synack

iOS App Reversing

a practical approach

patrick wardle

PATRICK WARDLE

Synack: director of research

NASA: autonomous software dev.

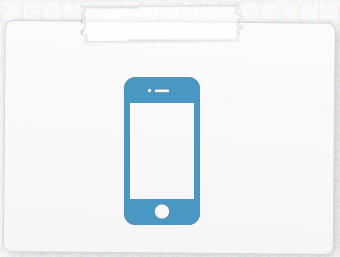
VRL: chief science officer

NSA: [redacted]

AN OUTLINE

THE TALK TODAY WILL COVER A SOLID AMOUNT OF MATERIAL

-> A TECHNICAL EXPLORATION OF REVERSING IOS APPS, FOCUSING ON UNCOVERING COMMON SECURITY ISSUES.



THE IOS ENVIRONMENT

A BREIF TECHNICAL OVERVIEW OF THE ENVIRONMENT IN WHICH IOS APPS EXECUTE

PREP'ING A REVERSING ENVIRONMENT

GETTING THE RIGHT TOOLS FOR THE JOB AND PREPARING IOS APPS FOR ANALYSIS

REVERSING TECHNIQUES

METHODS OF REVERSE-ENGINEERING IOS APPS

IOS APP VULNERABILITIES

COMMON CLASSES OF IOS APP VULNERABILITIES, AND APPS THAT ILLUSTRATE THESE VULNERABILITIES

"IF I HAVE SEEN FURTHER IT IS BY STANDING ON THE SHOULDERS OF GIANTS" (NEWTON)
...THIS TALK IS A COMPILATION OF PUBLIC INFORMATION, ENHANCED BY NOVEL TECHNIQUES & 'REAL-LIFE' EXAMPLES

synack

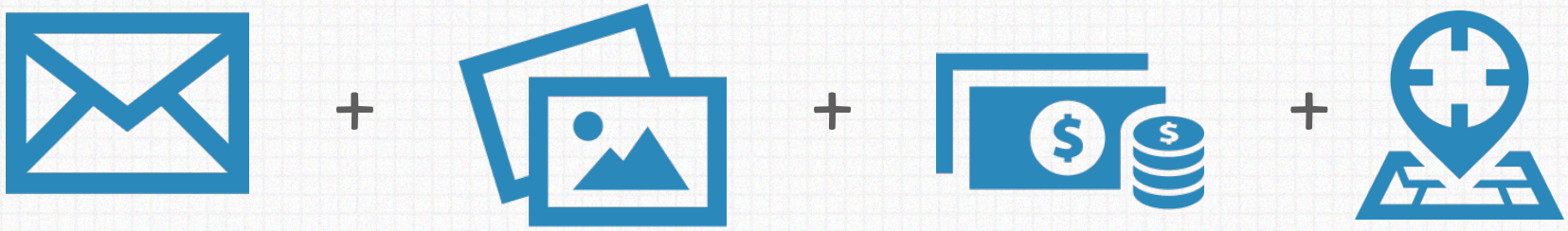
Background

...and why you should care about all of this

OUR iDEVICES

THINK ABOUT YOUR IPHONE/IPAD

-> IT CONTAINS A VAST AMOUNT OF PRIVATE AND HIGHLY SENSITIVE DATA



EMAIL + PHOTOS + FINANCIAL INFO + GEO-LOCATION

...THESE DEVICES ARE THE GATEWAYS INTO OUR DIGITAL LIVES

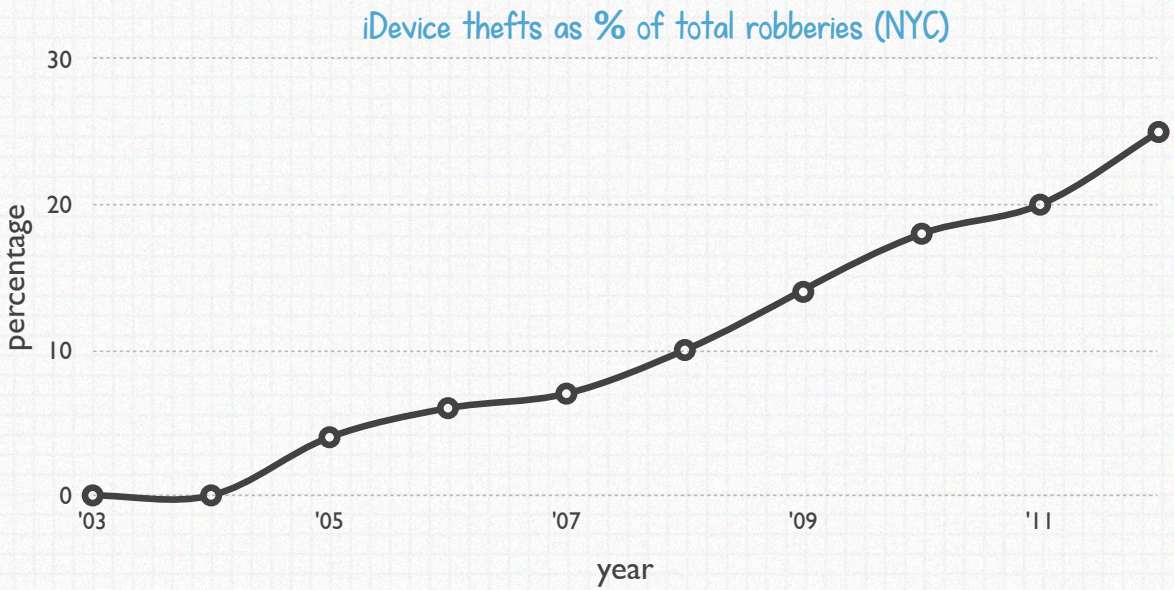
iDEVICE THEFT

STOLEN IDEVICES

THEY GET STOLEN OR LOST ALL THE TIME:
-> IN LARGER CITIES, "CELL PHONES COMPRISE 30-40% OF ALL ROBBERIES" (FCC.GOV)

THEFT

~50% no passcode lock



INSECURE REGARDLESS?

EVEN IN YOUR POSSESSION

-> STILL MANY POTENTIAL (MOSTLY APP-RELATED) THREATS THAT CAN LEAD TO SERIOUS PRIVACY ISSUES

TRANSMISSIONS

SENSITIVE DATA MAY BE TRANSMITTED INSECURELY

PROCESS UNTRUSTED DATA

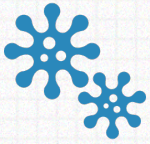
APPS PROCESS A LOT OF DATA FROM UNTRUSTED SOURCES (EXPLOITABLE?)

CLOSED-SOURCE

APPS ARE (GENERALLY) NOT OPEN-SOURCE

BACKUPS

A 'JUICY' TARGET FOR HACKERS (PCS/MACS AREN'T THAT HARD TO HACK)



"Fake Tor browser for iOS laced with adware, spyware, members warn"

IN THE NEWS

APPS ARE OFTEN SURPRISINGLY INSECURE

-> APPLE DOESN'T CHECK FOR, OR REALLY CARE ABOUT BUGGY/INSECURE APPS

so yah; lots of bugs!

BANKING

"Major security holes found in 90% of top mobile banking apps"

"Citi confirms critical bug in iPhone mobile banking app"

SOCIAL MEDIA

"Facebook for iOS Vulnerable to Credential Theft"

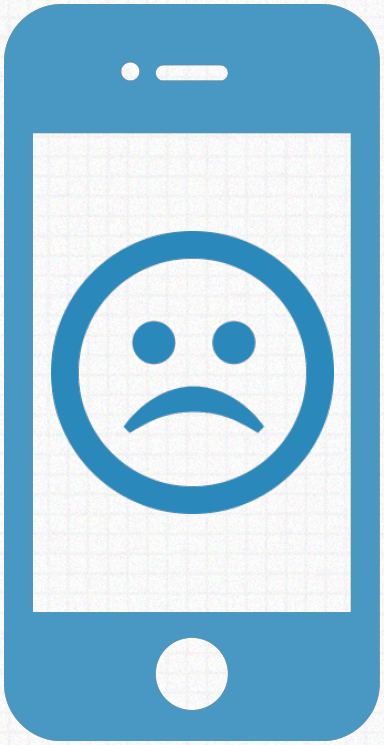
"Flaw in Tinder App let users track each other in real time"

ETC...

"Starbucks Stored iOS App Passwords And Location Data In Clear Text"

"Skype for iOS contains an XSS vulnerability that allows attackers steal information"

IN SHORT



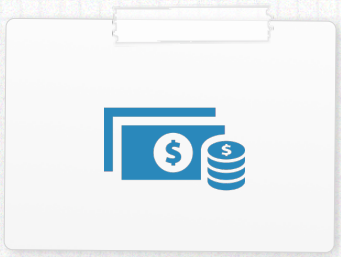
CLEARLY, IOS APPS ARE HORRIBLY INSECURE - WHAT TO DO?

→ REVERSE-ENGINEERING TO THE RESCUE!

reveal vulnerabilities

verify the app

raise security awareness



make some money!

synack

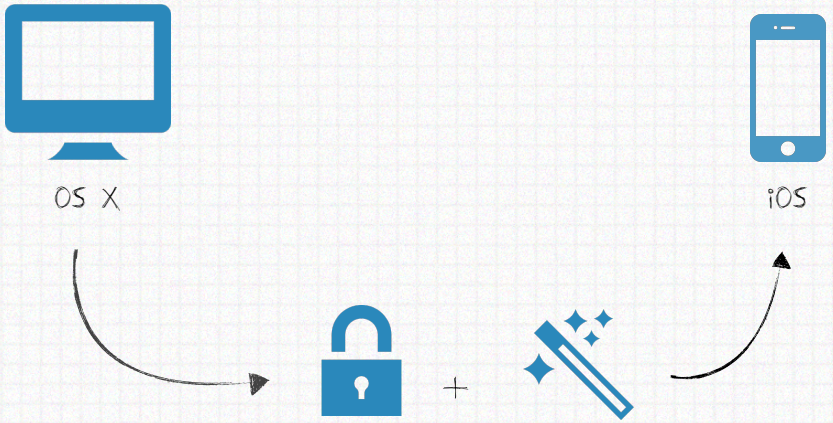
The iOS Environment

a brief technical overview

iOS

iOS IS DERIVED FROM OS X

-> BASICALLY JUST SLIMMED DOWN WITH EXTRA SECURITY



a jailbreak bypasses these

SECURITY++

- secure boot chain
- signed-code requirements
- anti-exploitation mechanisms
- encrypted storage
- sandboxed apps

APP SECURITY

iOS APP SECURITY

- signed (by Apple's signing certificate)
- encrypted
- run as limited user ("mobile")
- sandboxed
  - no dynamic code generation/execution
  - no access to other app's/processes
  - no direct access to hardware devices
  - confined to `/private/var/mobile/Applications/<app-GUID>`

iOS APPs

SO WHATS IN AN IOS APP?

-> DISTRIBUTED AS .IPA FILES, WHICH CONTAINS THE .APP BUNDLE

…just a zip file

.IPA

"IOS APP STORE PACKAGE"
OR
"IPHONE APPLICATION ARCHIVE"

installed: /private/var/mobile/Applications/

.APP

A 'BUNDLE' (DIRECTORY)
CONTAINS THE APP'S FILES

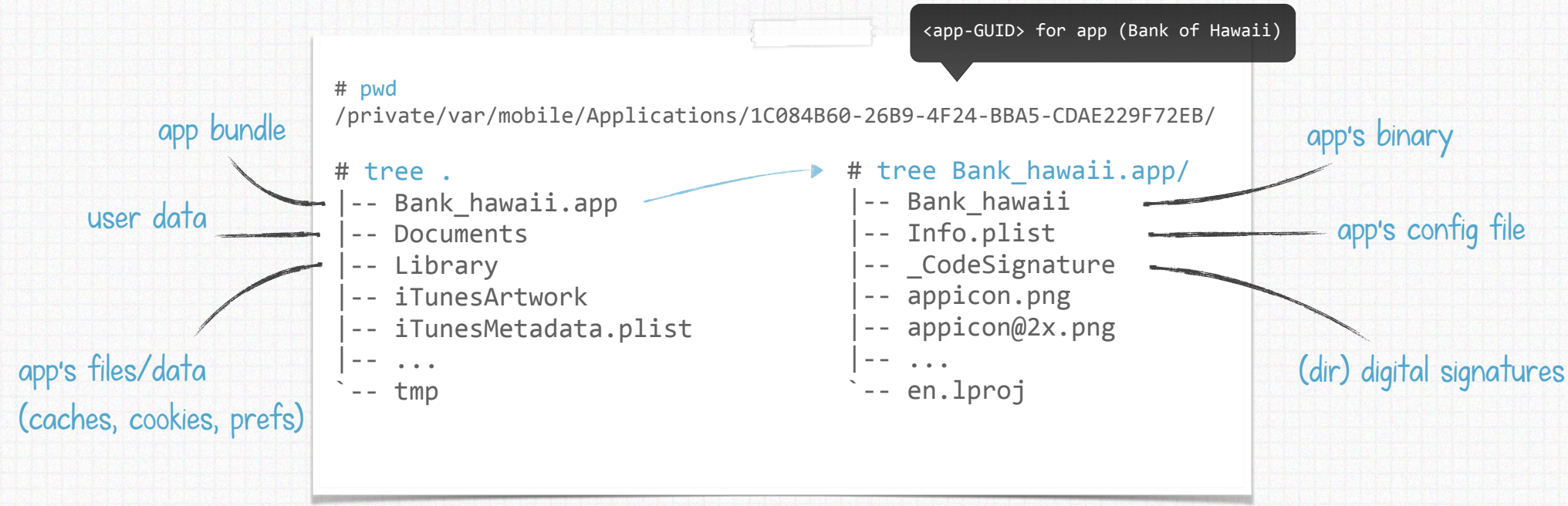
.APP's files

THE APP'S BINARY, IMAGES,
META-DATA, AND MORE

iOS APPs

THE APP'S ON-DISK LAYOUT

-> APPS HAVE A STANDARD LAYOUT THAT INCLUDES THE APP'S BUNDLE/BINARY, DATA, AND OTHER RESOURCES



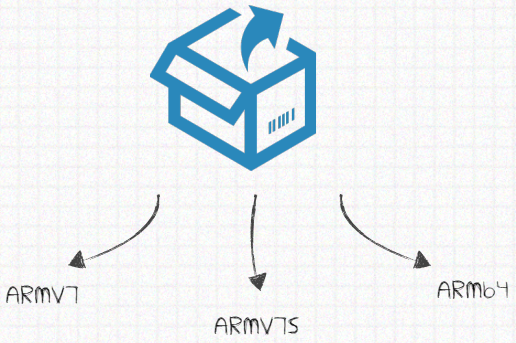
App Binary

THE APP BINARY IS "FAT"

-> ALLOWS A SINGLE DISTRIBUTABLE TO RUN ON MULTIPLE ARCHITECTURES

FAT BINARY

- standard format for iOS and OS X binaries
- contains multiple architecture-specific (mach-o) binary images




```
# less /usr/include/mach-o/fat.h

struct fat_header
{
   uint32_t magic;
   uint32_t nfat_arch;
};

struct fat_arch
{
   cpu_type_t cputype;
   cpu_subtype_t cpusubtype;
   uint32_t offset;
   uint32_t size;
   uint32_t align;
};
```

MACH-O BINARY

SO WHAT'S A MACH-O BINARY?

-> THE FILE FORMAT FOR (ARCHITECTURE-SPECIFIC) IOS/OS X BINARIES.
COMPRISED OF A HEADER, LOAD COMMANDS, AND THEN BINARY DATA/CODE.

```
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 HEADER

```
struct load_command
{
   uint32_t cmd;
   uint32_t cmdsize;
};

//load_command data
```

LOAD COMMANDS

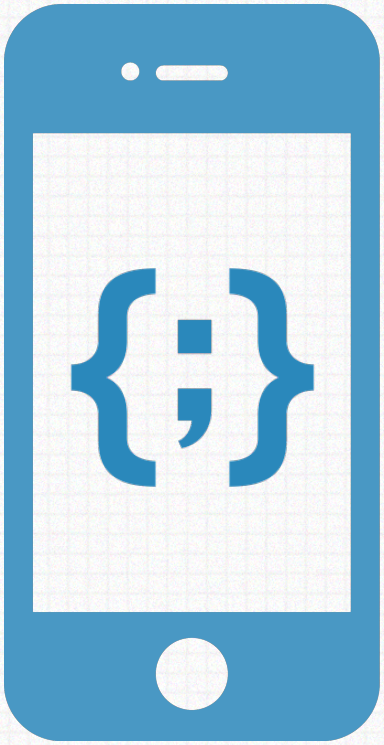
'INSTRUCTIONS' HOW TO SETUP/LOAD THE BINARY

MEMORY LAYOUT, THREAD CONTEXT, ETC

RAW DATA

SEGMENTS WITH CODE, DATA, ETC

OBJECTIVE-C



WHAT ARE IOS APPS WRITTEN IN?

OBJECTIVE-C

"Objective-C is a general-purpose, object-oriented programming language that adds Smalltalk-style messaging to the C programming language" (Wikipedia)

- a superset of C with classes/method, etc.
- the programming language used to create iOS/OS X Apps

```
//say hi!
NSLog(@"Hello, World");
```

OBJECTIVE-C

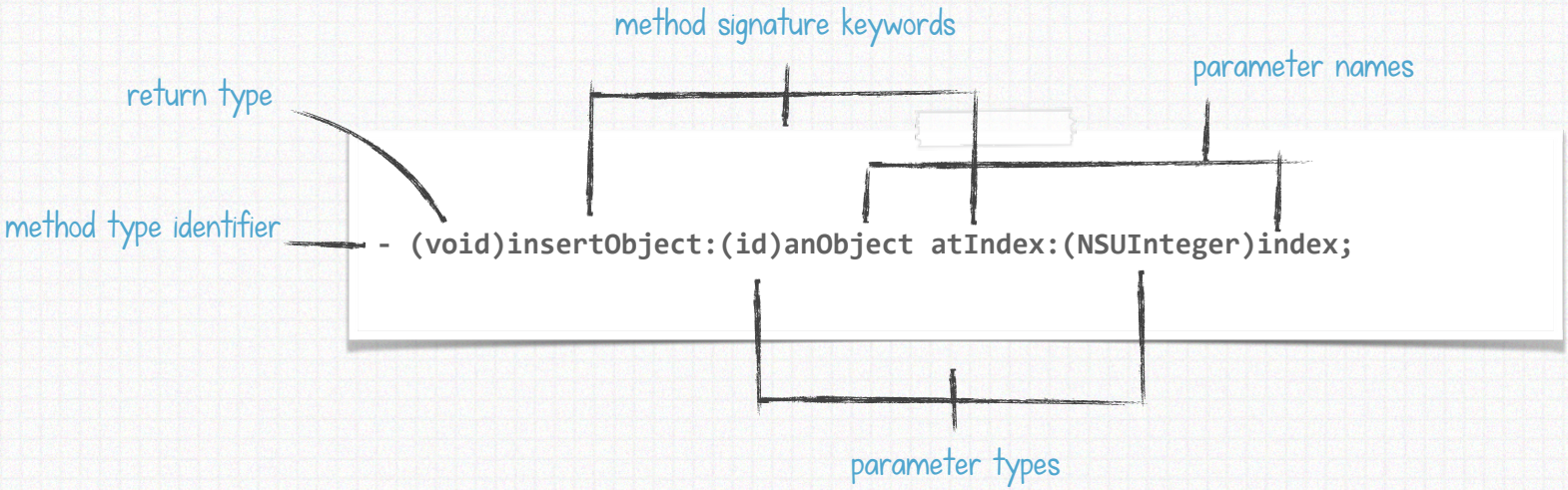
YES, THE SYNTAX IS QUITE 'ODD'

-> LOOKOUT FOR @'S AND [ ]
...AND TERMS SUCH AS 'MESSAGE PASSING' AND 'SELECTORS'

C++:
```
ObjectPtr->method(param1, param2);
```

Obj-C:
```
[ObjectPtr method:param1 p2:param2];
```

C++ vs. Objective-C



REVERSING OBJECTIVE-C

REVERSING IS SOMEWHAT NON-TRIVIAL

-> BEING AN OBJECT-ORIENTED LANGUAGE, STATIC ANALYSIS CAN BE CHALLENGING.

OBJECTIVE-C AND OBJC_MSGSEND

selector is the *name* of a method

```
//some Obj-C code
[ObjectPtr method:param1 p2:param2];
```

```
//compiler generates this code
objc_msgSend(ObjectPtr, @selector(method:p2:), param1, param2);
```

objc_msgSend
Sends a message to an instance of a class.

//method declaration
id objc_msgSend(id self, SEL op, ...)

self
A pointer to the instance of the class that is to receive the message.

op
The selector of the method that handles the message.

...
A variable argument list containing the arguments to the method.

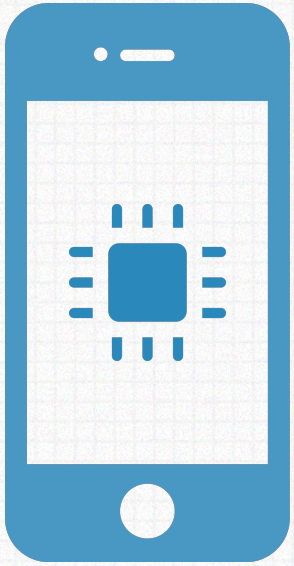
objc_msgSend

...ALL MESSAGES PASS THRU OBJC_MSGSEND

ARM Architecture

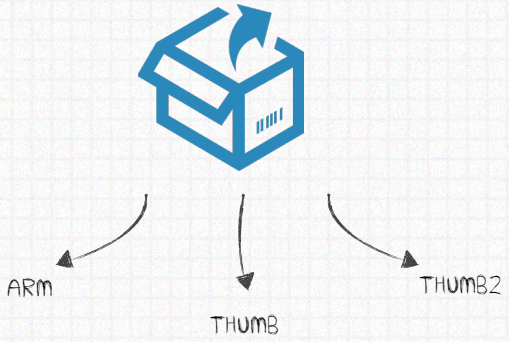
ARM POWERS MOBILE PROCESSORS EVERYWHERE

-> IOS DEVICES RUN ON PROCESSORS BASED ON THE ARM ARCHITECTURE



ARM IS:

- a RISC CPU strategy, with fixed length instructions
- "load/store" architecture
- encoded in various 'modes'



ARM: 4 byte instruction length

Thumb: 2 byte instruction length
-> subset of ARM instructions, encoded in 2-bytes
-> improves 'code density'

Thumb2: 2 or 4 byte instruction length
-> achieves 'code density' of Thumb and performance of ARM

'ARM32'

ARM ARCHITECTURE ON (MODERN) 32-BIT CPUS

-> THE ARMV7 PROCESSOR INSTRUCTION SET (IPHONE 3GS) WITH 32-BIT ADDRESS SPACE AND ARITHMETIC

[register]	[purpose]
R0-R12	general purpose registers
R13 (SP)	stack pointer
R14 (LR)	link register (return address)
R15 (PC)	program counter
CPSR	current program status register (processor mode, thumb bit, etc)

registers

memorizing this info is very helpful!

[register]	[purpose]
R0-R3	arguments
R4-R11	local variables/preserved
R0-R1	return value (from function)

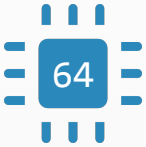
(function) calling convention

ARM64

ARM ARCHITECTURE ON 64-BIT CPUS

-> THE ARMV8 PROCESSOR INSTRUCTION SET (IPHONE 5S), WITH 64-BIT ADDRESS SPACE AND ARITHMETIC

go read: “ARM64 and You” (mike ash)



[register]	[purpose]
x0-x28	general purpose registers
x29 (FR)	frame register
x30 (LR)	link register (return address)
SP	stack pointer
PC	program counter

registers

[register]	[purpose]
x0-x7	arguments/return values
x9-x15	local variables
x19-x29	preserved

(function) calling convention

synack

Preparing a Reversing Environment

...getting some tools and some apps

REVERSING TOOLS

TOOLS FOR REVERSING IOS APPS

-> THERE ARE A MYRIAD OF TOOLS, FROM BASIC, TO ADVANCED, TO LARGELY COMPREHENSIVE

openssh
plutil
gdb
lsof
less
syslogd
vim
file
grep
sqlite3

BASIC TOOLS

COMMON *NIX/ OS X TOOLS PORTED TO IOS

```
# apt-get  install $(<tools.txt)
```

otool
class-dump
filemon
cycript
fileDP
burp
IDA Pro

ADVANCED TOOLS

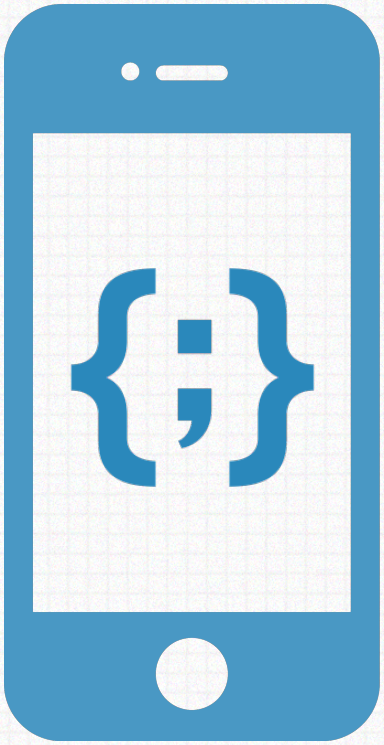
SLIGHT LEARNING CURVE, BUT NECESSARY FOR GETTING DOWN AND DIRTY!

SnoopIt (NESO)
IntroSpy (iSecPartners)
idb (Matasano)

COMPREHENSIVE

SUITES THAT PERFORM MANY TASKS

REVERSING TOOLS

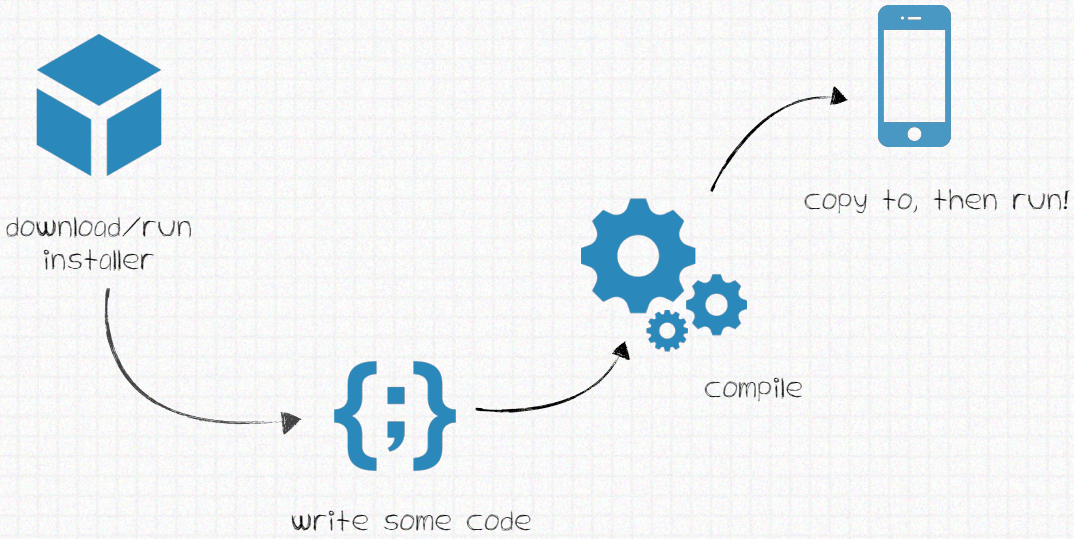


WHAT ABOUT WRITING YOUR OWN (OR FIXING OTHERS)?

find it at iosOpenDev.com

IOS OPEN-DEV

"sets up OS X and Xcode for 'open' development"



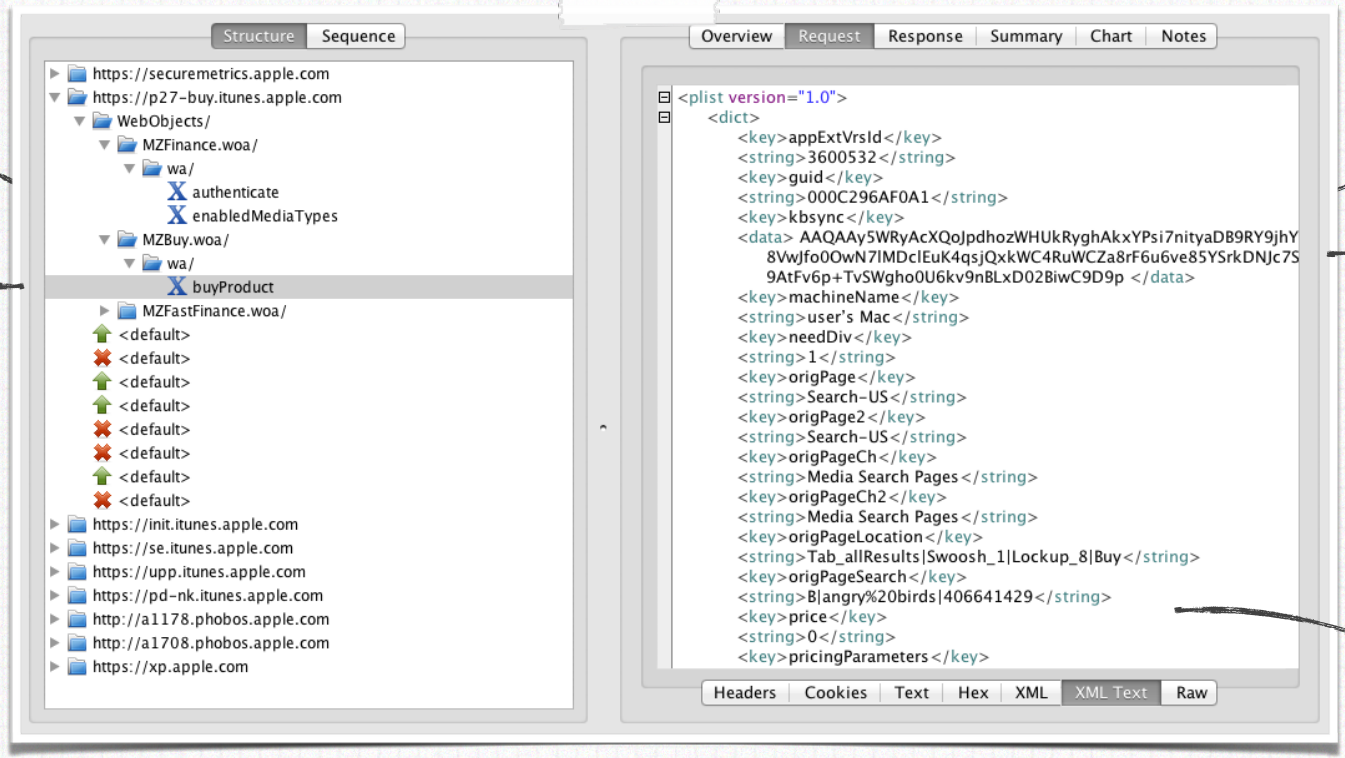
GETTING APPS

AUTOMATED APP GRABBING

-> SURE YOU CAN DO IT MANUALLY (WEB/ITUNES), BUT THAT DOESN'T SCALE WELL! HOW ABOUT DOING IT PROGRAMMATICALLY?

auth request

buy request



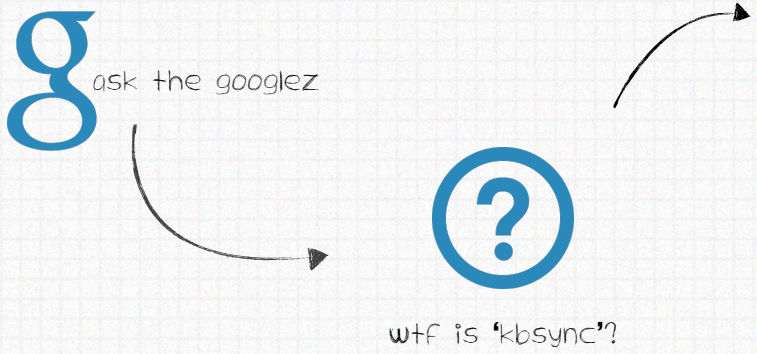
request data
hrmmm?

app name

GETTING APPS

AUTOMATED APP GRABBING

-> ALL THE PARAMETERS SEEM PRETTY MUCH SELF-EXPLANATORY, AND THUS EASY TO PROGRAMMATICALLY REPLICATE
....EXCEPT FOR THAT 'KBSYNC' PARAMETER.



.ru results; always interesting

Forum WASM.RU

"Reverse algorithm for computing the parameter "kbsync" in iTunes"

Requires reverse-engineering of the algorithm for calculating the parameter "kbsync" in iTunes.

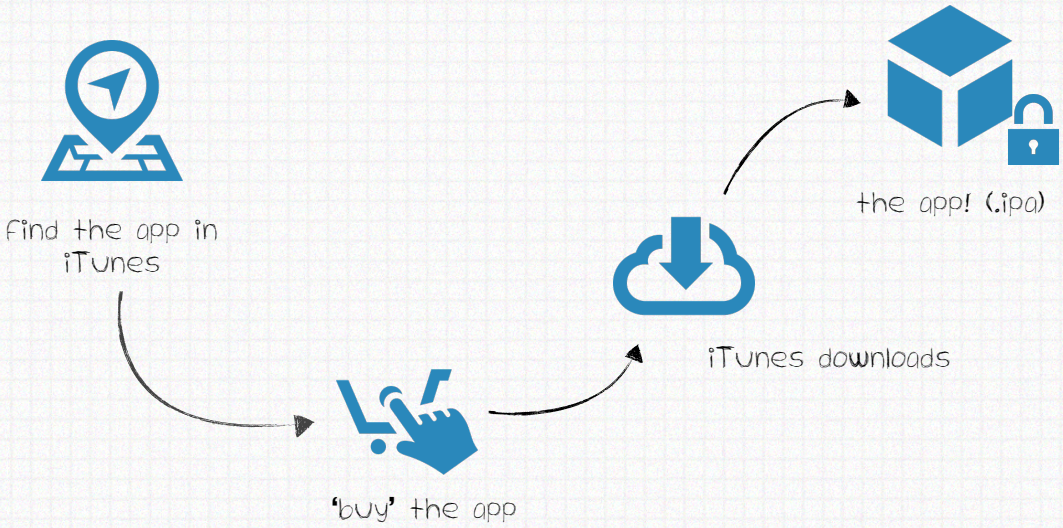
Project budget of $ 10,000;

(ab)USING iTUNES

LET ITUNES DO THE (HARD) WORK

iTunes

GRABBING APPS, IN TWO EASY STEPS



(ab)USING iTUNES

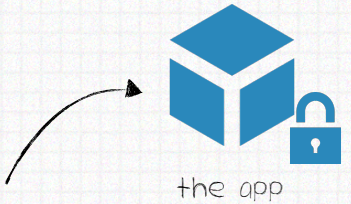
SO PROGRAMMATICALLY, HOW IS THIS DONE?

-> ACTUALLY, QUITE EASILY; APPLESCRIPT FTW :)

iTunes music store protocol

```
//AppleScript (open.scpt)
tell application “iTunes”

   open location item 1 of argv

end tell


//exec it
# osascript open.scpt itms://<app>
```

find the app



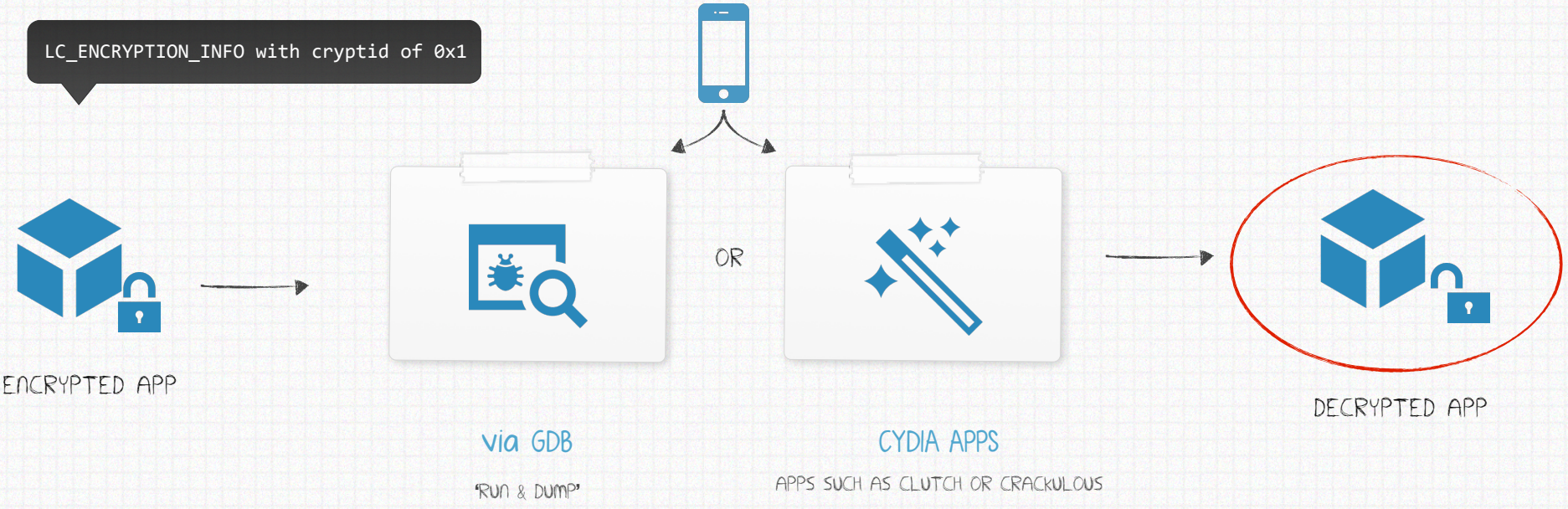
```
//AppleScript (open.scpt)
set elements to get entire contents of window 1

...

if (accessibility description of element as text) contains “Download”
then

   #trigger download
   click element

end if
```

click 'download' / 'buy'

APP DECRYPTION

REMOVING ENCRYPTION

APPS ARE ENCRYPTED WITH APPLE'S 'FAIRPLAY' DRM, WHICH SHOULD BE REMOVED TO ALLOW ANALYSIS



via GDB

'RUN & DUMP'

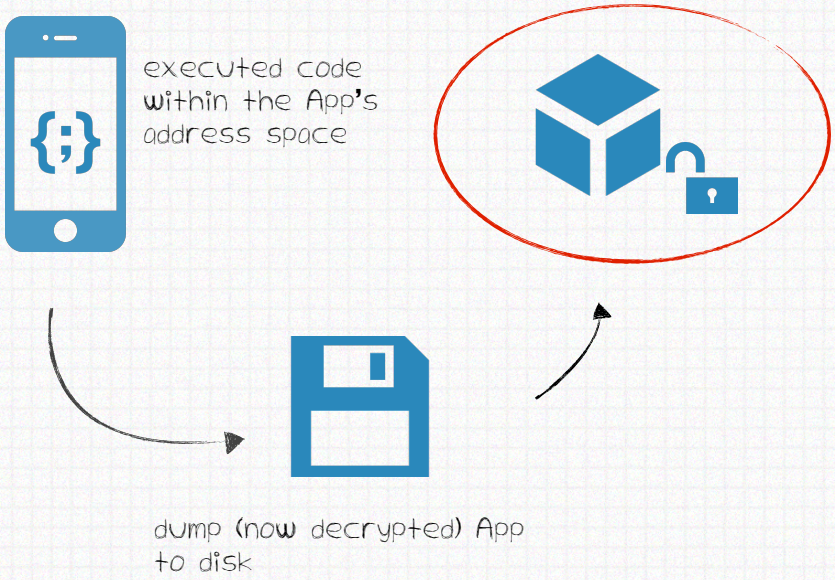
CYDIA APPS

APPS SUCH AS CLUTCH OR CRACKULOUS

APP DECRYPTION

REMOVING ENCRYPTION

-> WANT A METHOD THAT DOESN'T REQUIRE AN EXTERNAL PROGRAM/SCRIPTING (GDB) OR ISN'T CLOSED SOURCE (CYDIA APPS)



'dumpdecrypted.c' (by i0n1c)

1. to achieve code execution within that App's process space, launch the App with the `DYLD_INSERT_LIBRARIES` environment variable set:

   ```
   DYLD_INSERT_LIBRARIES=<decryptor>.dylib file.app/file
   ```

2. the dynamic library (`<decryptor>.dylib`), should export a constructor:

   ```
   __attribute__((constructor))
   ```

   this constructor should find the `LC_ENCRYPTION_INFO` load command then parse it in order to find, then dump the originally encrypted code.

synack

iOS Reversing Techniques

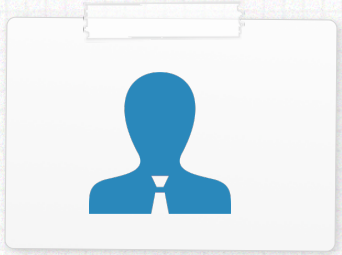
...methods to the madness

OTOOL

OTOOL: 'OBJECT FILE DISPLAYING TOOL'

-> DUMPING GENERAL INFORMATION ABOUT THE (DECRYPTED) APP'S BINARY

```
# otool -f | -h
```



Fat/Mach-O header

THE FAT BINARY/APP HEADERS

```
# otool -l
```

Load Commands

'INSTRUCTIONS' HOW TO SETUP/LOAD THE BINARY MEMORY LAYOUT, THREAD CONTEXT, ETC

```
# otool -L
```

Dependancies

FRAMEWORKS AND LIBRARIES IMPORTED BY THE APP

```
# otool -o
```

Objective-C Segment

CLASS NAMES, METHODS ETC,

Class-Dump

CLASS DUMP

-> PARSE/DISPLAY OBJECTIVE-C @INTERFACE DECLARATIONS

class-dump-z is the most accurate

command

interface

instance variables

```
# class-dump-z Bank_hawaii

@interface ASIHTTPRequest : XXUnknownSuperclass <NSCopying> {
    NSURL* url;
    NSString* username;
    NSString* password;
    ...
}

-(void)handleNetworkEvent:(unsigned long)event;
-(void)addBasicAuthenticationHeaderWithUsername:(id)username andPassword:(id)password;
-(void)attemptToApplyCredentialsAndResume;
-(BOOL)showAuthenticationDialog;
-(void)saveCredentialsToKeychain:(id)keychain;
-(void)saveProxyCredentialsToKeychain:(id)keychain;
...
@end
```

method declarations

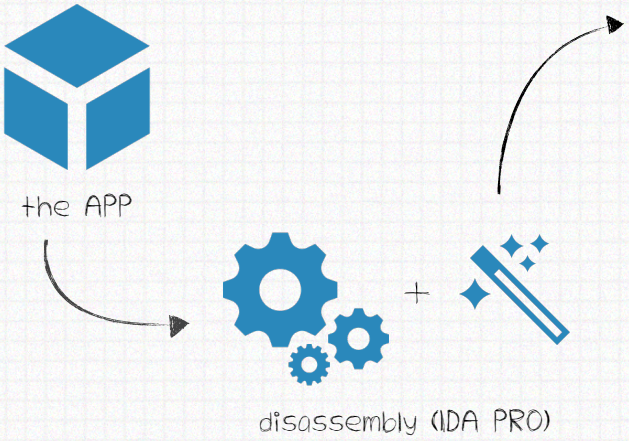
IDA PRO

IDA IS THE DE-FACTO REVERSING TOOL

-> LET'S LOOKS AN REVERSING A SMALL CHUNK OF OBJECTIVE-C CODE

```
NSString* now = [NSString stringWithFormat:@"now is: %@", [NSDate date]];
```

```
NSDate* date = objc_msgSend(classRef_NSDate, @selector("date"));
NSString* now = objc_msgSend(classRef_NSString, @selector("stringWithFormat:"), @"now is: %@", date);
```




```
R0: @"now is: 2014-03-14 03:13:37"
```

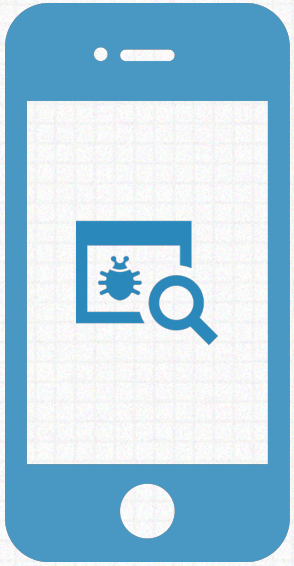
```
;load pointer to objc_msgSend info R9
__text:0000AEF6    MOV     R9, #(_objc_msgSend_ptr - 0xAF02)
__text:0000AEFE    ADD     R9, PC                          ; _objc_msgSend_ptr
__text:0000AEF0    LDR     R9, [R9]                        ; IMPORT _objc_msgSend

;load date object into R3
__text:0000AEF2    LDR     R3, [SP,#0x64+date]             ; load saved date

;load pointer to 'the time is: %@:' into R2
__text:0000AEF4    MOV     R2, #(cfstr_TheTimeIs - 0xAF00)
__text:0000AEFC    ADD     R2, PC                          ; "the time is: %@"

;load pointer to 'stringWithFormat:' into R1
__text:0000AEFE    MOV     R1, #(selRef_stringWithFormat_ - 0xAF0C)
__text:0000AF06    ADD     R1, PC                          ; selRef_stringWithFormat_
__text:0000AF08    LDR     R1, [R1]                        ; "stringWithFormat:"

;load pointer to NSString class into R0
__text:0000AF0A    MOV R0, #(classRef_NSString - 0xAF16)
__text:0000AF12    ADD R0, PC                              ; classRef_NSString
__text:0000AF14    LDR R0, [R0]                            ; IMPORT _OBJC_CLASS_$_NSString

;invoke objc_msgSend create formatted string
__text:0000AF08    BLX     R9                              ; objc_msgSend(classRef_NSString, ...);
```

```
NSDate* date = objc_msgSend(classRef_NSDate, @selector("date"));
```

Dynamic Analysis

DYNAMIC ANALYSIS OF IOS APPS

CAN BE FASTER (SIMPLER?) AND PROVIDE MORE INSIGHT INTO THE APP



network traffic

file-system I/O

debugging

instrumentation

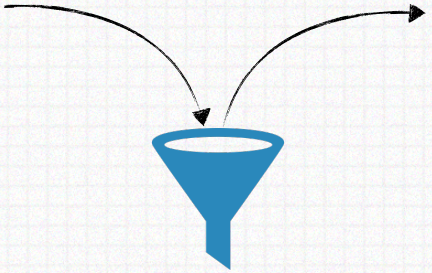
Network Analysis

CONCEPTUALLY, QUITE SIMPLE:

network traffic analysis

SNIFFING SOME TRAFFIC

execute the app



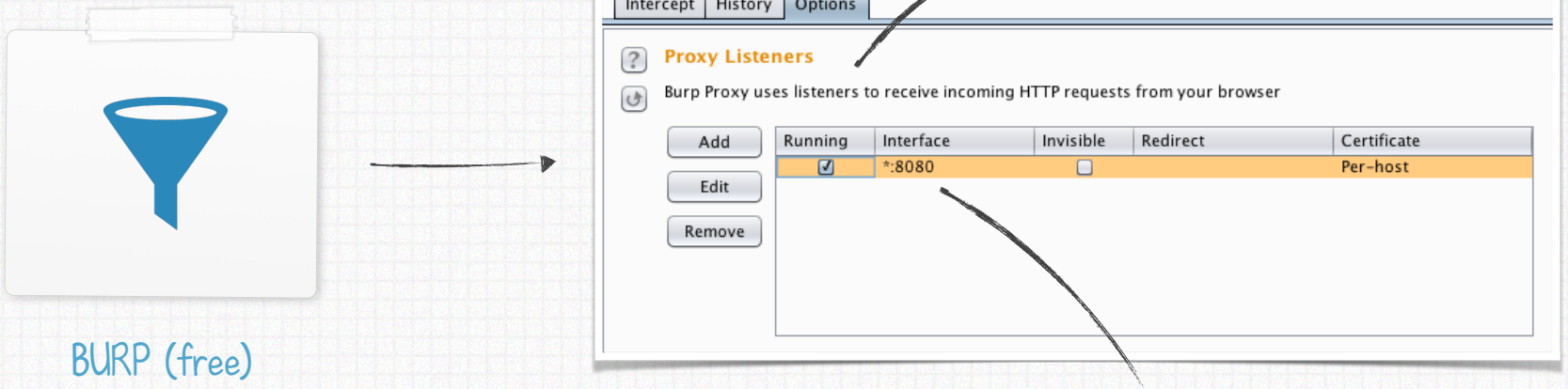
the proxy
collect/analyze :)

'the cloud'

Network Analysis

SO PRACTICALLY, HOW IS THIS DONE?

-> FIRST A PROXY SHOULD BE SETUP/CONFIGURED



BURP (free)

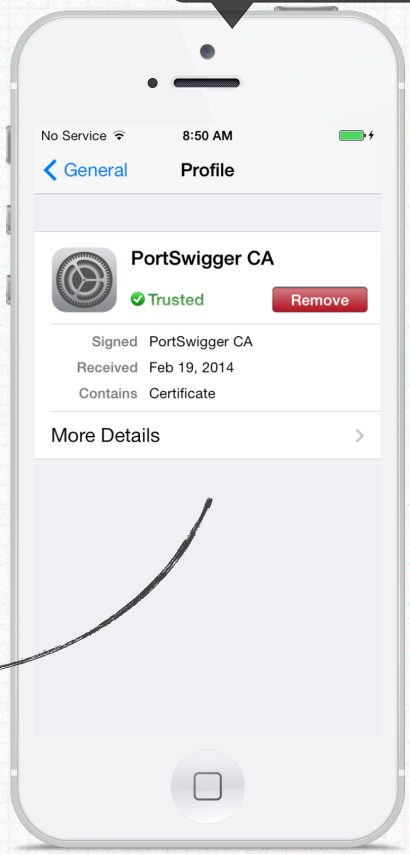
Network Analysis

SO PRACTICALLY, HOW IS THIS DONE?

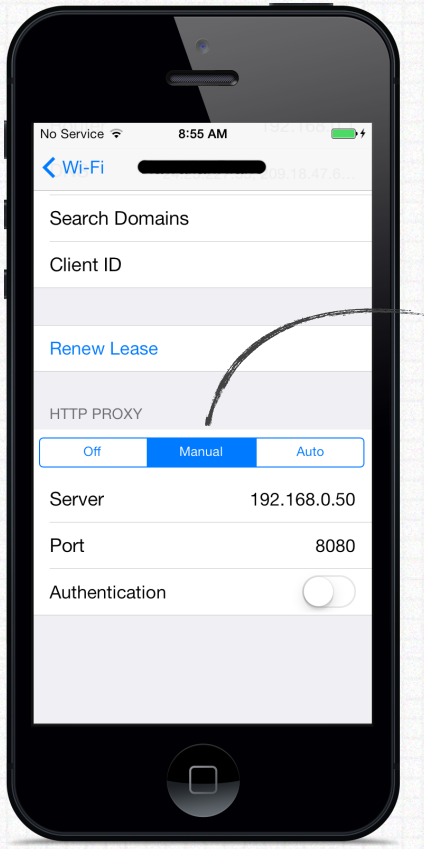
-> THEN THE DEVICE (IPHONE) HAS TO BE CONFIG'D

device

send cert via email



BURP profile



proxy settings

File-System I/O

AGAIN, CONCEPTUALLY, QUITE SIMPLE:

monitoring file-system I/O

MONITORING FILE-SYSTEM I/O

execute the app

passively monitor

file-system access

file event

File-System I/O

SO PRACTICALLY, HOW IS THIS DONE?

-> VIA A COMMAND-LINE FILEMON TOOL

```
//handle dropped events
if(fse->type == FSE_EVENTS_DROPPED)
{
  offInBuf += sizeof(kfs_event_a)+sizeof(fse->type);
  continue;
}
```

patch (for iOS 7)

filemon

create

```
# ./filemon

Bank_hawaii Created  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Application Support/analytics/analytics.db-journal
DEV: 1,3 INODE: 121171 MODE: 81a4 UID: 501 GID: 501 Arg64: 300649589561

Bank_hawaii Modified  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Application Support/analytics/analytics.db-journal
DEV: 1,3 INODE: 121171 MODE: 81a4 UID: 501 GID: 501 Arg64: 300650439368

Bank_hawaii Deleted  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Application Support/analytics/analytics.db-journal
DEV: 1,3 INODE: 121171 MODE: 81a4 UID: 501 GID: 501 Arg64: 300650449950

Bank_hawaii Created  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Preferences/com.fis.140SUB.plist.l0mitdo
DEV: 1,3 INODE: 121172 MODE: 8180 UID: 501 GID: 501 Arg64: 300677061026

Bank_hawaii Modified  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Preferences/com.fis.140SUB.plist.l0mitdo
DEV: 1,3 INODE: 121172 MODE: 8180 UID: 501 GID: 501 Arg64: 300677316493

Bank_hawaii Chowned  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Preferences/com.fis.140SUB.plist.l0mitdo
DEV: 1,3 INODE: 121172 MODE: 8180 UID: 501 GID: 501 Arg64: 300677321187

Bank_hawaii Renamed  /private/var/mobile/Applications/1C084B60-26B9-4F24-BBA5-CDAE229F72EB/Library/Preferences/com.fis.140SUB.plist.l0mitdo
DEV: 1,3 INODE: 121172 MODE: 8180 UID: 501 GID: 501
```

rename

App Debugging

CONCEPTUALLY, QUITE SIMPLE:

app debugging

MONITORING CODE EXECUTION

App Debugging

APP DEBUGGING WITH GDB

-> GDB IS THE DE FACTO DEBUGGER FOR IOS. HERE ARE SOME USEFUL HINTS/TIPS.

 # gdb -waitfor 'app name'

to debug an app, its easiest to wait for it, then attach.

attach to an app

(gdb) info shared

displaying all loaded modules (including the app's binary as the first module) displays ASLR deltas

view loaded modules

(gdb) x/hi <offset+base>

since Apple's GDB doesn't support the 'force-mode thumb', use the 'h' format letter to view thumb

disassemble thumb code

(gdb) po <address>

with the 'po' (print object) command, gdb can parse/display Objective-C objects!

print Objective-C objects

gdb (iOS 7): http://cydia.radare.org

App Instrumentation

CONCEPTUALLY, QUITE SIMPLE:

app instrumentation

INSTRUMENTING AN APP

execute the app

inject code/interpreter

BYPASS CLIENT-SIDE LOGIC
EXECUTE HIDDEN CODE
MANIPULATE THE APP RUNTIME
ETC

App Instrumentation

USING CYCRIPT TO INSTRUMENT AN APP

"ALLOWS DEVELOPERS TO EXPLORE AND MODIFY RUNNING APPLICATIONS ON EITHER IOS OR MAC OS X USING A HYBRID OF OBJECTIVE-C AND JAVASCRIPT SYNTAX THROUGH AN INTERACTIVE CONSOLE"

```
# dpkg -i cycript.deb
```

save/install

```
# cycript -p <appPID>
```

inject into a process

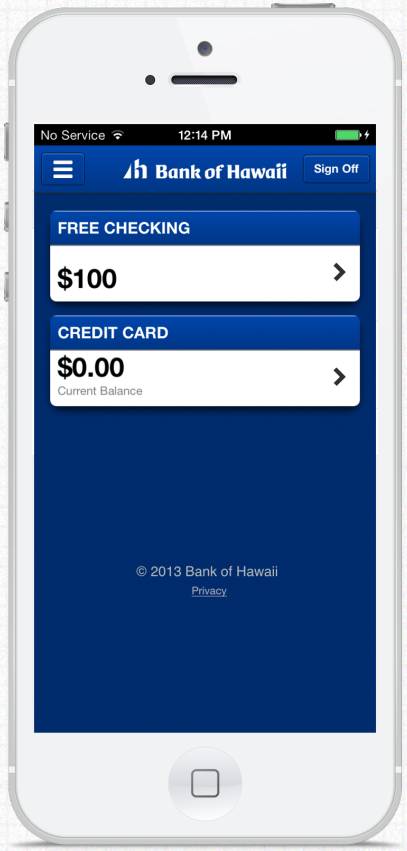
```
cy# *UIApp
{isa:#"UIApplication",_delegate:#"<TiApp: 0x17da6e10>",_touchMap:
0x17db2860,_exclusiveTouchWindows:...}

cy# UIApp.keyWindow.recursiveDescription
@"<UIWindow: 0x17dd82b0; frame = (0 0; 320 480); layer = <UIWindowLayer: 0x17dd8390>>
 | <TiRootView: 0x17dda240; frame = (0 20; 320 460); layer = <CALayer: 0x17dda320>>
 |  | <TiUIWindow: 0x17d04100; frame = (0 0; 320 460); layer = <CALayer: 0x17d040d0>>
```

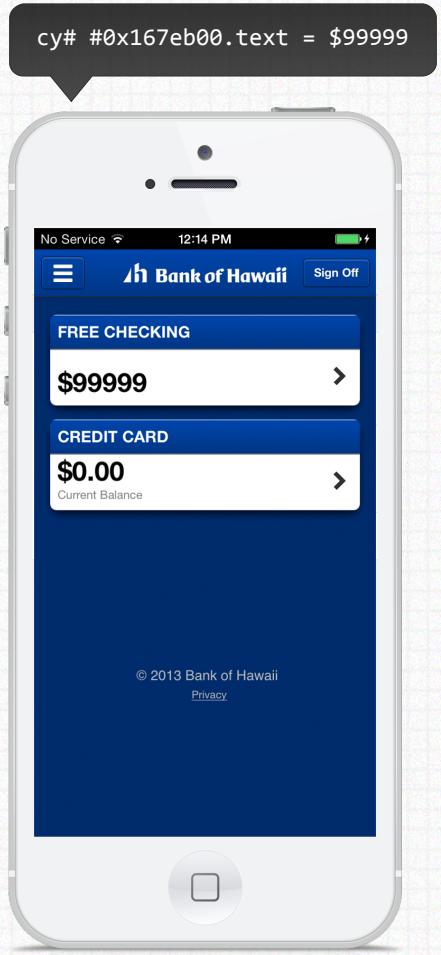
cycript console

App Instrumentation

USING CYCRIPT TO INSTRUMENT AN APP

cycript instrumentation





synack

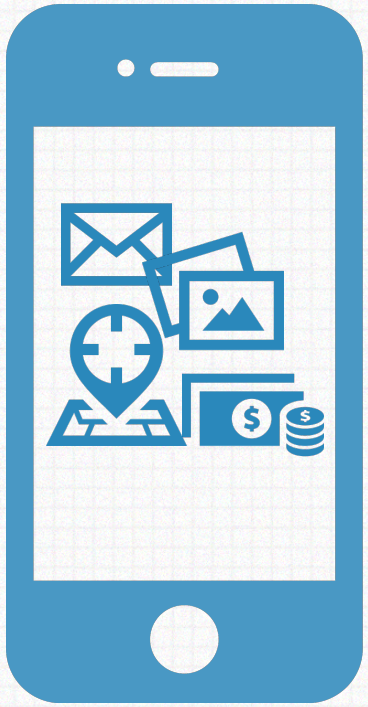
iOS App Vulnerabilities

...what to look for when reversing

THE MINDSET

THINK ABOUT IT THIS WAY

TARGETING MOBILE DEVICES IS UNIQUE - IT'S ALL ABOUT GAINING ACCESS TO SENSITIVE DATA



theft

network monitoring

back-ups

'shady' apps

NETWORK SECURITY

FIRST, HOW TO DO IT RIGHT?

-> ALL SENSITIVE NETWORK COMMUNICATIONS SHOULD BE SECURED AND ALL NETWORK INPUT SHOULD BE SANITIZED.

standard network/browser security practices

SECURED COMMS

SSL should be used (correctly) to prevent a myriad of issues such as sniffing or content injection.

INPUT SANITATION

content that is rendered (e.g. in a browser view) should be sanitized to prevent traditional 'browser security' issues.

NETWORK (in)SECURITY

SPOTTING A VULNERABILITY STATICALLY

-> DOES THE APP USE SSL AND DOES IT DO SO, 'CORRECTLY'?

iOS enables SSL for 'https://'

Statically verifying the (correct) use of SSL can be accomplished by examining the binary.

non-SSL (HTTP)

Unfortunately, allowing self-signed certificates makes the App vulnerable to man-in-the-middle attacks.

This can manifest in code in several ways:

```
//allow self signed certs
[NSURLRequest setAllowsAnyHTTPSCertificate:YES forHost:[[NSURL URLWithString:@"someURL"] host]];
```

[or]

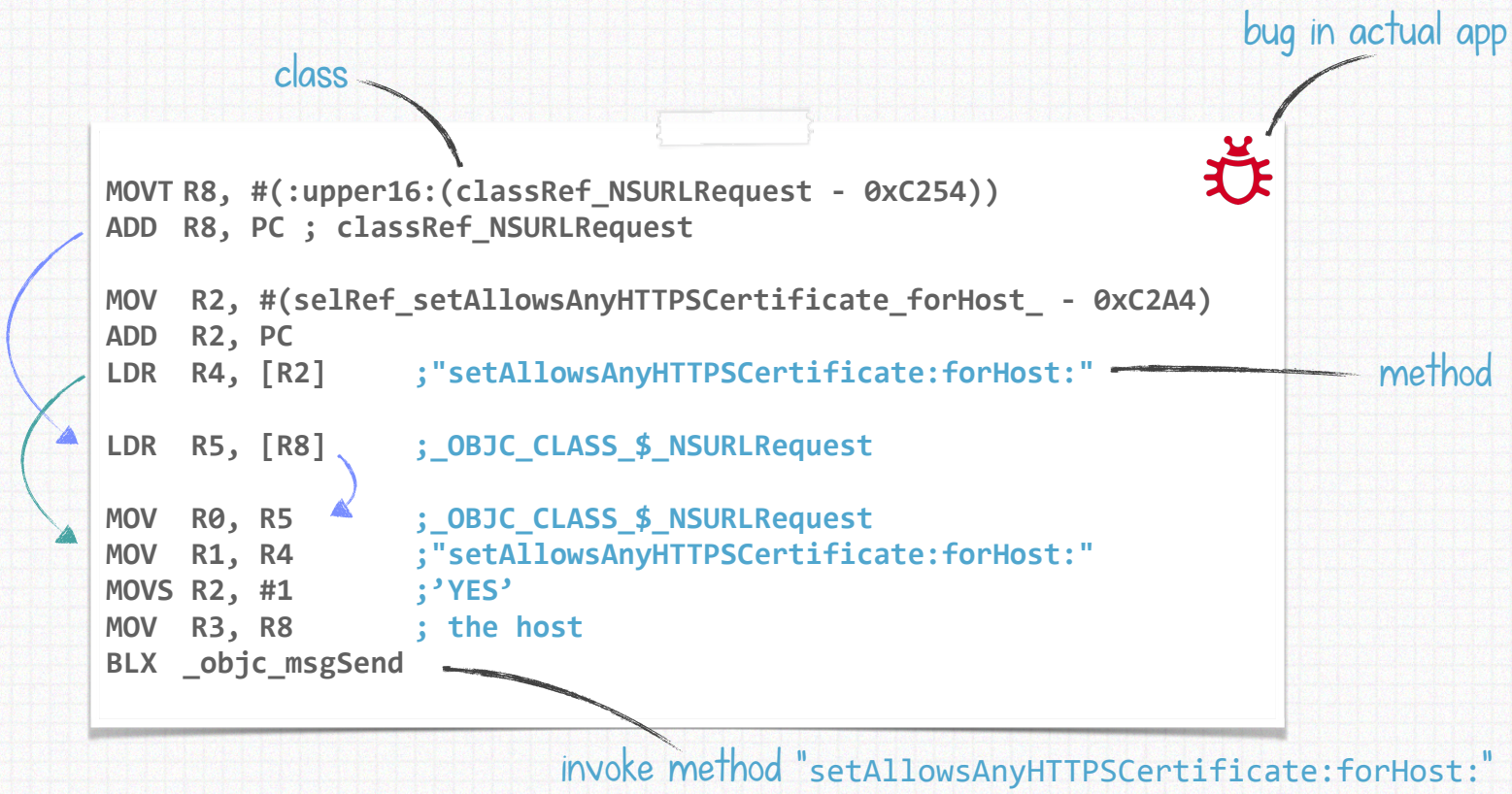
```
//implement the following category (iOS 5+)
-(void) connection: (NSURLConnection*) inConnection
        willSendRequestForAuthenticationChallenge: (NSURLAuthenticationChallenge*) inChallenge
```

'vulnerable' SSL

NETWORK (in)SECURITY

A BROKEN SSL IMPLEMENTATION

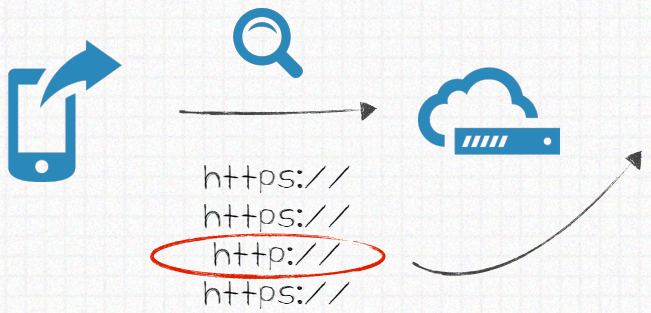
-> REMEMBER, DON'T ALLOW SELF-SIGNED CERTS (AND BETTER YET, PIN THEM).



NETWORK (in)SECURITY

SPOTTING A VULNERABILITY DYNAMICALLY

-> 'SNIFF' OR PROXY NETWORK TRAFFIC VIEW TO DETERMINE IF IT CAN BE MANIPULATED



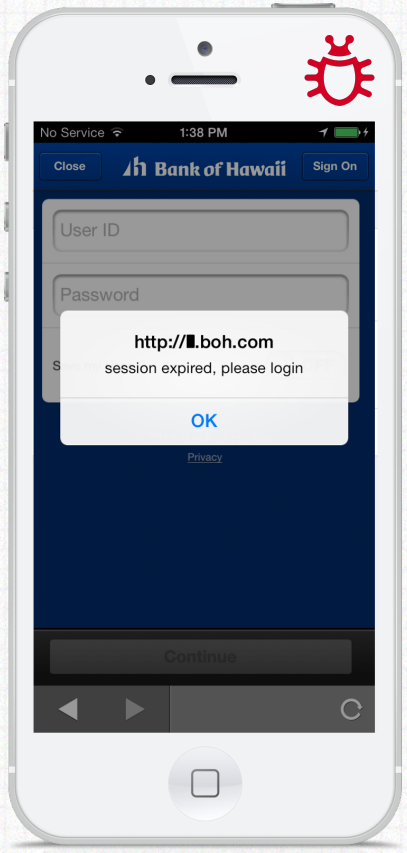
'mixed-content attack'

```
function displayLoginMsg()
 alert("session expired, please login");

<body onload="displayLoginMsg()">

<form>
<input name="userID" value="User ID">
<input name="password" value="Password">
<button onclick="stealCreds()">login</button>
...
```

malicious code injection

```
# tail -f /var/log/syslog

Mar 14 09:49:54 synack-1 Bank_hawaii: [DEBUG] New scheme: <NSMutableURLRequest: 0x18816fa0> { URL: http://xx.boh.com/ }
Mar 14 09:49:55 synack-1 Bank_hawaii: [DEBUG] New scheme: <NSMutableURLRequest: 0x175efe30> { URL: http://xx.boh.com/phoenix.zhtml }
```

UIWebView renders HTML/JS



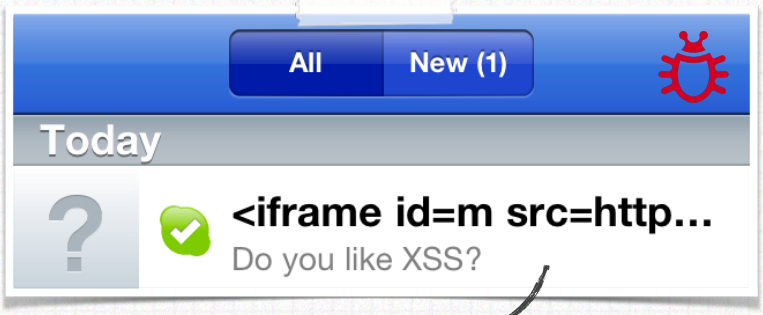
NETWORK (in)SECURITY

MORE NETWORK RELATED VULNERABILITIES?

OTHER COMMON VULNERABILITIES INCLUDE XSS OR EVEN SERVER-SIDE APIS.

Cross-site Scripting (XSS)

Since `UIWebViews` render all HTML and JS, it may be possible to perform a XSS if proper sanitation is not performed.



skype XSS in 'Full Name' (patched)

details: http://gibsonsec.org/snapchat

SERVER-SIDE API

Analyzing an app binary and/or its network traffic can reveal abusable server-side APIs.

/ph/find_friends "A single request (once logged in, of course!) to /ph/find_friends can find out whether or not a phone number is attached to an account"

```
{
    username: "<your account name>",
    timestamp: 1373207221,
    req_token: create_token(auth_token, 1373207221),
    countryCode: "US",
    numbers: "{\"3140001337\": \"Kevin Mitnick\"}"
}
```

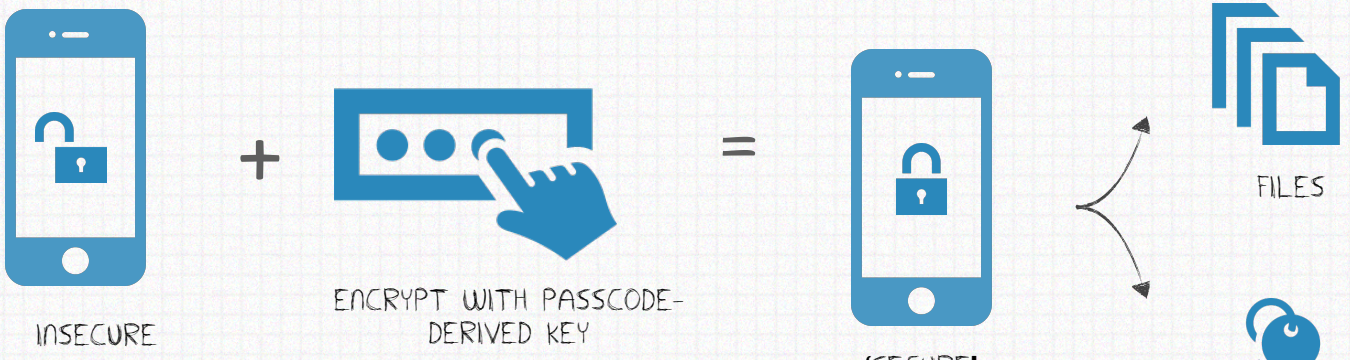
snapChat API abuse (4.6M phone #s, now patched)

SECURE DATA STORAGE

FIRST, HOW TO DO IT RIGHT?

THE DATA PARTITION (APPS, DOCUMENTS, CACHE, ETC) IS ENCRYPTED WITH A HARDWARE BASED FILE SYSTEM ENCRYPTION KEY. HOWEVER, WHEN THE DEVICE IS TURNED ON, THIS DATA IS DECRYPTED (SO IT IS AVAILABLE TO APPS, ETC). SOMEBODY WITH ACCESS TO THE DEVICE CAN ACCESS THIS DATA WITHOUT THE PASSCODE :(

use FileDP or Keychain Dumper

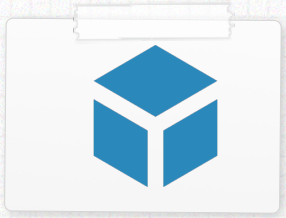


"While the device is locked, protected files [and keychain] are inaccessible even to the app that created them"

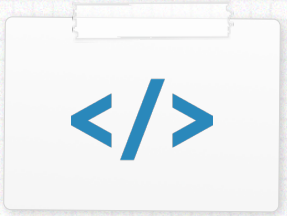
INSECURE DATA STORAGE

COUNTLESS APP STORE SENSITIVE DATA INSECURELY

-> THIS INCLUDES, USER NAMES, PASSWORDS, SESSION KEYS, GEOLOCATION DATA, ETC



the binary



property lists

databases

log files

STORAGE WITHIN THE BINARY

APPS MAY STORE SENSITIVE DATA WITHIN THEIR BINARY IMAGE

-> OFTEN WILL FIND CREDENTIALS, OR API KEYS, ETC WITHIN

within the binary

REST API

see: http://faptrackr.org

```
__cstring:000E91ED DCB "www.puffchat.me",0
__cstring:000E91FD DCB "POST",0
__cstring:000E9202 DCB "/v2/api/client/login",0
__cstring:000E9217 DCB "key=dl81Vh2uorfNdj2Rt2M4EylW91uUsQRZwhQ99g7K0MRXeMYePS", 0
```

Embedded 'secret' API Key (PuffChat)

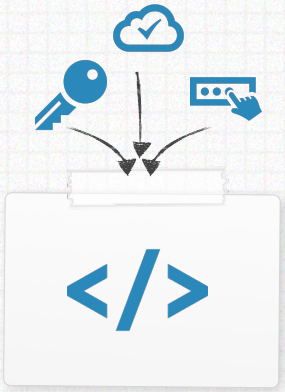
"We all know you can't keep a secret key secret in a binary, you can try and hide it but not only is it pretty futile, in this case it wasn't done at all."

STORAGE WITHIN A 'PLIST'

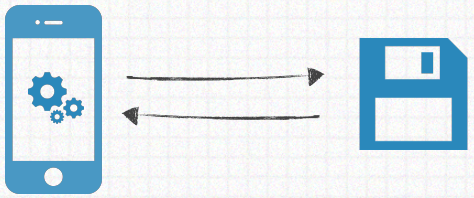
APPS MAY STORE SENSITIVE DATA WITHIN PROPERTY LISTS ('PLISTS')

-> OFTEN WILL FIND CREDENTIALS, SESSION KEYS, ETC WITHIN THE APP'S 'USER DEFAULTS' PLIST

```
# plutil -convert xml1 Info.plist
```



within plists



```
//store
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
[defaults setValue:@"someData" forKey:@"someKey"];

//retrieve
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
id persistedData = [defaults objectForKey:@"someKey"];
```

User Defaults

STORAGE WITHIN A 'PLIST'

SPOTTING A VULNERABILITY

-> SCOPE OUT THE DISASSEMBLY, OR SIMPLY DUMP THE 'USER DEFAULTS' PLIST

User Defaults in /private/var/mobile/Library/Preferences/

```
MOV     R1, #(selRef_standardUserDefaults-0x5917A)
ADD     R1, PC
LDR     R1, [R1]      ; "standardUserDefaults"

MOV     R0, #(classRef_NSUserDefaults-0x591A2)
ADD     R0, PC
LDR     R0, [R0]      ; _OBJC_CLASS_$_NSUserDefaults

BLX _objc_msgSend     ; [NSUserDefaults standardUserDefaults];


MOV     R3, #(cfstr_Sessionid_3-0x591D6)
ADD     R3, PC        ; "sessionCookie-PRODUCTION"

LDR     R2, [SP,#0xB4+sessionID]    ;session data

MOV     R1, #(selRef_setObject_forKey_-0x591D6)
ADD     R1, PC
LDR     R1, [R1]      ; "setObject:forKey:"

BLX _objc_msgSend     ; [userDefaults setObject: … forKey: …];
```

App Disassembly ('User Defaults')

by default, plist has NSFileProtectionNone!

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN">
<plist version="1.0">
<dict>
        <key>sessionCookie-PRODUCTION</key>
        <dict>
                <key>Created</key>
                <real>415574831</real>
                <key>Expires</key>
                <date>2014-03-31T21:27:11Z</date>
                <key>HttpOnly</key>
                <string>TRUE</string>
                <key>Name</key>
                <string>sessionid</string>
                <key>Secure</key>
                <string>TRUE</string>
                <key>Value</key>
                <string>v1t8qnt3plhnxl1ph9bms32xkem1856w</string>
        </dict>
</dict>
```

App's 'User Defaults' plist

STORAGE WITHIN A DATABASE

SQLITE IS A COMMON METHOD OF STORING DATA

-> OFTEN WILL FIND USER NAMES, PASSWORDS, OR OTHER SENSITIVE INFORMATION WITHIN DATABASE(S) CREATED BY THE APP,

within databases

```
//sqlite imports
/usr/lib/libsqlite3.dylib
   _sqlite3_exec
   _sqlite3_file_control
   _sqlite3_open_v2
```

more at: http://bas.bosschert.nl/steal-whatsapp-database

chats

```
# sqlite3  Documents/ChatStorage.sqlite
sqlite> .tables
ZWABLACKLISTITEM   ZWACHATSESSION     ZWAMEDIAITEM       Z_METADATA
ZWABROADCAST       ZWAGROUPINFO       ZWAMESSAGE         Z_PRIMARYKEY
ZWACHATPROPERTIES  ZWAGROUPMEMBER     ZWAMESSAGEWORD
sqlite> select ZPUSHNAME, ZTEXT from ZWAMESSAGE;
|Colby|1394645540-45|Brooooo|
||Hey d00d|
|Colby|1394645540-57|Sweet! Giving up on telegrams broken security? :p|


|Mark|Is this supposed to be secret?|
||not from people who know how to reverse apps!|
|Mark|Haha|
```

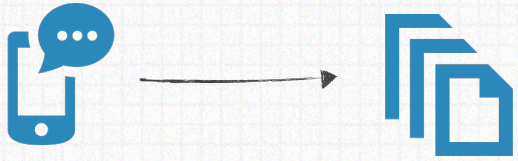
WhatsApp chat history

'STORAGE' WITHIN A LOGFILE

FOR DEBUGGING PURPOSES, MANY APPS 'STORE' (LEAK) SENSITIVE INFORMATION IN LOGS FILES

-> OFTEN WILL FIND USER NAMES, PASSWORDS, SESSION DATA, OR SENSITIVE INFORMATION WITHIN LOG FILES CREATED BY THE APP,

within log files



```
//create a file path (within the 'Documents/' dir)
NSString *documentsDirectory =
   [NSSearchPathForDirectoriesInDomains (NSDocumentDirectory, NSUserDomainMask, YES) objectAtIndex:0];
NSString *fileName = [documentsDirectory stringByAppendingPathComponent:@"logfile.txt"];

//create the file
[[NSFileManager defaultManager] createFileAtPath:fileName contents:nil attributes:nil];

//start writing data to file
NSFileHandle *file = [NSFileHandle fileHandleForWritingAtPath:fileName];
[file writeData:[@"some logging data" dataUsingEncoding:NSUTF8StringEncoding]];
```

creating/writing to a log file

'STORAGE' WITHIN A LOGFILE

SPOTTING A VULNERABILITY

-> SCOPE OUT THE DISASSEMBLY, OR SIMPLY RUN FILEMON AND DUMP THE LOG FILE(S)

```
BLX     NSSearchPathForDirectoriesInDomains

MOV     R1, #(selRef_objectAtIndex_ - 0xABF6C)
ADD     R1, PC ; selRef_objectAtIndex_
LDR     R1, [R1] ; "objectAtIndex:"
BLX     _objc_msgSend        ;[NSSearchPathForDirectoriesInDomains… objectAtIndex:0]

MOV     R1, #(selRef_createFileAtPath_contents_attributes_ - 0xAC638)
ADD     R1, PC
LDR     R1, [R1] ; "createFileAtPath:contents:attributes:"
BLX     _objc_msgSend        ;[[NSFileManager defaultManager] createFileAtPath:…]

MOV     R1, #(selRef_fileHandleForWritingAtPath_ - 0xAC670)
ADD     R1, PC
LDR     R1, [R1] ; "fileHandleForWritingAtPath:"
BLX     _objc_msgSend        ;[NSFileHandle fileHandleForWritingAtPath:fileName];
```

also: tail -f /var/log/syslog

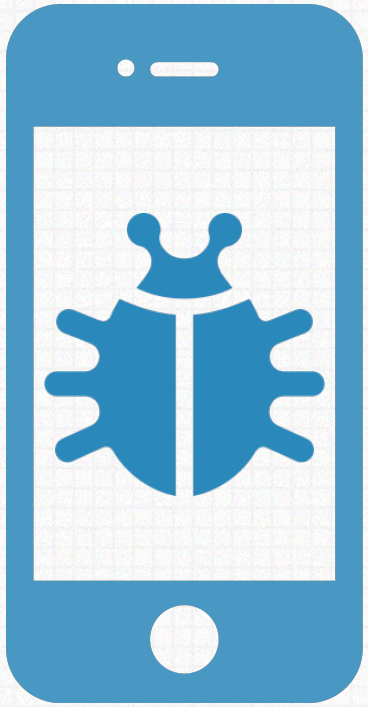
Creating a log file

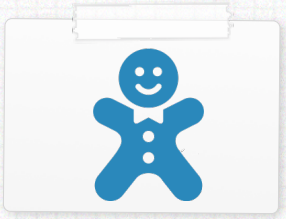
```
+0000 PUSHER: Subscribing to channel: [member-136746]
+0000 PUSHER: Will authorize channel with request: <NSMutableURLRequest: 0x17784090> { URL: https://app.              .com/pusher/auth }
+0000 PUSHER: Authorization headers: {
    Cookie = "csrftoken=XexecXibPCy3iLL8Jy0GQlF4xoTVgycV; sessionid=v1t8qnt3plhnxl1ph9bms32xkem1856w";
}
```

OTHER BUGZ

THERE ARE MANY OTHER PLACES WHERE IOS APP VULNERABILITIES CAN POP UP

-> SOME INCLUDE OS LEVEL DESIGN 'ISSUES', WHILE OTHERS ARE RESULT OF POORLY DESIGNED APPS

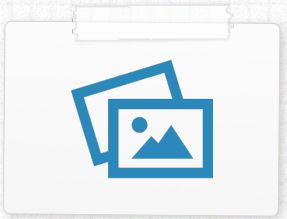




cookies (binary)

cache'd requests/responses

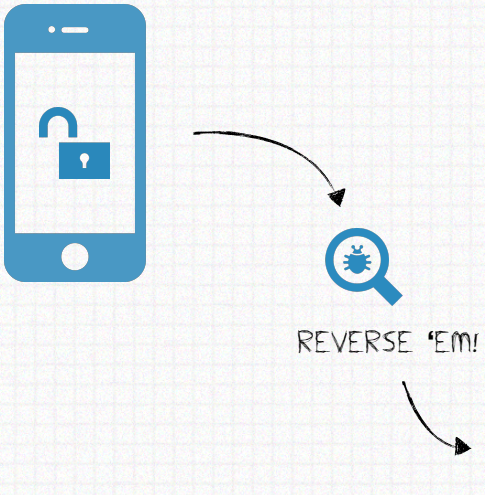
"inter-app" comms



screen shots (OS)

SO GO FORTH!

(INSECURE) APPS ARE EVERYWHERE...



Currently, the security in iOS apps is generally an afterthought. Using the techniques described in this presentation, you should be able to reverse any iOS app and hopefully find some interesting security vulnerabilities.

QUESTIONS/ANSWERS

Q&A

and mahalo for your attention!

CONTACT US

patrick@synack.com

www.synack.com