Google

Code

Archive

iphone-dataprotection - README.wiki

Export to GitHub

Supported devices

- (iPhone 2G)
- iPhone 3G
- iPhone 3GS
- iPad 1
- iPhone 4 GSM
- iPhone 4 GSM rev A
- iPhone 4 CDMA

Newer devices are NOT supported.

It is not possible to bruteforce passcode or fix boot-loops on A5+ devices (anything newer than iPhone 4)

Requirements

- Mac OS X 10.8/10.9
- Mercurial to download the tools from the repository
- Xcode with iOS SDK (open Xcode at least once to accept the license agreement)
- redsn0w 0.9.15b3 (downloaded by build.py)
- Supported IPSW for the target device in the data/ipsw folder (downloaded by build.py)
- A few Python modules: PyCrypto, M2crypto, construct, progressbar, pyasn1 (see below)

Mac OS X is only required to build the custom ramdisk. Once this is done, Windows can be used to boot the ramdisk and interact with it, either through ssh or using the python scripts provided. Linux is not supported.

OSX Python dependencies

``` sudo easy_install M2crypto construct progressbar setuptools pyasn1 sudo ARCHFLAGS='-arch i386 -arch x86_64' easy_install pycrypto

# see http://chandlerproject.org/Projects/MeTooCrypto for other OSX versions

curl -O http://chandlerproject.org/pub/Projects/MeTooCrypto/M2Crypto-0.21.1-py2.7-macosx-10.9-intel.egg sudo easy_install M2Crypto-0.21.1-py2.7-macosx-10.9-intel.egg ```

## Windows dependencies

- redsn0w: https://sites.google.com/a/iphone-dev.com/files/home/redsn0w_win_0.9.15b3.zip
- iTunes. It is not required to install everything, you can extract iTunes64Setup.exe and only install AppleApplicationSupport.msi and AppleMobileDeviceSupport64.msi
- Python 2.7: https://www.python.org/ftp/python/2.7.6/python-2.7.6.msi
- PyCrypto: http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-py2.7.exe
- M2Crypto: http://chandlerproject.org/pub/Projects/MeTooCrypto/M2Crypto-0.21.1.win32-py2.7.msi
- setuptools: https://pypi.python.org/pypi/setuptools#windows-7-or-graphical-install

```
C:\Python27\Scripts\easy_install.exe construct progressbar pyasn1
```

# Building custom ramdisk & kernel (Mac OS X)

After installing Mercurial and Xcode, open Terminal and run the following commands :

```
osx109(~) $ sudo xcodebuild -license osx109(~) $ hg clone https://code.google.com/p/iphone-dataprotection/ osx109(~) $ cd iphone-dataprotection osx109(~/iphone-dataprotection) $ ./build.py Usage: ./build.py ipad1|iphone3gs|iphone4|iphone4cdma|iphone4gsmA|ipt2 osx109(~/iphone-dataprotection) $ ./build.py iphone4
```

The build script should give the following output :

```
Using data/ipsw/iPhone3,1_5.0_9A334_Restore.ipsw Decrypting kernelcache.release.n90 Unpacking ... Doing CSED patch Doing getxattr system patch Doing nand-disable-driver patch Doing task_for_pid_0 patch Doing IOAES gid patch Doing AMFI patch Doing AppleIOPFMI::_fmiPatchMetaFringe patch Doing _PE_i_can_has_debugger patch Doing IOAESAccelerator enable UID patch Added IOFlashControllerUserClient::externalMethod patch at file offset 0x5f08f4 Patched kernel written to data/boot/kernel_iPhone3,1_5.0_9A334.patched Decrypting 018-7923-347.dmg Decrypted ramdisk written to data/boot/ramdisk_iPhone3,1_5.0_9A334.dmg Rebuilding ramdisk_tools Found iOS SDK at /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS7.0.sdk/
```
```

```
clang -arch armv7 -Wall -Wno-pointer-sign -isysroot
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS7.0.sdk/
-DHGVERSION="\"4152dc4f6484\"" -O3 -I. -framework CoreFoundation -framework IOKit -framework Security
-miphoneos-version-min=4.0 -o device_infos device_infos.c device_info.c IOAESAccelerator.c
AppleEffaceableStorage.c AppleKeyStore.c bsdcrypto/pbkdf2.c bsdcrypto/sha1.c bsdcrypto/key_wrap.c
bsdcrypto/rijndael.c util.c IOKit.c registry.c ioflash/ioflash.c ioflash/IOFlashPartitionScheme.c
kernel_patcher.c codesign -s - --entitlements entitlements.plist device_infos clang -arch armv7 -Wall
-Wno-pointer-sign -isysroot
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS7.0.sdk/
-DHGVERSION="\"4152dc4f6484\"" -O3 -I. -framework CoreFoundation -framework IOKit -framework Security
-miphoneos-version-min=4.0 -o restored_external restored_external.c device_info.c remote_functions.c
plist_server.c AppleKeyStore.c AppleEffaceableStorage.c IOKit.c IOAESAccelerator.c util.c registry.c
AppleKeyStore_kdf.c bsdcrypto/pbkdf2.c bsdcrypto/sha1.c bsdcrypto/rijndael.c bsdcrypto/key_wrap.c
ioflash/ioflash.c ioflash/IOFlashPartitionScheme.c kernel_patcher.c codesign -s - --entitlements
entitlements.plist restored_external clang -arch armv7 -Wall -Wno-pointer-sign -isysroot
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS7.0.sdk/
-DHGVERSION="\"4152dc4f6484\"" -O3 -I. -framework CoreFoundation -framework IOKit -framework Security
-miphoneos-version-min=4.0 -o bruteforce systemkb_bruteforce.c AppleKeyStore.c
AppleEffaceableStorage.c IOKit.c IOAESAccelerator.c util.c registry.c AppleKeyStore_kdf.c
bsdcrypto/pbkdf2.c bsdcrypto/sha1.c bsdcrypto/rijndael.c bsdcrypto/key_wrap.c device_info.c
ioflash/ioflash.c ioflash/IOFlashPartitionScheme.c kernel_patcher.c codesign -s - --entitlements
entitlements.plist bruteforce clang -arch armv7 -Wall -Wno-pointer-sign -isysroot
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS7.0.sdk/
-DHGVERSION="\"4152dc4f6484\"" -O3 -I. -framework CoreFoundation -framework IOKit -framework Security
-miphoneos-version-min=4.0 -o ioflashstoragekit ioflash/ioflash.c ioflash/ioflash_kernel.c
ioflash/ioflashstoragekit.c ioflash/IOFlashPartitionScheme.c util.c codesign -s - --entitlements
entitlements.plist ioflashstoragekit Non-fat file: ramdisk_tools/restored_external is architecture:
armv7 resizing ramdisk... hdiutil will segfault if ramdisk was already resized, thats ok Attaching
ramdisk /dev/disk1 /Volumes/ramdisk Unpacking ssh.tar.gz on ramdisk... bin/: Already exists bin/cat:
Already exists bin/launchctl: Already exists bin/ln: Already exists bin/mkdir: Already exists bin/mv:
Already exists bin/rm: Already exists etc/: Already exists sbin/: Already exists usr/: Already exists
usr/bin/: Already exists usr/lib/: Already exists var/: Already exists tar: Error exit delayed from
previous errors. ^^ This tar error message is okay Adding/updating ramdisk_tools binaries on
ramdisk... "disk1" unmounted. "disk1" ejected. Build OK, running boot.py
```

The boot.py script is used to launch redsn0w with the right command line options

```
Using data/ipsw/iPhone3,1_5.0_9A334_Restore.ipsw Use nand-disable boot flag ? [y/n] n Boot args: -v
rd=md0 amfi=0xff msgbuf=409600 Command line: redsn0w_mac_0.9.15b3/redsn0w.app/Contents/MacOS/redsn0w
-i data/ipsw/iPhone3,1_5.0_9A334_Restore.ipsw -k data/boot/kernel_iPhone3,1_5.0_9A334.patched -r
data/boot/ramdisk_iPhone3,1_5.0_9A334.dmg -a "-v rd=md0 amfi=0xff msgbuf=409600" Launching redsn0w...
```

Follow redsn0w instructions to place the device in DFU mode.

On Windows, copy the contents of the data/boot and data/ipsw folders from the OSX machine, and run boot.bat.

SSH access (Mac OS X/Windows)

After redsn0w is done, the ramdisk should boot in verbose mode. Once "OK" appears on the screen, the custom ramdisk has successfully started. The device is now accessible using ssh over usbmux. Run the following command (in a separate terminal window) to setup port redirections:

```
osx109(~/iphone-dataprotection) $ ./tcprelay.sh #use tcprelay.bat on Windows Forwarding local port 2222 to remote port 22
```

SSH is now accessible at localhost:2222.

```

# if ~/.ssh/id_rsa.pub exists on the host, it was copied to the ramdisk so no password is required.

# otherwise root password is alpine

ssh -p 2222 root@localhost ```

# Python scripts (Mac OS X/Windows)

The demo_bruteforce.py script connects to the custom restored_external daemon on the ramdisk, collects basic device information (serial number, UDID, etc.), unique device keys (keys 0x835 and 0x89B), downloads the system keybag and tries to bruteforce the passcode (4 digits only). If the bruteforce is successfull it will also download the keychain database.

```
python python_scripts/demo_bruteforce.py
```

The results are stored in a plist file named after the device's data parititon volume identifier (a folder named after the device UDID is also created). This plist file is required for the other python scripts to operate correctly. For instance, the keychain database contents can be displayed using keychain_tool.py.

```
python python_scripts/keychain_tool.py -d UDID/keychain-2.db UDID/DATAVOLUMEID.plist
```

A shell script is provided to create a dd image of the data parititon that will be placed in the device UDID directory.

```
./dump_data_partition.sh
```
```

The image file can be opend using the modified HFSExplorer that will decrypt the files "on the fly". To decrypt it permanently (for use with standard tools), the emf_decrypter.py script can be used. Both tools depend on the aforementioned plist file being in the same directory as the disk image.

```

# do a dry run to avoid crashing halfway

python python_scripts/emf_decrypter.py --nowrite UDID/data_DATE.dmg

# if no errors then decrypt the image in place

python python_scripts/emf_decrypter.py UDID/data_DATE.dmg ```

Finally, the HFS journal file can be carved to search for deleted files. Keep in mind that only a very few number of files (or even none at all) can be recovered that way. `python python_scripts/emf_undelete.py UDID/data_DATE.dmg`

# NAND acquisition & deleted files recovery

For NAND access it is recommanded to use the nand-disable boot flag.

Once the ramdisk is booted, run the ios_examiner script without parameters: `python python_scripts/ios_examiner.py Connecting to device : 0000000000000000000000000000000000000000 Device model: iPhone 4 GSM UDID: 0000000000000000000000000000000000000000 ECID: 000000000000 Serial number: 00000000000 key835: 00000000000000000000000000000000 key89B: 00000000000000000000000000000000 Chip id 0x3294e798 banks per CE physical 1 NAND geometry : 16GB (4 CEs (1 physical banks/CE) of 4100 blocks of 128 pages of 8192 bytes data, 12 bytes metdata) Searching for special pages... Found DEVICEUNIQUEINFO, NANDDRIVERSIGN, DEVICEINFOBBT special pages in CE 0 NAND signature 0x43313131 flags 0x10006 withening=1, epoch=1 Effaceable generation 204 Effaceable CRC OK Found effaceable lockers in ce 3 block 1 page 96 Lockers : BAG1, DONE, Dkey, LwVM Found DEVICEUNIQUEINFO, serial number=00000000000 Using VSVFL VSVFL context open OK YaFTL context OK, version=CX01 maxIndexUsn=137419 context usn=137419 LwVM header CRC OK cprotect version : 4 (iOS 5) iOS version: 5.1.1 Keybag state: locked (iPhone4-data) /`

At this point you can enter commands in the ios_examiner shell.

``` (iPhone4-data) / bruteforce Enter passcode or leave blank for bruteforce:

Passcode "" OK Keybag state: unlocked Save device information plist to [0000000000.plist]: iphone4.plist (iPhone4-data) / nand_dump iphone4_nand.bin Dumping 16GB NAND to iphone4_nand.bin 100% |##| NAND

dump time : 0:48:51.799000 SHA1: 0000000000000000000000000000000000000000 (iPhone4-data) / exit ```

You can then relaunch ios_examiner with the nand image and device plist files as parameters: `python python_scripts/ios_examiner.py iphone4_nand.bin iphone4.plist`

Use the "undelete" command to recover deleted files from the nand image. The undelete feature is still experimental but should give better results than the HFS journal carving technique. The "dd" command can be used to extract the data partition as a regular dd image.

ios_examiner commands

- system/data: switch between system and data partition
- dd filename.dmg: dump current partition to filename.dmg
- cd/ls/pwd: browse current partition
- open/plist filename: display file
- xxd filename [length]: show hexdump
- ptable: display partition table
- xattr filename: display extended attributes
- cprotect filename: display decoded cprotect attribute
- protected_files: lists files that use protection classes != NSProtectionNone (files that cannot be accessed without the passcode)
- nand_dump nanddump.bin: dumps NAND memory of currently connected device to nanddump.bin
- bruteforce: asks for system keybag passcode or tries to bruteforce 4-digit passcodes
- keybag: display system keybag state
- keychain: display keychain items
- undelete: try to recover deleted files (only YaFTL is supported)
- img3: extract img3s (NAND-only devices)
- reboot: reboot connected device
- help: list available commands

FAQ

The following AppleBCMWLANCore error message appears on screen after booting the ramdisk `AppleBCMWLANCore:handleIOKitBusyWatchdogTimout(): Error, no successful firmware download after 60000 ms!! Giving up...` This is normal, Wi-Fi is not initialized when booting the ramdisk environment.

Photorec only recovers existing files, even after running emf_decrypter

Unallocated space cannot be decrypted with emf_decrypter, because each file uses a different encryption key.