each bundle to be installed, nonce, ECID
Step 3. signed (Cryptographic measurement + nonce +ECID)
iTunes or Device

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 !

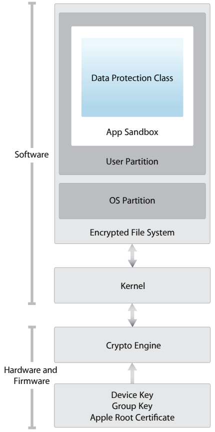

Orange F 11:32
Cancel Find Friends
"Facebook" Would Like to Access Your Contacts
Don't Allow OK


Orange F 11:30
Settings Privacy
Location Services On
Contacts
Calendars
Reminders
Photos
Bluetooth Sharing
Twitter
Facebook


Orange F 11:30
Privacy Contacts
Facebook ON
Applications that have requested access to your contacts will appear here.

iPhone Security Architecture

Does it make any sense now ?



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

Diagram from Apple iOS Security Document.

Android-powered smartphones

Software running on Android-powered smartphones is :

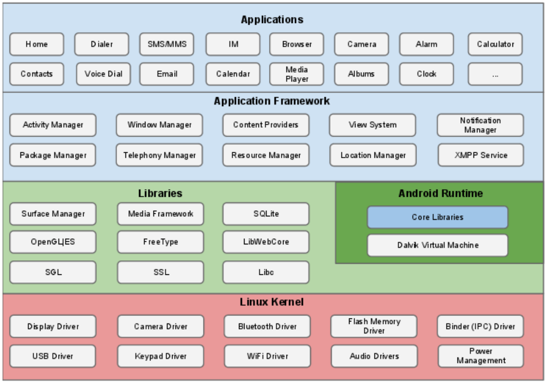
- Immutable code in Boot ROM
- Firmware
- Bootloader
- Android (Linux kernel, system modules, services, apps)
- Third-party Apps (No restriction for the source !)

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.

Android Software Stack

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



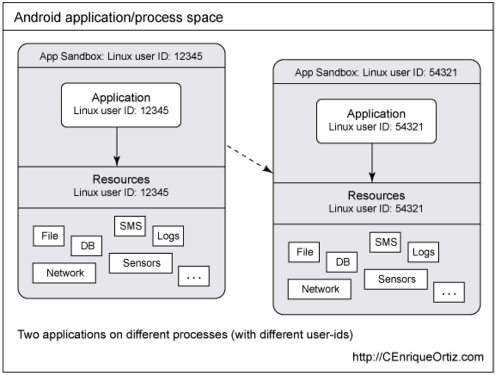
Taken from Android Open Source website.

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 unlike iPhones, it doesn't extend till Apps in any case.
- FileSystem Encryption
  - is performed in the kernel using dm-crypt after Android v 3.0.
  - Not on by default.
  - Some custom ROM builders had even removed this feature completely...strange enough !
- 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.
  - Latest versions use ASLR.

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.

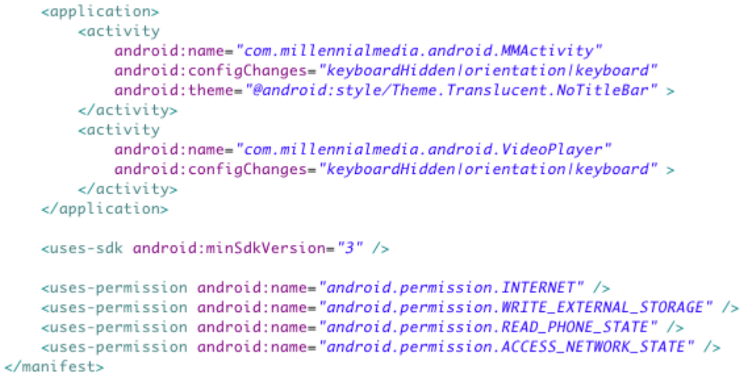


Android Security features (3)

- 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.
  - OTA mechanism isn't described ; but 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.

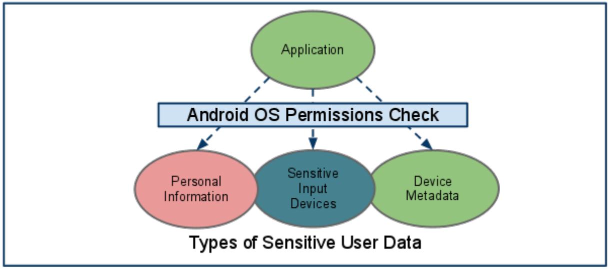


```
        <application>
            <activity
                android:name="com.millennialmedia.android.MMActivity"
                android:configChanges="keyboardHidden|orientation|keyboard"
                android:theme="@android:style/Theme.Translucent.NoTitleBar" >
            </activity>
    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

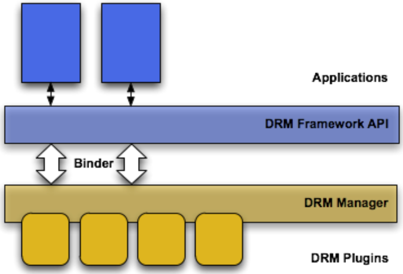


Taken from Android Open Source website.

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

Comparision between security measures employed in iPhone and Android-powered Smartphones (1)

- Secure Boot Chain
  - iPhone secure boot chain extends to apps whereas in case of Android-powered smarthpones, at max, it extends to kernel and system apps.
  - Moreover, in Android-powered smartphones, it depends on device manufacturer. For example, Google's nexus series smartphones have unlocked bootloader and any custom-prepared system image can be flashed whereas other manufacturers (Samsung, HTC etc.) sell their devices in conjuction with cellular service providers and bootloader is generally locked to prevent user to bypass the restrictions put by them.

Comparision between security measures employed in iPhone and Android-powered Smartphones (2)

- Data protection features
  - In iPhone flash storage, all user data always (by default) remains encrypted and is, at least, protected by UID whereas encryption is not on by default in Android.
  - iPhone prevents user data in different protection classes to balance security and availabilty. Data in certain protection classes isn't accessible when the screen is passcode locked whereas Android decrypts whole user partition first time it boots up by the passcode provided by the user.
  - In Android, all or part of the user data can be accessed by adb depending on the fact that it is "Rooted" or not even when the device is lost with screen lock. It's not the case with iPhone.
  - It's more complex to hack/bypass the data protection feature in iPhone by apps running as privileged user as it makes use of both hardware and software stack of the device.

Comparision between security measures employed in iPhone and Android-powered Smartphones (3)

- App Sandboxing
  - In Android, each app runs in a separate process with different user id (except the apps from the same developer if the developer wants to and specifies it in the app's manifest file) whereas all iPhone apps runs in different process with same unprivileged user id (user called mobile).
  - In iPhone, sandboxd daemon installs a different sandbox profile at the start of each process by implementing a policy module for TrustedBSD mandatory access control (MAC) framework.
- App access to standard APIs
  - In Android, app requested permissions are allowed/denied at install time whereas iPhone prompts the user to allow/deny access to private data at runtime. Additionally, in case of iPhone, application verification is done by Apple after developers request their apps to be put on Apple App store. However, it's not public how the verification is done.

Comparision between security measures employed in iPhone and Android-powered Smartphones (4)

- System upgrades
  - iPhone system software upgrades prevents use to downgrade by making use of device-specific upgrade. It's possible to downgrade Android in case of side-loaded upgrade if one is available.
  - OTA upgrades must be secured in both systems ; however, Android OTA upgarde is not documented.
- What if you lose Android-powered smartphone or iPhone ?
  - If device is passcode locked : Android keeps the decryption key used to encrypt the user data in the memory till the device is not powered off whereas iPhone wipes it out from the memory 10 seconds after user locks the phone rendering the data inaccessible in certain protection classes (classes which are protected by user passcode !)
  - If flash memory is taken out of the device to read the data : iPhone non-volative flash memory data won't be accessible in any case if someone takes it out and tries to read the data whereas on Android, it won't be accessible only if the user has explicitely switched on the encryption feature.

Outline

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.

Modifying the default software stack of iPhone (2)

Jailbreaking (Behind the scene) Reference : iOS Hacker's Handbook

Case of A4 Soc devices (iPhone 4, 3GS, iPad 1) having Bootrom exploit !

- *Step 1. Exploiting the BootROM :*
  - The vulnerability exploited is a heap-based buffer overflow in the USB DFU stack of the bootrom.
- *Step 2. Booting the custom ramdisk :*
  - When the ramdisk is booted, the kernel executes modified /sbin/launchd.
  - It mounts both partitions as readable and writable.
  - Eventually, an executable called *jailbreak* takes over and perfoms all of the following steps.
- *Step 3. Jailbreaking the FileSystem :*
  - To survive reboot and add additional services, one requires to modify the root filesystem.
  - (Re)mounting the root filesystem with read and write permission is usually the first step after acquiring root permissions.
  - To persist these changes across reboots, /etc/fstab is changed to mount system partition as readable and writable

Modifying the default software stack of iPhone (3)

Jailbreaking (Behind the scene) Contd...

- *Step 4. Installing the AFC2 service :*
  - The Apple File Connection (AFC) is a file transfer service.
  - lockdownd daemon (A daemon is a program which keeps running in the background !) provides two services named com.apple.afc and com.apple.crashreportcopymobile to access /var/mobile/Media and /var/mobile/Library/Logs/CrashReporter via USB.
  - com.apple.afc2 service is installed by adding the below lines in the lockdownd configuration file (/System/Library/Lockdown/Services.plist).

```
<key>com.apple.afc2</key>
<dict>
   <key>AllowUnactivatedService</key>
   <true />
   <key>Label</key>
   <string>com.apple.afc2</string>
   <key>ProgramArguments</key>
   <array>
      <string>/usr/libexec/afcd</string>
      <string>--lockdown</string>
      <string>-d</string>
      <string>/</string>
   </array>
</dict>
```

Modifying the default software stack of iPhone (4)

Jailbreaking (Behind the scene) Contd...

- *Step 5. Installing base utilities :*
  - Apple doesn't ship the iPhone with a UNIX shell.
  - It comes with launchctl executable in /bin/ directory and awd_ice3, DumpBasebandCrash, powerlog and simulatecrash executables in /usr/bin/ directory.
  - So, all these basic utility tools like mv, cp, tar, gzip, gunzip, ldid etc. are installed.
- *Step 6. Application Stashing and Bundle Installation :*
  - A new directory /var/stash is created on the data partition and some directories like /Applications, /Library/Ringtones, /Library/Wallpaper are moved to /var/stash directory from root filesystem.
  - The moved directories from root filesystsem are then repalced by symbolic links to the new location in /var/stash.
  - In the end, bundles (tar archives) like Cydia are unpacked into the /Applications directory and registered in a systemwide installation cache stored in /var/mobile/Library/Caches/com.apple.mobile.installation.plist.

Security implications of Jailbreaking (1)

- If your phone is lost/stolen, think of AFC2 service installed during Jailbreaking !
  - Someone can just copy all your data !
  - and additionally, 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 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 (implicitly requires knowing all the internals !)

Security implications of Jailbreaking (2)

DEMO : An example of compromised keychain on a jailbreaked phone

- A process can get access to all keychain items stored on the device.
- That process requires com.apple.keystore.access-keychain-keys and com.apple.keystore.device entitlement.
- Accessing keychain-2.db directly doesn't reveal stored items keys ; AppleKeyStore KeyUnwrap selector must be called to get stored items keys
- Process must run as root

Modifying the default software stack of Android-powered Smartphones

In Android-powered smartphones, software stack is normally modified to get "Root" (privileged user) access to the phone and is known as "Rooting" the phone.

- 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, for example, removing cellular restrictions, evaluating the security and performing some malicious activities.
- How do you "Root" Android-powered smartphone ?
  - If your device has an unlocked bootloader
  - "ro.secure" system property value. The value of this property is set at boot time from default.prop file in the root directory.
  - And hacking one of system process running in privileged mode e.g. z4root, gingerbreak...to execute arbitrary code (Well, the arbitrary code is normally mounts /system in read-write mode and installs su command.)

Security implications of "Rooting"

- If "Rooted" Android-powered smartphone is lost/stolen, ALL user data on the device is at risk if adb access is enabled. Even if Android encryption feature is ON !
- "Rooting" normally involves flashing custom ROM and few custom ROM builders had even removed Encryption option from the device ! 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.

Outline

What are we doing at present ?

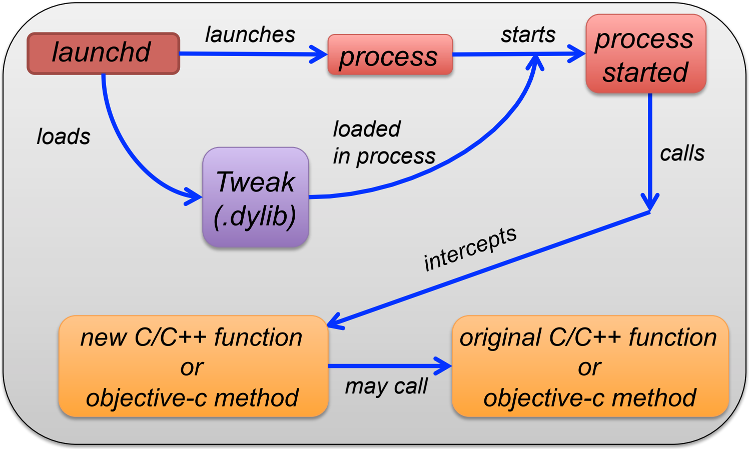
- We are developing a logger for iOS and Android capable of monitoring activities all across the device especially to track access to user's private information by Apps.
  - iOS Logger makes use of objective-c runtime and trampoline to dynamically analyze the application's behaviour. It runs on iPhones having modified software stack with privileged user (root) access and disabled signature verification of code by kernel.
  - Android logger modifies the Android framework source code to intercept the calls made by Apps to standard APIs. (Development in progress...late started !)

iOS Logger (1)

- Our logger logs access to personal data
  - Contacts
  - Geographic location
  - various device and user accounts
  - Calendar
  - Photos and Videos
  - UDID and Device Name
  - Voice memos etc.
- It also logs various user related activities and events (also works as SPYWARE)
  - Phone calls (Incoming and outgoing)
  - Sending and receiving SMS, iMessage and Emails
  - Internet navigation using Safari/Youtube apps (Also, all raw network data BSD_SOCKETS ! Every single byte...)
  - Taking a picture, start/stop video capturing etc.

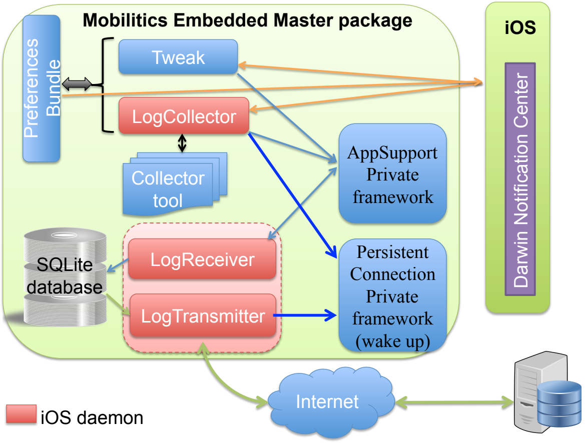
iOS Logger (2)

- How we are doing runtime dynamic analysis ?
- iOS Tweak (Trampoline + objective-c runtime)



iOS Logger Design

Three tools + Tweak (a dylib loaded at the start of each process and replaces our implemented methods/functions with the original ones) + Preference bundle


Mobilitics Embedded Master package
Preferences Bundle
Tweak
LogCollector
Collector tool
AppSupport Private framework
SQLite database
LogReceiver
LogTransmitter
Persistent Connection Private framework (wake up)
iOS
Darwin Notification Center
Internet
iOS daemon

Questions/Discussion/Comments/Feedback

- Any questions or something you want to discuss ?
- Comments/feedback on the lecture ?
- I recently got interested in Jailbreaking activity itself ; not merely using it. If someone is interested ? My Email : jagdish.achara@inria.fr

THANKS