SWISS **CYBER STORM** 2011

THE SWISS CONFERENCE ON **IT SECURITY** TECHNOLOGY

12.-15. MAY 2011 / **RAPPERSWIL (CH)**

iOS applications reverse engineering

Julien Bachmann – julien@scrt.ch

SCRT Information Security

SCRT
INFORMATION SECURITY
SWITZERLAND

WWW.SCRT.CH

SYSTEM CERTIFICATION STANDARD SGS

Agenda

- Motivations
- The architecture
  - Mach-O
  - Objective-C
  - ARM
- AppStore binaries
  - Find'em
  - Decrypt'em
  - Reverse'em
- What to look for
  - Where to start
  - Remote connections
  - Data protection
- Conclusion

Preamble

- Security engineer @ SCRT
- Areas of interest focused on reverse engineering, software vulnerabilities and OS internals
- Not an Apple fanboy but like all the cool kids... ;)

- Goals of this presentation is to give a state of the art, in 45minutes, of my knowledge about iOS applications reverse engineering
- Motivate people to do more research in user/kernel-land iOS reverse engineering

Motivations

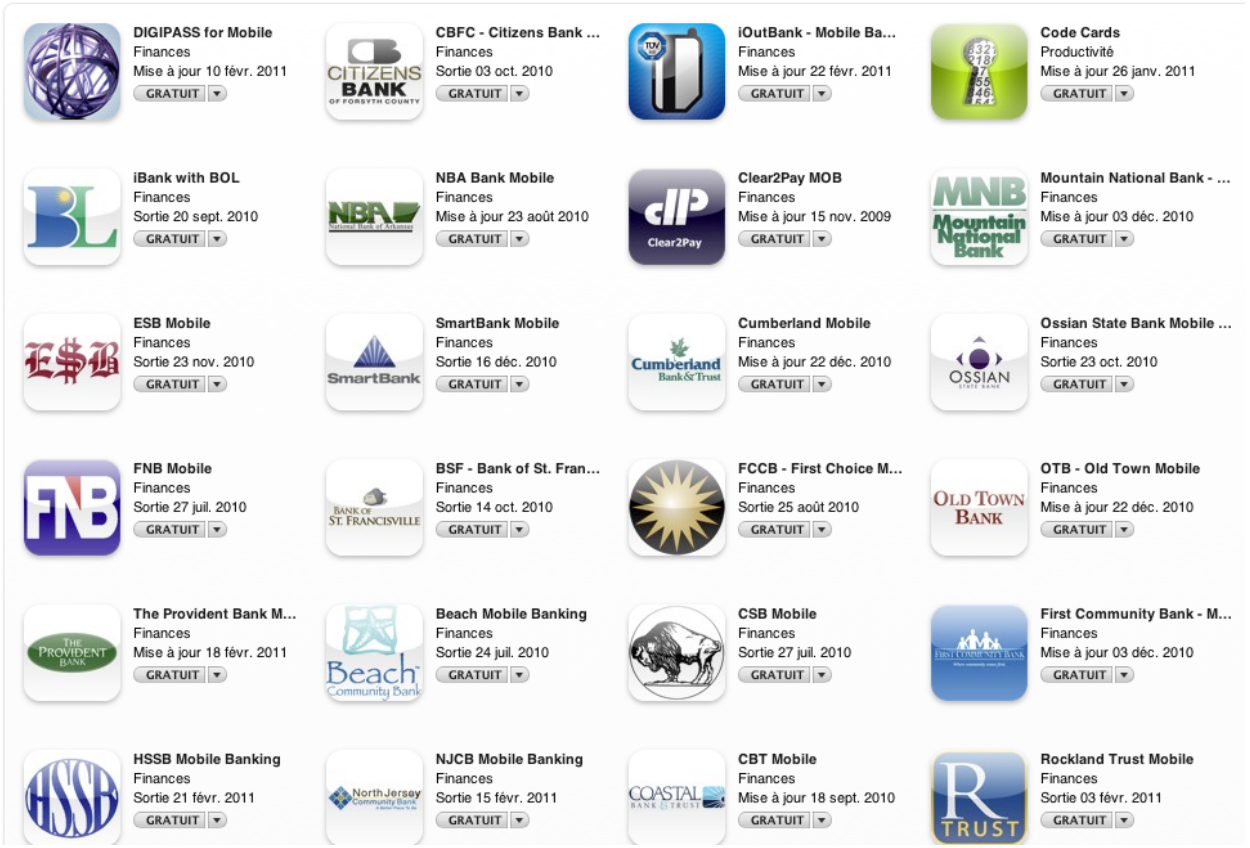
A few numbers

- +160 millions iOS users
- +400 000 applications available
- +10 billion downloads

→ (modestly) large user base

e-banking applications

Tous/toutes les Apps pour iPhone pour « e-banking »



Applications review

- Apple defined a review process
  - 10% of the applications are classified as dangereous
- Cases of applications not « compliant » with their description

Storm8 case

iPhone developer accused of stealing user data via app backdoor

Fri, 06 Nov 2009

The complaint seeks class action status so other users of Storm8 games

Nick Spence

A US iPhone developer behind a series of popular games has been accused of stealing users data via a backdoor in the applications.

Now, what if you want to...

- check an external app ?
- verify that your application is secure ?
- check what kind of information an attacker can get from your application ?

Best reason ever...

- Because it's fun to learn how to reverse new things !

The architecture

Mach-O

- File format for
  - Executables
  - Libraries
  - Core dumps

Mach-O

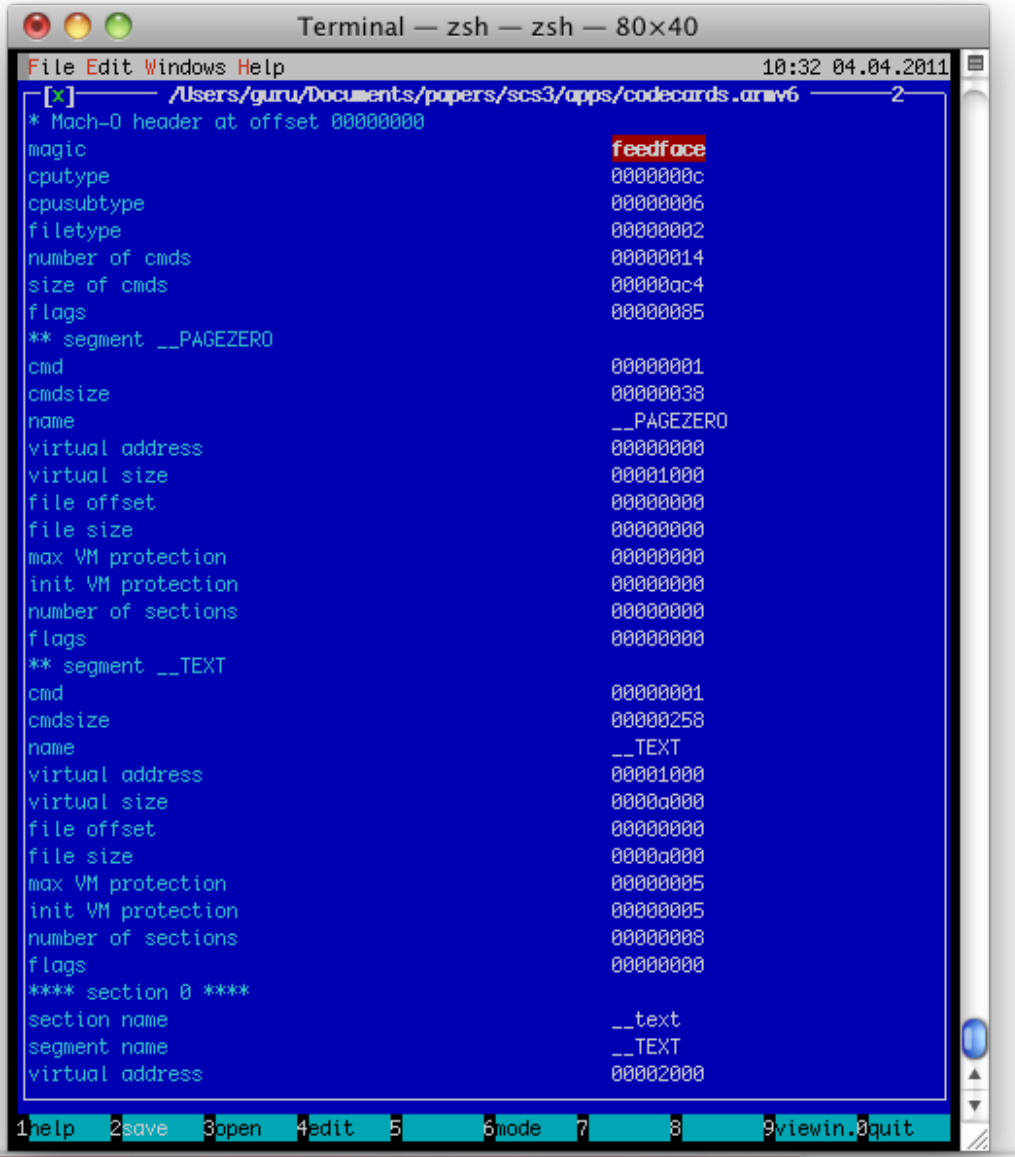
- Contains three parts
  - Header
  - Load commands
  - Data

Mach-O

- Header

```
struct mach_header
{
  uint32_t      magic;
  cpu_type_t    cputype;
  cpu_subtype_t cpusubtype;
  uint32_t      filetype;
  uint32_t      ncmds;
  uint32_t      sizeofcmds;
  uint32_t      flags;
};
```

Mach-O

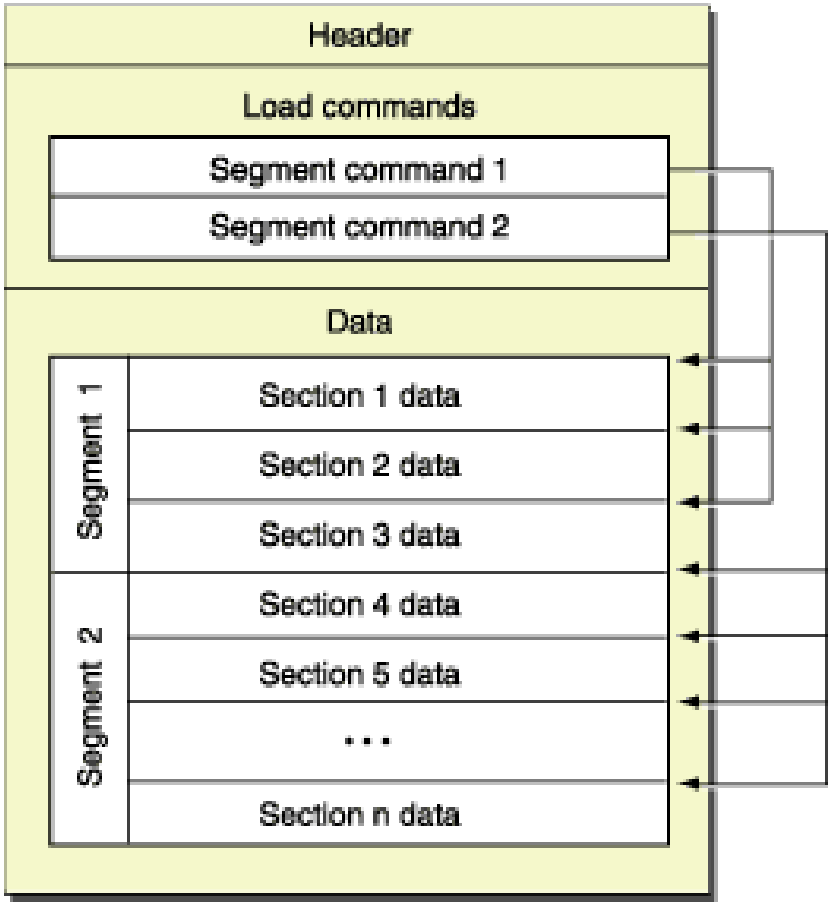

Terminal — zsh — zsh — 80×40
File Edit Windows Help
10:32 04.04.2011
/Users/guru/Documents/papers/scs3/apps/codecards.armv6
* Mach-O header at offset 00000000
magic feedface
cputype 0000000c
cpusubtype 00000006
filetype 00000002
number of cmds 00000014
size of cmds 00000ac4
flags 00000085
** segment __PAGEZERO
cmd 00000001
cmdsize 00000038
name __PAGEZERO
virtual address 00000000
virtual size 00001000
file offset 00000000
file size 00000000
max VM protection 00000000
init VM protection 00000000
number of sections 00000000
flags 00000000
** segment __TEXT
cmd 00000001
cmdsize 00000258
name __TEXT
virtual address 00001000
virtual size 0000a000
file offset 00000000
file size 0000a000
max VM protection 00000005
init VM protection 00000005
number of sections 00000008
flags 00000000
**** section 0 ****
section name __text
segment name __TEXT
virtual address 00002000
1help 2save 3open 4edit 5 6mode 7 8 9viewin.0quit

Mach-O

- Load commands
  - Indicates memory layout
  - Locates symbols table
  - Main thread context
  - Shared libraries

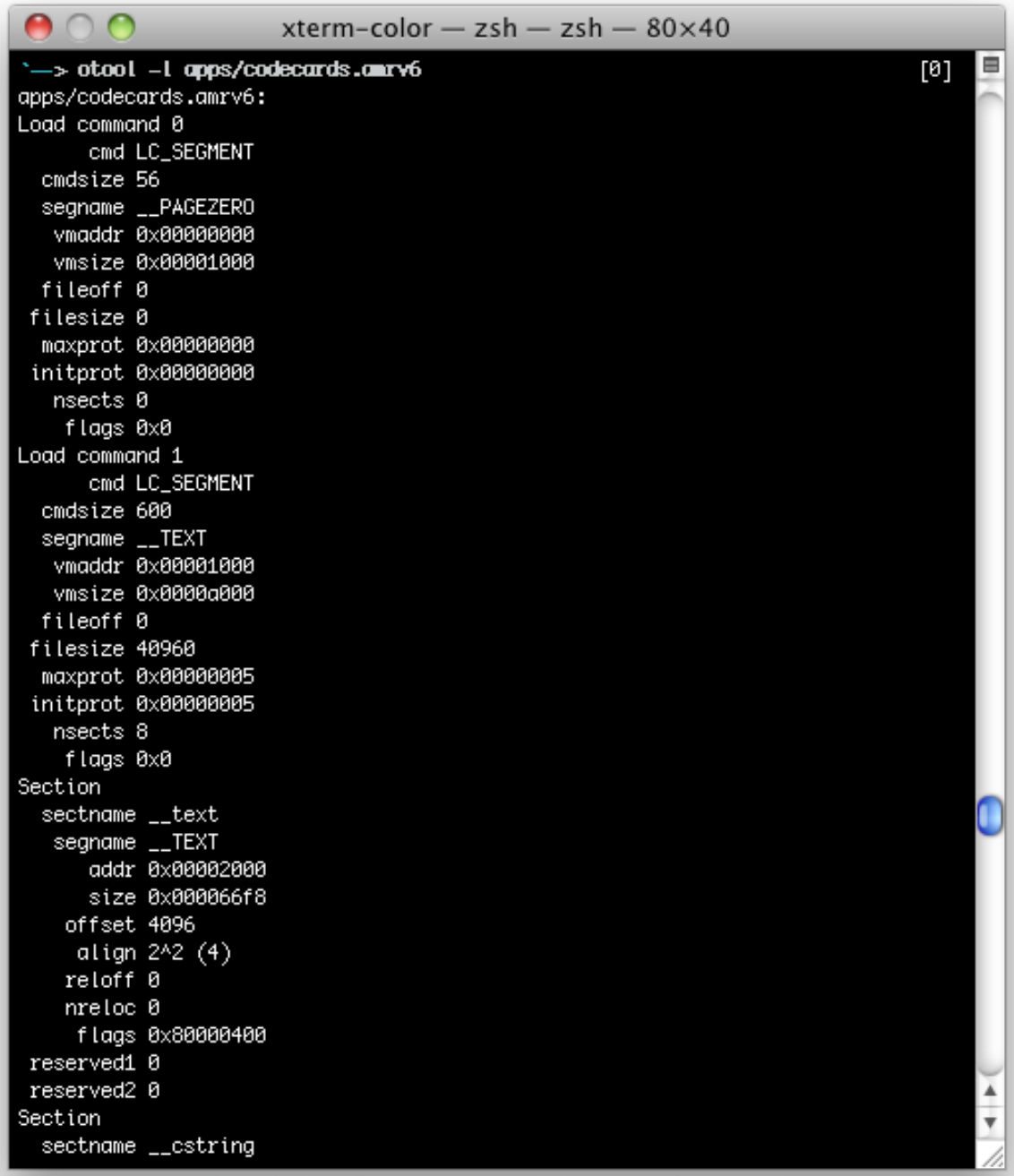
Mach-O

- Data
  - Segments containing sections
  - __PAGEZERO
  - __TEXT
    - Executable code and r--
  - __DATA
    - rw-
  - __OBJC
  - ...



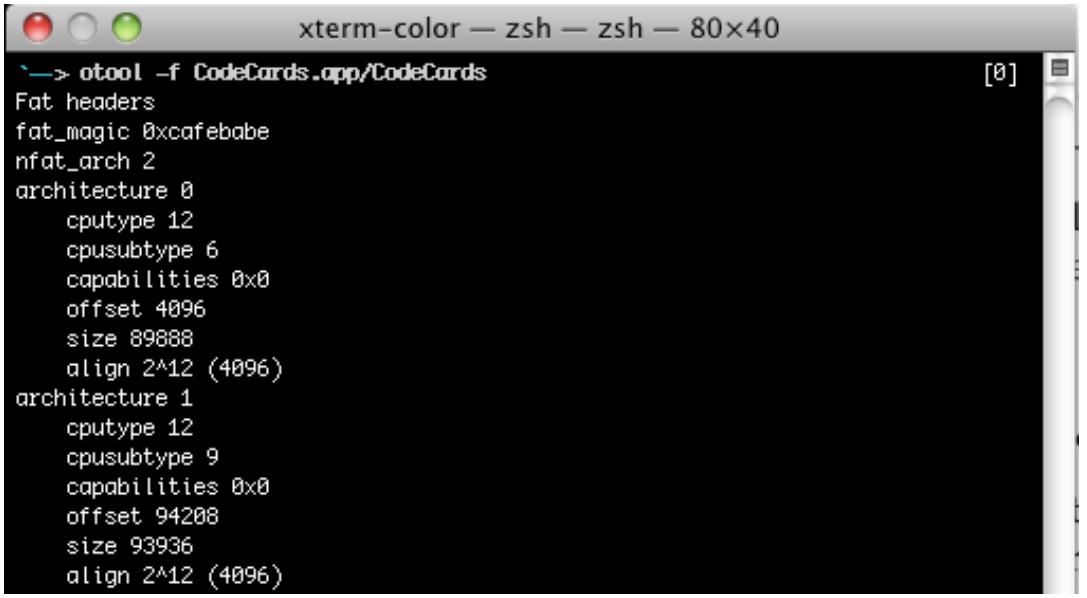
Mach-O

- objdump ?
  - Forget about it
  - Introducing : otool !



Mach-O

- Universal / FAT files
  - Supports multiples architectures
  - For OSX
    - Universal
    - PowerPC, x86 and x86_64
  - For iOS
    - FAT
    - armv6, armv7



Objective-C

- Programming language
- Superset of the C language
- Object oriented
  - Class method calls differ from C++

Calling methods

- C++
  - ObjectPointer->Method(param1, param2)
- Obj-C
  - [ObjectPointer Method:param1 param2Name:param2]

Looking more closely

- [ObjectPointer Method]
  - objc_msgSend(ObjectPointer, @selector(Method))
- Selector
  - C string
  - objc_msgSend(ObjectPointer, "Method")

ARM

- RISC
- load-store architecture
- Fixed-length 32-bit instructions
- 3-address instruction formats

Registers

- User-level programs
  - 15 general-purpose 32-bit registers : r0 → r14
  - PC = r15
  - Current program status register (N, Z, C, V flags, etc.)

Load-store architecture

- Instructions can be classified into 3 groups
  - Data transfer (load-store)
  - Data processing
  - Control flow

Data transfer instructions

- Load from memory
  - LDR r0, [r1] → r0 = mem[r1]
- Store to memory
  - STR r0, [r1] → mem[r1] = r0

Data processing instructions

- Simple
  - ADD r0, r1, r2 → r0 = r1 + r2
- Immediate operands
  - ADD r1, r1, #1 → r1 = r1 + 1
- Shifted register operands
  - ADD r3, r2, r1, LSL #3 → r3 = r2 + (r1 << 3)

Control flow instructions

- Branch instructions
  - B LABEL
  - BAL LABEL
- Conditional branches
  - BXX LABEL
    - BEQ, BNE, BPL, BMI, …
- Conditional execution
  - CMP r0, #5 → if (r0!= 5)
  - ADDNE r1, r1, r0 r1 = r1 + r0

Control flow instructions

- Branch and link instructions
  - BL SUBROUTINE → r14 = @next instr + jmp SUBR
  - PUSH {r0-r5, LR}
  - …
  - POP {r0-r5, PC}

Calling convention

- Arguments values
  - r0 → r3
- Local variables
  - r4 → r11
- Return value
  - r0

Summing it up

- Objective-C
  - [ObjectPointer Method:42]
- C++
  - ObjectPointer->Method(42)
- Pseudo C
  - objc_msgSend(ObjectPointer, "Method", 42)
- ARM assembly
  -
```
LDR     R1, =off_1337   ; offset to "Method"
MOV     R2, #0x2a
MOV     R0, R4          ; R4 contains Object @
BL      _objc_msgSend
```

AppStore binaries

First of all

- Forget about the simulator
  - Binaries compiled for x86 not ARM
  - Need to use a jailbroken iOS device
  - Tools to install
    - SSH
    - GDB
    - ...

Find'em

- Downloaded from the AppStore as .ipa
  - ZIP file
  - ~/Music/iTunes/iTunes Music/Mobile Applications/
- On iOS devices
  - /var/mobile/Applications/<UUID>/<AppName>.app/

Content of <AppName>.app*

```
`—> ls -l CodeCards.app                                                    [0]
total 720
-rwxr-xr-x  1 guru  staff  188144 Mar 23 16:32 CodeCards
drwxr-xr-x  6 guru  staff     204 Mar 23 16:32 CodeCards.momd
-rw-r--r--  1 guru  staff    2564 Mar 23 16:32 CodeResources
-rw-r--r--  1 guru  staff    7090 Mar 23 16:32 Icon-72.png
-rw-r--r--  1 guru  staff    5535 Mar 23 16:32 Icon.png
-rw-r--r--  1 guru  staff   11987 Mar 23 16:32 Icon@2x.png
-rw-r--r--  1 guru  staff     856 Mar 23 16:32 Info.plist
-rw-r--r--  1 guru  staff    5001 Mar 23 16:32 NoPhoto.png
-rw-r--r--  1 guru  staff       8 Mar 23 16:32 PkgInfo
-rw-r--r--  1 guru  staff     485 Mar 23 16:32 ResourceRules.plist
drwxr-xr-x  4 guru  staff     136 Mar 23 16:32 SC_Info
drwxr-xr-x  5 guru  staff     170 Mar 23 16:32 Settings.bundle
-rw-r--r--  1 guru  staff    1587 Mar 25 10:57 XML.Binary.Info.plist
drwxr-xr-x  3 guru  staff     102 Mar 23 16:32 _CodeSignature
-rw-r--r--  1 guru  staff    3994 Mar 23 16:32 delete.png
drwxr-xr-x  4 guru  staff     136 Mar 23 16:32 en.lproj
-rw-r--r--  1 guru  staff  107772 Mar 23 16:32 launch.png
drwxr-xr-x  4 guru  staff     136 Mar 23 16:32 lv.lproj
drwxr-xr-x  4 guru  staff     136 Mar 23 16:32 ru.lproj
-rw-r--r--  1 guru  staff    3684 Mar 23 16:32 thick.png
-rw-r--r--  1 guru  staff    2080 Mar 23 16:32 world.png
```

*after download from the device to workstation. Owner set to mobile:mobile on iOS

FAT binaries

- Binary might contain multiple versions
  - Need to extract the one corresponding to our device

```
$ lipo -thin armv6 CodeCards.app/CodeCards -output codecards.armv6
$ file codecards.armv6
codecards.armv6: Mach-O executable acorn
$ otool -h codecards.armv6
codecards.armv6:
Mach header
      magic cputype cpusubtype  caps    filetype ncmds sizeofcmds      flags
 0xfeedface      12          6  0x00          2    20       2756  0x00000085
```

Decrypt'em

- Encrypted using "FairPlay like" method
  - Each executable page is encrypted with AES and a MD5 checksum is computed
- How to know if a binary is encrypted ?
  - LC_ENCRYPTION_INFO
    - cryptid → 1 if the binary is encrypted
    - cryptoffset → offset of the encrypted data
    - cryptsize → size of the encrypted data

LC_ENCRYPTION_INFO

```
`—> otool -l codecards.armv6| grep LC_ENCRYPTION_INFO -B1 -A4
Load command 10
          cmd LC_ENCRYPTION_INFO
      cmdsize 20
    cryptoff  4096
    cryptsize 36864
    cryptid   1
```

Unpack the binary

- Use a script that automates the process
  - crackulous
  - Not leet enough;)
- "unpack your app in 5 steps and achieve peace"
  - Launch GDB
  - Set a breakpoint
  - Run the application
  - Extract the unencrypted executable code
  - Patch the architecture specific binary

Where do I set the breakpoint ?

- Execution steps
  - FAT binary is run
  - Architecture specific binary is mapped in memory
  - Executable code is decrypted
  - Branch to *start* symbol
- Get *start*'s address

```
`—> nm codecards.armv6 | grep start -B5                                [0]
0000bcd0 s _OBJC_IVAR_$_addEditCardsController.sections
0000bcdc s _OBJC_IVAR_$_addEditCardsController.webView
0000bca4 s _OBJC_IVAR_$_addEditCardsController.webViewController
0000c31c s _OBJC_IVAR_$_cardsListController.checkPin
0000c318 s _OBJC_IVAR_$_cardsListController.data
0000c320 s _OBJC_IVAR_$_cardsListController.startup
--
         U _objc_msgSendSuper2
         U _objc_msgSend_stret
         U _objc_setProperty
0000b044 s _pvars
         U dyld_stub_binder
00002000 T start
```

GDB, set, run

```
Milkmixs-iPhone: root# gdb --quiet CodeCards
(gdb) b *0x2000
Breakpoint 1 at 0x2000
(gdb) x /10i 0x2000
0x2000: cmnne   r9, #3604480    ; 0x370000
0x2004: strbgt  r2, [r8, #-125]
0x2008: adclt   r7, r0, #153    ; 0x99
0x200c: ldrbcc  r2, [r5, -r6, lsl #17]
0x2010: tst     r5, #159744     ; 0x27000
0x2014: subsmi  pc, r6, #8960   ; 0x2300
0x2018: stc     9, cr0, [r8, #424]!
0x201c: cfstr64pl       mvdx15, [r9], #168
0x2020: strtls  r12, [sp], #-3417
0x2024: mvnsle  r9, #4653056    ; 0x470000
(gdb) r
...
```

« Breakpoint reached capt'ain »

```
Breakpoint 1, 0x00002000 in ?? ()
(gdb) x /10i 0x2000
0x2000: ldr     r0, [sp]
0x2004: add     r1, sp, #4      ; 0x4
0x2008: add     r4, r0, #1      ; 0x1
0x200c: add     r2, r1, r4, lsl #2
0x2010: bic     sp, sp, #7      ; 0x7
0x2014: mov     r3, r2
0x2018: ldr     r4, [r3], #4
0x201c: cmp     r4, #0  ; 0x0
0x2020: bne     0x2018
0x2024: blx     0x202c
```

Extract the executable code

- Useful information
  - start
  - cryptsize
-
```
(gdb) dump memory decrypted.bin 0x2000 (0x2000 + 36864)
```

Patch the architecture specific binary

- Locate LC_ENCRYPTION_INFO
  - Mach-O header parser
  - Hexadecimal editor
- Replace cryptid
  - 1 → 0
- Replace encrypted code with unpacked one

Locate LC_ENCRYPTION_INFO

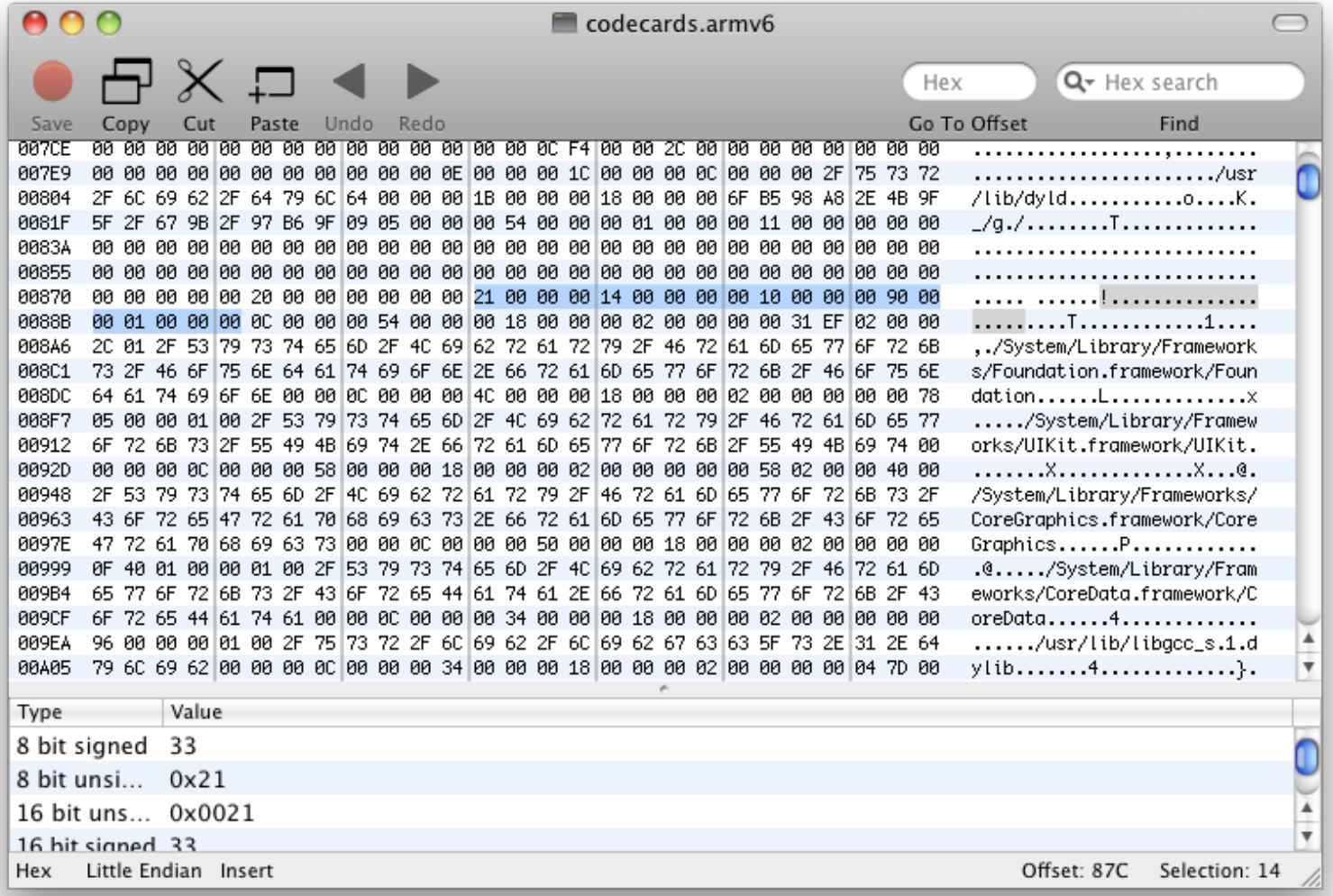
- Mach-O header parser

```
#include <mach-o/loader.h>
#include <mach-o/nlist.h>
#include <mach-o/fat.h>
```

- Search for the load command in the binary

```
cmd LC_ENCRYPTION_INFO          (21000000h)
      cmdsize 20                (14000000h)
 cryptoff   4096                (00100000h)
 cryptsize 36864                (00900000h)
 cryptid    1                   (01000000h)
```

Locate LC_ENCRYPTION_INFO



Modified LC_ENCRYPTION_INFO

```
$ otool -l codecards.decrypted.armv6| grep LC_ENC -A4
 cmd LC_ENCRYPTION_INFO
     cmdsize 20
 cryptoff  4096
 cryptsize 36864
 cryptid   0
```

Replace encrypted code

```
#!/usr/bin/env python

import sys

if len(sys.argv) != 4:
    print "usage: binpatch.py <file to patch> <offset> <blob>"
    exit (1)

topatch = open(sys.argv[1], "r+b")
offset = int(sys.argv[2])
blob = open(sys.argv[3], "rb")

topatch.seek(offset)
topatch.write(blob.read())
topatch.close()
```

Reverse'em

- Retrieve classes declarations
  - class-dump
- Resolve objc_msgSend calls
  - Useless call graph
  - Need to patch the disassembly

class-dump

```
@interface CodeCardsAppDelegate : NSObject <UIApplicationDelegate>
{
    UIWindow *window;
    mainController *controller;
    db *db1;
}

- (BOOL)application:(id)arg1 didFinishLaunchingWithOptions:(id)arg2;
- (void)applicationDidEnterBackground:(id)arg1;
- (void)applicationWillResignActive:(id)arg1;
- (void)applicationWillTerminate:(id)arg1;
- (void)dealloc;

@end
```

First look at the disassembly

```
CODE16


; Attributes: bp-based frame

_main
PUSH    {R4-R7,LR}
ADD     R7, SP, #0xC
MOVS    R4, R0
MOVS    R6, R1
LDR     R0, =(off_CA3C - 0x203C)
LDR     R1, =(off_C668 - 0x203E)
ADD     R0, PC
ADD     R1, PC
LDR     R0, [R0]
LDR     R1, [R1]
BLX     _objc_msgSend
LDR     R1, =(off_C664 - 0x204A)
ADD     R1, PC
LDR     R1, [R1]
BLX     _objc_msgSend
LDR     R3, =(cfstr_Codecardsappde - 0x2058)
MOVS    R1, R6
MOVS    R2, #0
ADD     R3, PC          ; "CodeCardsAppDelegate"
MOVS    R5, R0
MOVS    R0, R4
BLX     _UIApplicationMain
LDR     R1, =(off_C660 - 0x2064)
ADD     R1, PC
LDR     R1, [R1]
MOVS    R4, R0
MOVS    R0, R5
BLX     _objc_msgSend
MOVS    R0, R4
POP     {R4-R7,PC}
; End of function _main
```

objc_msgSend

- As stated before
  - objc_msgSend(<ref to object>, @selector(method), …)
  - ARM calling convention
    - arg1 → r0
    - arg2 → r1
- Backtrace calls to objc_msgSend
  - By hand
  - Using Zynamics IDAPython scripts

objc_helper.py

```
CODE16

; Attributes: bp-based frame

_main
PUSH    {R4-R7,LR}
ADD     R7, SP, #0xC
MOVS    R4, R0
MOVS    R6, R1
LDR     R0, =(off_CA3C - 0x203C)
LDR     R1, =(off_C668 - 0x203E)
ADD     R0, PC
ADD     R1, PC
LDR     R0, [R0]
LDR     R1, [R1]
BLX     __NSAutoreleasePool_alloc_
LDR     R1, =(off_C664 - 0x204A)
ADD     R1, PC
LDR     R1, [R1]
BLX     ___init_
LDR     R3, =(cfstr_Codecardsappde - 0x2058)
MOVS    R1, R6
MOVS    R2, #0
ADD     R3, PC          ; "CodeCardsAppDelegate"
MOVS    R5, R0
MOVS    R0, R4
BLX     _UIApplicationMain
LDR     R1, =(off_C660 - 0x2064)
ADD     R1, PC
LDR     R1, [R1]
MOVS    R4, R0
MOVS    R0, R5
BLX     ___release_
MOVS    R0, R4
POP     {R4-R7,PC}
; End of function _main
```

What to look for

Where to start

- Locate the main class
  - UIApplicationDelegate
    - applicationDidFinishLaunching
    - ApplicationDidFinishLaunchingWithOptions
- Views
  - UI*ViewController
    - viewDidLoad

applicationDidFinishLaunching

```
@interface CodeCardsAppDelegate : NSObject <UIApplicationDelegate>
{
    UIWindow *window;
    mainController *controller;
    db *db1;
}

- (BOOL)application:(id)arg1 didFinishLaunchingWithOptions:(id)arg2;
- (void)applicationDidEnterBackground:(id)arg1;
- (void)applicationWillResignActive:(id)arg1;
- (void)applicationWillTerminate:(id)arg1;
- (void)dealloc;
```

```
; Attributes: bp-based frame

__CodeCardsAppDelegate_application_didFinishLaunchingWithOptions__

var_2C= -0x2C
var_28= -0x28
var_24= -0x24
var_20= -0x20

PUSH    {R4-R7,LR}
MOV     R6, R11
MOV     R5, R10
MOV     R4, R8
PUSH    {R4-R6}
ADD     R7, SP, #0x18
SUB     SP, SP, #0x1C
LDR     R1, =(off_C6D8 - 0x2188)
MOV     R8, R0
LDR     R0, =(off_CA40 - 0x218C)
ADD     R1, PC
LDR     R5, =(cfstr_Checkregionfor - 0x219A)
ADD     R0, PC
LDR     R1, [R1]
LDR     R0, [R0]
BLX     __NSUserDefaults_standardUserDefaults_
LDR     R1, =(off_C6D4 - 0x219C)
LDR     R2, =(cfstr_Yes - 0x21A0)
ADD     R5, PC          ; "checkRegionFormat"
ADD     R1, PC
MOVS    R3, R5
ADD     R2, PC          ; "YES"
LDR     R1, [R1]
LDR     R6, =(cfstr_Applelanguages - 0x21A6)
ADD     R6, PC          ; "AppleLanguages"
MOVS    R4, R0
LDR     R0, =(off_CA44 - 0x21AC)
ADD     R0, PC
LDR     R0, [R0]
BLX     __NSDictionary_dictionaryWithObject_forKey__
LDR     R1, =(off_C6D0 - 0x21B6)
ADD     R1, PC
LDR     R1, [R1]
MOVS    R2, R0
MOVS    R0, R4
BLX     ___registerDefaults__
LDR     R1, =(off_C6CC - 0x21C6)
MOVS    R0, R4
ADD     R1, PC
LDR     R1, [R1]
MOV     R11, R1
BLX     ___synchronize_
LDR     R1, =(off_C6C8 - 0x21D6)
```

Remote connections

- HTTP(S)
  - NSURL
  - ...
- Sockets
  - CFSocketCreate
  - ...

```
MOV     R0, R8
BLX     __NSObject_encode__
LDR     R1, =off_19864
LDR     R2, =cfstr_Http10_0_2_1Ju ; "http://10.0.2.1/julien/iphone.php?data=%@"
LDR     R1, [R1]
MOVS    R3, R0
LDR     R0, =off_19E1C
LDR     R0, [R0]
BLX     __NSString_stringWithFormat__
LDR     R1, =off_1985C
LDR     R4, [R1]
LDR     R1, =off_19860
LDR     R1, [R1]
MOVS    R2, R0
LDR     R0, =off_19E20
LDR     R5, [R0]
LDR     R0, =off_19E24
LDR     R0, [R0]
BLX     __NSURL_URLWithString__
LDR     R1, =0x40340000
MOVS    R3, #4
MOVS    R2, R0
MOVS    R0, #0
STR     R0, [SP,#0x1C+var_1C]
STR     R1, [SP,#0x1C+var_18]
MOVS    R1, R4
MOVS    R0, R5
BLX     __NSURLRequest_requestWithURL_cachePolicy_timeoutInterval__
LDR     R1, =off_19858
MOVS    R3, #0
STR     R3, [SP,#0x1C+var_14]
STR     R3, [SP,#0x1C+var_1C]
LDR     R1, [R1]
ADD     R3, SP, #0x1C+var_14
MOVS    R2, R0
LDR     R0, =off_19E28
LDR     R0, [R0]
BLX     __NSURLConnection_sendSynchronousRequest_returningResponse_error__
ADD     SP, SP, #0xC
POP     {R2}
MOV     R8, R2
POP     {R4-R7,PC}
; End of function __SpyPhoneAppDelegate_applicationDidFinishLaunching__
```

Data protection

- Accessing the KeyChain using JB tools
  - *Lost iPhone ? Lost Passwords ! **
- Protect KeyChain content
  - Using passcode
  - setAttributes ofItemAtPath → *NSFileProtectionComplete*
  - SecItemAdd → *kSecAttrAccessibleWhenUnlocked*

* http://www.sit.fraunhofer.de/forschungsbereiche/projekte/Lost_iPhone.jsp

Data protection

```
PUSH            {R7,LR}
ADD             R7, SP, #0
SUB             SP, SP, #0x1C
LDR             R3, =(_kSecClassGenericPassword_ptr - 0x2ADE)
LDR.W           R12, =(_kSecAttrService_ptr - 0x2AE0)
LDR             R0, =(off_1ADA0 - 0x2AEA)
ADD             R3, PC
ADD             R12, PC
LDR             R3, [R3]
LDR.W           R12, [R12]
LDR             R1, =(off_1A7D0 - 0x2AF2)
ADD             R0, PC
LDR.W           LR, [R3]
LDR             R3, =(_kSecClass_ptr - 0x2AF6)
ADD             R1, PC
LDR             R0, [R0]
ADD             R3, PC
LDR             R1, [R1]
LDR             R3, [R3]
LDR             R3, [R3]
STR             R2, [SP,#0x1C+var_1C]
LDR.W           R12, [R12]
STR             R2, [SP,#0x1C+var_14]
LDR             R2, =(_kSecAttrAccount_ptr - 0x2B0C)
STR.W           R12, [SP,#0x1C+var_18]
ADD             R2, PC
LDR             R2, [R2]
LDR             R2, [R2]
STR             R2, [SP,#0x1C+var_10]
LDR             R2, =(_kSecAttrAccessibleAfterFirstUnlock_ptr - 0x2B16)
ADD             R2, PC
LDR             R2, [R2]
LDR             R2, [R2]
STR             R2, [SP,#0x1C+var_C]
LDR             R2, =(_kSecAttrAccessible_ptr - 0x2B20)
ADD             R2, PC
LDR             R2, [R2]
LDR             R2, [R2]
STR             R2, [SP,#0x1C+var_8]
MOVS            R2, #0
STR             R2, [SP,#0x1C+var_4]
MOV             R2, LR
BLX             __NSMutableDictionary_dictionaryWithObjectsAndKeys__
SUB.W           SP, R7, #0
POP             {R7,PC}
```

Conclusion

Conclusion

- *This is a revolution !*
- This presentation was only an introduction
  - Lot of work/ideas around iOS
  - Grab your debugger and disassembler and work on it
  - I'm open to discuss it around a few beers