Downgrading iOS: SHSH Blobs & APTickets

By: iH8sn0w

Jailbreakcon 2012
September 29, 2012

Who Am I?

- Steven (iH8sn0w)
- From Toronto, Ontario (Canada)
- 17 years old
- Involved in the iOS Jailbreak community since 2009
- Known for sn0wbreeze, iREB, iFaith, f0recast

History

- 2007 - iPhone OS 1.0 is released with the iPhone 2G
  - Firmware was pre-signed.
- 2008 - iPhone OS 2.0 is released with the iPhone 3G
  - Firmware is still pre-signed.
- 2009 - iPhone OS 3.0 is released with the iPhone 3GS
  - Firmware is pre-signed but 3GS requires an additional set of signatures unique per device.

SHSH Blobs

- In development since iPhone OS 2.0 days
- Initially introduced with the iPhone 3GS
- ~17 files within an IPSW require unique signatures.
    - LLB, iBoot, Kernel, DeviceTree, Ramdisk, Apple Logo, etc.
- Upon restore, iTunes phones home.
- Apple replies back.
- Blobs are encrypted during restore with this key:

DB 1F 5B 33 60 6C 5F 1C 19 34 AA 66 58 9C 06 61

SHSH Blobs - HW vs SW

- iPhone 3GS' bootrom was shipped **with** the SHSH check.
- iPod Touch 2G [MB & MC] bootrom was shipped **without** the SHSH check. Software based SHSH blob enforcement done via LLB/iBSS.
- Devices that enforce SHSH blobs at a hardware level will always enter DFU mode when the LLB SHSH Blob validation fails.
- iPhone 3G joined the software SHSH enforcement in iOS 4.0.

History (cont.)

- 2010 - iOS 4.0 is released with the iPhone 4
  - Firmware is partially pre-signed but requires SHSH blobs.
  - iPhone 3G/iPod Touch 2G starts to fully enforce SHSH blobs at a software level. (From LLB/iBSS/etc).
- 2011 - iOS 5.0 is released with the iPhone 4S
  - Firmware is partially pre-signed but requires SHSH blobs and an APTicket.

APTickets

- Evil.
- Been in development since iOS 4.x.x
- APTicket replaces majority of the SHSH protocol.
- Generated with a NONCE.

17 SHSH blobs → 2 SHSH blobs

What happened to the other 15 SHSH blobs?

APTickets (cont.)

- An APTicket contains SHA-chunks for those 15 remaining images. (Boot Logos, iBoot, DevTree, Kernel).
- iPhone 3G 4.x.x deja-vu (Enforcing new protocol at sw level)
- Only LLB & iBSS are SHSH signed.
- LLB/iBSS stops caring about SHSH blobs upon execution. Only wants APTickets.
- iBSS refuses to execute any image until a valid APTicket is received for the random NONCE.

APTickets - NONCES

- Similar to how Baseband Tickets are issued.
- A random 72-byte hexadecimal string is generated upon every 5.x.x+ iBoot image execution. [iBSS/iBEC/iBoot/LLB]
- This gets passed along to the TSS request prior to iTunes initializing a restore.
- APTicket received from Apple is for the unique 72-byte hexadecimal string.
- This kills the replay attack with Saurik's TSS@Home server or locally with TinyUmbrella.
- The attack would only work if the device generates the EXACT NONCE that the user has cached.

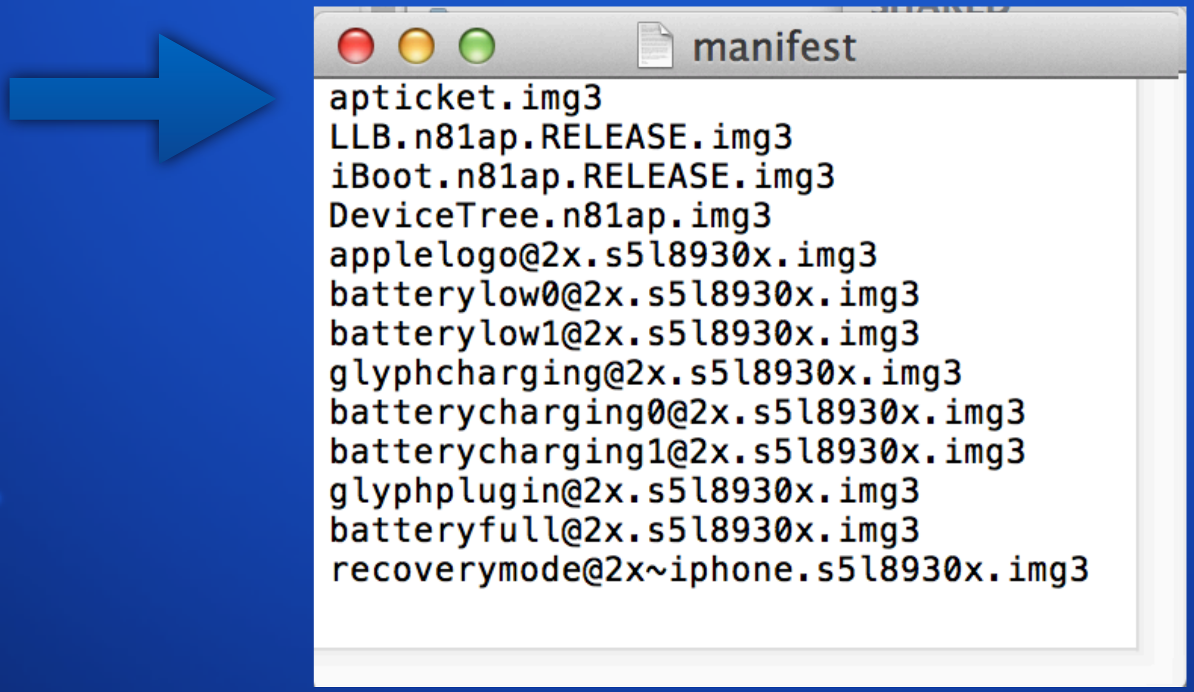
APTickets - ODDS

1 in....

497, 323, 236, 409, 786, 642, 155, 382,
248, 146, 820, 840, 100, 456, 150, 797,
347, 717, 440, 463, 976, 893, 159, 497,
012, 533, 375, 533, 056

APTickets - Counterattacks

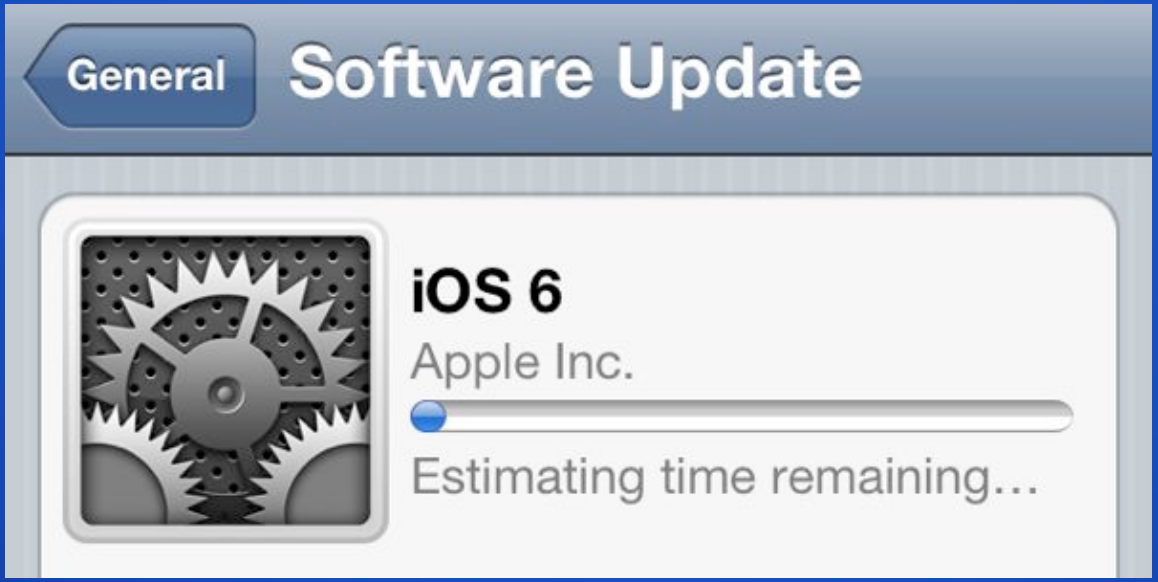
- The APTicket used during a restore is flashed in an Img3 container with the tag 'SCAB'.
- Similar to all other flash images (LLB, iBoot, Boot Logos)
- iOS 5 restores do not error out when no APTicket is provided.
- APTicket can easily be flashed if pre-packaged in an Img3 container and added to the manifest flash file.



```
apticket.img3
LLB.n81ap.RELEASE.img3
iBoot.n81ap.RELEASE.img3
DeviceTree.n81ap.img3
applelogo@2x.s5l8930x.img3
batterylow0@2x.s5l8930x.img3
batterylow1@2x.s5l8930x.img3
glyphcharging@2x.s5l8930x.img3
batterycharging0@2x.s5l8930x.img3
batterycharging1@2x.s5l8930x.img3
glyphplugin@2x.s5l8930x.img3
batteryfull@2x.s5l8930x.img3
recoverymode@2x~iphone.s5l8930x.img3
```

APTickets - Counterattacks (cont.)

- Cannot apply to A5(X) devices due to NONCE requirement upon restore. (or can it?)
- Apple conveniently introduced OTA updates the same time they introduced APTickets.



iOS 6
Apple Inc.
Estimating time remaining…

- But how can this benefit us?

APTickets - 5.1.1 to 5.0.1 Loophole

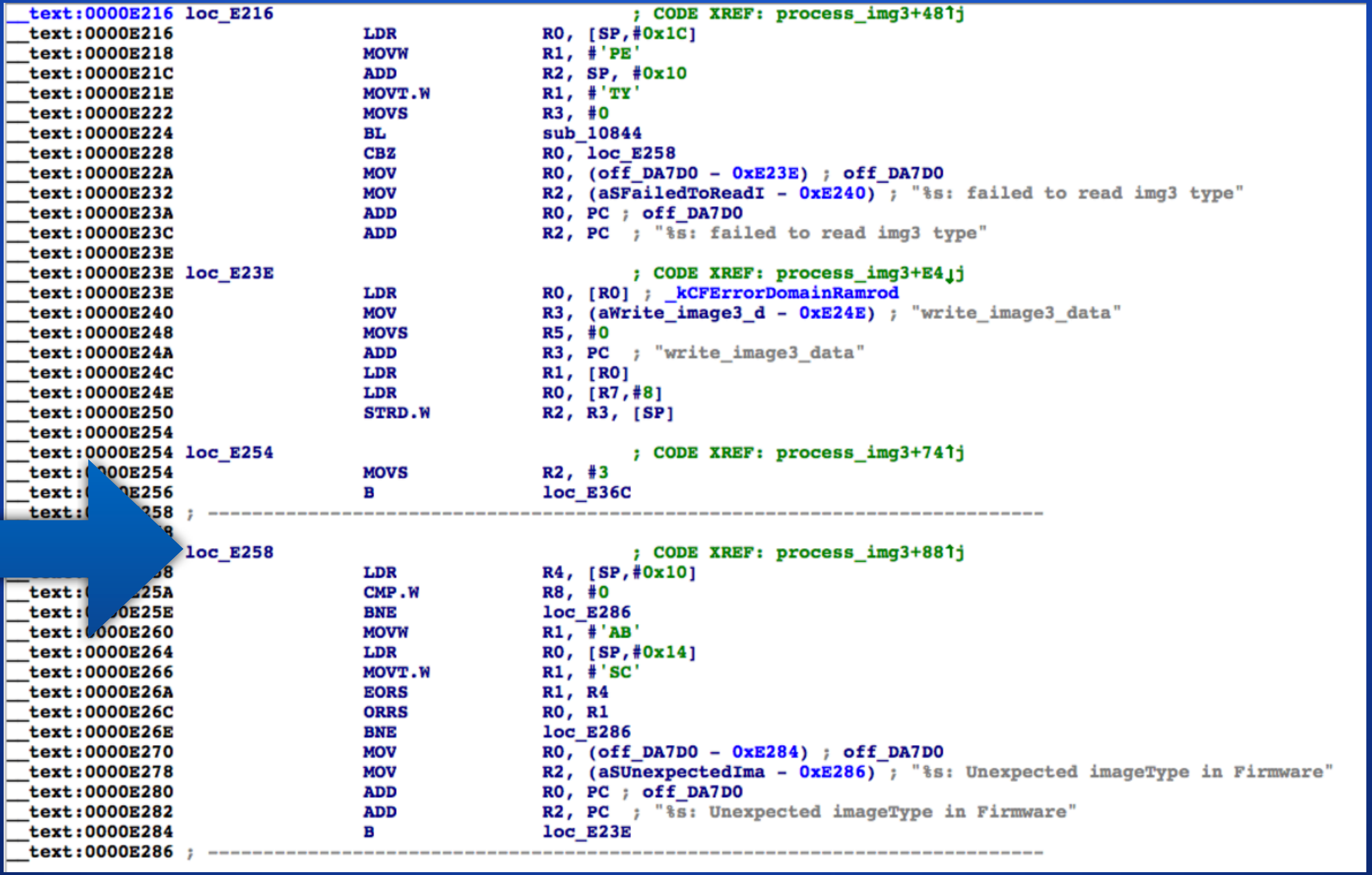
- Recovery Mode accepts iBEC (restore kick-starter) images equal or greater than the flashed APTicket version on the device **without** wanting another unique APTicket based on the generated NONCE.
- Once an iDevice is in iBEC, the device can re-flash the current running firmware.
- A 5.0.1 cached APTicket can be pre-packaged into an Img3 container and flashed by using the 'manifest trick'. Along with flashing the 5.0.1 SHSH signed LLB and 5.0.1 iBoot. All three of these images will be flashed as replacements for the 5.1.1 images.
- When the device restores, it will immediately enter the 5.0.1 flashed iBoot.
- Again, equal or higher iBEC rule still applies. A 5.0.1 iBEC can now be uploaded and a full 5.0.1 restore can be initiated.

APTickets - 4.3.x to 5.1.1 Bootstrap Loophole

- iPad 2 owners whom saved their 4.3.x SHSH blobs can always enter DFU, upload an SHSH signed 4.3.x iBSS and bootstrap to a 5.1.1 SHSH signed iBEC. Device is ready for a 5.1.1 restore.
- 3G models may suffer with network issues.
- This only works because NONCE enforcement is not in 4.3.x iBoot images.

APTickets - 6.0 Nightmare

- Restores immediately fail if no APTicket is provided after iTunes boots the ramdisk.
- Apple now purposely checks the ‘TYPE’ tag within every image that it flashes to ensure its not an APTicket.



```
__text:0000E216 loc_E216                                ; CODE XREF: process_img3+48↑j
__text:0000E216                 LDR             R0, [SP,#0x1C]
__text:0000E218                 MOVW            R1, #'PE'
__text:0000E21C                 ADD             R2, SP, #0x10
__text:0000E21E                 MOVT.W          R1, #'TY'
__text:0000E222                 MOVS            R3, #0
__text:0000E224                 BL              sub_10844
__text:0000E228                 CBZ             R0, loc_E258
__text:0000E22A                 MOV             R0, (off_DA7D0 - 0xE23E) ; off_DA7D0
__text:0000E232                 MOV             R2, (aSFailedToReadI - 0xE240) ; "%s: failed to read img3 type"
__text:0000E23A                 ADD             R0, PC ; off_DA7D0
__text:0000E23C                 ADD             R2, PC  ; "%s: failed to read img3 type"
__text:0000E23E
__text:0000E23E loc_E23E                                ; CODE XREF: process_img3+E4↓j
__text:0000E23E                 LDR             R0, [R0] ; _kCFErrorDomainRamrod
__text:0000E240                 MOV             R3, (aWrite_image3_d - 0xE24E) ; "write_image3_data"
__text:0000E248                 MOVS            R5, #0
__text:0000E24A                 ADD             R3, PC  ; "write_image3_data"
__text:0000E24C                 LDR             R1, [R0]
__text:0000E24E                 LDR             R0, [R7,#8]
__text:0000E250                 STRD.W          R2, R3, [SP]
__text:0000E254
__text:0000E254 loc_E254                                ; CODE XREF: process_img3+74↑j
__text: 00E254                  MOVS            R2, #3
__text:  E256                   B               loc_E36C
__text:   258 ; ---------------------------------------------------------------------------
         8
                loc_E258                                ; CODE XREF: process_img3+88↑j
         8                      LDR             R4, [SP,#0x10]
__text:   25A                   CMP.W           R8, #0
__text:  0E25E                  BNE             loc_E286
__text: 000E260                 MOVW            R1, #'AB'
__text:0000E264                 LDR             R0, [SP,#0x14]
__text:0000E266                 MOVT.W          R1, #'SC'
__text:0000E26A                 EORS            R1, R4
__text:0000E26C                 ORRS            R0, R1
__text:0000E26E                 BNE             loc_E286
__text:0000E270                 MOV             R0, (off_DA7D0 - 0xE284) ; off_DA7D0
__text:0000E278                 MOV             R2, (aSUnexpectedIma - 0xE286) ; "%s: Unexpected imageType in Firmware"
__text:0000E280                 ADD             R0, PC ; off_DA7D0
__text:0000E282                 ADD             R2, PC  ; "%s: Unexpected imageType in Firmware"
__text:0000E284                 B               loc_E23E
__text:0000E286 ; ---------------------------------------------------------------------------
```

History (cont.)

- 2012 - iOS 6.0 is released with the iPhone 5
  - Firmware is partially pre-signed but requires an APTicket.

Where are things now?

- A5(X) devices still running 5.x.x can essentially "re-restore" to the same firmware they are running as long as they have their SHSH blobs & APTicket.
- A5 devices with 4.3.x & 5.x.x SHSH Blobs can perform the 4.3.x iBSS to 5.x.x iBEC Bootstrap Loophole.
- A4 devices can obviously be downgraded if 5.x.x SHSH Blobs and APTickets are cached due to limera1n.
- APTicket checks have improved in 6.0, but it can be better. ;)
- SHSH Blobs will no longer be present after the 4S stops being supported. Purely APTickets.

Q&A