Identifying Back Doors, Attack Points, and Surveillance Mechanisms in iOS Devices

JONATHAN ZDZIARSKI
JONATHAN@ZDZIARSKI.COM

@JZDZIARSKI

Whois NerveGas

- Worked as dev-team member on many of the early jailbreaks until around iOS 4.
- Author of five iOS-related O'Reilly books including "Hacking and Securing iOS Applications"
- Designed all of the iOS forensics techniques used in law enforcement and commercial products today
- Consulted closely with federal and local law enforcement agencies and US military on high profile projects and criminal cases
- Trained law enforcement worldwide in iOS forensics and penetration arts

iOS Operating System

- Subject of interest among forensics, law enforcement, and criminal communities
- As leaked by Der Spiegel, iOS was targeted by NSA for targeted collection
- Later found more evidence of C&C capabilities in DROPOUTJEEP leaks via **close access methods**
- Attacked for everything from cases of national security to nude photos of marginally attractive celebrities
- A number of forensic techniques exist to acquire data

What This Talk Is

- Overview of a number of undocumented high-value forensic services running on every iOS device
  - How they've evolved
  - What kind of data they provide
- Examples of forensic artifacts acquired that should never come off the device without user consent
- Surveillance mechanisms to bypass personal security (intended for enterprises), but make potential targets
- Suspicious design omissions in iOS that make collection easier

What This Talk Is NOT

- A talk about fun 0days and how we can have a little temporary fun with them for a few days.
  - The content discussed here has been around for many years, and are low level operating system components
  - Apple is well aware of these components, and has clearly been updating them and supporting them for reasons unknown
  - I have emailed both Tim Cook and Steve Jobs at various times to ask for an explanation about these services, citing them as "back doors", and have received no reply
  - I *have* received replies from Tim Cook about Apple's crummy warranty service, so I know he gets my email

Centralized Control

- Apple has worked hard to make iOS devices reasonably secure against typical attackers
- Apple has worked hard to ensure that Apple can access data on end-user devices on behalf of law enforcement
- To their credit, iPhone 5* + iOS 7 is more secure from everybody except Apple (and .gov)
- Apple's Law Enforcement Process Guidelines:
  - https://www.apple.com/legal/more-resources/law-enforcement/

Law Enforcement Process

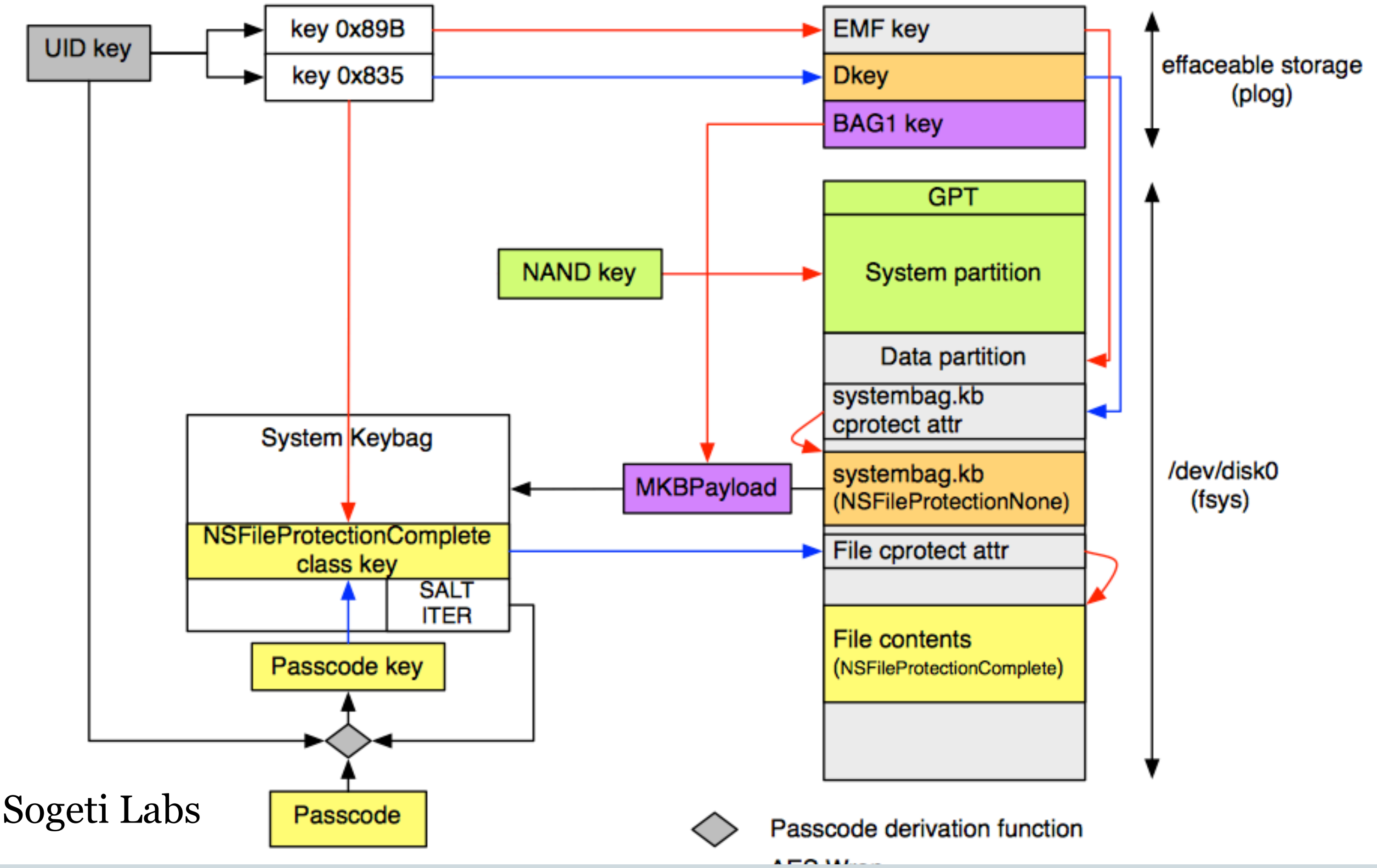
- Requires a warrant for actual content from iCloud, iTunes, or from the device itself
- A subpoena appears good enough for "metadata"
- Recent changes will notify all customers unless a confidentiality order is included; so most agencies are now getting confidentiality orders with every warrant.
- When provided with the physical device, Apple will retrieve and return NSProtectionNone data from passcode locked devices; rumors of a PIN brute forcer are floating around, but I'm told this practice stopped around iOS 5.
- Process is now taking about four months on average, and costs about $1,000, so LE is looking for streamlined / inexpensive tools to collect evidence.

Apple Law Enforcement Process

Extracting Data from Passcode Locked iOS Devices

Upon receipt of a valid search warrant, **Apple can extract certain categories of active data from passcode locked iOS devices**. Specifically, the **user generated active files** on an iOS device that are contained in Apple's native apps and for which the **data is not encrypted using the passcode** ("user generated active files"), can be extracted and provided to law enforcement on external media. Apple can perform this data extraction process on iOS devices running iOS 4 or more recent versions of iOS. Please note the only categories of user generated active files that can be provided to law enforcement, pursuant to a valid search warrant, are**: SMS, photos, videos, contacts, audio recording, and call history**. Apple cannot provide: email, calendar entries, or any third-party App data.

iOS 4 Storage Encryption Overview


UID key
key 0x89B
key 0x835
EMF key
Dkey
BAG1 key
effaceable storage (plog)
GPT
System partition
NAND key
Data partition
systembag.kb cprotect attr
System Keybag
MKBPayload
systembag.kb (NSFileProtectionNone)
NSFileProtectionComplete class key
File cprotect attr
SALT ITER
/dev/disk0 (fsys)
File contents (NSFileProtectionComplete)
Passcode key
Passcode
Passcode derivation function

Courtesy of Sogeti Labs

Encryption in iOS 7: Not Much Changed

- Almost all native application / OS data is encrypted with a key **not married to the passcode**, but rather encrypted with a **hardware deduced key (NSProtectionNone)**
- As of iOS 7, third party documents are encrypted, **but Library and Caches folders are usually not**
- Once the device is **first unlocked** after reboot, most of the data-protection encrypted data can be accessed until the device is shut down
  - Screen Lock != Encrypted
- The undocumented services running on every iOS device help make this possible
- Your device is almost always at risk of spilling **all** data, since it's almost always authenticated, even while locked.

Law Enforcement Technologies

- Latest commercial forensics tools perform deep extraction using these services
- Tablet forensics in the field can acquire a device at a routine traffic stop, or during arrest – before device can be shut down (leaving encryption unlocked)
- Federal agencies have always been interested in black bag techniques (compromised docking stations, alarm clocks, etc).
- Snowden Docs: Computer infiltration was used

Undocumented Services

- Accessed through *lockdownd*, requiring pairing authentication. (Explain Pairing)
- MACTANS talk demonstrated how easy Juice Jacking can be to establish pairing
  - iOS 7 trust dialog helps, but third party accessories are making people stupid again … and people are naturally stupid too
- Law enforcement agencies moving to tablet devices for pairing and acquisition in the field; USB thumb drive to scan computers for pairing records
- Der Spiegel outlined black bag techniques to access a target’s computer, where pairing records live

Der Spiegel

- “The documents state that it is possible for the NSA to tap most sensitive data held on these smart phones, including **contact lists**, **SMS traffic**, **notes** and **location information** about where a user has been. In the internal documents, experts boast about successful access to iPhone data in instances where the NSA is able to **infiltrate the computer a person uses to sync their iPhone**. Mini-programs, so-called "scripts," then enable additional access to at least 38 iPhone **features**.”

Undocumented Services

- Bypasses "Backup Encryption" mechanism provided to users
- Can be accessed both via USB and wirelessly (WiFi, maybe cellular); networks can be scanned for a specific target
- If device has not been rebooted since user last entered PIN, can access all data encrypted with **data-protection** (third party app data, etc)
- Other (more legitimate) services enable software installation, APN installation (adding proxy servers) for continued monitoring

Undocumented Services

- Most services are not referenced by any known Apple software (we’ve looked)
- The raw format of the data makes it impossible to put data back onto the phone, making useless for Genius Bar or carrier tech purposes (cpio.gz, etc)
- The personal nature of the data makes it very unlikely as a debugging mechanism
- Bypassing backup encryption is deceptive
- Services are available **without** developer mode, eliminating their purpose as developer tools

DROPOUTJEEP

- DROPOUTJEEP describes techniques, most of which are possible with Apple's undocumented services
- SMS messaging suggests either jailbreak or baseband code

DROPOUTJEEP

(TS//SI//REL) DROPOUTJEEP is a STRAITBIZARRE based software implant for the Apple iPhone operating system and uses the CHIMNEYPOOL framework. DROPOUTJEEP is compliant with the FREEFLOW project, therefore it is supported in the TURBULENCE architecture.

(TS//SI//REL) DROPOUTJEEP is a software implant for the Apple iPhone that utilizes modular mission applications to provide specific SIGINT functionality. This functionality includes **the ability to remotely push/pull files from the device**, **SMS retrieval, contact list retrieval, voicemail**, **geolocation**, hot mic, camera capture**, cell tower location**, etc. Command, control, and data exfiltration can occur over SMS messaging or a **GPRS data connection**. All communications with the implant will be covert and encrypted.

(TS//SI//REL) The initial release of DROPOUTJEEP will focus on **installing the implant via close access methods**. A remote installation capability will be pursued for a future release.

Starting Services

- Connect to lockdownd (tcp:62078) via usbmux or TCP
- Authenticate with intercepted / generated pairing record
- Invoke “StartService” command with name of the service we wish to start
- Profit*

- * A number of commercial law enforcement forensic manufacturers have started tapping these services:
  - Cellebrite
  - AccessData (Mobile Phone Examiner)
  - Elcomsoft

Open Source!

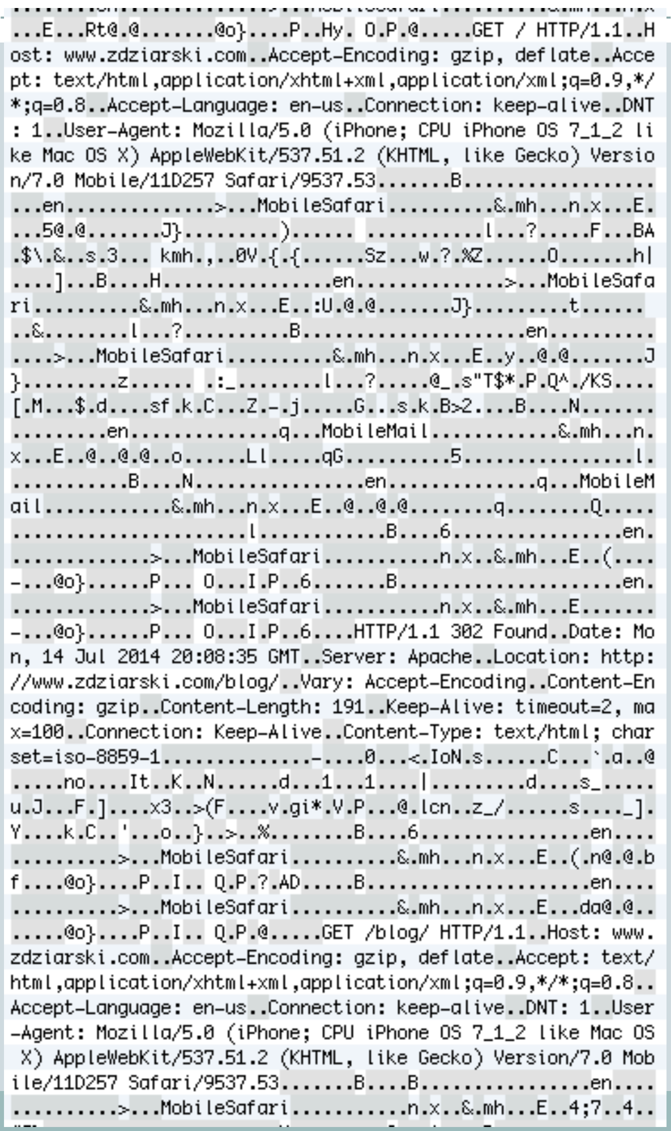
- Nearly all *lockdownd* protocols have been documented in the libimobiledevice project (libimobiledevice.org).
- Been around since 2009 but many of these services haven't been re-examined since then; initially benign
- A number of private tools and source are out there as well to take advantage of these services

com.apple.pcapd

- Immediately starts libpcap on the device
- Dumps network traffic and HTTP request/response data traveling into and out of the device
- **Does not** require developer mode; is active on every iOS device
- Can be targeted via WiFi for remote monitoring
- No visual indication to the user that the packet sniffer is running.

WHY DO WE NEED A PACKET SNIFFER RUNNING ON 600 MILLION PERSONAL IOS DEVICES?

com.apple.pcapd



```
...E...Rt@.@.......@o}....P..Hy. O.P.@.....GET / HTTP/1.1..H
ost: www.zdziarski.com..Accept-Encoding: gzip, deflate..Acce
pt: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8..Accept-Language: en-us..Connection: keep-alive..DNT
: 1..User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_2 li
ke Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Versio
n/7.0 Mobile/11D257 Safari/9537.53.......B..................
...en..............>...MobileSafari..........&.mh...n.x...E.
...5@.@.......J}.........)...... ...........l...?.....F...BA
.$\.&..s.3... kmh.,..0V.{.{......Sz...w.?.%Z......O.......h|
....]...B....H................en..............>...MobileSafa
ri..........&.mh...n.x...E..:U.@.@.......J}.........t......
..&.........l...?..........B.......................en.........
....>...MobileSafari..........&.mh...n.x...E..y..@.@.......J
}.........z...... .:_.........l...?.....@_.s"T$*.P.Q^./KS....
[.M...$.d....sf.k.C...Z.-.j.....G...s.k.B>2....B....N.......
.........en...............q...MobileMail............&.mh...n.
x...E..@..@.@..o......Ll.....qG..........5.................l.
..........B....N................en..............q...MobileM
ail............&.mh...n.x...E..@..@.@........q.........Q.....
......................l............B....6.................en.
.............>...MobileSafari...........n.x..&.mh...E..(....
-...@o}......P... O...I.P..6.......B......................en.
.............>...MobileSafari...........n.x..&.mh...E.......
-...@o}......P... O...I.P..6....HTTP/1.1 302 Found..Date: Mo
n, 14 Jul 2014 20:08:35 GMT..Server: Apache..Location: http:
//www.zdziarski.com/blog/..Vary: Accept-Encoding..Content-En
coding: gzip..Content-Length: 191..Keep-Alive: timeout=2, ma
x=100..Connection: Keep-Alive..Content-Type: text/html; char
set=iso-8859-1..............-....0...<.IoN.s......C...`.a..@
.....no....It..K..N......d...1...1....|.........d....s_.....
u.J...F.]....x3..>(F....v.gi*.V.P...@.lcn..z_/......s...._].
Y....k.C..'...o..}..>..%........B....6.................en....
..........>...MobileSafari..........&.mh...n.x...E..(.n@.@.b
f....@o}....P..I.. Q.P.?.AD.....B......................en....
..........>...MobileSafari..........&.mh...n.x...E...da@.@..
.....@o}....P..I.. Q.P.@.....GET /blog/ HTTP/1.1..Host: www.
zdziarski.com..Accept-Encoding: gzip, deflate..Accept: text/
html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8..
Accept-Language: en-us..Connection: keep-alive..DNT: 1..User
-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_2 like Mac OS
 X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mob
ile/11D257 Safari/9537.53.......B....B.................en....
..........>...MobileSafari...........n.x..&.mh...E..4;7..4..
```

Example from iOS 7.1.2

Developer Mode NOT turned on

Packet sniffing now available on 600 million iOS devices ☺

com.apple.mobile.file_relay

- Biggest forensic trove of intelligence on the device
- Found in /usr/libexec/mobile_file_relay on device
- Provides *physical* artifacts vs. *logical* (databases; deleted records can be recovered)
- Transmits large swaths of raw file data in a compressed cpio archive, based on the data source requested.
- Completely **bypasses Apple's backup encryption** for end-user security.
- Once thought benign, has evolved considerably, even in iOS 7, to expose much personal data.
- Very intentionally placed and intended to dump data from the device by request

com.apple.mobile.file_relay

- File Relay sources in iOS v2:

  AppleSupport
  Network
  WiFi
  UserDatabases
  CrashReporter
  SystemConfiguration

com.apple.mobile.file_relay

- File Relay sources in iOS 7:

Accounts
AddressBook
AppleSupport
AppleTV
Baseband
Bluetooth
CrashReporter
CLTM
Caches
CoreLocation
DataAccess
DataMigrator
demod
Device-o-Matic

EmbeddedSocial
FindMyiPhone
GameKitLogs
itunesstored
IORegUSBDevice
HFSMeta
Keyboard
Lockdown
MapsLogs
MobileAsset
MobileBackup
MobileCal
MobileDelete
MobileInstallation

MobileMusicPlayer
MobileNotes
NANDDebugInfo
Network
Photos
SafeHarbor
SystemConfiguration
tmp
Ubiquity
UserDatabases
VARFS
VPN
Voicemail
WiFi
WirelessAutomation

com.apple.mobile.file_relay

- Accounts A list of email, Twitter, iCloud, Facebook etc. accounts configured on the device.
- AddressBook A copy of the user's address book SQLite database; deleted records recoverable.
- Caches The user cache folder: suspend screenshots (last thing you were looking at), shared images, offline content, clipboard/pasteboard, map tile images, keyboard typing cache, other personal data

com.apple.mobile.file_relay

- CoreLocation GPS logs; cache of locations taken at frequent intervals (com.apple.routined)
  - fileslockCache_encryptedA.db and cache_encryptedA.db
  - Similar to the old consolidated.db database from iOS 4
  - Timestamps span ~60 days on my phone

com.apple.mobile.file_relay

- HFSMeta (New in iOS 7!) A complete metadata disk sparseimage of the iOS file system, sans actual content.
  - Timestamps, filenames, sizes, creation dates of **all** files
  - When device was last activated / wiped
  - All applications installed on a device and filenames of all documents (e.g. Dropbox documents, etc)
  - The filenames of all email attachments on the device
  - All email accounts configured on a device
  - Host IDs and timestamps of all devices paired with the device
  - Phone numbers and timestamps of everyone for whom an SMS draft was saved
  - Timeline of activity based on timestamp data

com.apple.mobile.file_relay

- Keyboard A copy of the keyboard autocorrect cache
  - DynamicDictionary-4: First half contains all recent typed content from all applications, consolidated and in the order it was typed
  - DynamicDictionary-5: Improved, contains words and word counts only
- MobileCal, MobileNotes Complete database images of the user's calendar, alarms, and notes databases in SQLite format (deleted records recoverable).
- Photos Complete dump of user's photo album (not just camera roll) stored on the device

com.apple.mobile.file_relay

- UserDatabases (Been around since v2) dump of address book, calendar, call history, SMS database, email metadata (envelope index); SQLite databases (deleted records recoverable)
- VARFS (predecessor to HFSMeta) virtual file system metadata dump in statvfs format.
- Voicemail Copy of user's voicemail database **and audio files** (AMR format)

com.apple.mobile.house_arrest

- Originally used to allow iTunes to copy documents to/from third party applications
- Even though iTunes doesn’t permit it through GUI, the service allows access to the Library, Caches, Cookies, Preferences folders as well
- These folders provide highly sensitive account storage, social/Facebook caches, photos and other data stored in “vaults”, and much more.

Example: Twitter

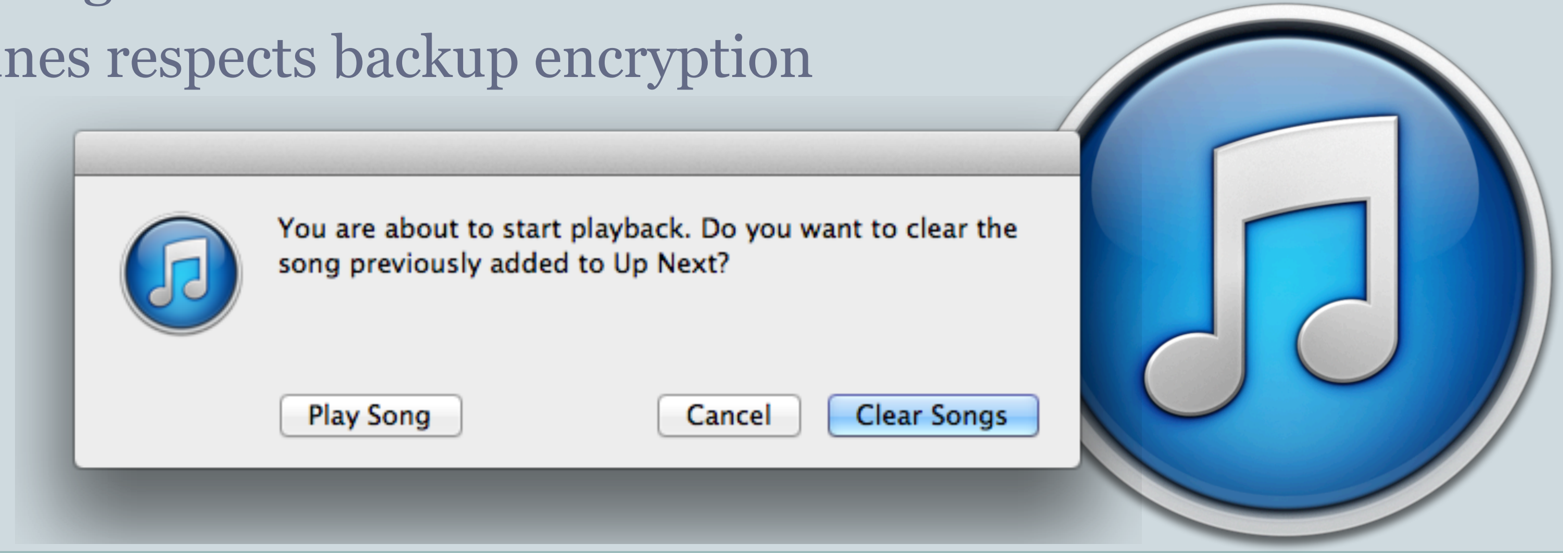
- Recent photos from my stream
- Most recent timeline
- Private message database; numerous deleted messages recovered
- Screenshots of my last use of Twitter
- OAuth tokens (when combined with consumer key/ secret, can be used to spy on all future correspondence remotely)

Example: Photo Vaults

- Copies of the actual photos the vaults are “protecting”
- Configuration files including the PIN, or a hash of the PIN
- Occasionally, developer will actually encrypt files
- Sometimes encryption keys or PIN dumped to syslog

Theories

- Maybe iTunes or Xcode use them? No.
  - iTunes uses com.apple.mobilesync, backup2, and other facilities, but none use file relay or pcap
  - iTunes uses house_arrest, but only for accessing Documents; there's no need to allow access to Library, Cache, or other privileged folders
  - iTunes respects backup encryption

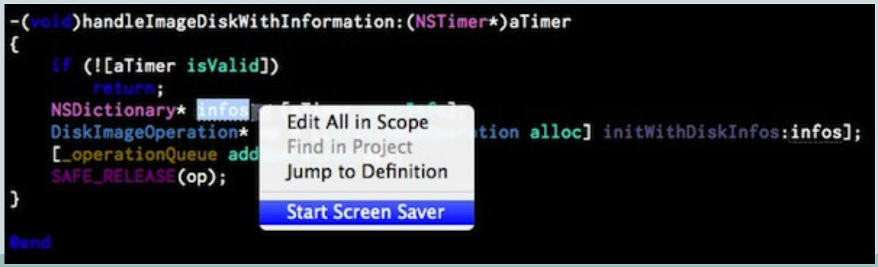

You are about to start playback. Do you want to clear the song previously added to Up Next?

Theories

- Maybe for Genius Bar or Apple Support? No.
  - Data is in too raw a format to be used for tech support
  - Can't be put back onto the phone in any way
  - Tech support use shouldn't call for bypassing backup password
  - Data is far too personal in nature for mere tech support

Theories

- Maybe for Developers for Debugging? No.
  - Actual developer tools live on the developer image, and are only available when Developer Mode is enabled
  - Xcode does not provide a packet sniffing interface for developers
  - Developers don't need to bypass backup encryption
  - Developers don't need access to such sensitive content
  - Apple wants developers to use the SDK APIs to get data
  - There are no docs to tell developers about these "features"



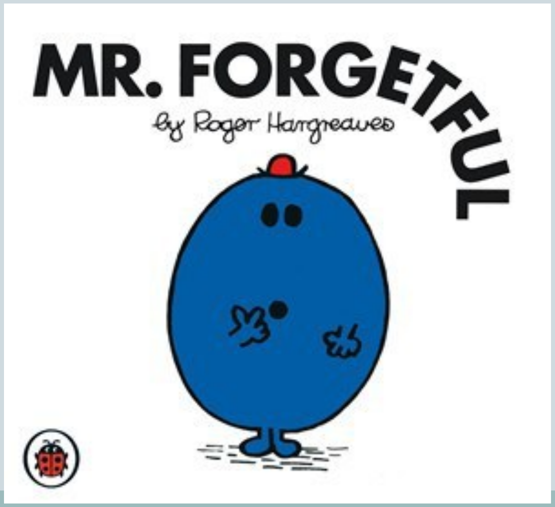
```
-(void)handleImageDiskWithInformation:(NSTimer*)aTimer
{
    if (![aTimer isValid])
        return;
    NSDictionary* infos
    DiskImageOperation*                    tion alloc] initWithDiskInfos:infos];
    [_operationQueue add
    SAFE_RELEASE(op);
}

@end
```

Theories

- Maybe for Engineering / Debugging? No.
  - Not all 600 million devices need debugging always on
  - By preventing localhost connections, Apple must know these services are being abused by malware
  - You still wouldn't need to bypass backup encryption
  - Engineering wouldn't need access to such personal data

Theories

- Maybe old debug code they forgot was in there? No.
  - Apple has been maintaining and enhancing this code, even with iOS 7; they know it's there
  - Have emailed Apple's CEOs and gotten no response
  - It's not buried; it's listed in Services.plist
  - While house_arrest security issues might be "bugs", file relay and pcap most certainly aren't



The More Benign Services

- While more benign, the following services are good attack targets for forensic artifacts:
- com.apple.iosdiagnostics.relay Provides detailed network usage per-application on a per-day basis
- com.apple.mobile.installation_proxy Given an enterprise certificate, can use this to load custom software onto the device (which can run invisibly and in the background)
- com.apple.syslog_relay: Syslog, provides a lot of details about what the device is doing, and often leaks user credentials from 3rd party apps via NSLog()

Invisible Malware

- Installing invisible software that backgrounds is still easy to do in iOS 7
- Apple made a crucial security improvement in iOS 7: prevented socket connections to localhost / local IP
  - Prior to this, I had spyware running invisibly that could dump a phone and send its contents remotely anywhere. (never released for obvious reasons)
- This stopped a number of privately used spyware apps in their tracks; they can not connect to localhost:62078
- Future spyware: phones attacking other phones on the network (zomg zombies)

Invisible Malware

- Info.plist:

```
<key>SBAppTags</key>
<array>
<string>hidden</string>
</array>

<key>UIBackgroundModes</key>
<array>
<string>voip</string>
</array>
```

Backgrounding Malware

```
[ [ UIApplication sharedApplication ]
setKeepAliveTimeout: 600 handler:^(void)
{
    /* Do bad things in background */
}
```

In iOS 7, you can still capture:

- All socket connections (netstat data)
- Process information (ps data)
- A number of personal files on the device
- Launch some very closely-held-to-the-vest userland exploits

But Wait. I paid $600 for a Fingerprint Reader

- Fingerprint reader: Doesn’t add any additional encryption beyond basic PIN
- Has shown to be spoofed with the right equipment
- Allows GUI access, therefore allowing pairing, therefore allowing forensic dumps

- Oh, and… there’s a bypass switch for pairing anyway

Pairing Bypass

- Added for supervised devices to be accessible (e.g. employee dies, leaves on bad terms, criminal investigation).
- Devices try to call home when first configured to download automatic configurator profile. (likely used for large-scale MDM rollouts).
- An electronic alternative to interdiction could be deployed by spoofing Apple's certificates and configuring / pairing the device out of the box.
- OR by penetrating a targeted organization, supervisor records can be used to pair with and access **any** device they're supervising.

MCCloudConfiguration

- Deny all pairing
- Allow pairing, but prompt the user
- Allow pairing with no user prompt (and while locked)
- Allow pairing with a challenge/response

Pairing Bypass

```
               ; Check -[ MCProfileConnection hostMayPairWithOptions:challenge: ]
__text:0001938E                 LDR.W           R0, [R8,#0xC]
__text:00019392                 BL              sub_5754
__text:00019396                 CMP             R0, #0
__text:00019398                 BNE.W           loc_19AA8
__text:0001939C                 LDR.W           R1, [R8,#0x1C]
__text:000193A0                 ADD             R2, SP, #0x7E8+var_420
__text:000193A2                 ADD             R3, SP, #0x7E8+out
__text:000193A4                 MOV             R0, R4
__text:000193A6                 BL              sub_1F100

               ; Pairing is explicitly forbidden by MC
__text:000193AA                 CMP             R0, #0
__text:000193AC                 BEQ.W           loc_19AB0

               ; Pairing is allowed by MC, but with challenge/response
__text:000193B0                 LDRB.W          R0, [SP,#0x7E8+out]
__text:000193B4                 CMP             R0, #0
__text:000193B6                 BNE.W           loc_19AC2


               ; Pairing is allowed by MC while locked / untrusted without
               ; any challenge/response (pairing security is bypassed)

__text:000193BA                 LDRB.W          R0, [SP,#0x7E8+var_420]    <- Profit
__text:000193BE                 CMP             R0, #0
__text:000193C0                 BNE.W           loc_19B06

~~               ; Pairing is allowed while locked / untrusted if the device~~
~~               ; doesn’t support it~~
__text:000193C4                 MOV             R0, #(cfstr_Hasspringboa_1 - 0x193D0) ; "HasSpringBoard"
__text:000193CC                 ADD             R0, PC  ; "HasSpringBoard"
__text:000193CE                 BLX             _MGGetBoolAnswer
__text:000193D2                 CMP             R0, #1
__text:000193D4                 BNE.W           loc_19B06

               ; Actual pairing security routines (check device lock, whether
               ; user has pressed “Trust”, and so on)

__text:000193D8                 MOVS            R0, #0
__text:000193DA                 BLX             _MKBGetDeviceLockState
```

In Pseudocode

```
if (mc_allows_pairing_while_locked || device_has_no_springboard_gui)
{
    goto **skip_device_lock_and_trust_checks**; /* Skip security */
}

/* Pairing Security */

if (device_is_locked == true) {
    if (**setup_has_completed**) {
        if (user_never_pushed_trust) {
            error(PasswordProtected);
        }
    }
}
```

Calling Home

- On setup, teslad connects to https://iprofiles.apple.com
  - /resource/certificate.cer
  - /session and /profile
  - Capable of downloading MCCloudConfiguration
- Could be used for electronic interdiction, either with technology or secret FISA order
- MCCloudConfiguration affects pairing bypass
- Build in mechanism to bypass SSL validation. WTF.
  - MCTeslaConfigurationFetcher checks for MCCloudConfigAcceptAnyHTTPSCertificate

Calling Home

- Once configured, a new cloud configuration can be downloaded via periodic check-in
- -[MCProfileConnection retrieveCloudConfiguration FromURL:username:password:anchorCertificates: completionBlock:]
  - Great attack surface if you can get past the SSL
  - Not necessary if you have a secret FISA order

Questions for Apple

- Why is there a packet sniffer running on 600 million personal iOS devices instead of moved to the developer mount?
- Why are there undocumented services that bypass user backup encryption that dump mass amounts of personal data from the phone?
- Why is most of my user data *still* not encrypted with the PIN or passphrase, enabling the invasion of my personal privacy by YOU?
- Why is there still no mechanism to review the devices my iPhone is paired with, so I can delete ones that don't belong?

Pairing Locking

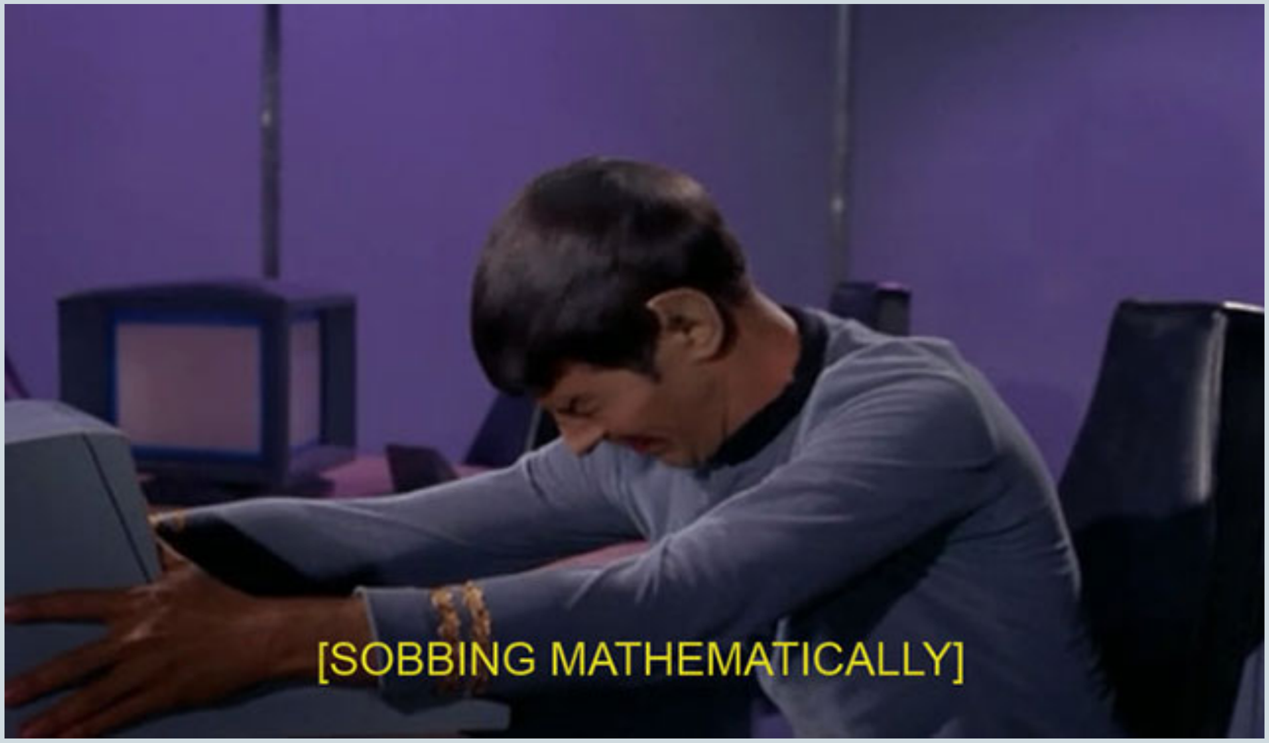
```
               ; Check -[ MCProfileConnection hostMayPairWithOptions:challenge: ]
__text:0001938E                 LDR.W           R0, [R8,#0xC]
__text:00019392                 BL              sub_5754
__text:00019396                 CMP             R0, #0
__text:00019398                 BNE.W           loc_19AA8
__text:0001939C                 LDR.W           R1, [R8,#0x1C]
__text:000193A0                 ADD             R2, SP, #0x7E8+var_420
__text:000193A2                 ADD             R3, SP, #0x7E8+out
__text:000193A4                 MOV             R0, R4
__text:000193A6                 BL              sub_1F100

               ; Pairing is explicitly forbidden by MC

__text:000193AA                 CMP             R0, #0     <- HOW DO WE MAKE THIS WORK?
__text:000193AC                 BEQ.W           loc_19AB0

               ; Pairing is allowed by MC, but with challenge/response
__text:000193B0                 LDRB.W          R0, [SP,#0x7E8+out]
__text:000193B4                 CMP             R0, #0
__text:000193B6                 BNE.W           loc_19AC2


               ; Pairing is allowed by MC while locked / untrusted without
               ; any challenge/response (pairing security is bypassed)
__text:000193BA                 LDRB.W          R0, [SP,#0x7E8+var_420]
__text:000193BE                 CMP             R0, #0
__text:000193C0                 BNE.W           loc_19B06

_______________; Pairing is allowed while locked / untrusted if the device
_______________; doesn't support it
__text:000193C4                 MOV             R0, #(cfstr_Hasspringboa_1 - 0x193D0) ; "HasSpringBoard"
__text:000193CC                 ADD             R0, PC  ; "HasSpringBoard"
__text:000193CE                 BLX             _MGGetBoolAnswer
__text:000193D2                 CMP             R0, #1
__text:000193D4                 BNE.W           loc_19B06

               ; Actual pairing security routines (check device lock, whether
               ; user has pressed "Trust", and so on)

__text:000193D8                 MOVS            R0, #0
__text:000193DA                 BLX             _MKBGetDeviceLockState
```

Apple Configurator

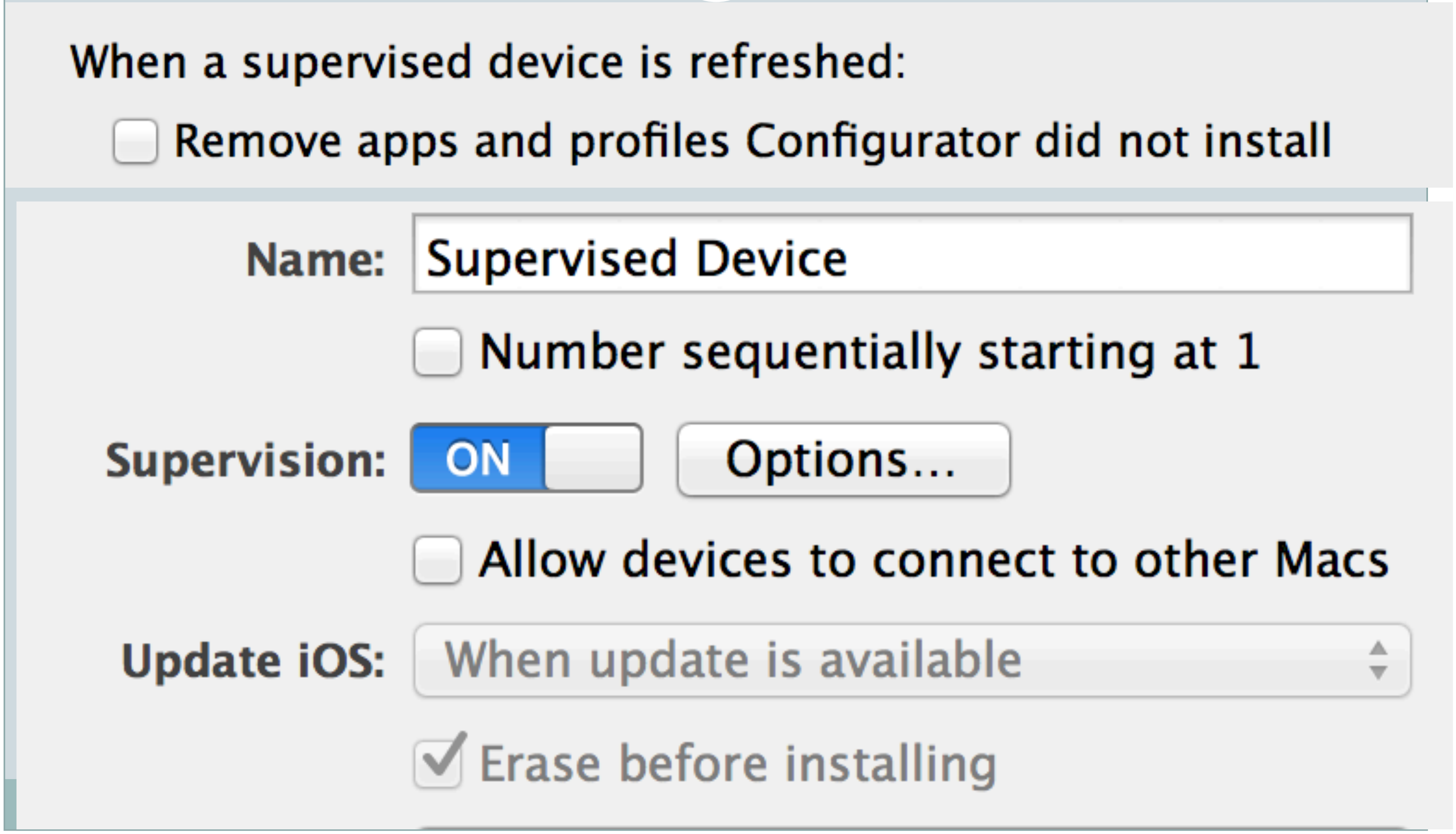
- Free in the Mac App Store
- Allows you to set enterprise MDM restrictions on your device
- Can be used to prevent pairing **even when unlocked**
- Pair once with your desktop, then never again… OR (if you're paranoid) delete all pairing records and prevent any comms.
- Won't help you if device sent to Apple; should still use a **complex passphrase**
- Removable later if you change your mind

Forensics Tools

- Every commercial forensics tool, after pair locking with Configurator:



Pair Locking with Configurator


When a supervised device is refreshed:

☐ Remove apps and profiles Configurator did not install

Name: Supervised Device

☐ Number sequentially starting at 1

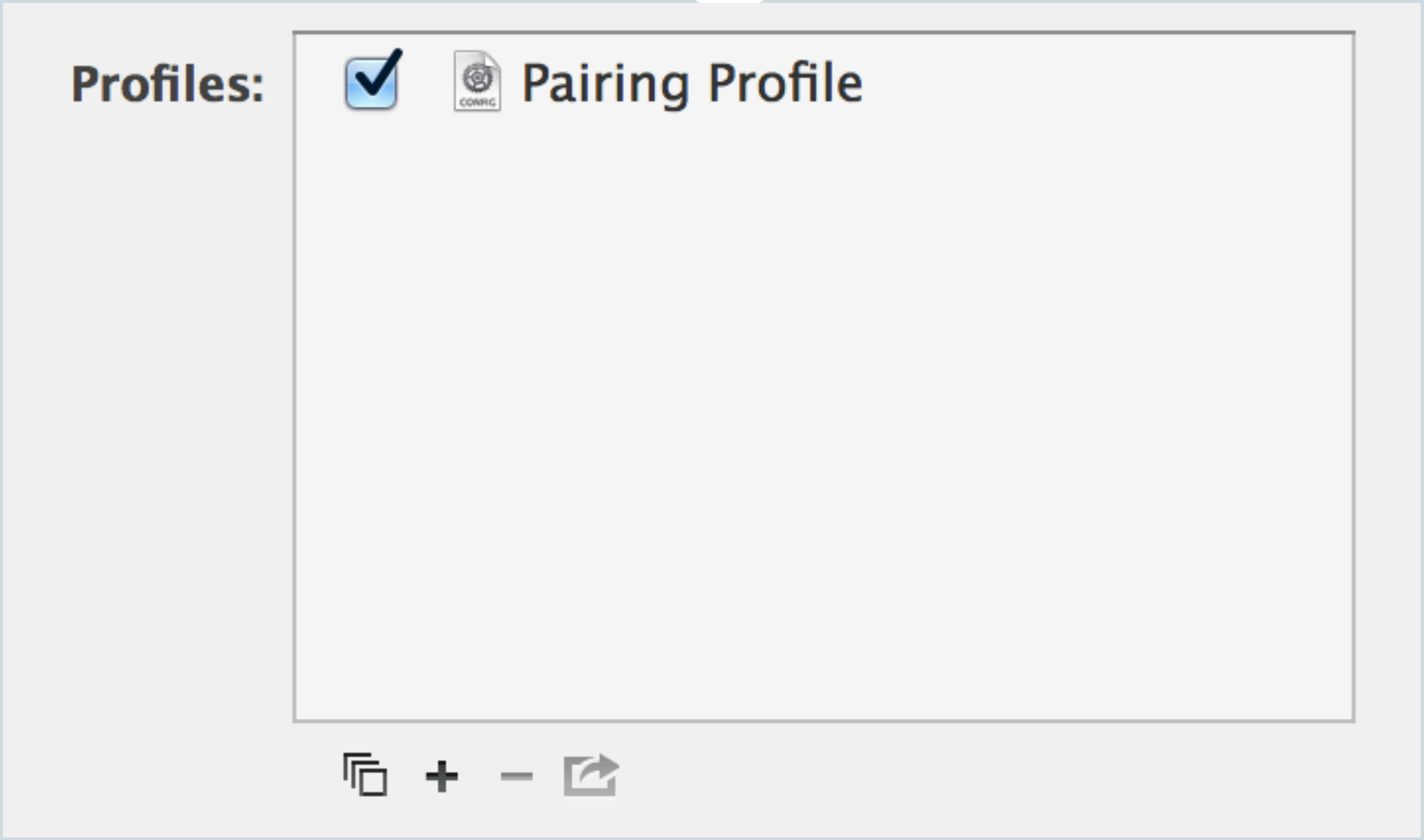
Supervision: ON

☐ Allow devices to connect to other Macs

Update iOS: When update is available

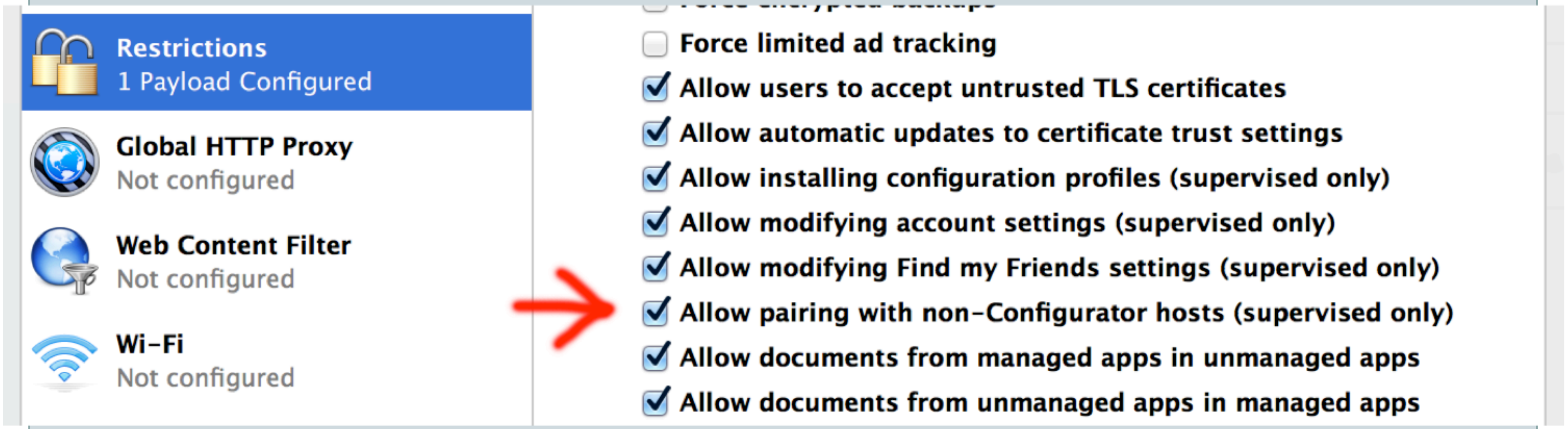
✓ Erase before installing

Pair Locking with Configurator



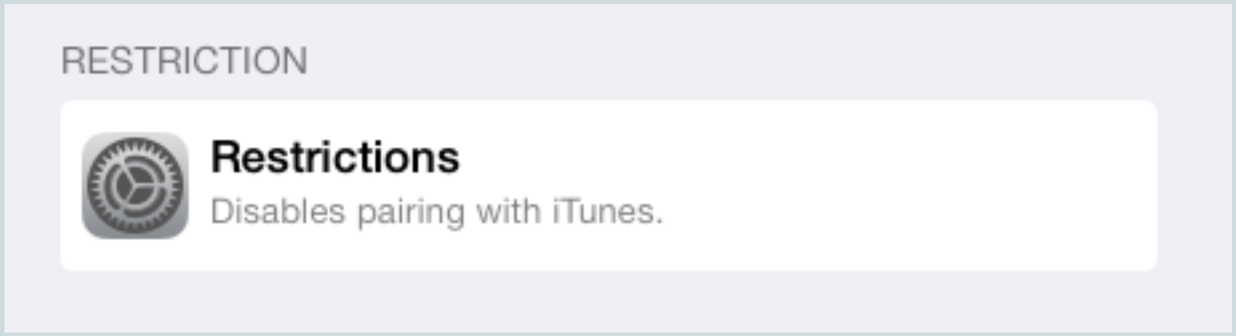
Profiles: [✓] Pairing Profile

Pair Locking with Configurator


Restrictions
1 Payload Configured
Global HTTP Proxy
Not configured
Web Content Filter
Not configured
Wi-Fi
Not configured

- [ ] Force limited ad tracking
- [x] Allow users to accept untrusted TLS certificates
- [x] Allow automatic updates to certificate trust settings
- [x] Allow installing configuration profiles (supervised only)
- [x] Allow modifying account settings (supervised only)
- [x] Allow modifying Find my Friends settings (supervised only)
- [x] Allow pairing with non-Configurator hosts (supervised only)
- [x] Allow documents from managed apps in unmanaged apps
- [x] Allow documents from unmanaged apps in managed apps

Pair Locking with Configurator



RESTRICTION

Restrictions

Disables pairing with iTunes.

Design Suggestions

- Asymmetric cryptography to allow encryption of incoming SMS, Photos, etc. without requiring decryption
- File system equivalent of “session keys” for memory resident processes (CommCenter) to uniquely decrypt shadow copy of certain data (AddressBook)
- Add boot password to encapsulate existing FS encryption; makes stronger / complex passwords less inconvenient
- When pairing, encrypt all keys and EscrowBag sent from phone using backup password, so can’t be used without something you know.

Summary

- Apple is dishing out a lot of data behind our backs
- It's a violation of the customer's trust and privacy to bypass backup encryption
- There is no valid excuse to leak personal data or allow packet sniffing without the user's knowledge and permission.
- Much of this data simply should never come off the phone, even during a backup.
- Apple has added many conveniences for enterprises that make tasty attack points for .gov and criminals
- Overall, the otherwise great security of iOS has been compromised… by Apple… by design.

Thank You

Questions?

@JZdziarski