



2008

Primer on Reversing Jailbroken iPhone Native Applications



Shub-Nigurrath

ARTeam

May 2008

DISCLAIMER

All code included with this tutorial is free to use and modify; we only ask that you mention where you found it. This tutorial is also free to distribute in its current unaltered form, with all the included supplements.

All the commercial programs used within this tutorial have been used only for the purpose of demonstrating the theories and methods described. No distribution of patched applications has been done under any media or host. The applications used were most of the times already been patched by other fellows, and cracked versions were available since a lot of time. ARTeam or the authors of the papers shouldn't be considered responsible for damages to the companies holding rights on those programs. The scope of this document as well as any other ARTeam tutorial is of sharing knowledge and teaching how to patch applications, how to bypass protections and generally speaking how to improve the RCE art. We are not releasing any cracked application.

VERIFICATION

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: <http://releases.accessroot.com>



**TABLE OF CONTENTS**

Disclaimer	2
Verification	2
Forewords	5
 1 JAILBREAKING YOUR IPHONE POWER	 6
1.1 Fast and practical jailbreaking procedure	7
1.1.1 Using iPlus to jailbreak	8
1.1.2 Using ZiPhone to jailbreak	11
1.2 File system structure	13
 2 DISASSEMBLING THE FIRST NATIVE PRE-INSTALLED APPLICATIONS	 14
2.1 Essential Notes on the Mach-O file format	15
2.2 Essential Notes on the Objective-C Programming Model	19
2.3 Essential Notes on the reversing of objective-C: the role of objc_msgsend	20
2.4 Where to go deeper	22
2.5 Continue to reverse our first application	22
2.6 CFStrings objects and normal strings	23
2.7 Selectors-handlers map	24
 3 UNLOCKING FOR REAL THE IPHONE/IPOD TOUCH	 27
3.1 ScreenShots	29
3.2 Connect your PC to the device	29
 4 REVERSING SOME APPLICATIONS	 32
4.1 Go down with the first application: AcceleroLog	32
4.1.1 Adjust the Application so as IDA can handle it correctly	33
4.1.2 Patching the Application	37
4.1.3 How to calculate the opcode of a branch in ASM	39
4.2 Going on with the second application: ShowTime	41
4.2.1 Version 1.1b	42



4.3	Going on with the third application: iSim	45
4.3.1	Find the correct installer URL through the .plist file	46
4.3.2	Reversing the target	46
4.3.3	Changing the class of a Section from DATA to CODE	47
4.3.4	Patching the Application	49
4.4	Going on with the Fourth application: Camerapro	50
4.4.1	Find the correct installer URL through the .plist file	50
4.4.2	Reversing the target	51
4.5	Going on with the FIFTH application: PocketMoney	51
4.6	Going on with the Sixth application: Softick Solitaire	52
4.6.1	Approaching the Application	53
4.6.2	Manually Finding the reference to a given string	54
4.6.3	Patching the Application	54
4.7	Where to find other native applications	57
5	CONCLUSIONS	58
5.1	References	59
5.2	History	60

FOREWORDS



This tutorial is another primer I decided to write (similarly to what I did for Symbian), following my early experiences in the iPhone/iPood Touch world. We are talking of the iPhone (and the little brother iPood Touch), the Apple telephone which doesn't support Java, nor Flash, cannot be used like a modem, do not fully support Bluetooth, do not allow to install third party applications (officially at least), which cost is very high and that you can only be used with those telecom carriers chosen by Apple just with EDGE network.

With all these limitations the iPhone should have disappeared from the market, instead it is alive and kicking, collecting records: it conquered in a very short timeframe about a third of the overall smartphones market and it is responsible of about ¾ of the overall online traffic of the mobile devices. The iPhone is not only an iPood with advanced multimedia features, equipped with WiFi, but it has a real powerful system under the wood, a complete operative system with a full web browser (Safari) and applications, built to go on the Web 2.0.

Once unlocked (using techniques I will not explain with much in details) a new world opens up: a lot of native programs (not web) are awaiting your patches and customizations. It's a completely different world for those of you already accustomed to the Win32 environment, but also has some differences for those already reversing in the Mac world. The OS is an OSX, build up from FreeBSD, but it isn't the MAC OSX and the processor is an ARM, then RISC assembler not easy to handle at all (like for Symbian phones). The programming technologies are anyway the same already used for MAC: COCOA/Darwin and underneath objective-C.

At the moment all the existing native applications are built using the unofficial SDK (but the official SDK is announced soon) and are most of the times, but not always, completely free. Users must before jailbreak their devices and it's not a thing anyone could/would do. Most of the applications are just experiments meanwhile the real SDK is not here, just few are donation-ware and even less are shareware.

I will assume you have already got the basics of the ARM disassembler and IDA things. If not I suggest to read the "Primer on Reversing Symbian S60 Applications"[12] or the "Symbian Symphony For 4 Crackmes And A Commercial Program"[13] I already released, at least just about the details on the ARM platform and assembler there explained.

All along the document I spread a lot of links to references and further readings, keep attention to them to dive some aspects..

BTW Do you like the new tutorials look?

Your Favourite Neighbourhood Shubby



1 JAILBREAKING YOUR IPHONE POWER

When I did for the first time the jailbreak of the iPhone it was a quite complex and risky procedure: you had to downgrade the firmware to version 1.1 and then break it, using a failure in the handling of TIFF¹ images of that firmware. Starting from this you had to upgrade backward (or forward) to the latest firmware. The procedure was, theoretically simple: downgrade to firmware 1.1, exploit of a TIFF handling bug, which gave permission to execute arbitrary code as native. The hole was large enough to be finally able to install any application. Upgrade where done on an already compromised system, which was then able then to re-create the backdoor on newest firmware too.

On the net you can find several guides which explain completely what you have to do.

As described in [1] the original procedure used for the first time to unlock the firmware was possible with some collaboration of an Apple employee (rumours). Anyway what is sure is that the distribution file of the firmware iPhone1,1_1.0.2_1C28 was incredibly easy to decrypt.

You should know that the firmware is downloaded from the apple site as files with the **ipsw** extension. These files are indeed zip files which contains among other things an encrypted MAC disk image, in the **dmg** format². This format was well encrypted using AES, and theoretically very hard to decrypt. What is the strange is that the encryption key was clearly buried inside the **dmg** file as clear string. It was possible to run a bat file like the following to find it³:

```
strings -q %1|agrep "[0-9a-fAF]*$" |awk "{ if (length($1) == 72) print; }"
```

which in other words finds a string made of literals and numbers long 72 chars then prints it.

Then this string was used to decrypt the file using vfdcrypt, using this line:

```
vfdcrypt.exe -i 694-5298-5.dmg -k
7d5962d0b582ec2557c2cade50de90f4353a1c1de07b74212513fef9cc71fb890574bfe5 -o
dmg_image_decrypted.dmg
```

The vfdcrypt sources are available on the net [2] and the early lines of the source code tell:

```
* Copyright (c) 2006
* Ralf-Philipp Weinmann <ralf@coderpunks.org>
* Jacob Appelbaum <jacob@appelbaum.net>
* Christian Fromme <kaner@strace.org>
* Decrypt a AES-128 encrypted disk image given the encryption key and
* the hmacshalkey of the image. These two keys can be found out by running
* hdiutil attach with -debug on the disk image.
```

This is a tool that was developed for Macs, not for iPhones! This makes me wondering that Apple was indeed willing to have jailbroken iPhones⁴

Anyway, using this procedure is even possible to decrypt the whole dmg file on a Windows PC (you must install some dlls from the cgywin distribution and find some other things but it's not difficult)⁵.

¹ <http://blog.metasploit.com/2007/10/cracking-iphone-part-1.html>

² The ONLY tool I found able to read iPhone dmg files is TransMac 8.x (www.asy.com), all the others, including PowerISO, fails.

³ To be precise, these files contain the string "encrcdsa" at the beginning then are Mac Encrypted Sparsedisk

⁴ <http://iphonejtag.blogspot.com/2008/01/iphone-secret-key.html>