23-25 October 2012
HACK.LU
It can only be attributable to human error.

GOTO: H[a]CK

SOGETI

Hacking iOS Applications

Is your company data safe when stored on idevices ?

Mathieu RENARD - @GOTOHACK
mathieu.renard[-at-]gotohack.org
mathieu.renard[-at-]sogeti.com

Working with a regular device
[Not jailbroken]

Attack vectors : Regular device

USB: AFC

Applications

Network

Bluetooth

WiFi

Simcard

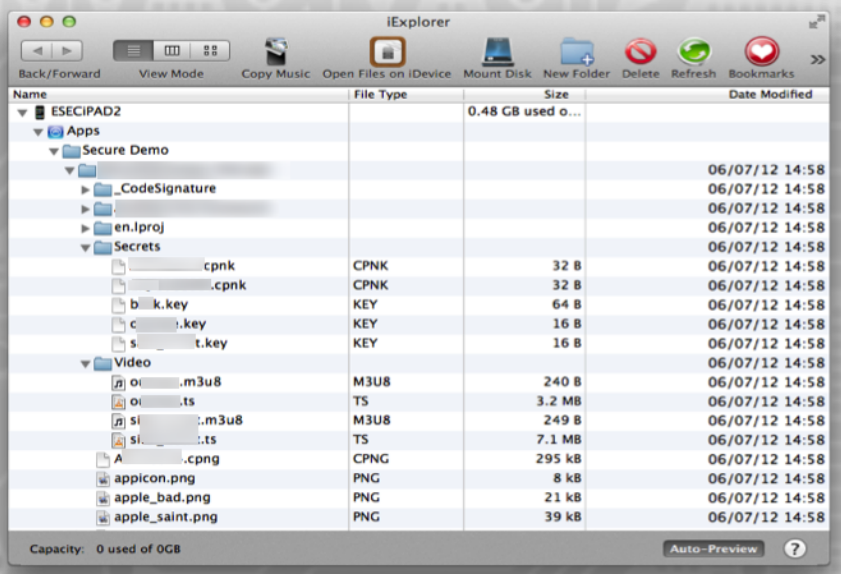
Backups

Baseband

Abusing AFC protocol

SOGETI

- AFC (Apple File Connection)
  - Service running on all iDevices
  - Handled by /usr/libexec/afcd
  - Used by iTunes to exchange files
  - AFC clients can access certain files only
    - Files located in the Media folder
    - User installed applications folders
  - Implemented in libiMobileDevice

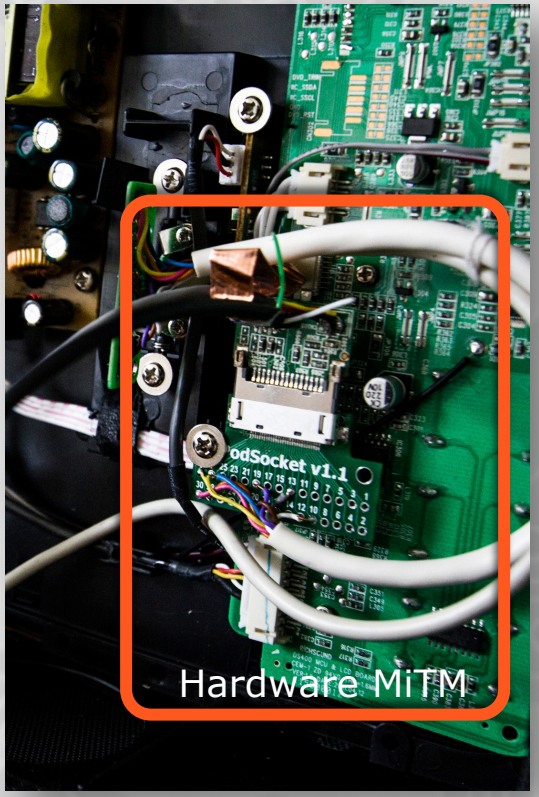


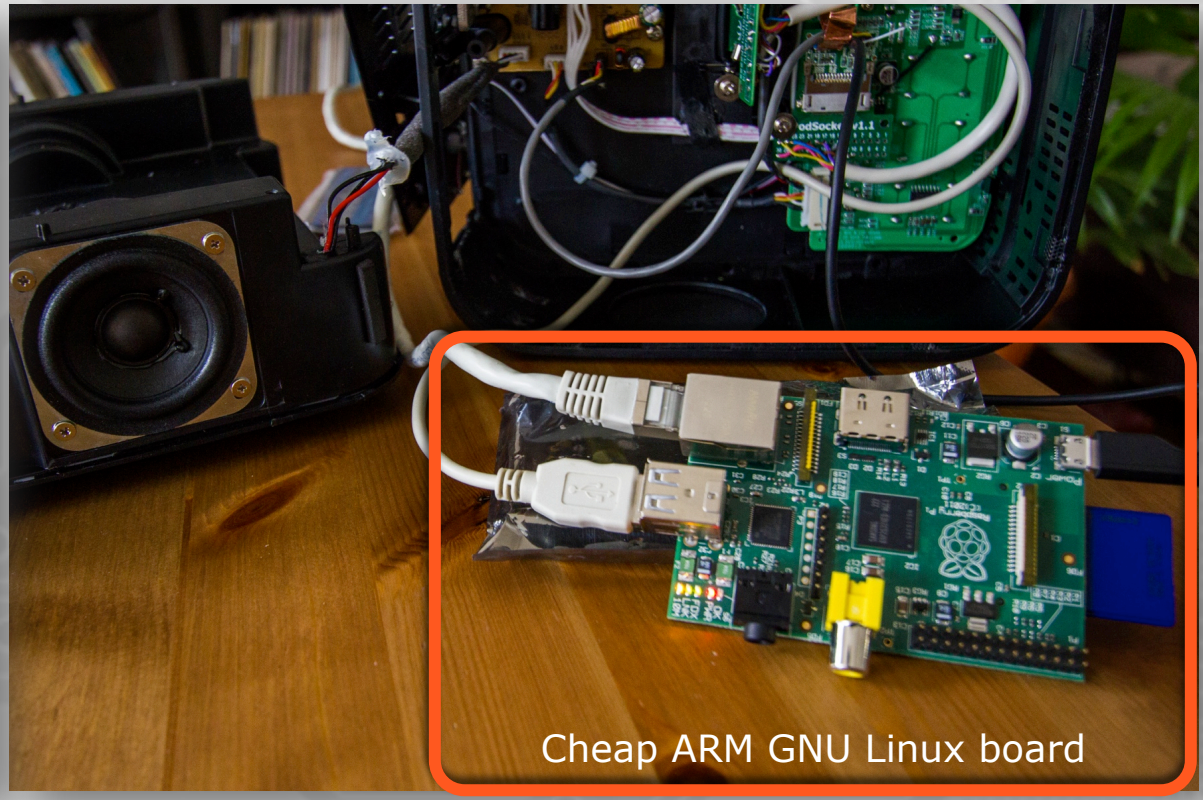
- What you can do
  - Access to default pref file
  - Access app resources
  - **Only if the iDevice unlocked**

iPown Dock & Evil Maid…

- Nowadays Dock station are used a lot…
  - Hotel room
  - Supermaket
  - ….

Hardware backdoor
How I put the evil inside...





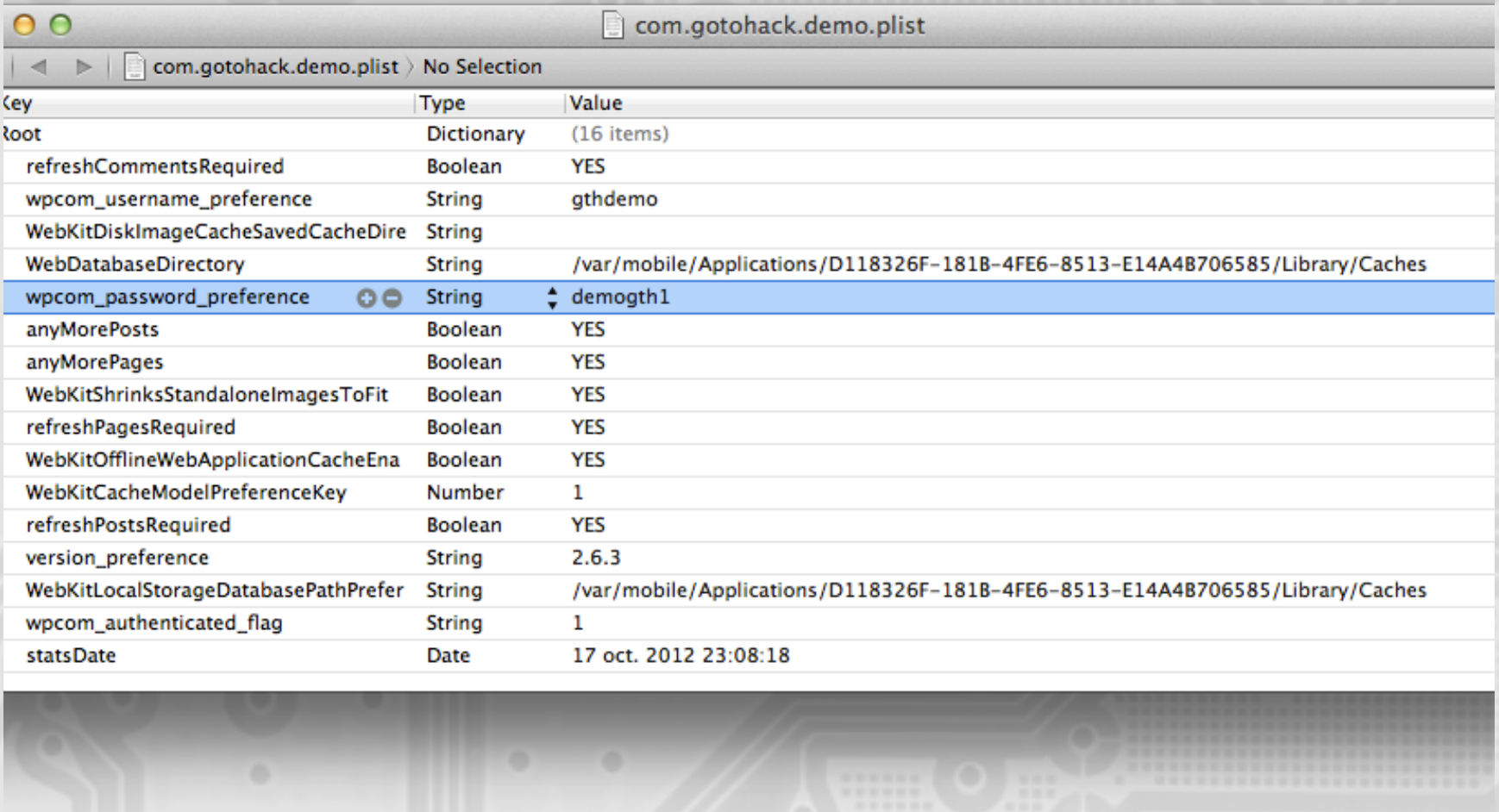
This dock station is now powered by

iPown Dock & Evil Maid…

SOGETI

Demo

Unsecure credential storage



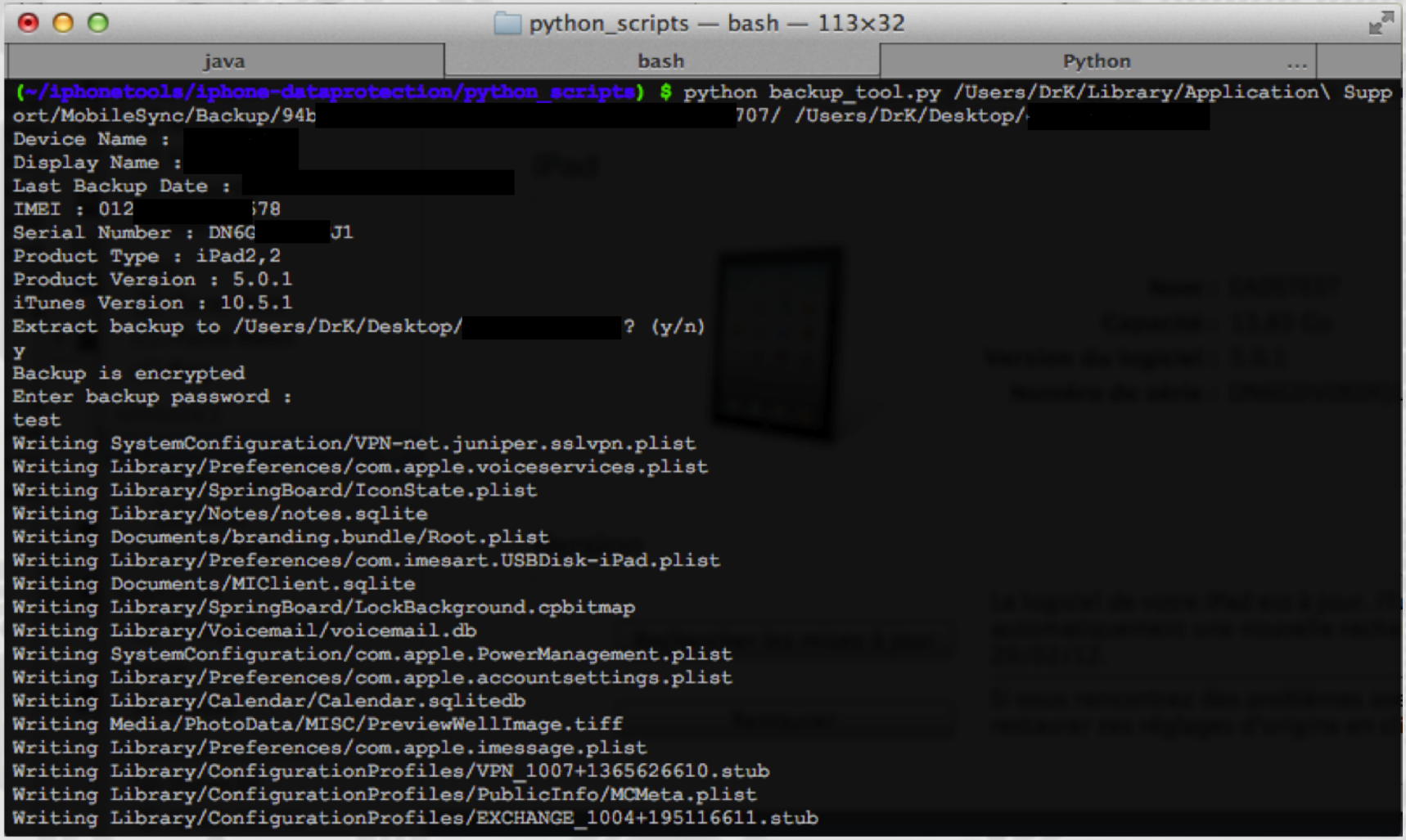
Key	Type	Value
Root	Dictionary	(16 items)
refreshCommentsRequired	Boolean	YES
wpcom_username_preference	String	gthdemo
WebKitDiskImageCacheSavedCacheDire	String	
WebDatabaseDirectory	String	/var/mobile/Applications/D118326F-181B-4FE6-8513-E14A4B706585/Library/Caches
wpcom_password_preference	String	demogth1
anyMorePosts	Boolean	YES
anyMorePages	Boolean	YES
WebKitShrinksStandaloneImagesToFit	Boolean	YES
refreshPagesRequired	Boolean	YES
WebKitOfflineWebApplicationCacheEna	Boolean	YES
WebKitCacheModelPreferenceKey	Number	1
refreshPostsRequired	Boolean	YES
version_preference	String	2.6.3
WebKitLocalStorageDatabasePathPrefer	String	/var/mobile/Applications/D118326F-181B-4FE6-8513-E14A4B706585/Library/Caches
wpcom_authenticated_flag	String	1
statsDate	Date	17 oct. 2012 23:08:18

Having fun with backups

Backup storage
- %APPDATA%/Apple Computer/MobileSync/Backup/<udid>
- Can be password protected
- Encrypted (AES-256 CBC)
- Filenames : SHA1 hashes

Using iPhoneDataProtection Framework
- Developed by Jean SIGWALD – Sogeti ESEC Lab
- Bruteforce backup password [require some scripting skills] [Extremely slow]
  - I do recommend Elcomsoft Phone Password Breaker (35 000 pwd/s on GPU)
- Extract backup content
- Extract keychain stored data

Having Fun With backups



iOS Keychain

- Almost the only place to store critical data:
  - Crypto keys
  - Credentials
  - …

- Apple defined 6 values to define when a keychain item should be readable
  - **kSecAttrAccessibleAfterFirstUnlock**
  - **kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly**
  - **kSecAttrAccessibleAlways**
  - **kSecAttrAccessibleAlwaysThisDeviceOnly**
  - **kSecAttrAccessibleWhenUnlocked**
  - **kSecAttrAccessibleWhenUnlockedThisDeviceOnly**

iOS Keychain

SOGETI

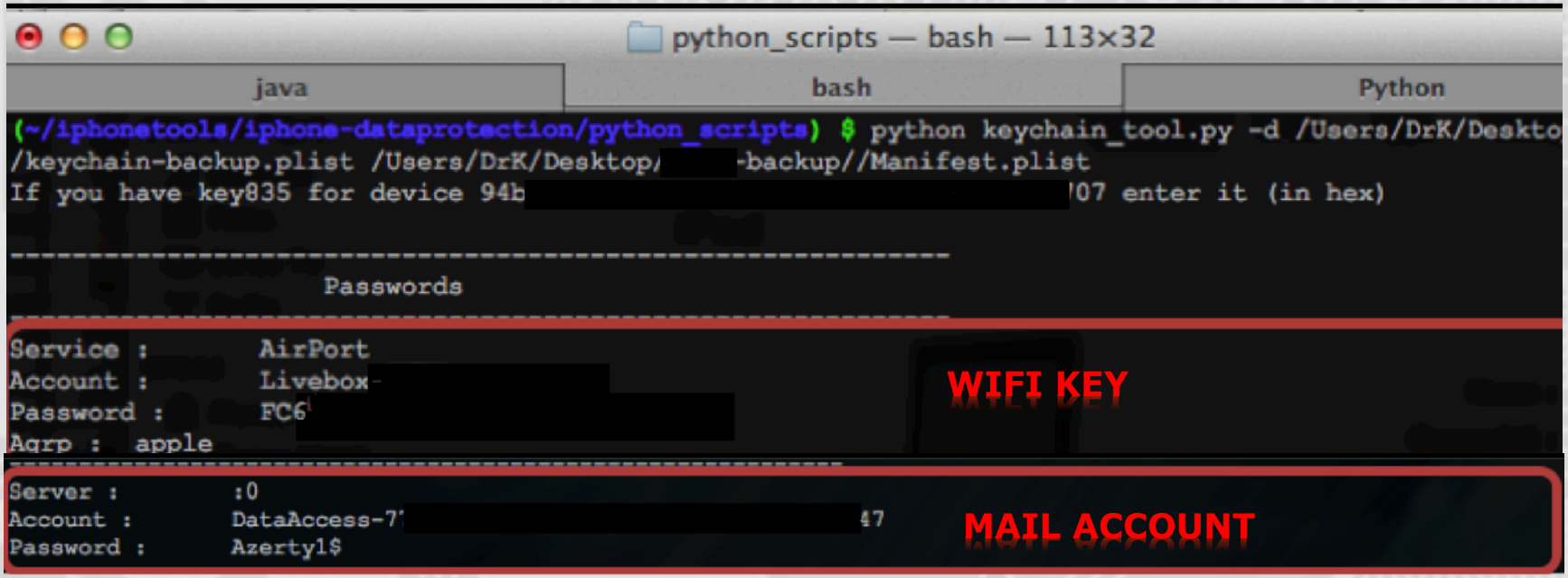
Protection class for built-in application items

Can be extracted without jailbreak

Item	Accessibility
Wi-Fi passwords	Always
IMAP/POP/SMTP accounts	AfterFirstUnlock
Exchange accounts	Always
VPN	Always
LDAP/CalDAV/CardDAV accounts	Always
iTunes backup password	WhenUnlockedThisDeviceOnly
Device certificate & private key	AlwaysThisDeviceOnly

Extraction requires the 0x835 hardware key => Jailbreak is mandatory

Extracting Keychain data



Analyzing network connexion

SOGETI

Remote virtual interface

- When enabled all network traffic is mirrored to this interface
  - No need to jailbreak the device
  - Does not allow SSL interception

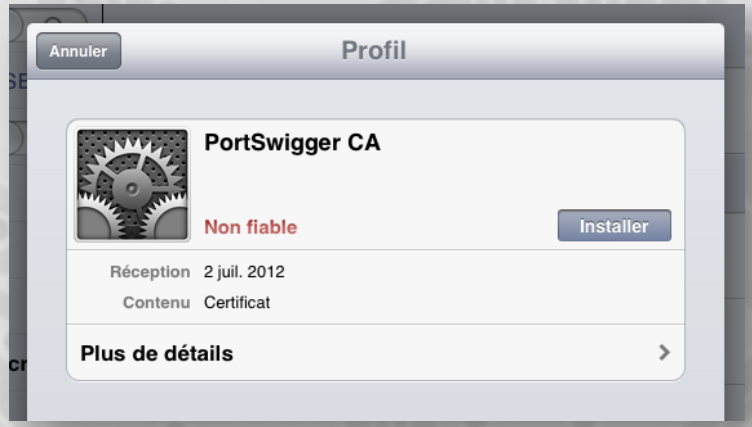
- **Mac OS**
  - Connect the device over usb
  - Get the device ID
  - Launch rvictl –s <UID>
  - Launch wireshark

- **Other OS**
  - com.apple.pcapd & usbmux

```
$ tcpdump -n -t -i rvi0 -q tcp
tcpdump: WARNING: rvi0: That device doesn't support promiscuous mode
(BIOCPROMISC: Operation not supported on socket)
tcpdump: WARNING: rvi0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on rvi0, link-type RAW (Raw IP), capture size 65535 bytes
IP 192.168.1.66.55101 > 192.168.1.64.51712: tcp 117
IP 192.168.1.64.51712 > 192.168.1.66.55101: tcp 0
```

HTTPS trafic interception

- Like other web applications
  - Launch your proxy (Burp, Charles, Paros,…)
  - Setup the proxy on the device
  - If the application check for certificate validity
  - Extract your proxy CA and install it on the device
    - Link-it on a web page
    - Download the CA and install it



Let's Jailbreak* the device

*BIG UP For the Jailbreak dream team!

Attack vectors : Jailbroken device

USB: AFC

Applications

Network

Bluetooth

WiFi

Simcard

Backups

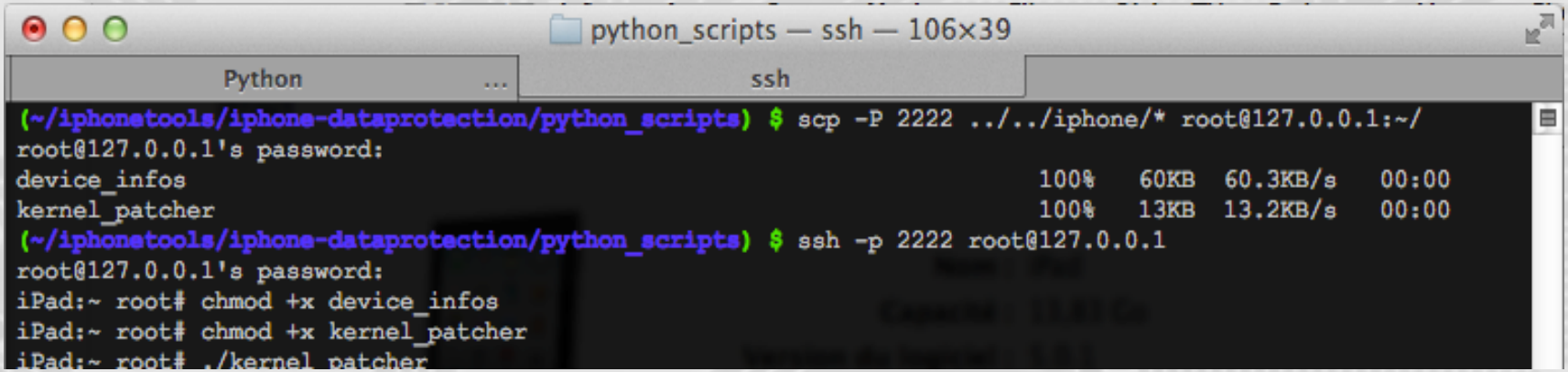
Baseband

Jailbroken device

Jailbreaking allows
- root access to the operating system
- downloading & installing new apps
  - Additional applications (ssh, gdb, …)
  - Retrieve application and data stored on the device
  - Retrieve all data stored in the Keychain
    - We can extract the 0x835 hardware key
- Decrypting and reversing the application

Getting the 0x835 Key

- Getting 0x835 key on jailbroken device
  - Kernel_patcher
    - By default accessing to the hardware keys form user land is forbidden)
  - Device_info
    - Extracting hardware keys



Reversing iOS Applications

iOS Binaries : ARM

ARM7

ARM7

ARM7s

RISC
Load-store architecture
32-bit (ARM) & 16-bit (Thumb) instruction sets

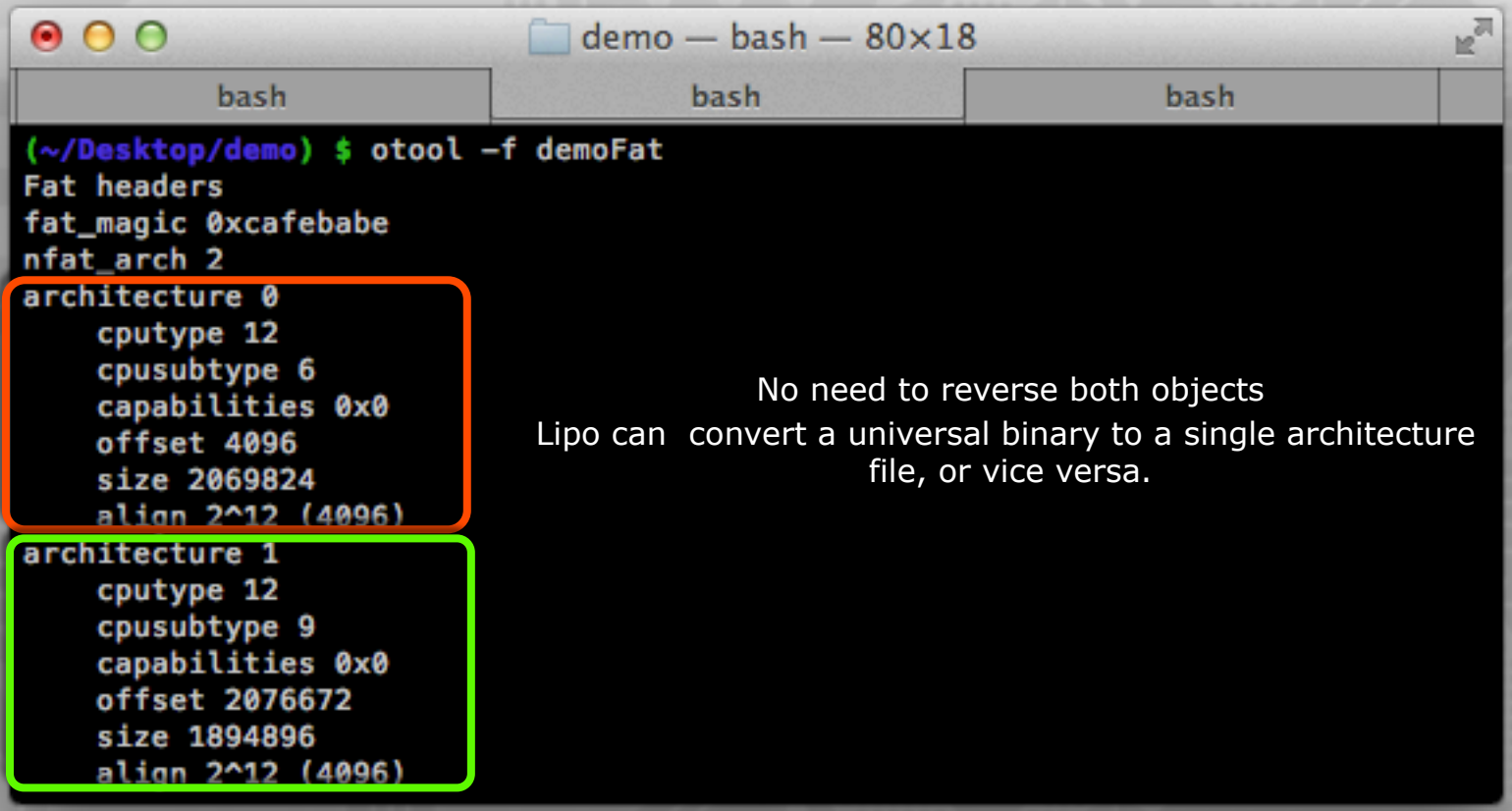
Registers
- R0-R3 > Used to pass params
- R7 > Frame pointer
- R13 > SP, Stack Pointer
- R14 > LR, Link register
- R15 > PC, Program counter

CPSR Current Program Status Register
- N > Negative
- Z > Zero
- C > Carry
- V > Overflow

iOS Binaries : Fat & Thin

SOGETI

- Some executable are fat binaries
  - They contain multiple mach objects within a single file
    - Each one for a different architecture or platform



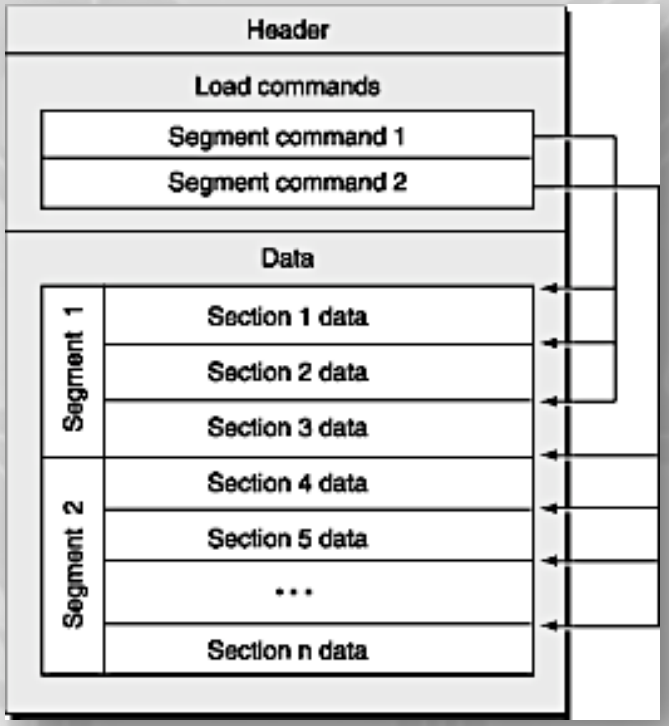
No need to reverse both objects

Lipo can convert a universal binary to a single architecture file, or vice versa.

iOS Binaries : Mach-O

SOGETI

- Contains three parts
  - **Header**
  - **Load commands**
  - **Data**



- Header
  - Magic
  - Cputype
  - Cpusubtype
  - Filetype
  - Ncmds
  - Sizeofcmds
  - Flags

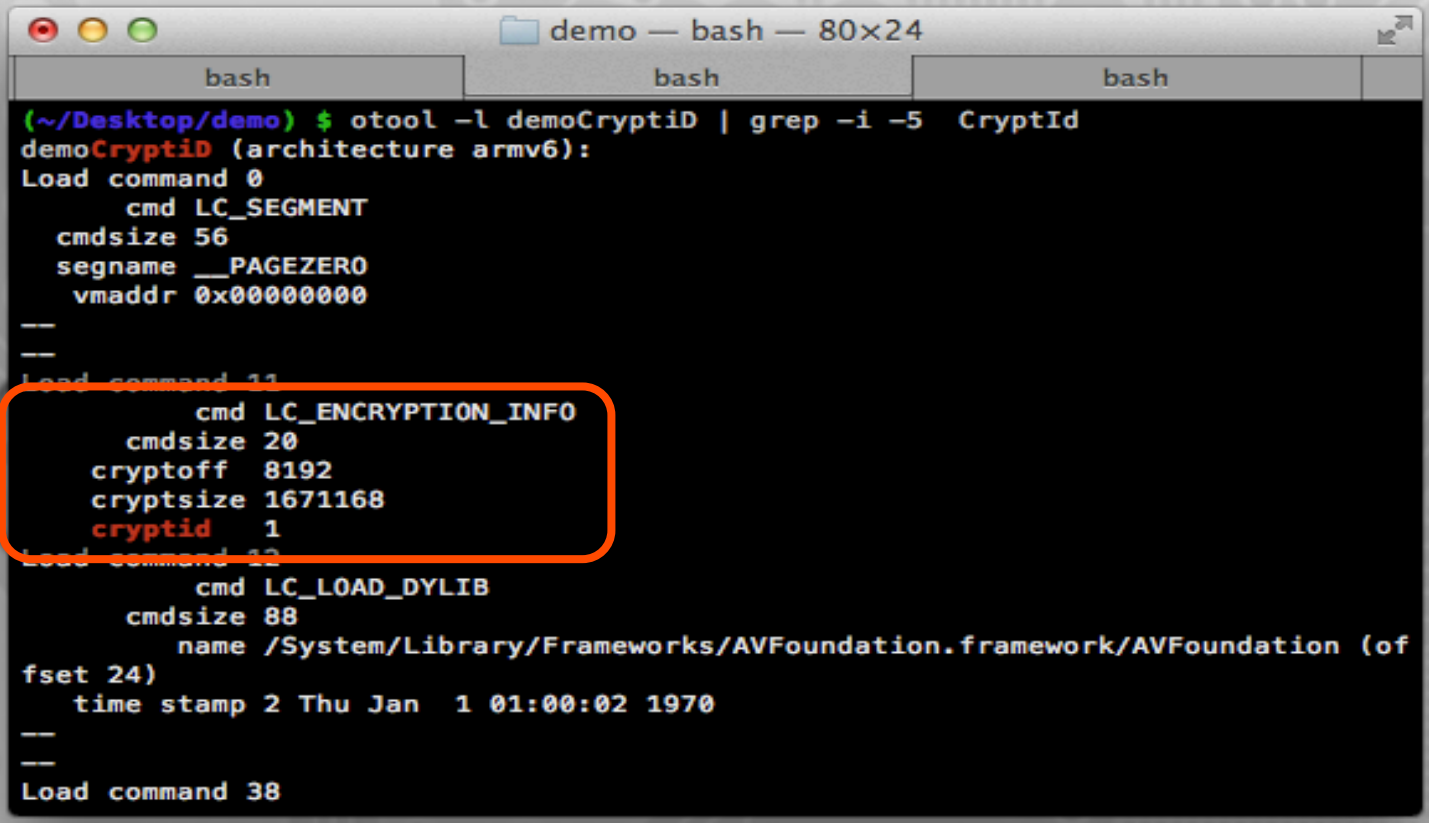
- Data
  - Segments sections
  - __PAGEZERO
  - __TEXT
  - __DATA
  - Rw-
  - **__OBJC**
  - ...

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

iOS Binaries : Cryptid

SOGETI

- Load commands & cryptid



Defeating Fairplay Encryption

- Manually using GDB
  - Launch GDB
  - Set a breakpoint
  - Run the application
  - Extract the unencrypted executable code
  - Patch the architecture specific binary

```
$CryptSize=1671168
$CryptOff=8192
echo -e "set sharedlibrary load-rules \".*\" \".*\" none\r\n\
set inferior-auto-start-dyld off\r\n\
set sharedlibrary preload-libraries off\r\n\
set sharedlibrary load-dyld-symbols off\r\n\
dump memory dump.bin $(($CryptOff + 4096)) $(($CryptSize + $CryptOff + 4096))\r\n\
kill\r\n\
quit\r\n" > batch.gdb

gdb -q -e demoCryptId -x batch.gdb -batch
```

Defeating Fairplay Encryption

SOGETI

Lamers way : Using Crackulous (Angel)

- With only one click
  - Decrypt apps & Unset CryptID
  - Provide fully functional cracked ipa
  - Generate credit file.
  - Automatic uploading
  - Automatic submission
- Bug
  - **Does not handle Thin binaries**

Defeating Fairplay Encryption

SOGETI

The smart way : Dumpdecrypted (i0n1c)

```
iPod:~ root# DYLD_INSERT_LIBRARIES=dumpdecrypted.dylib /var/mobile/Applications/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/Scan.app/Scan
mach-o decryption dumper

DISCLAIMER: This tool is only meant for security research purposes, not for application crackers.

[+] Found encrypted data at address 00002000 of length 1826816 bytes - type 1.
[+] Opening /private/var/mobile/Applications/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/Scan.app/Scan for reading.
[+] Reading header
[+] Detecting header type
[+] Executable is a FAT image - searching for right architecture
[+] Correct arch is at offset 2408224 in the file
[+] Opening Scan.decrypted for writing.
[-] Failed opening. Most probably a sandbox issue. Trying something different.
[+] Opening /private/var/mobile/Applications/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/tmp/Scan.decrypted for writing.
[+] Copying the not encrypted start of the file
[+] Dumping the decrypted data into the file
[+] Copying the not encrypted remainder of the file
[+] Closing original file
[+] Closing dump file
```

Analyzing __OBJC Segment

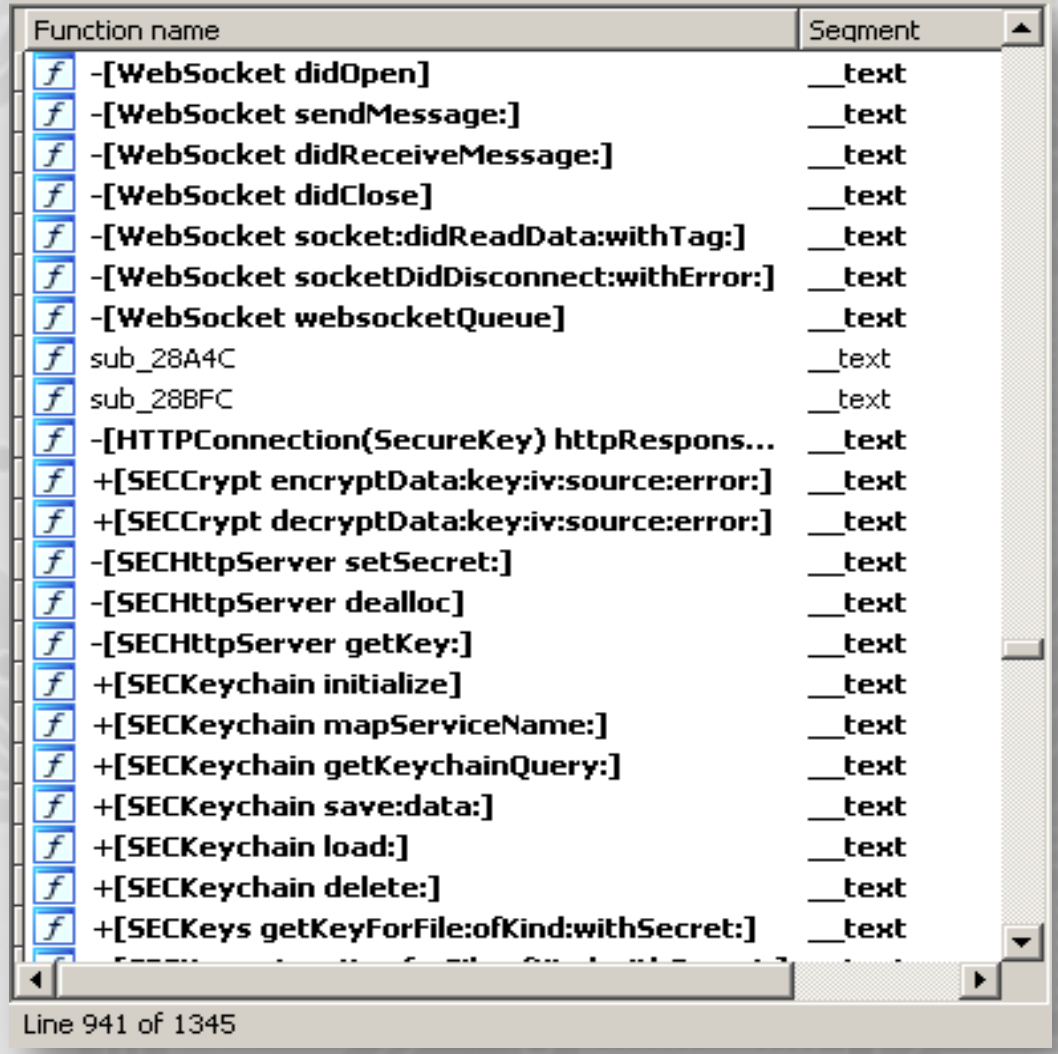
- __OBJC
  - **__objc_classlist** : list of all classes for which there is an implementation in the binary.
  - **__objc_classref** : references to all classes that are used by the application.

- By parsing these section it is possible to retrieve classes and methods prototypes

Introducing Classdump


```
@property(nonatomic) __weak UITextField *textPass1; // @synthesize textPass1=_textPass1;
- (void).cxx_destruct;
- (void)authentifierAndChangeView:(id)arg1;
- (void)didReceiveMemoryWarning;
- (void)viewDidLoad;
- (BOOL)isPasswordValid:(id)arg1;
- (id)mainViewController;
- (id)dao;

@end

@interface SqliteDao : NSObject
{
    struct sqlite3 *database;
    NSString *databasePath;
}

@property struct sqlite3 *database; // @synthesize database;
@property(retain) NSString *databasePath; // @synthesize databasePath;
- (void).cxx_destruct;
- (void)closeDatabase;
- (int)openCryptoDatabase:(id)arg1;
- (void)addUser:(id)arg1:(id)arg2:(id)arg3:(id)arg4:(id)arg5:(id)arg6;
- (id)getPassword:(id)arg1;
- (int *)getNombreUsers;

@end
```

Introducing IDA Pro



Objective-C

Calling convention

- C++
  - ObjectPointer->Method(param1, param2)
- Objective-C
  - [ObjectPointer Method:param1 param2Name:param2
    - **objc_msgSend(ObjectPointer, @selector(Method))**
- ARM calling convention
  - Arg1: **ObjectPointer** → r0
  - Arg2: **@selector(Method)** → r1
- Backtracing calls to objc_msgSend
  - By hand
  - Using Zynamics IDAPython script
  - IDA Pro > 6.1

Where to start ?

Where to start ?
- Locate the main class
  - UIApplicationDelegate
    - ApplicationDidFinishLaunching
    - ApplicationDidFinishLaunchingWithOptions
  - Locate views inititialisation
    - UI*ViewController
      - » ViewDidLoad

Where to look ?
- URL > NSURL*
- Socket > CFSocket*
- Keychain > ksecAttr*, SecKeychain*
- Files Handling > NSFileManager*
- Crypto > CCCrypt*

Hooking? Yes there is an app for that...

Hooking made easy: MobileSubstrate

MobileSubstrate

- Allows developers to provide run-time patches
  - MobileLoader will first load itself into the run application using DYLD_INSERT_LIBRARIES
  - Looks for all dynamic libraries in the directory /Library/MobileSubstrate/DynamicLibraries/ and load them.

- MobileHooker is used to replace system functions
  - MSHookMessageEx()
    - Replace the implementation of the Objective-C message [class selector] by replacement, and return the original implementation..
  - MSHookFunction()
    - like MSHookMessageEx() but is for C/C++ functions.

DEMO: Stealing Crypto keys

CCCrypt(3cc) API

```
CCCrypt(CCOperation op, CCAlgorithm alg, CCOptions options, const void *key,
        size_t keyLength, const void *iv, const void *dataIn, size_t dataInLength,
        void *dataOut, size_t dataOutAvailable,size_t *dataOutMoved);
```

Hooking

```
CCCryptorStatus hk_CCCrypt(CCOperation op, CCAlgorithm alg, CCOptions options,
                           const void *key, size_t keyLength, const void *iv,
                           const void *dataIn, size_t dataInLength, void *dataOut,
                           size_t dataOutAvailable, size_t *dataOutMoved){

    NSLog(@"CryptoTheft> CCCrypt(%d,%d,%d,%s,%s)", op, alg, options, key, iv);
    return old_CCCrypt(op, alg, options, key, keyLength, iv, dataIn, dataInLength, dataOut,
                       dataOutAvailable, dataOutMoved);
}

__attribute__((constructor)) static void initialize() {
    MSHookFunction(CCCrypt, hk_CCCrypt, (void**)&old_CCCrypt);
}
```

The Truth about Jailbreak detection

[The Good, The Bad, The ….]

Jailbreak detection classic checking for shell [The good]

SOGETI

Checking for shell

```
+ (BOOL)doShell {
        if (system(0)) {
                return YES;
        }
    return NO;
}
```

Bypassing the check

```
static int (*old_system)(char *) = NULL;
int st_system(char * cmd){
                if (!cmd){
                                return nil;
                }
                return old_system(cmd);
}
 __attribute__((constructor)) static void initialize() {
                NSLog(@"StealthJBInitialize!");
                MSHookFunction(system, st_system, &old_system);
}
```

Jailbreak detection Classics
Jailbreak files detection [The bad]

SOGETI

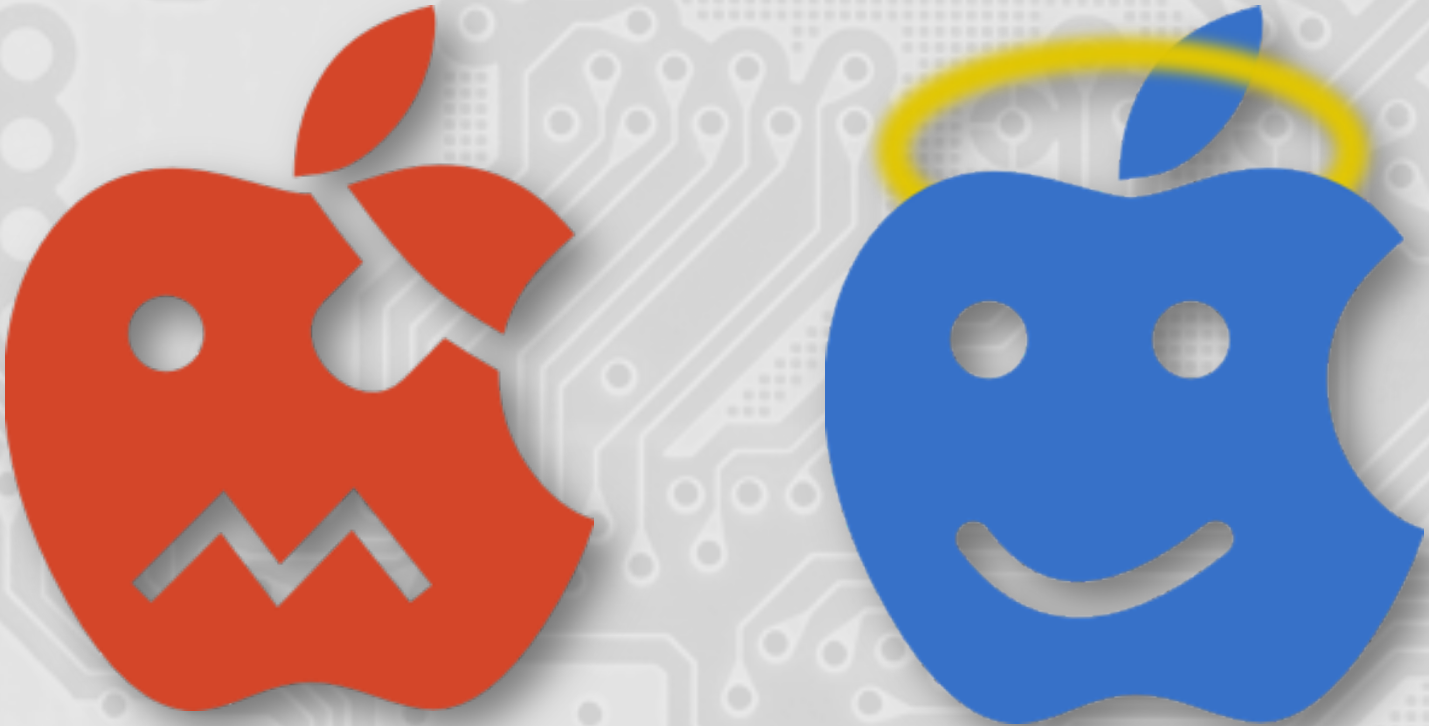
- Checking for jailbreak files (Cydia, SSH, MobileSubstrate, Apt, …)

```
+ (BOOL)doCydia {
    if ([[NSFileManager defaultManager]
      fileExistsAtPath: @"/Applications/Cydia.app"]){
      return YES;
    }
    return NO;
}
```

- Bypassing the check (hooking NFSFileManager)

```
void* (*old_fileExistsAtPath)(void* self, SEL _cmd,NSString* path) = NULL;
void* st_fileExistsAtPath(void* self, SEL _cmd, NSString* path){
   if ([path isEqualToString:@"/Applications/Cydia.app"){
      NSLog(@"=>hiding %@", path);
      return 0;
   }
   return old_fileExistsAtPath(self,_cmd,path);
}

__attribute__((constructor)) static void initialize() {
            MSHookMessageEx([NSFileManager class], @selector(fileExistsAtPath:),
                                    (IMP)st_fileExistsAtPath, (IMP *)&old_fileExistsAtPath);
}
```

DEMO: Bypassing jailbreak detection



Jailbreak detection classics [The ...]

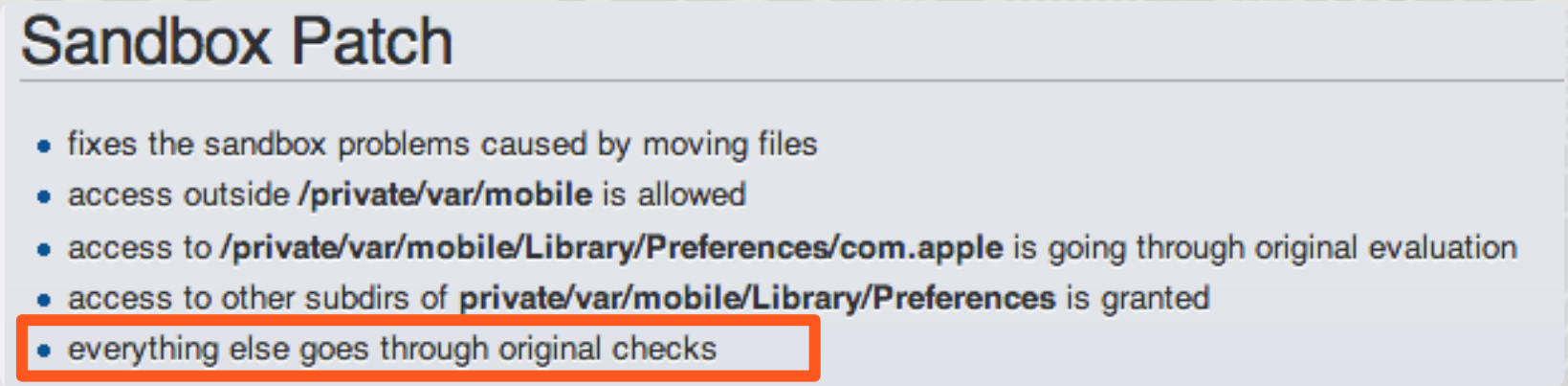
- Sandbox check using fork
- Documented in some books and blog posts
  - If the process can fork, the device is jailbroken.

```
+(BOOL) doFork () {
    int res = fork();
    if (!res) {
        exit(0);
    }

    if (res >= 0) {
                #if TARGET_IPHONE_SIMULATOR
                    NSLog("fork_check -> Running on the simulator!");
                    return 0;
                #else
                    return 1;
                #endif
    }
    return 0;
}
```

Jailbreak detection classics [The fail!]

SOGETI

From the iphonewiki



For further info see

- https://github.com/comex/datautils0/blob/master/sandbox.S

Jailbreak detection classics [The fail!]

- Sandbox check using fork
- Not working!
  - The sandbox patch does’nt affect this part of the sandbox!

```
+(BOOL) doFork () {
    int res = fork();
    if (!res) {
        exit(0);
    }

    if (res >= 0) {
            #if TARGET_IPHONE
                NSLog("fork
                return 0;
            #else
                retu
            #endif
    }
    return 0;
}
```

Security Worst Practices

Having fun localy with Apple media player DRM

Hardcoded crypto key...

```
v31 = objc_msgSend(&OBJC_CLASS___NSString, "stringWithFormat:", CFSTR("HiddenTreasures"));
v32 = objc_msgSend(&OBJC_CLASS___NSData, "dataFromBase64String:", v31);
v33 = objc_msgSend(&OBJC_CLASS___NSBundle, "mainBundle");
v34 = objc_msgSend(v33, "pathForResource:ofType:", v13, CFSTR("cpng"));
v35 = objc_msgSend(&OBJC_CLASS___NSData, "dataWithContentsOfFile:", v34);
v36 = objc_msgSend(&OBJC_CLASS___FBEncryptorAES, "decryptData:", v35, v32);
v37 = objc_msgSend(&OBJC_CLASS___UIImage, "imageWithData:", v36);
v38 = objc_msgSend(&OBJC_CLASS___UIImageView, "alloc");
v39 = objc_msgSend(v38, "initWithImage:", v37);
__asm { VMOV.F32        S14, #9.0 }
v207 = 1126957056;
v206 = 1134297088;
__asm
{
  VADD.F32        D7, D8
  VSTR            S14
}
v205 = 1114898432;
v40 = v39;
objc_msgSend(v39, "setFr
objc_msgSend(v5, "addSubv
objc_msgSend(v40, "release
```

FAIL

```
1  from C...Cipher import AE
2  fr...64 im...  b64encode,
3  ...  binasci
4  ...  sys
5
6  SECRET_KEY = b6...enTre...  # = Padding
7  print len(SECRE...EY) ,...  ...SECRET_KEY)
8  secret=SECRET_K... + ("\x00"...  ...(SECRET_KEY)))
9  IV = "\x00" * 16  # Defa...
10
11 ...for_decry...  ...new(secret, AES.MODE_CBC, IV)
12
13 def DecryptWit...ecrypt_cipher, encrypted_data):
14     decode...pted_data = encrypted_data
15     dec...data = decrypt_cipher.decrypt(encrypted_data)
16     ...decrypted_data
17
18 ...open(sys.argv[1],"rb").read()
19 print "Input File >",sys.argv[1],"Size >",len(i)
20 print "AES Key >",binascii.hexlify(secret)
21 decrypted_data = DecryptWithAES(cipher_for_decryption, i+("\x00" * (16-(len(i)%16))))
22 outfile = sys.argv[1]+'.dec.png'
23 o = open(outfile,'wb+')
24 o.write(decrypted_data)
25 print "decrypted File >",outfile,"Size >",len(i)
```

Secure browser… Really ?

DEMO: Authentication Bypass

- Secure sandbox
  - Designed to meet Government Security requirements standards
  - All data at rest being encrypted.
- User password is securely stored in an encrypted database
  - But…

Defensives Measures

Defensives Measures
How to slow down the analysis…

Antidebug technics
- Old School GDB Killer : PTRACE_DENY_ATTACH
- Checking the P_TRACED flag

Anti Hooking technics
- Validating address space : Using dladdr() & Dl_info structure
- Inlining

Obfuscation
- No public tools for Objective C code obfuscation.
- Objective C is a dynamic language,
  - Based on message passing paradigm,
  - Most of bindings are resolved run time
  - It is always possible for attacker to track, intercept and reroute calls, even with obfuscated names.
- Manually implementing obfuscation can slow down attackers analysis
  - Renaming classes and methods
  - Dynamic string generation

Conclusion

Conclusion

Regarding security most of iOS applications are not mature!

Developers should follow the following recommendation in order to mitigate the risks.

- Do not rely only on iOS security
- Do not store credential using standardUserDefaults method.
- Encrypt data even when stored in the keychain
- Do not store encryptions keys on the device
- Check your code, classes, functions, methods integrity
- Detect the jailbreak (At less try do to it)
- Properly implement cryptography in applications
  - simple implementation are the most secure
- Remove all debug information from the final release
- **Minimize use of Objective-C for critical functions & security features.**

Thank you for Listening
Questions ?

mathieu.ranard[-at-]sogeti.com - http://esec-pentest.sogeti.com
mathieu.renard[-at-]gotohack.org - http://www.gotohack.org