Crowd Security Intelligence

(download slides)

syn.ac/t2_reversingApps

@patrickwardle

Reversing iOS Apps

a practical approach

PATRICK WARDLE

Synack: dir. of R&D

NASA: autonomous software dev.

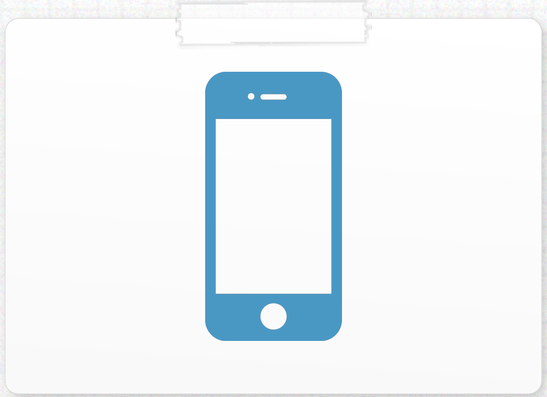
VRL: chief science officer

NSA: [redacted]

AN OUTLINE

THE TALK TODAY WILL COVER A SOLID AMOUNT OF MATERIAL

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



THE IOS ENVIRONMENT

PREP'ING A REVERSING ENVIRONMENT

REVERSING TECHNIQUES

IOS APP VULNERABILITIES

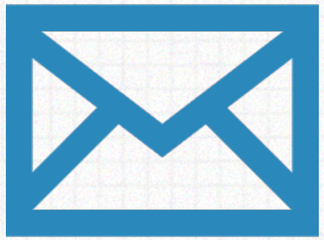
Background

...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

+

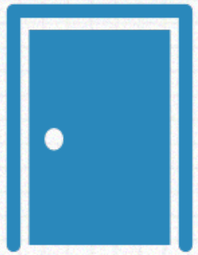
PHOTOS

+

FINANCIAL INFO

+

GEO-LOCATION



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

iDEVICE THEFT

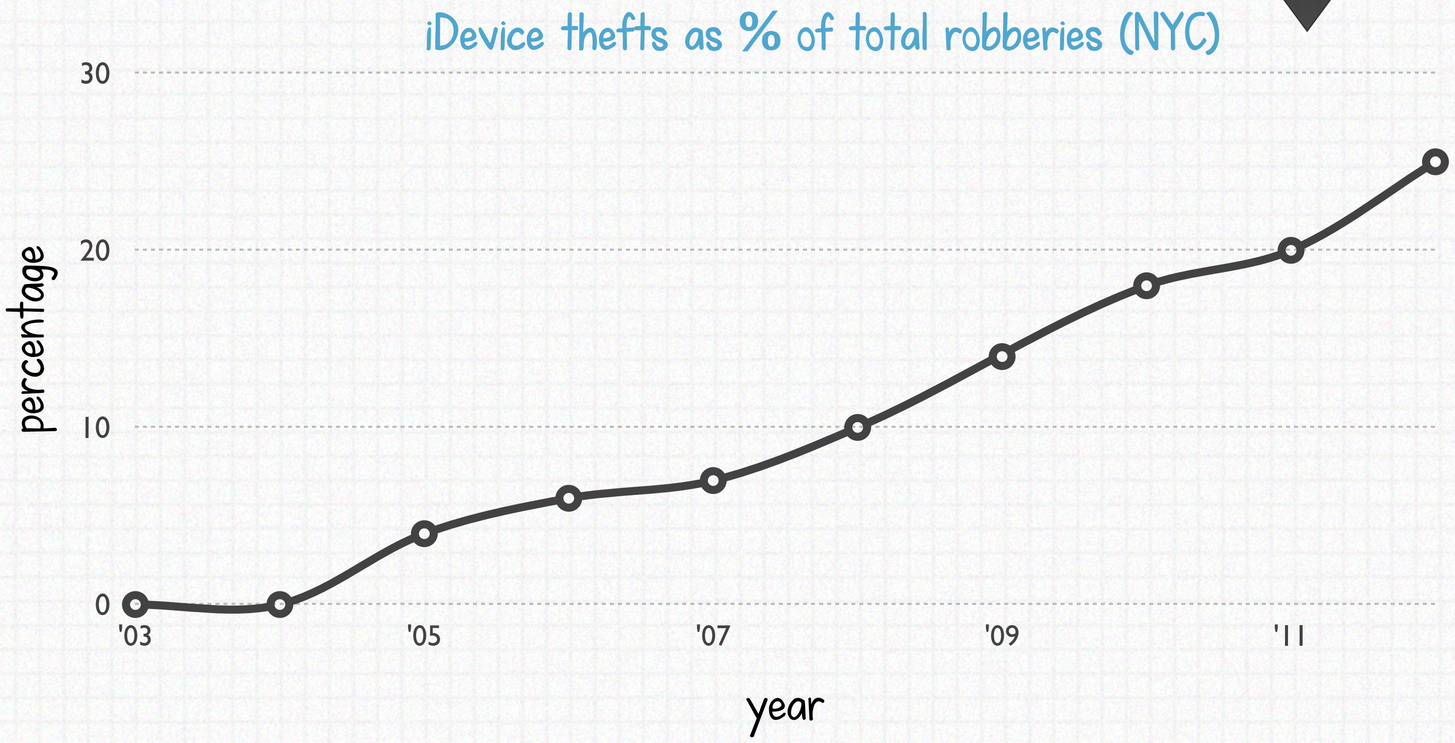
STOLEN IDEVICES

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

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

TRANSMISSIONS

sensitive data may be transmitted insecurely

PROCESS UNTRUSTED DATA

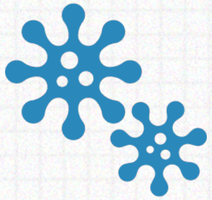
app process a lot of data from untrusted sources (attack surface)

CLOSED-SOURCE

apps are generally not open-source

BACKUPS

a juicy target for hackers (PC's 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

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"

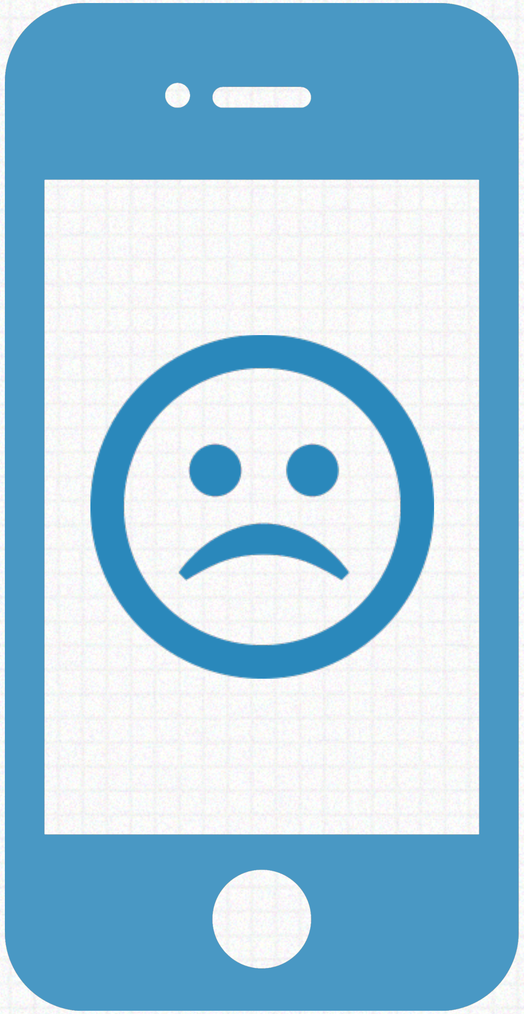
ETC...

"starbucks stored iOS app passwords and location data in clear text"

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

Synack

IN SHORT



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

REVERSE-ENGINEERING TO THE RESCUE!

REVEAL VULNERABILITIES

VERIFY THE APP

SECURITY AWARENESS++

MAKE SOME MONEY!

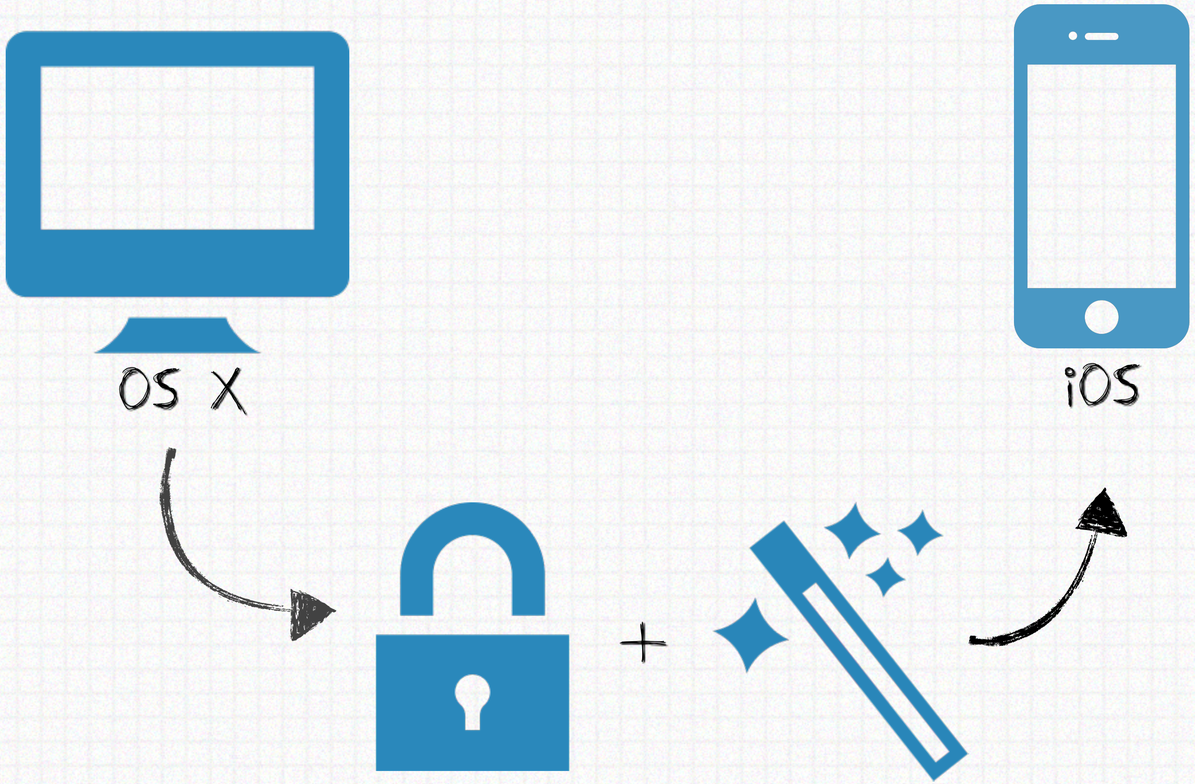
The iOS Environment

a brief technical overview

iOS

iOS is derived from OS X

-> basically just slimmed down with extra security



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

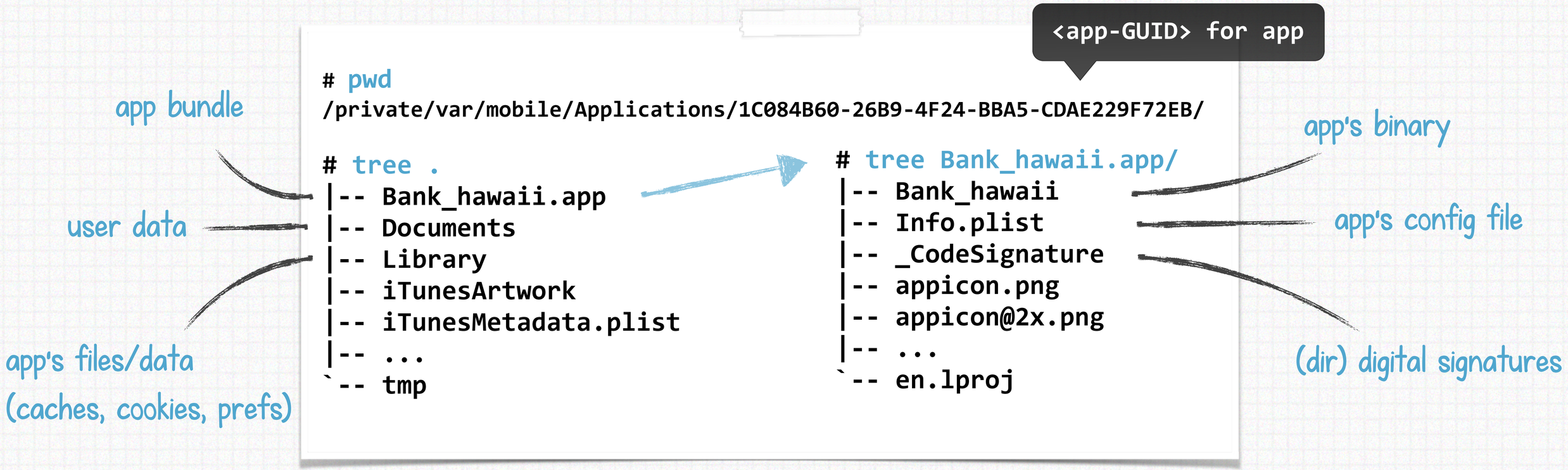
.APP's FILES

the app's binary, images,
meta-data, and more

iOS APPs

THE APP'S ON-DISK LAYOUT

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



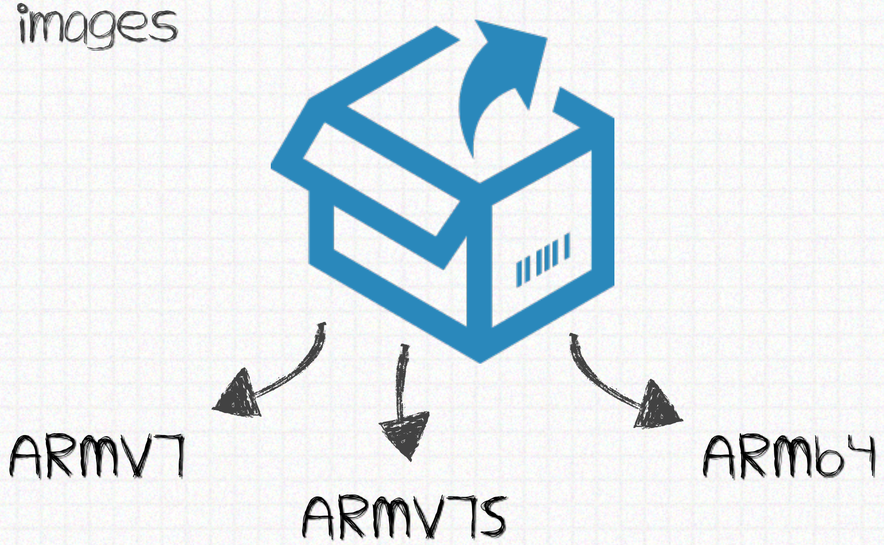
APP BINARY

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

describes binary's meta-data/layout

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

//load_command data
```

LOAD COMMANDS

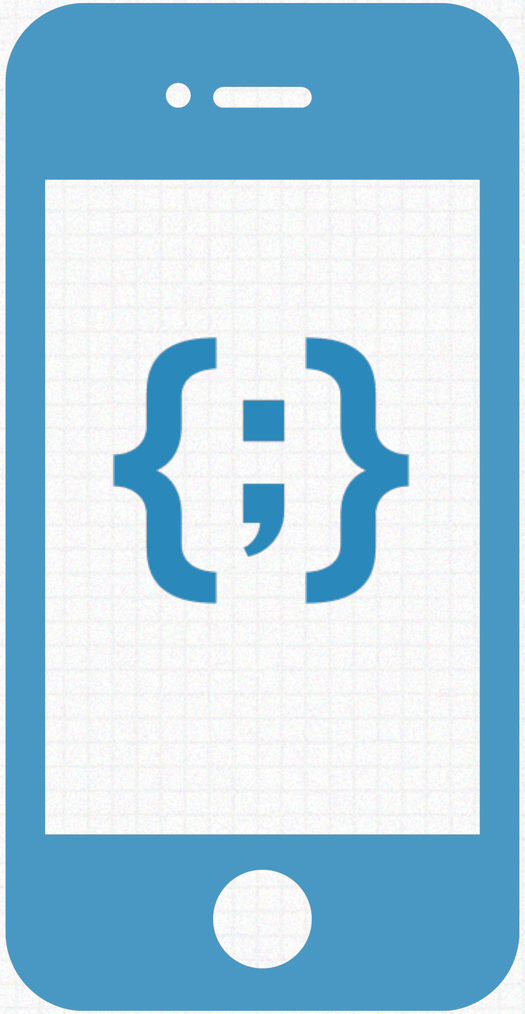
'instructions' how to setup/load the binary memory layout, thread context, etc.

RAW DATA

segments with code, data, etc

Synack

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/methods, etc.
- the programming language used to create iOS/OS X apps

```
//say hi!
NSLog(@"Hello, T2'ers");
```

OBJECTIVE-C

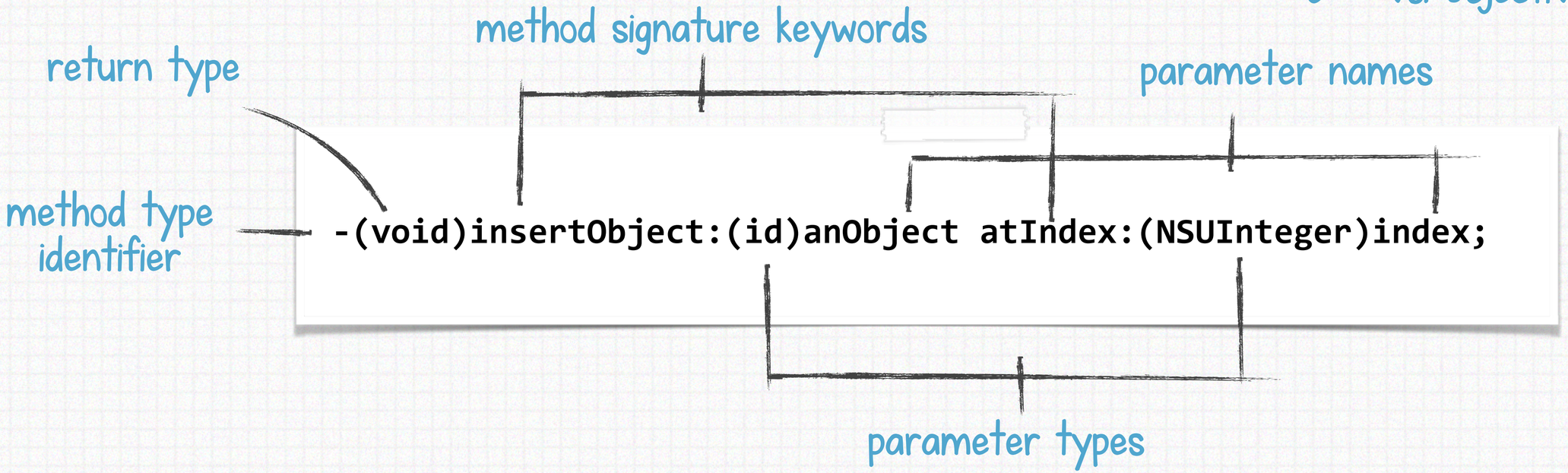
YES, THE SYNTAX IS QUITE 'ODD'

-> LOOKOUT FOR **@** AND **[ ]**

...AND TERMS SUCH AS 'MESSAGE PASSING' AND 'SELECTORS'

C++ vs. Objective-C

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


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



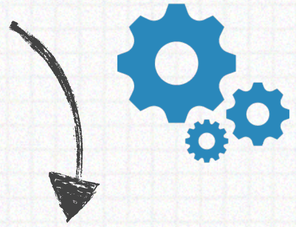
REVERSING OBJECTIVE-C

REVERSING IS SOMEWHAT NON-TRIVIAL

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

OBJECTIVE-C AND OBJC_MSGSEND

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



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

selector is the *name* of a method

```
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.
```

...ALL MESSAGES PASS THRU OBJC_MSGSEND

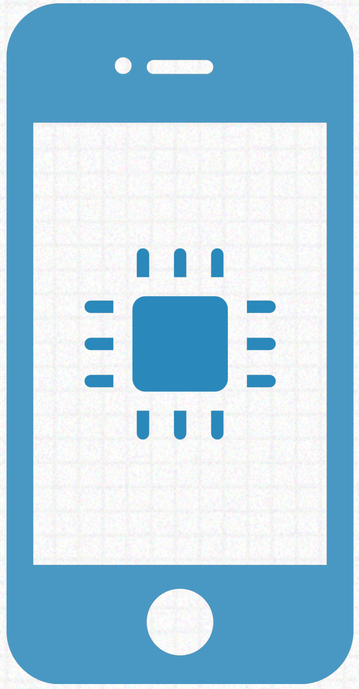
OBJC_MSGSEND

Synack

ARM ARCHITECTURE

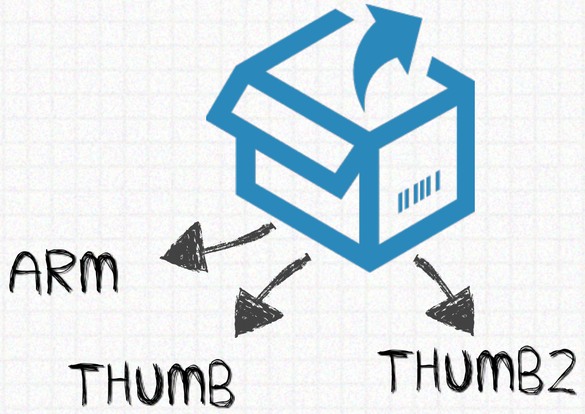
ARM POWERS MOBILE PROCESSORS EVERYWHERE

-> 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'

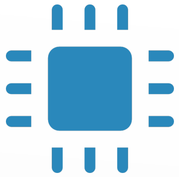
Thumb2: 2 or 4 byte instruction length
-> 'code density' of Thumb w/ performance of ARM

'ARM32'

ARM ARCHITECTURE ON (MODERN) 32-BIT CPUS

-> THE **ARMV7** PROCESSOR INSTRUCTION SET (IPHONE 3GS), 32-BIT ADDRESS SPACE & 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

memorize this info!

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

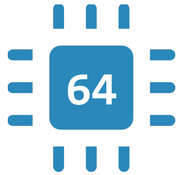
(FUNCTION) CALLING CONVENTION

ARM64

ARM ARCHITECTURE ON 64-BIT CPUS

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

[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

read: “ARM64 and You” (mike ash)

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

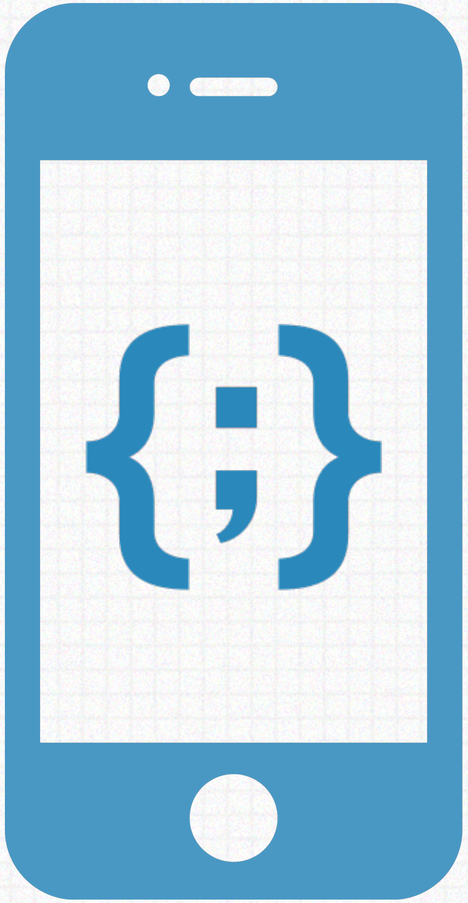
slight learning curve, but necessary for getting down & dirty!

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

COMPREHENSIVE

suites that perform that perform many tasks

REVERSING TOOLS



WHAT ABOUT WRITING YOUR OWN (OR FIXING OTHERS)?

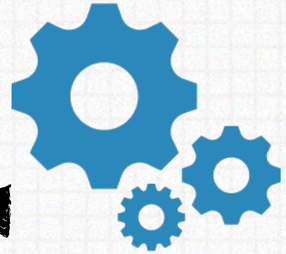
find it at iosOpenDev.com

IOS OPEN-DEV

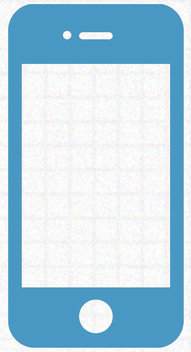
"sets up OS X and Xcode for 'open' development"

download/run installer

write some code



compile



copy to, then run!

Synack

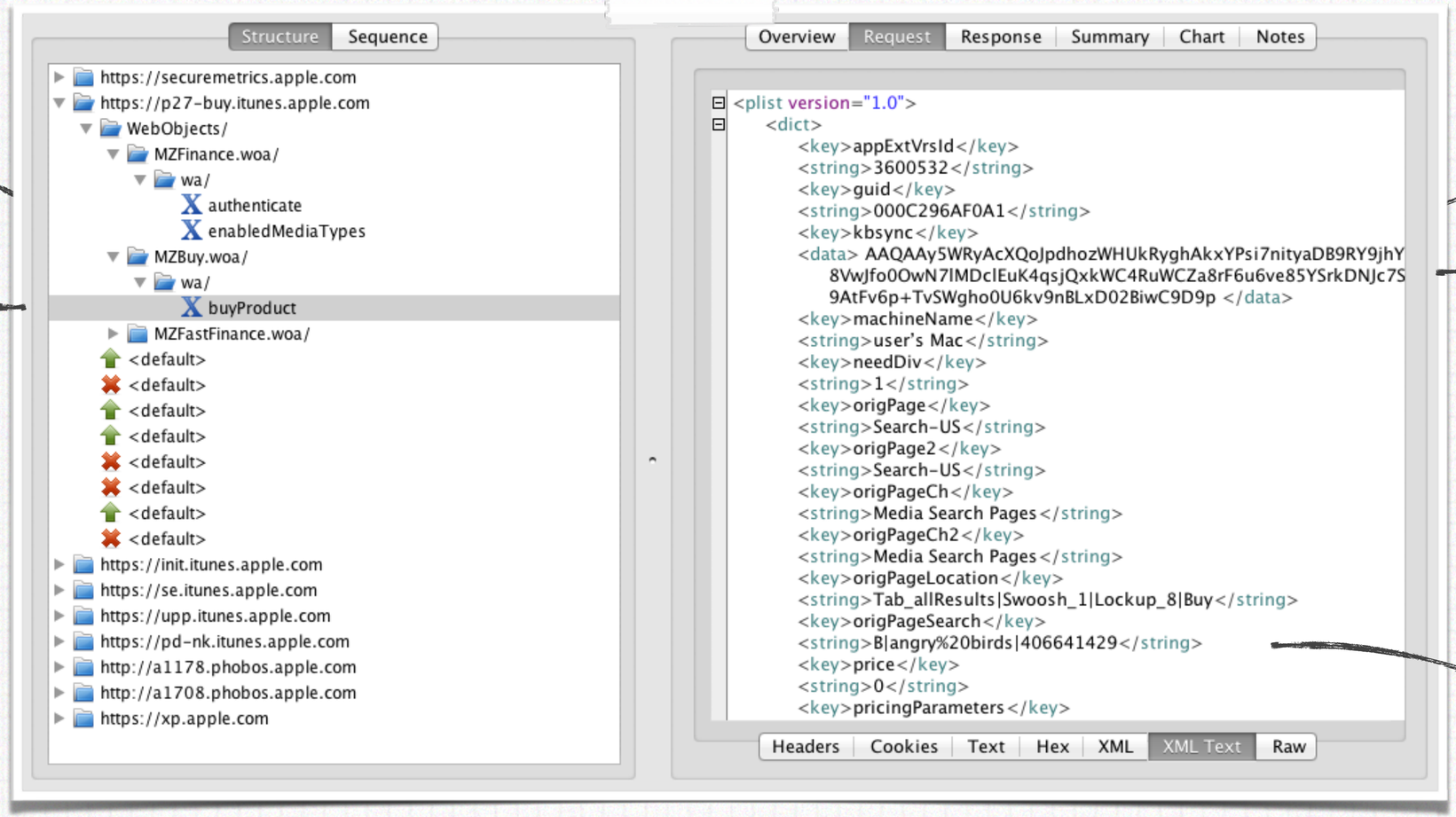
GETTING APPS

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?

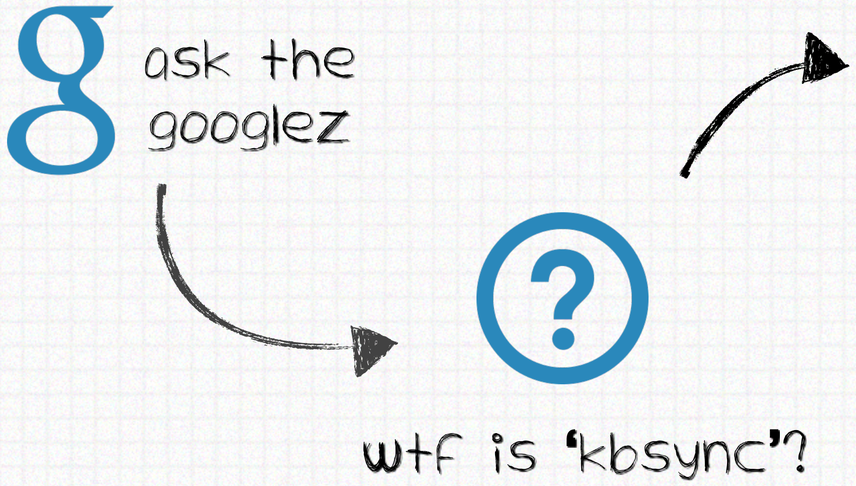
app name

Synack

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

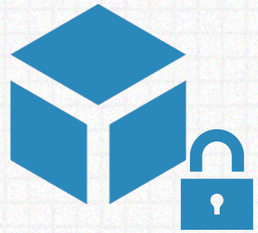
GRABBING APPS, IN TWO EASY STEPS

iTUNES

Find the app in iTunes

'buy' the app

iTunes downloads



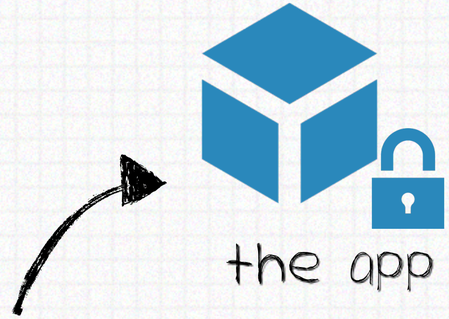
the app! (.ipa)

(ab)USING iTUNES

SO PROGRAMMATICALLY, HOW IS THIS DONE?

-> ACTUALLY, QUITE EASILY: APPLESCRIPT FTW :)

itms -> 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

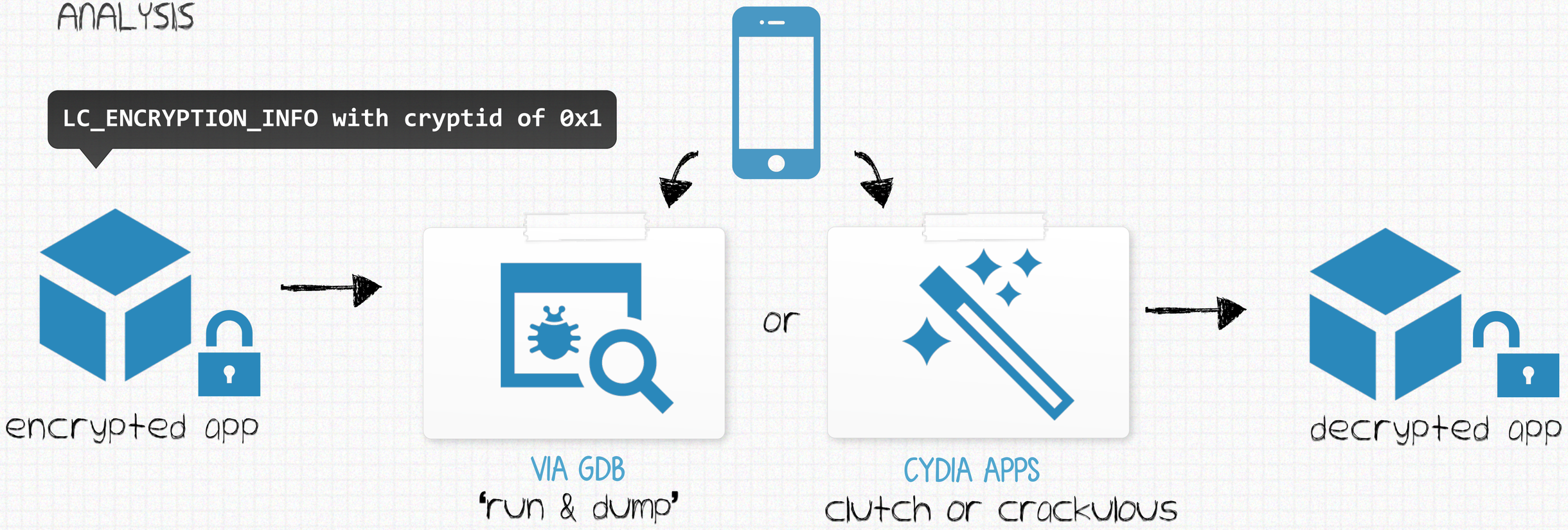
```
//AppleScript (open.scpt)
set elements to get entire contents of
    window 1
...

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



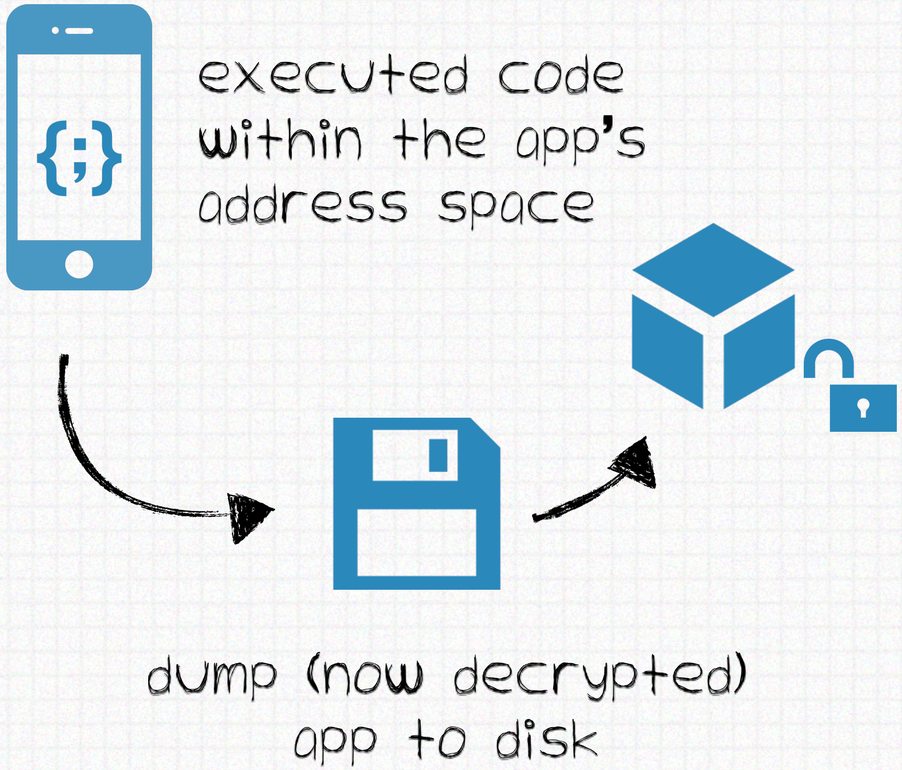
VIA GDB
'run & dump'

CYDIA APPS
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 application'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

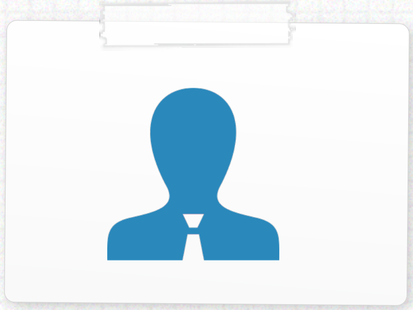
...methods to the madness

OTOOL

OTOOL: 'OBJECT FILE DISPLAYING TOOL'

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

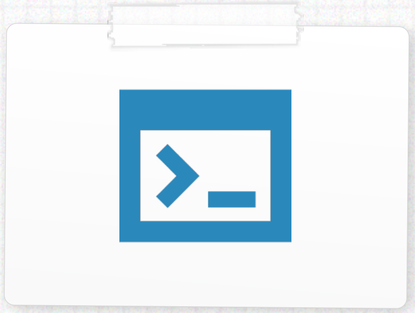
`# otool -f | -h`



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, the most accurate

interface

instance variables

method declarations

```
# class-dump-z Bank_hawaii

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

-(void)handleNetworkEvent:(unsigned long)event;
-(void)addBasicAuthenticationHeaderWithUsername:(id)name andPassword:(id)passwd;
-(void)attemptToApplyCredentialsAndResume;
-(void)saveCredentialsToKeychain:(id)keychain;
...
@end
```

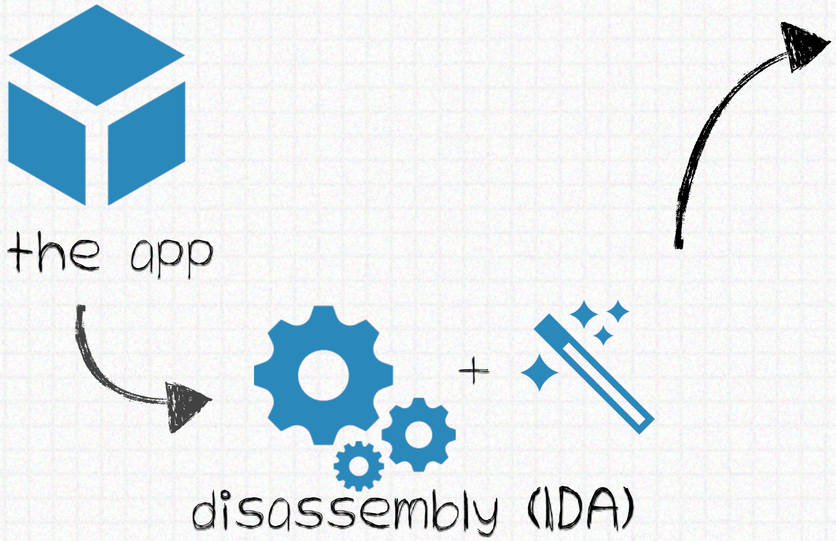
IDA PRO

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);
```




```
;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, ...);
```

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

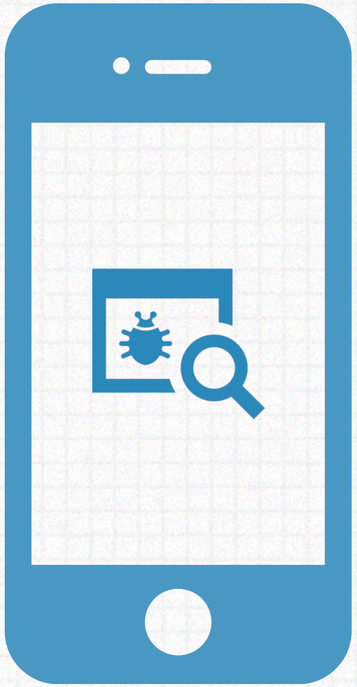
```
NSDate* date = objc_msgSend(classRef_NSDate, @selector("date"));
```

Synack

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

Synack

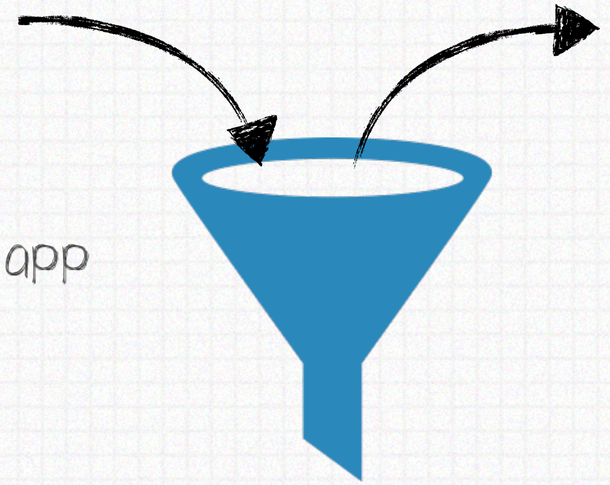
NETWORK ANALYSIS

CONCEPTUALLY, QUITE SIMPLE:

→ SNIFFING SOME TRAFFIC

NETWORK TRAFFIC ANALYSIS

execute the app



the proxy
(collect & analyze)

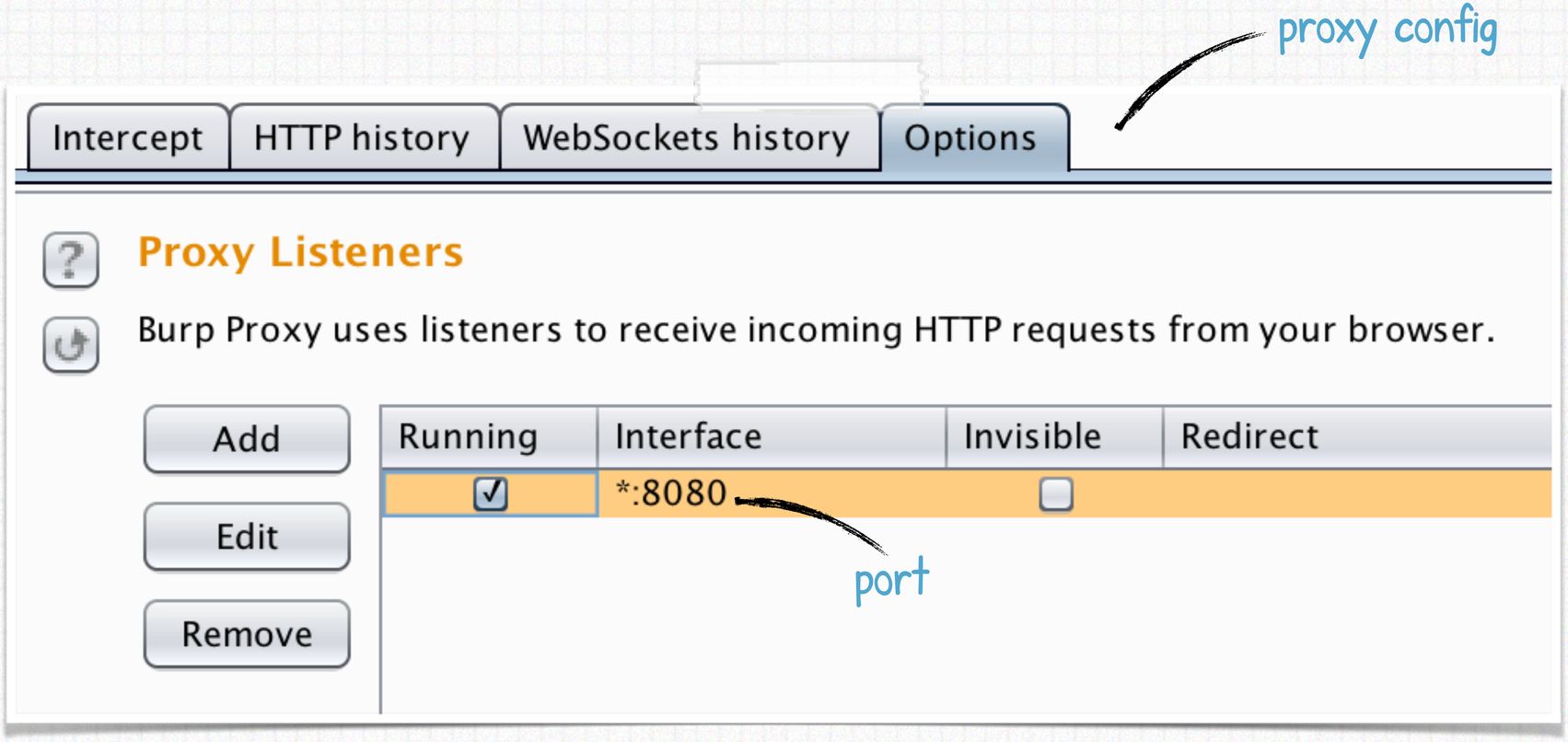
internet/"cloud"

NETWORK ANALYSIS

SO PRACTICALLY, HOW IS THIS DONE?

-> FIRST A PROXY SHOULD BE SETUP/CONFIGURED

BURP



NETWORK ANALYSIS

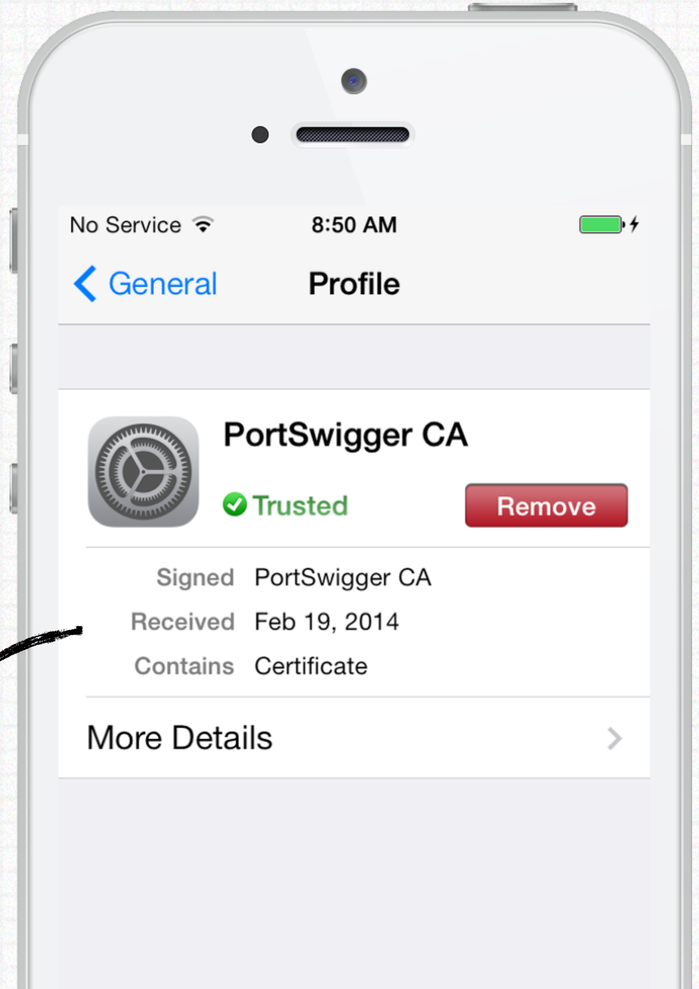
SO PRACTICALLY, HOW IS THIS DONE?

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

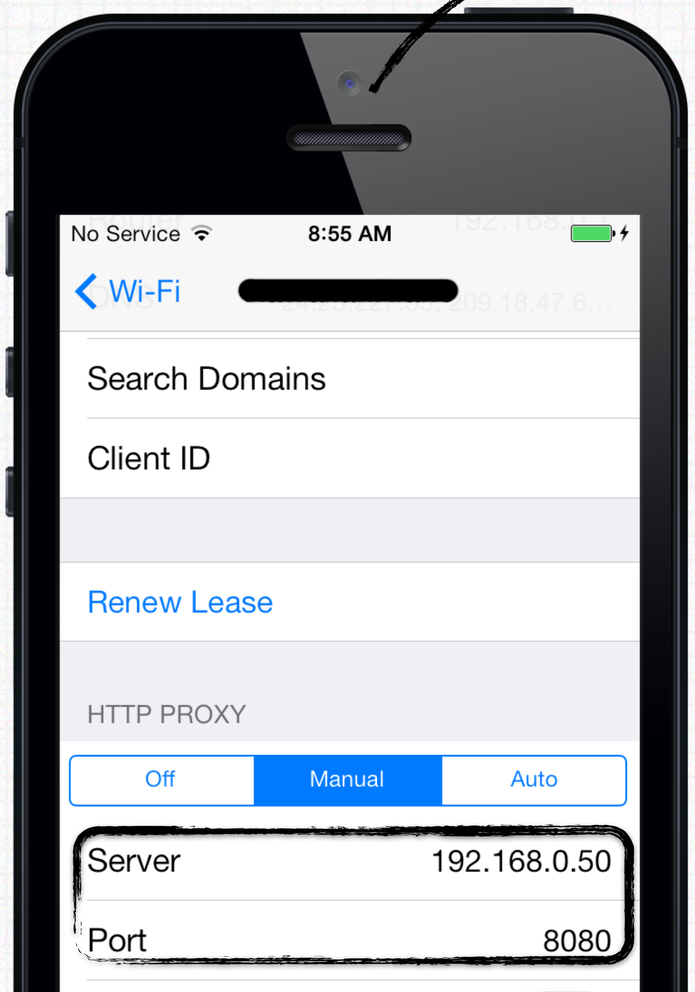
DEVICE

send cert via email

BURP profile



iOS proxy settings



Synack

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

capture/analyze file event(s)

Synack

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);
}
```

PATCH (FOR IOS 7+)

FILEMON
(newosxbook.com)

create

```
# ./filemon

Bank_hawaii Created  /Application Support/analytics/analytics.db-journal
DEV: 1,3 INODE: 121171 MODE: 81a4 UID: 501 GID: 501 Arg64: 300649589561

Bank_hawaii Deleted /Application Support/analytics/analytics.db-journal
DEV: 1,3 INODE: 121171 MODE: 81a4 UID: 501 GID: 501 Arg64: 300650449950

Bank_hawaii Created /Preferences/com.fis.140SUB.plist.l0mitdo
DEV: 1,3 INODE: 121172 MODE: 8180 UID: 501 GID: 501 Arg64: 300677061026

Bank_hawaii Renamed /Preferences/com.fis.140SUB.plist.l0mitdo
DEV: 1,3 INODE: 121172 MODE: 8180 UID: 501 GID: 501
```

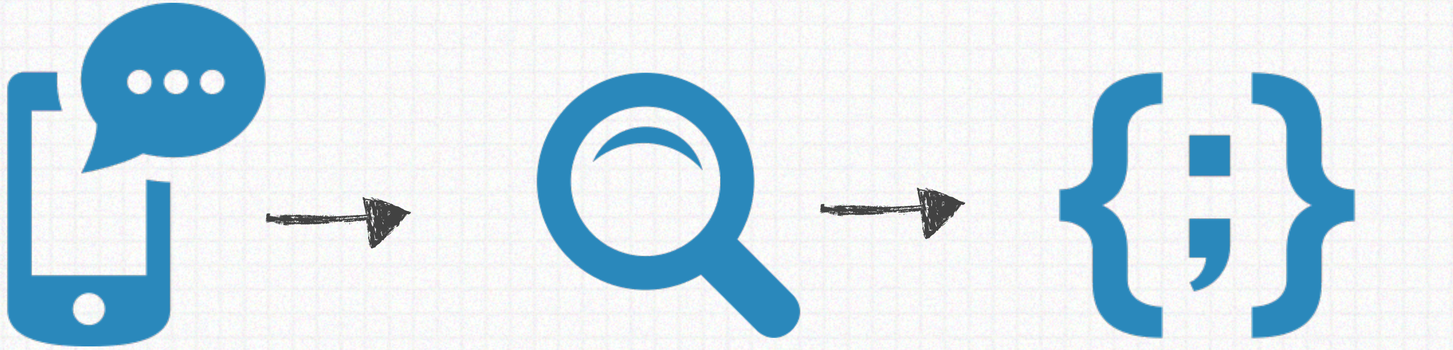
rename

APP DEBUGGING

CONCEPTUALLY, QUITE SIMPLE:

→ MONITORING CODE EXECUTION

APP DEBUGGING



APP DEBUGGING

APP DEBUGGING WITH GDB

-> GDB IS THE DE FACTO DEBUGGER FOR IOS

```
# 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 (iOS 7+): cydia.radare.org

```
(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

APP INSTRUMENTATION

CONCEPTUALLY, QUITE SIMPLE:

APP INSTRUMENTATION

INSTRUMENTING AN APP

execute the app

inject code or interpreter

bypass client-side logic
execute hidden code
manipulate the app runtime

APP INSTRUMENTATION

USING CYCRIPT

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

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

save/install

```
# cycript -p <pID>
```

inject into a process

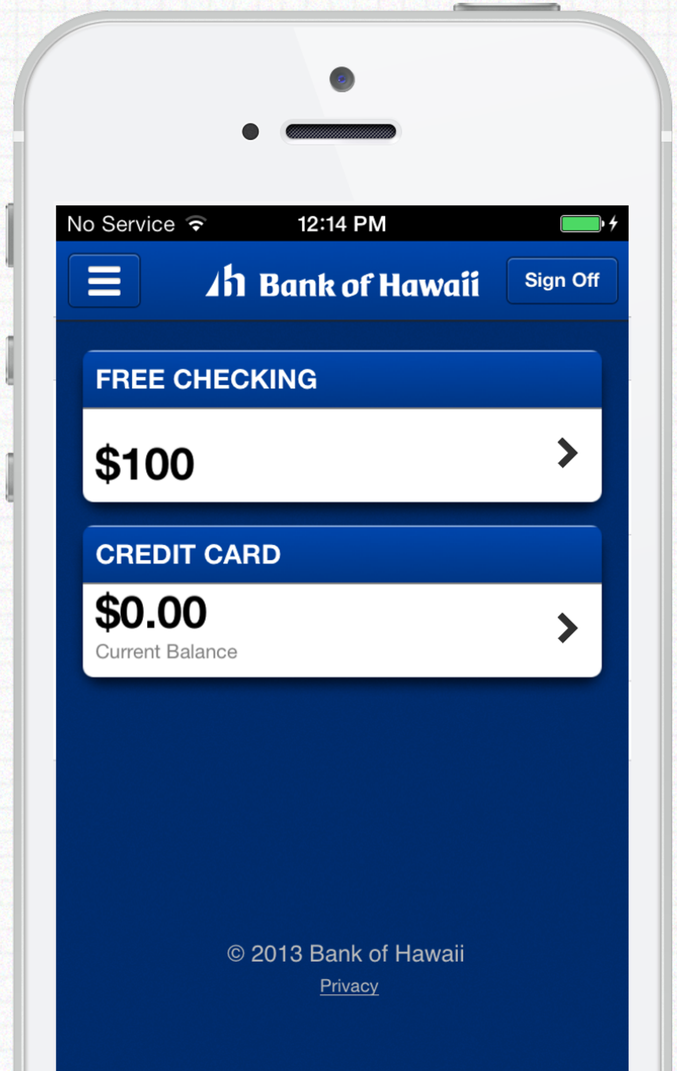
```
cy# *UIApp
{isa:#"UIApplication",_delegate:#"<TiApp: 0x17da6e10>”,
        _touchMap:0x17db2860,_exclusiveTouchWindows:...}

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

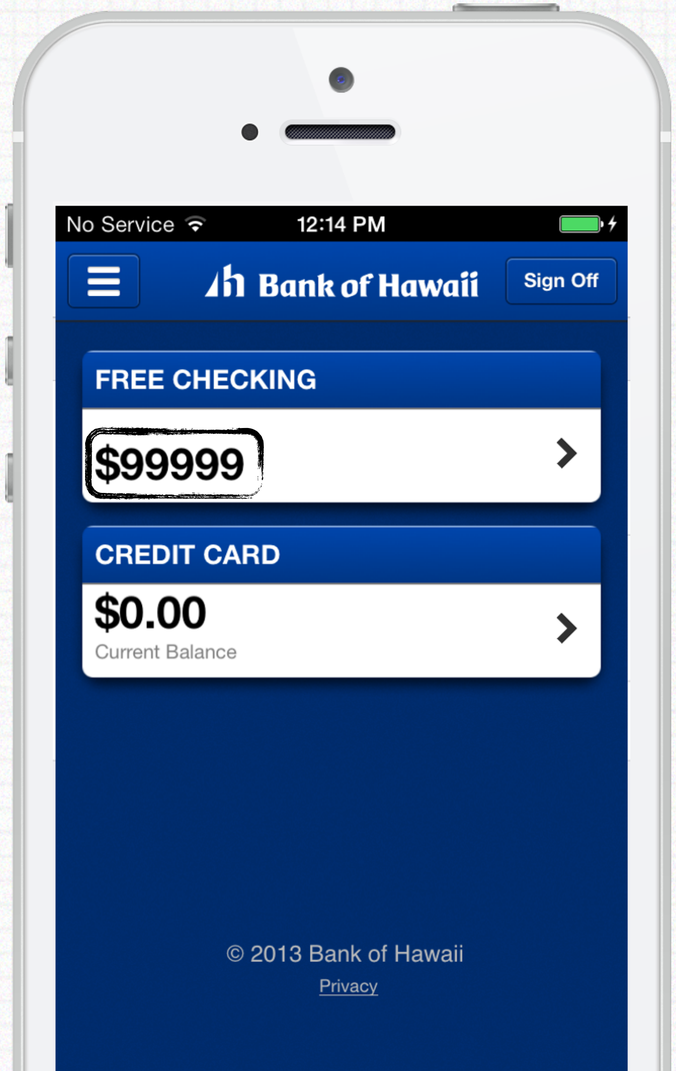
CYCRIPT CONSOLE

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
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

Synack

NETWORK (in)SECURITY

A BROKEN SSL IMPLEMENTATION

-> REMEMBER, DON'T ALLOW SELF-SIGNED CERTS!

bug in actual app

class

```
MOVT R8, #(:upper16:(classRef_NSURLRequest - 0xC254))
ADD  R8, PC ; classRef_NSURLRequest

MOV  R2, #(selRef_setAllowsAnyHTTPSCertificate_forHost_ - 0xC2A4)
ADD  R2, PC
LDR  R4, [R2]      ;"setAllowsAnyHTTPSCertificate:forHost:"

LDR  R5, [R8]      ;_OBJC_CLASS_$_NSURLRequest

MOV  R0, R5        ;_OBJC_CLASS_$_NSURLRequest
MOV  R1, R4        ;"setAllowsAnyHTTPSCertificate:forHost:"
MOVS R2, #1        ;'YES'
MOV  R3, R8        ; the host
BLX  _objc_msgSend
```

method

invoke method "setAllowsAnyHTTPSCertificate:forHost:"

Synack

NETWORK (in)SECURITY

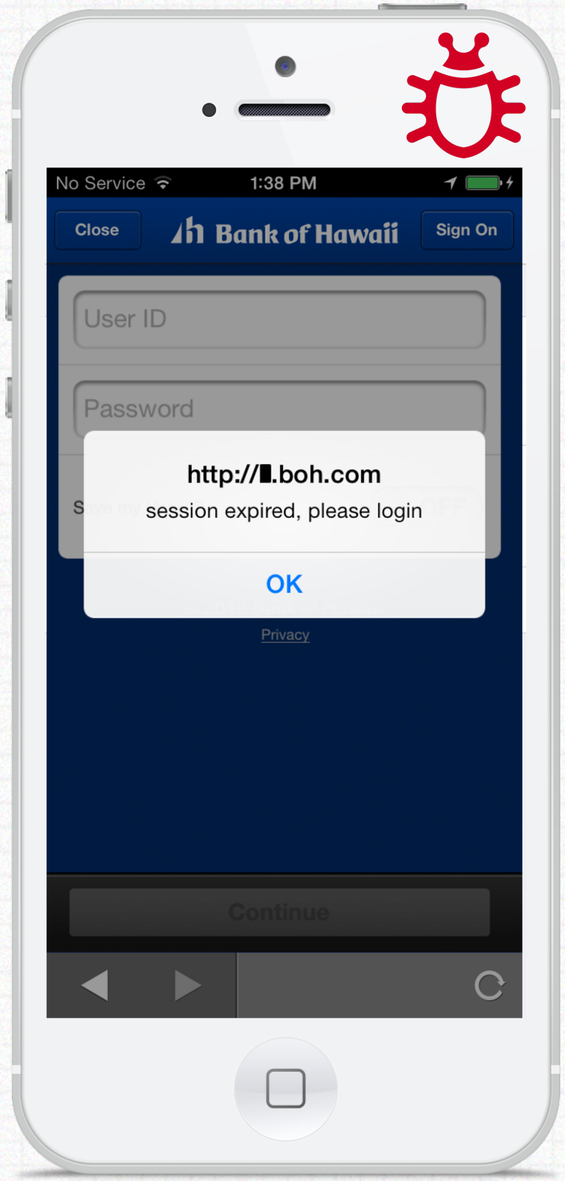
SPOTTING A VULNERABILITY DYNAMICALLY

-> 'SNIFF'/PROXY NETWORK TRAFFIC - CAN IT 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

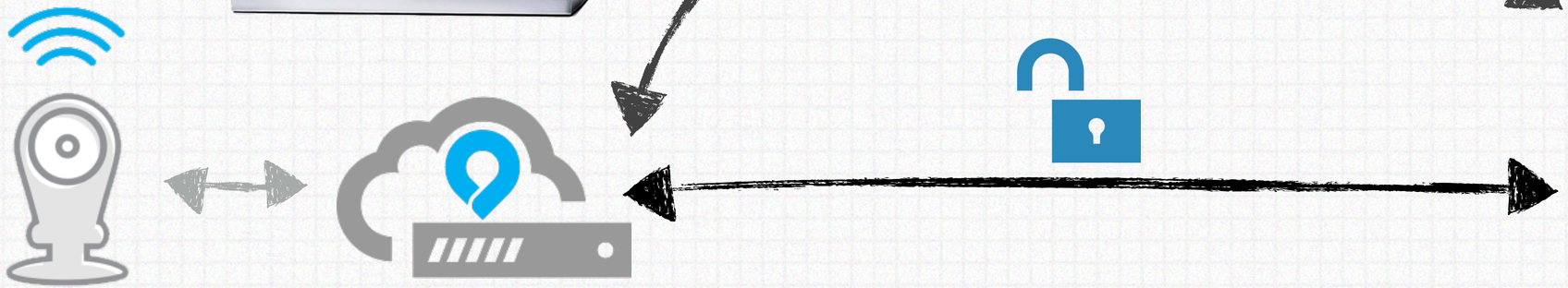


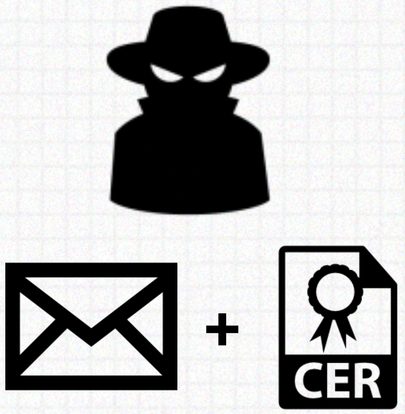

```
# tail -f /var/log/syslog

Bank_hawaii: [DEBUG] New scheme: <NSMutableURLRequest: 0x18816fa0> { URL: http://xx.boh.com/ }
Bank_hawaii: [DEBUG] New scheme: <NSMutableURLRequest: 0x175efe30> { URL: http://xx.boh.com/phoenix.zhtml }
```

NETWORK (in)SECURITY

PIN YOUR SSL CERTS!

-> ONLY TRUST YOUR CERT TO PREVENT MITM ATTACKS





HACKER'S CERT

NON-JAILBROKEN DEVICE

Synack

NETWORK (in)SECURITY

PIN YOUR SSL CERTS!

-> ONLY TRUST YOUR CERT TO PREVENT MITM ATTACKS

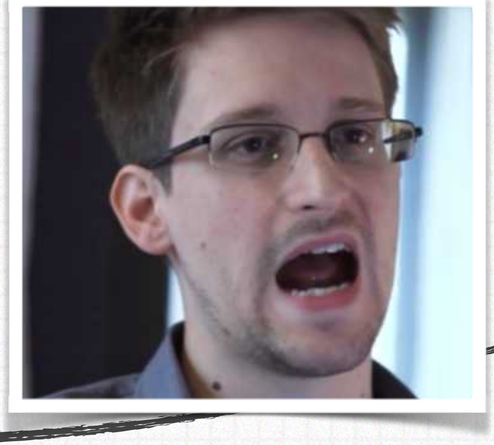
Host	Method	URL
https://www.dropcam.com	POST	/api/v1/login.login
https://www.dropcam.com	POST	/api/v1/notification_targets.register_apple_device

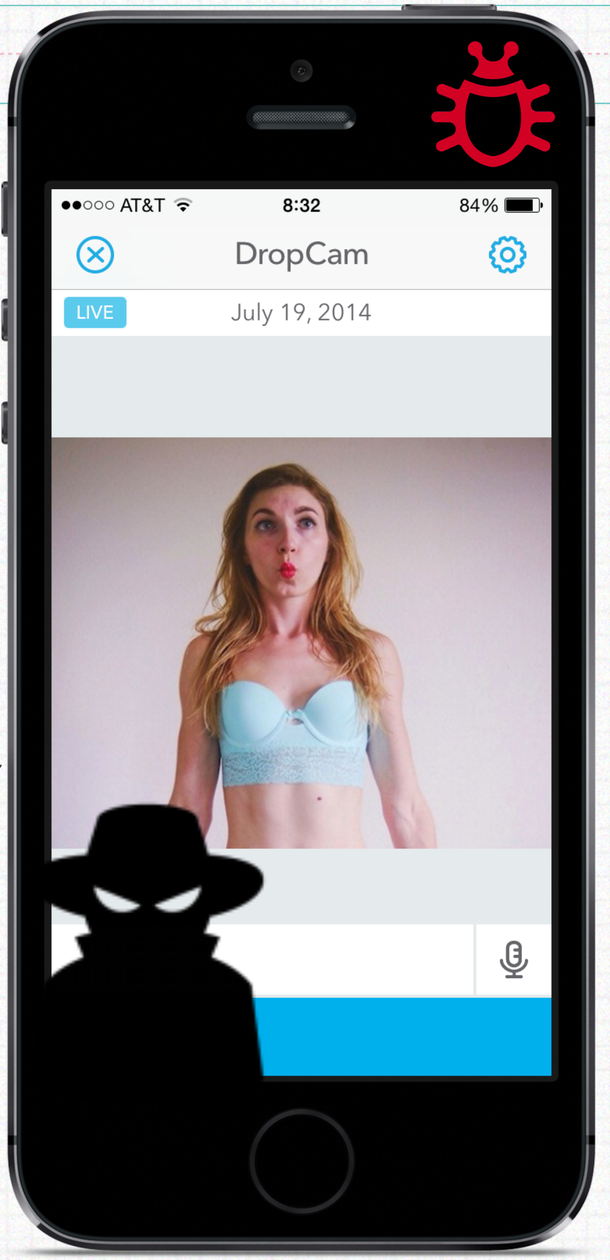
Raw | Params | Headers | Hex

```
POST /api/v1/login.login HTTP/1.1
Host: www.dropcam.com
User-Agent: Dropcam/3.3.2 (iPhone; iOS 7.1.1; Scale/2.00) Darwin

client=model%3DiPhone5%252C1%26client%3Diphone%26app_version%3D3.3.2%26device_name%3Dpatrick
2527s%2520iPhone%26system name%3DiPhone%2520OS%26system_version%3D7.1.1&password=[redacted]
&username=[redacted]
```

no dual-factor auth

no 'shared session' alert





Synack

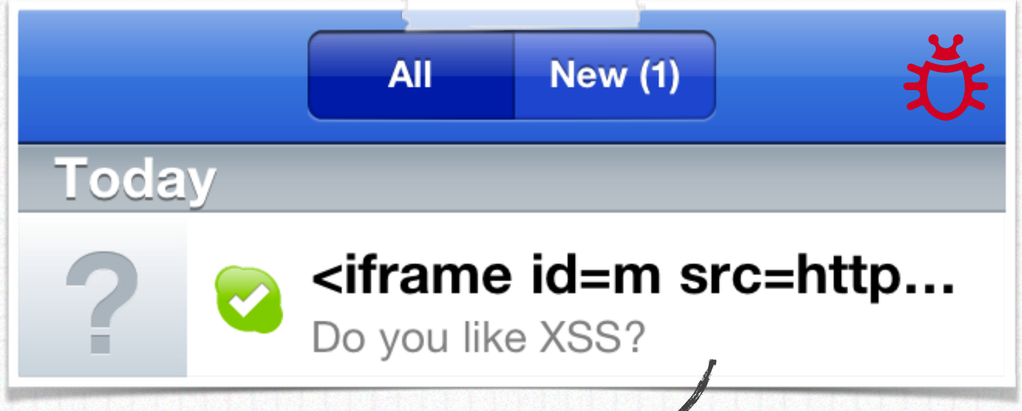
NETWORK (in)SECURITY

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)

see: 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: "FI",
  numbers: "{\"3140001337\": \"Mikko Hyppönen\"}"
}
```

SNAPCHAT API ABUSE (4.6M PHONE #S)

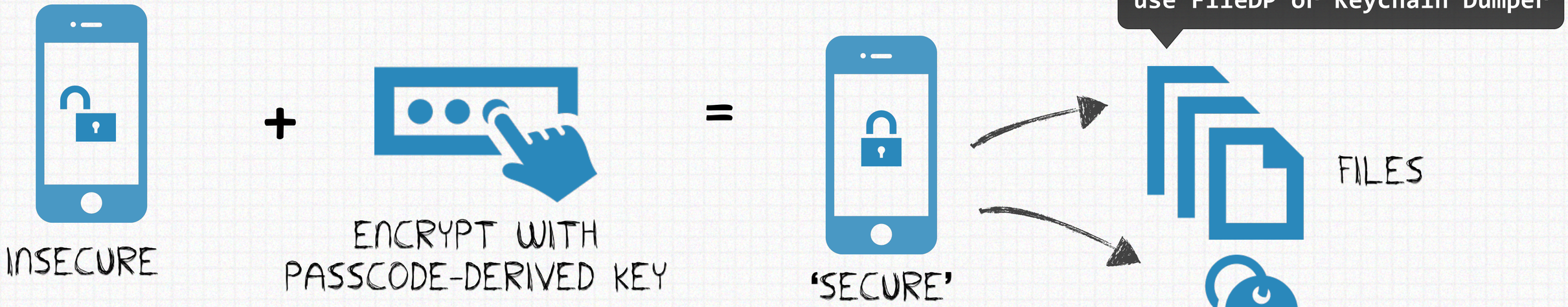
Synack

SECURE DATA STORAGE

FIRST, HOW TO DO IT RIGHT?

-> USE THE DATA PROTECTION APIS & IOS KEYCHAIN

The encrypted data partition is decrypted at boot. Somebody with access to the device can access this data without the passcode



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

Synack

INSECURE DATA STORAGE

COUNTLESS APP STORE SENSITIVE DATA INSECURELY

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

THE BINARY

PROPERTY LISTS

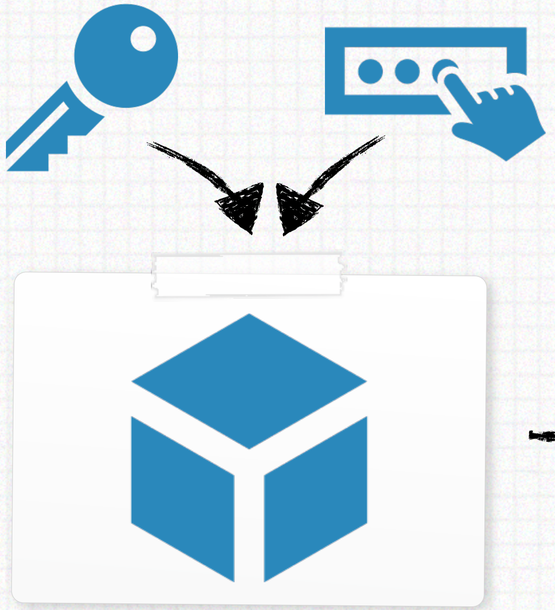
DATABASES

LOG FILES

STORAGE WITHIN THE BINARY

APPS MAY STORE SENSITIVE DATA WITHIN THEIR BINARY IMAGE

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



WITHIN THE BINARY

REST API

```
000E91ED "www.puffchat.me"
000E91FD "POST"
000E9202 "/v2/api/client/login"
000E9217 "key=dl81Vh2uorfNdj2Rt2M4EylW91uUsQRZwhQ99g7K0MRXeMYePS”
```

see: faptrackr.org

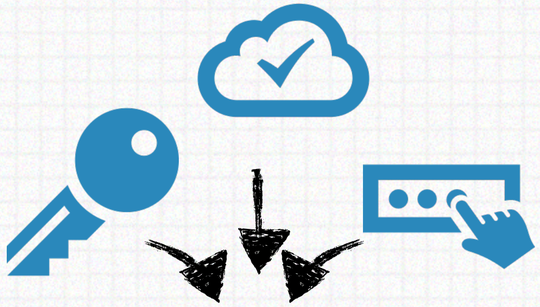
EMBEDDED "SECRET" API KEY (PUFFCHAT)

"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."

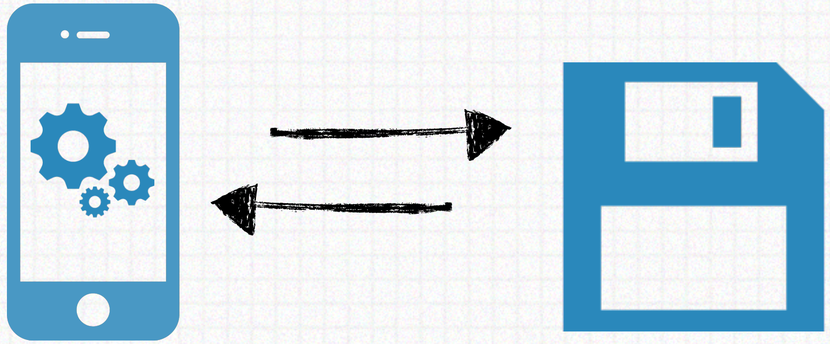
STORAGE WITHIN A 'PLIST'

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

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



WITHIN PLISTS



```
# plutil -convert xml1 <appID>.plist
```

```
//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 DUMP THE 'USER DEFAULTS' PLIST

User Defaults in the app's /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')

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

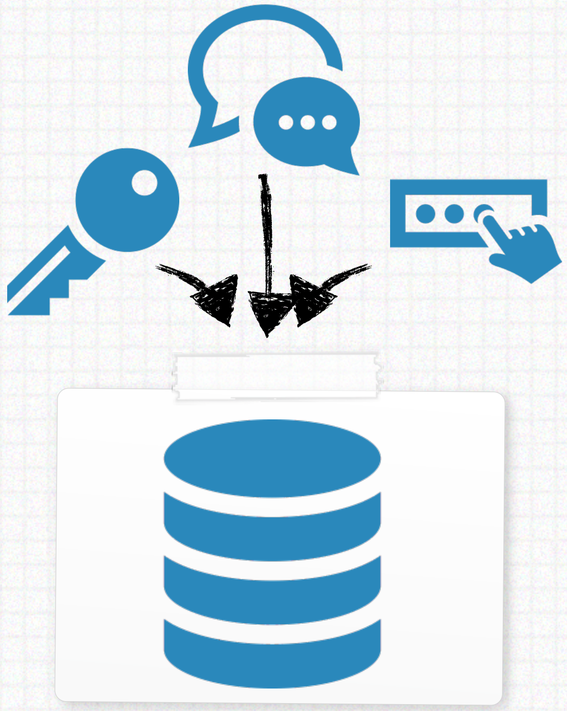
Synack

STORAGE WITHIN A DATABASE

SQLITE IS A COMMON METHOD OF STORING DATA

-> MAY FIND USER CREDENTIALS, OR OFTEN OTHER SENSITIVE INFO WITHIN APP'S DATABASES

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



WITHIN DATABASES

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|
```

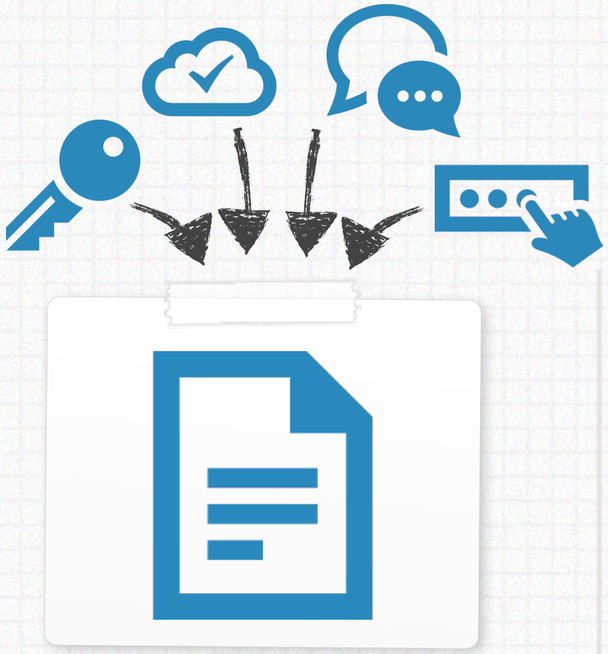
WHATSAPP CHAT HISTORY

Synack

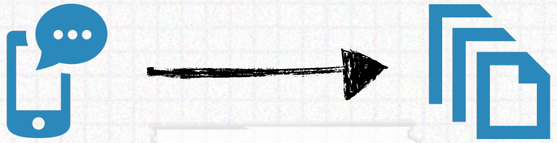
'STORAGE' WITHIN A LOGFILE

MANY APPS 'STORE' (LEAK) DEBUG/SENSITIVE INFO IN LOGS FILES

-> MAY FIND USER CREDS, SESSION DATA, ETC. 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];

//init file name
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";
```

Synack

GEOLOCATION

APPS OFTEN MAKE USE OF A USER'S LOCATION

-> THIS SHOULD BE TREATED WITH CARE & SECURED!

sniff with burp

iOS location services default to the highest level of accuracy

Original request | Edited request | Response

Raw | Params | Headers | Hex

```
POST /2.0/nearbyProfiles HTTP/1.1
Host: primus.grindr.com
Accept: */*
Accept-Encoding: gzip, deflate
Content-Type: application/json; charset=utf-8
Accept-Language: en;q=1, fr;q=0.9, de;q=0.8, ja;q=0.7, nl;q=0.6, it;q=0.5
Cookie:
Session-Id=[redacted]
Content-Length: 69
Connection: keep-alive
Proxy-Connection: keep-alive
User-Agent: Grindr/2.0.21 (iPhone; iOS 7.1.1; Scale/2.00)

{"lat":30.0599153,"lon":31.2620199,"filter":{"page":1,"quantity":50}}
```

REPORTING THE USER'S PRECISE LAT/LONG

Synack

GEOLOCATION

USING A USER'S LOCATION ISN'T CAN BE VERY USEFUL

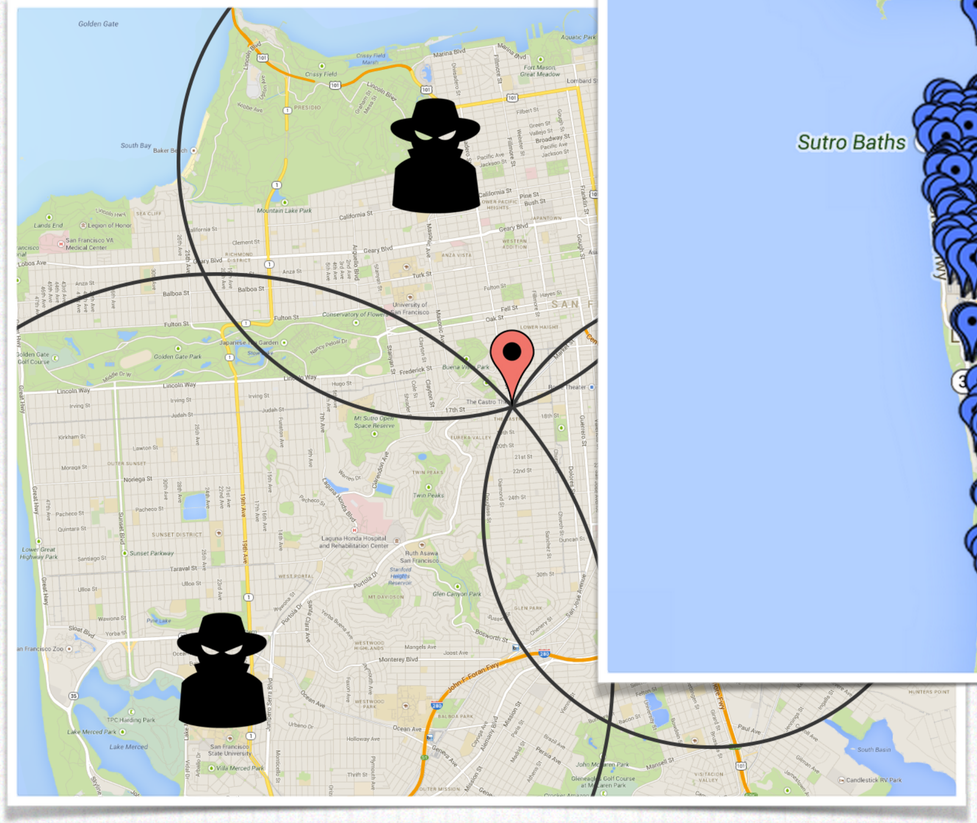
-> TO THE APP, OR AN ATTACKER!!

GRINDR; DOING IT WRONG

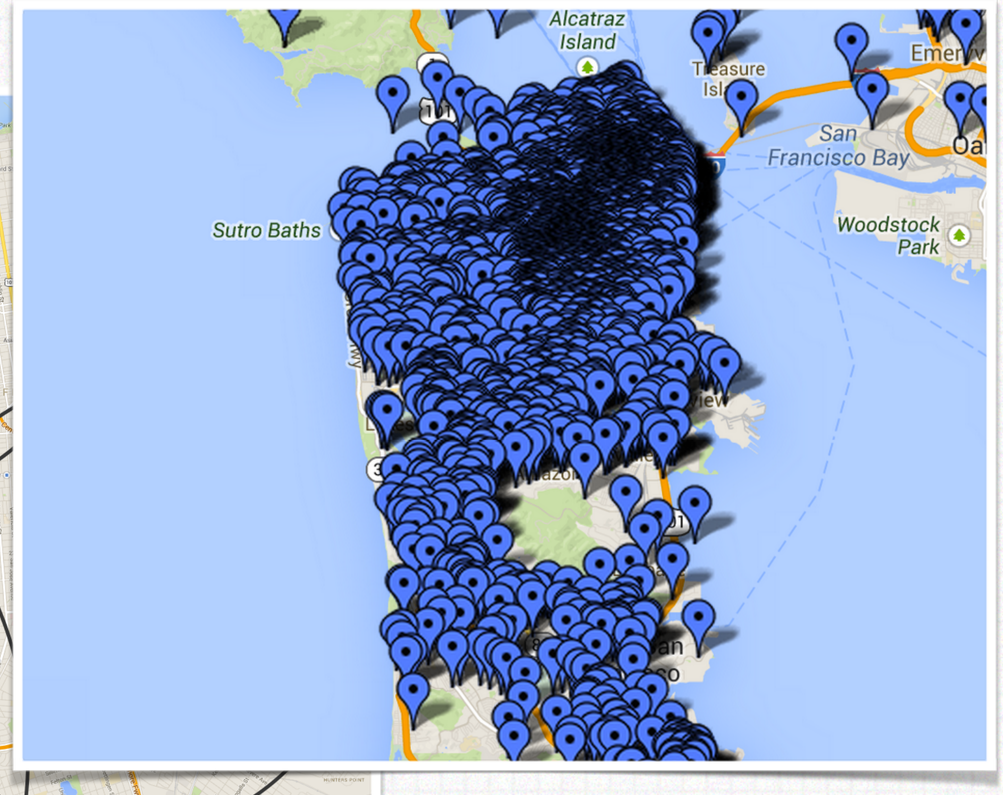
precise relative distances
location spoofing
anonymous non-limited APIs

"egyptian cops using grindr to hunt gays"

yikes!



TRILATERATION



GRINDR USER'S IN SF

Synack

GEOLOCATION

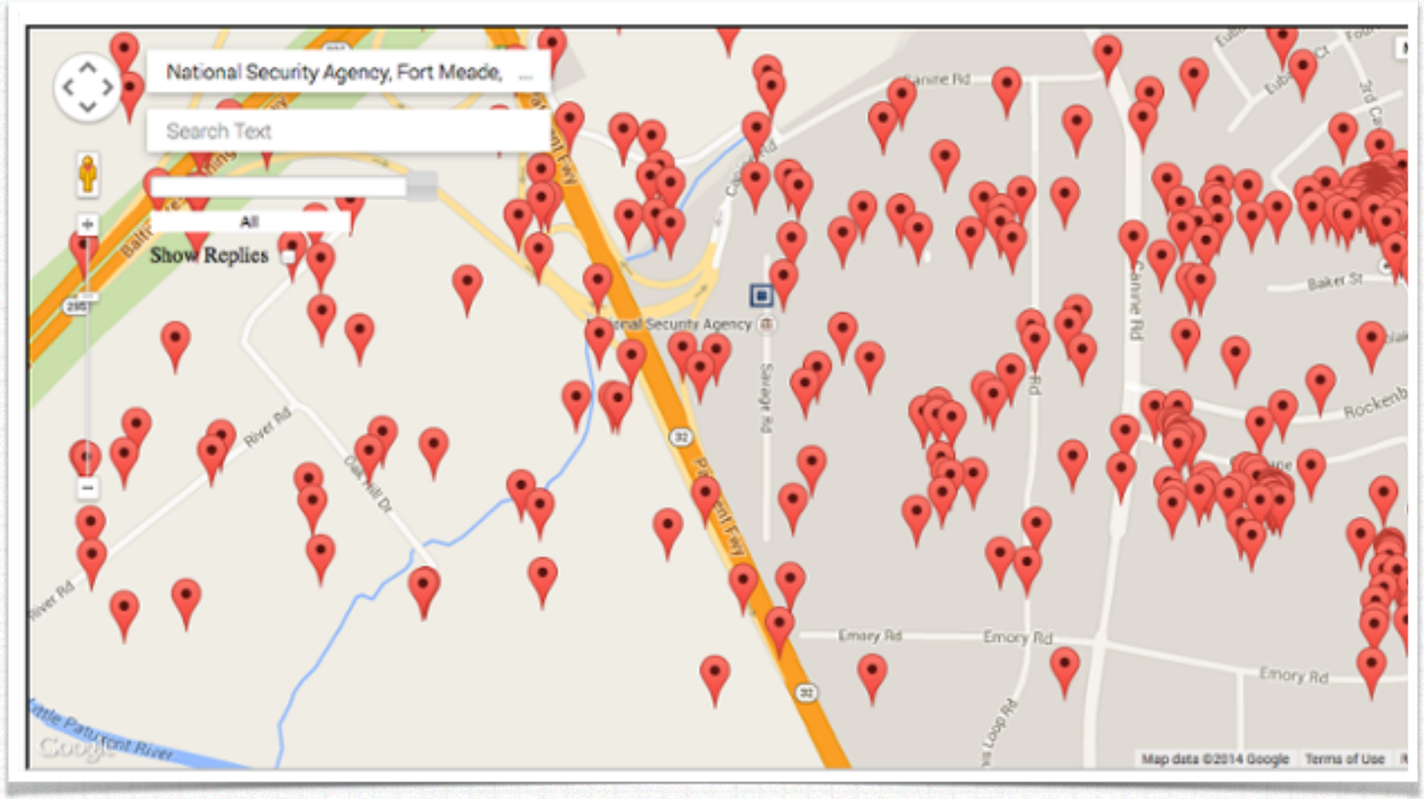
AN APP MAY ALLOW YOU TO OPT OUT OF GEOLOCATION

-> BUT SHOULD YOU TRUST IT? (PROBABLY NOT!)

WHISPER; ALSO DOING IT WRONG

users monitored (opt'd out of geo)
user data indefinitely stored
information shared with US DOD

"Revealed: how Whisper app tracks 'anonymous' users"
(the guardian)



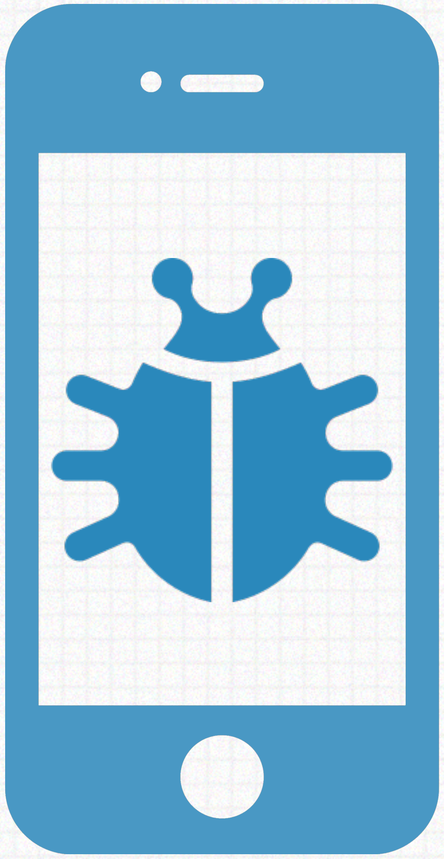
WHISPER MAP (USERS NEAR NSA)

Synack

OTHER BUGZ

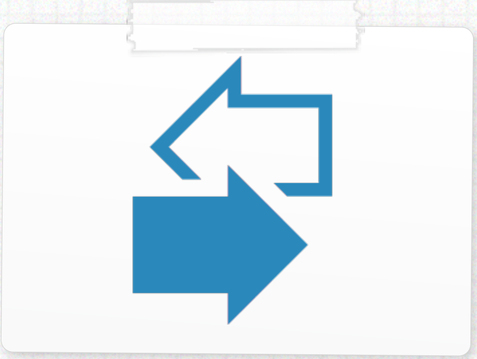
MANY OTHER PLACES WHERE IOS APP VULNERABILITIES CAN POP UP

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

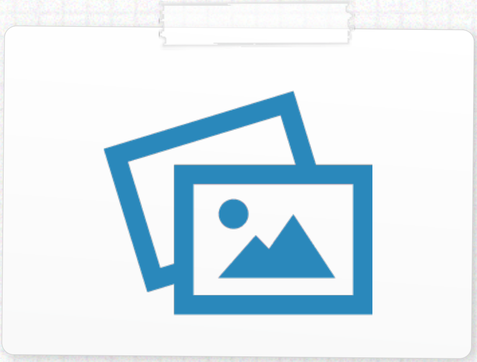


COOKIES (BINARY)

CACHE'D REQUESTS/RESPONSES



"INTER-APP" COMMS

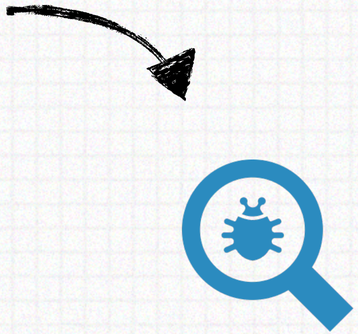


SCREEN SHOTS (OS)

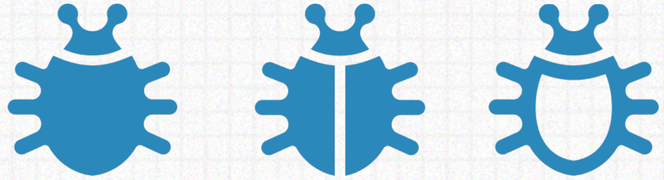
SO GO FORTH!

INSECURE APPS ARE EVERYWHERE

-> AND DON'T APPEAR TO BE GOING AWAY ANYTIME SOON :/



reverse' em



...find bugz

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

patrick@synack.com

@patrickwardle

syn.ac/t2_reversingApps

Synack

Crowd Security Intelligence

CREDITS (IMAGES/ICONS)

thezooom.com
deviantart.com (FreshFarhan)

iconmonstr.com
flaticon.com