Intro

- My perspective is that of a penetration tester (not developer)
- Info here is ideally of use to both testers and developers
- Assumes little to no iOS knowledge
- Focus is app security, not OS security
- Takeaways: be able fix or break your own or others’ iOS apps
Intro to iOS

It’s an OS, but with an i

- High-level API, “Cocoa Touch”
- Development in XCode
  - So yes, you need a Mac
- iOS Simulator (not emulator)
  - Compiles iOS apps to native code to run locally
- Applications written primarily in Objective-C
Objective-C

How to spot it from a very long way away

- C + Smalltalk…ish
- Uses “infix” notation:
  - `[Object messagePassedToObject:argument];`
- It is not to everyone’s tastes
- Some of us have very refined tastes
Objective-C in 1 slide
Defining Interfaces

```objective-c
@interface Classname : NSParentObject {
  SomeType aThing; // instance variables
}
+(type)classMethod:(vartype)myVariable;
-(type) instanceMethod:(vartype)myVariable;
@end
```

These go in `.h` files, and define the structure of objects (like C structs).
import "NSParentClass.h"

@interface Classname : NSParentClass {
    @public NSURL *blorg;
    @private NSString *gurgle;
}

@property (readonly) NSURL *blorg;
@property (copy) NSString *gurgle;

This is the “2.0” way to declare interfaces.
Infix and dot notation

@implementation Classname
@synthesize blorg;  // generates set/get methods
@synthesize gurgle;

Instance *myInstance = [[Instance alloc] init];

[myInstance setGurgle:@"eep"];  // infix notation
myInstance.gurgle = @"eep";  // dot notation

This is the “implementation”, stored in .m files. @synthesize creates getter/setter methods for properties.
Objective-C Not subclassing

Categories

- Simple method for adding functionality to classes without subclassing
- Just define a new `@interface` and `implementation` with new methods

```objective-c
@interface NSURL (CategoryName)
- (BOOL) isPurple;
{
    if ([self isColor:@"purple"])
        return YES;
    else
        return NO;
}
@end
```
Memory Management
Retain/Release

- No garbage collection in iOS
- Must track with “retain” and “release” methods

```objective-c
Classname *myClass = [[Classname alloc] init]; // Retain count: 1
// Can be shortened to
// [Classname new];

[myClass release];
```
Testing Setup

XCode

```
@implementation ReachabilityAppDelegate

-(void) configureTextField: (UITextField*) textField imageView: (UIImage*) imageView
{
    NetworkStatus netStatus = [curReach currentReachabilityStatus];
    BOOL connectionRequired= [curReach connectionRequired];
    NSString* statusString= @"";
    switch (netStatus) {
    case NotReachable:
        {
            statusString = @"Access Not Available";
            imageView.image = [UIImage imageNamed: @"stop-32.png"] ;
            //Minor interface detail- connectionRequired may return yes, connectionRequired= NO;
            break;
        }
    
```
Testing Setup

Intercepting secure communications

- Standard proxy intercept won’t work
- Cert errors are a hard failure
- Options:
  - Change source to use HTTP
  - Use device + cert for proxy
  - Use simulator with → proxy → real site
Stunnel config

; SSL client mode
client = yes

; service-level configuration
[https]
accept = 127.0.0.1:80
connect = 10.10.1.50:443
TIMEOUTclose = 0
## Proxy Config

![Burp Suite Proxy Configuration](image)

### Proxy Listeners

<table>
<thead>
<tr>
<th>running</th>
<th>port</th>
<th>loopback only</th>
<th>support invisible</th>
<th>redirect</th>
<th>cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>8080</td>
<td>✓</td>
<td>✓</td>
<td>127.0.0.1</td>
<td>per-host</td>
</tr>
</tbody>
</table>

To add a new listener, complete the relevant details and click "add".

- **local listener port**: 8080
- **listen on loopback interface only**: ✓
- **support invisible proxying for non-proxy-aware clients**: ✓

**redirect to host**: 127.0.0.1
**redirect to port**: 80
The Sandbox Mechanism

Seatbelt

- aka “Seatbelt”
- Based upon TrustedBSD MAC framework
- Unlike Android’s UID-based segregation, apps run as one user
- Seatbelt policies provide needed segregation. Probably.
- Sandbox policies now compiled and rolled into the kernel
- On jailbroken devices, sandbox no longer applies
The Sandbox Mechanism

Jailbreaking

- On jailbroken devices, sandbox no longer applies
- However, devs for sideloaded apps can voluntarily hop into one

Documented profiles for OSX:

- kSBXProfileNoNetwork (= "nonet")
- kSBXProfileNoInternet (= "nointernet")
- kSBXProfilePureComputation (= "pure-computation")
- kSBXProfileNoWriteExceptTemporary (= "write-tmp-only")
- kSBXProfileNoWrite (= "nowrite")

¹http://iphonedevwiki.net/index.php/Seatbelt
The Sandbox Mechanism

Jailbreak Detection

- No more official Apple jailbreak detection API
- If you must determine whether a device is jailbroken, some possible checks:
  - /bin/bash
  - /bin/ssh
  - /private/var/lib/apt
- But discriminating against jailbroken devices is not necessarily a great idea
- And Apple app review may flag it
Binary Analysis

- Useful for black box testing or self-testing
- Disassembly of Mach-O binary format quite clean
- Several useful tools: otool, otx, class-dump
- Use for reversing other applications, or finding what info would be available to a third party
- Obfuscation is generally pretty futile, but especially in ObjC
- Encrypted binaries easily dumped\(^2\)

\(^2\)http://www.246tnt.com/iPhone/
**Testing Setup**

**Binary Analysis**

`otool`

```bash
otool -toV /Applications/iCal.app/Contents/MacOS/iCal/Applications/iCal.app/
    Contents/MacOS/iCal
Objective-C segment
Module 0x22b52c
   ...
   Class Definitions
defs[0] 0x00204360
       isa 0x0020a560
       super_class 0x001a5f44 CALCanvasItem
       name 0x001c6574 CALCanvasAttributedString
       ...
   ivars 0x00224300
       ivar_count 13
       ivar_name 0x001a54e2 _text
       ivar_type 0x001a53d0 "NSMutableAttributedString"
       ivar_offset 0x0000012c
       ivar_name 0x001a54e8
```

David Thiel (iSEC Partners)
Binary Analysis

http://otx.osxninja.com/

-(BOOL)[NSString(NSStringExtras) isFeedURLString]

+0 00003488 55
+1 00003489 89e5
+3 0000348b 53
+4 0000348c 83ec14
+7 0000348f 8b5d08
+10 00003492 c744240844430700

feed:
+18 0000349a a180a00700

_web_hasCaseInsensitivePrefix:
+23 0000349f 89442404
+27 000034a3 891c24
+30 000034a6 e850420800
Binary Analysis

class-dump

http://iphone.freecoder.org/classdump_en.html (or via Cydia)

class-dump-x /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator3.0.sdk/Applications/MobileSafari.app

    < snip >

    @protocol CALCanvasTextProtocol
    - (id)attributes;
    - (id)foregroundColor;
    - (float)fontSize;
    @end

    @protocol CALDetachmentDelegate
    - (int) decideDetachmentFor:(id)fp8 withOccurrence:(id)fp12;
    @end
Static Analysis
XCode & Clang

- Clang analyzer merged into XCode
- "Build & Analyze" option
- Identifies memory leakage, use-after-free, etc.
- Note: in some recent XCode versions, Analyzer results only show for device SDK builds. Meh
Static Analysis

Output

```objective-c
NetworkStatus netStatus = [curReach currentReachabilityStatus];
BOOL connectionRequired = [curReach connectionRequired];
NSString* statusString = @"";
switch (netStatus) {
    case NotReachable:
        NSString *myString = [[NSString alloc] init];
        statusString = @"Access Not Available";
        imageView.image = [UIImage imageNamed: @"stop-32.png"];
        // Minor interface detail: connectionRequired may return yes, even when the host is unreachable.
        connectionRequired = NO;
        [myString release];
        [myString release];
        break;
    case ReachableViaWWAN:
        statusString = @"Reachable WWAN";
        imageView.image = [UIImage imageNamed: @"WWAN5.png"];
        break;
    case ReachableViaWiFi:
        statusString = @"Reachable Wifi";
        imageView.image = [UIImage imageNamed: @"Airport.png"];
        break;
}
```
Keyboard Caching

- Keyboard cache used for form autocompletion
- /root/Library/Keyboard/dynamic-text.dat
- Already disabled for password fields
- Should be disabled for any potentially sensitive fields
- Set `UITextField` property `autocorrectionType = UITextAutocorrectionNo`
Networking

TLS and NSURL Handling

- Standard method for working with URLs
- SSL/TLS handled properly! Bypassing failed verification not allowed by default.
- So, of course, people turn it off
Networking

TLS and NSURL Handling

- Check for `NSURLSession` verification bypass via `setAllowsAnyHTTSPSCertificate`
- SSL verification bypass via `NSURLSession` delegation
  - Search for `continueWithoutCredentialForAuthenticationChallenge`\(^3\)
- Extra bonus stupid: Define category method to slip by Apple’s private API checks\(^4\)

\(^3\)[http://stackoverflow.com/questions/933331/](http://stackoverflow.com/questions/933331/)

Networking

NSS streams

- Good for non-HTTP traffic or going slightly lower-level

```objective-c
// First we define the host to be contacted
NSHost *myhost = [NSHost hostWithName: @"www.conglomco.com"];

// Then we create
[NSStream getStreamsToHost:myhost
    port:443
    inputStream:&MyInputStream
    outputStream:&MyOutputStream];

[MyInputStream setProperty:NSStreamSocketSecurityLevelTLSv1
    forKey:NSStreamSocketSecurityLevelKey];
```
Networking

CFStreams

- Slightly lower-level still
- Security defined by `kCFStreamPropertySSLSettings`
- Has sad set of constants 😞

```c
CFStringRef kCFStreamSSLLevel;
CFStringRef kCFStreamSSLLAllowsExpiredCertificates;
CFStringRef kCFStreamSSLLAllowsExpiredRoots;
CFStringRef kCFStreamSSLLAllowsAnyRoot;
CFStringRef kCFStreamSSLLValidatesCertificateChain;
 CFStringRef kCFStreamSSLPeerName;
```
A few ways data is stored (and potentially exposed):

- SQLite
- Core Data
  - Internally, SQLite
- Cookie management
- Caches
- plists
Anatomy of an App

- ~/Library/Application Support/iPhone Simulator/Applications/(appID)
- ./Documents → properties, logs
- ./Library/Caches → cachey things
- ./Library/Caches/Snapshots → screenshots of your app
- ./Library/Cookies → cookie plists
- ./Library/Preferences → various preference plists
- ./Library/WebKit → WebKit local storage
- ./Appname.app → app resources: binary, graphics, nibs, Info.plist
- ./tmp → tmp
Cookies

- Manipulated by the URL loading system
- Can alter `cookieAcceptPolicy` to:
  - `NSHTTPCookieAcceptPolicyNever`
  - `NSHTTPCookieAcceptPolicyOnlyFromMainDocumentDomain`
- Note that this may affect other running applications
  - In OS X, cookies and cookie policy are shared among apps
  - In iOS, only cookie policy is shared
SQLite and SQL injection

Dynamic SQL

```objective-c
NSString *uid = [myHTTPConnection getUID];
NSString *statement = [NSString stringWithFormat:@"SELECT username FROM users
where uid = '%@',uid];
const char *sql = [statement UTF8String];
```
const char *sql = "SELECT username FROM users where uid = ?";
sqlite3_prepare_v2(db, sql, -1, &selectUid, NULL);
sqlite3_bind_int(selectUid, 1, uid);
int status = sqlite3_step(selectUid);
Caching

- HTTP & HTTPS requests cached by default
- Can be prevented by `NSURLSession` delegate

```objective-c
-(NSCachedURLResponse *)connection:(NSURLSession *)connection
    willCacheResponse:(NSCachedURLResponse *)cachedResponse
{
    NSCachedURLResponse *newCachedResponse = cachedResponse;
    if ([[[[cachedResponse response] URL] scheme] isEqual:@"https")
    {
        newCachedResponse = nil;
    }
    return newCachedResponse;
}
```
Geolocation

Best Practices

- Use least degree of accuracy necessary
- Check for graceful handling of `locationServicesEnabled` and `authorizationStatus` method responses
- If you don’t want to handle subpoenas from divorce lawyers:
  - Don’t log locally
  - Anonymize server-side data
  - Prune logs
Geolocation

Accuracy Settings

Several accuracy constants:

```objective-c
CLLocationAccuracy kCLLocationAccuracyBestForNavigation;
CLLocationAccuracy kCLLocationAccuracyBest;
CLLocationAccuracy kCLLocationAccuracyNearestTenMeters;
CLLocationAccuracy kCLLocationAccuracyHundredMeters;
CLLocationAccuracy kCLLocationAccuracyKilometer;
CLLocationAccuracy kCLLocationAccuracyThreeKilometers;
```
The Keychain is where secret stuff goes

- Argh! Do not store this data in `NSUserDefaults`!

- Encrypted with device-specific key
  - Apps “can’t read”, not included in backups

- Simpler API than OS X: `SecItemAdd`, `SecItemUpdate`, `SecItemCopyMatching`

- Not available in simulator for pre-4.0

[Image: cause it’s got keys in it, see]
The Keychain

Key protection

- Pass an appropriate \texttt{kSecAttrAccessible} value to \texttt{SecItemAdd}:

```c
CFTypeRef kSecAttrAccessibleWhenUnlocked;
CFTypeRef kSecAttrAccessibleAfterFirstUnlock;
CFTypeRef kSecAttrAccessibleAlways;
CFTypeRef kSecAttrAccessibleWhenUnlockedThisDeviceOnly;
CFTypeRef kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly;
CFTypeRef kSecAttrAccessibleAlwaysThisDeviceOnly;
```
The Keychain

Shared keychains

- For using the same keychain among different apps\(^5\)
- Used by setting `kSecAttrAccessGroup` on init
- Apps must have same keychain-access-groups
- Apps can only have one access group
- On jailbroken phone...all bets off

\(^5\) [http://useyourloaf.com/blog/2010/4/3/keychain-group-access.html]
The Keychain

Certificates

- On device, can be installed via e-mail, Safari or iTunes sync
- On older simulators, no such luck
- Certs still verified, but no way to install new ones
  - Since they’re stored in the Keychain
- Stubs necessary for detecting simulator vs. device
Data Protection
Improving file and keychain protection

- By default, data encrypted with “hardware” key
- In iOS 4, “hardware” key can supplemented with PIN
- Developers can also mark files as “protected”
- Files encrypted, unreadable while device is locked
Data Protection

Usage

- 2 methods for enabling
  - Pass `NSDataWritingFileProtectionComplete` to `writeToFile` method of `NSData` object
  - Set `NSFileProtectionKey` to `NSFileProtectionComplete` on `NSFileManager` object
  - Again, data not accessible when device is locked
    - Check for data availability before use
    - Clean up when `UIApplicationProtectedDataWillBecomeUnavailable`

---

Entropy

How does it work?

- Using Cocoa, not /dev/random
- Gathered via `SecRandomCopyBytes`
  - Again, does not work in simulator
- Obviously, `rand()`, `random()`, `arc4random()` are all non-starters

```c
int result = SecRandomCopyBytes(kSecRandomDefault, sizeof(int), (uint8_t*) &randomResult);
```
Probably most security-relevant API in iOS 4.0

- Use `beginBackgroundTaskWithExpirationHandler` method to initiate background tasks
  - Needs matching `endBackgroundTask` method

- Remaining task time stored in `backgroundTimeRemaining` property
Backgrounding

Concerns

- Note: app is snapshotted upon backgrounding
- Prior to this, application should remove any sensitive data from view
  - Use splash screen or set hidden or alpha properties of UIWindow
Detect state transitions

Key state transition methods:

application:didFinishLaunchingWithOptions:
applicationDidBecomeActive:
applicationWillResignActive:
applicationDidEnterBackground:
applicationWillEnterForeground:
applicationWillTerminate:
Apps can register their own URL handlers — added by editing the plist, usually from XCode.

Called just like any URL, with multiple parameters, e.g.

```objective-c
openURL:[NSURL URLWithString:@"myapp://?foo=urb&blerg=gah"];
```

Can be called by app or web page

- Without user confirmation...

- Params accessible to receiving app via a delegate
Deprecation of delegation method:

```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url
```

New method:

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
  sourceApplication:(NSString *)sourceApplication
  annotation:(id)annotation
```

- Allows for determining calling application, receives data in plist form
- Obviously, sanitization is key here, especially given...
IPC

URL handler conflicts

- What happens if two apps use the same handler?
  - If an Apple app uses it: Apple app launches
  - Third-party apps: “Undefined”

“If your URL type includes a scheme that is identical to one defined by Apple, the Apple-provided application that handles a URL with that scheme (for example, “mailto”) is launched instead of your application. If a URL type registered by your application includes a scheme that conflicts with a scheme registered by another third-party application, the application that launches for a URL with that scheme is undefined.”

- May go to the last claiming app...ew.
- Hence: be wary of passing private data in app URLs
IPC

Push Notifications

- Registering for notifications:

```objectivec
[[UIApplication sharedApplication] registerForRemoteNotificationTypes: 
(UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound)];
```

- Receiving notifications:

```objectivec
-(void)application:(UIApplication *)application 
didReceiveRemoteNotification:(NSDictionary *)userInfo

-(BOOL)application:(UIApplication *)application 
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
```

- Check for validation of userInfo and launchOptions
Copy/Paste

Pasteboards

- Obligatory dig at Apple re: copy/paste debacle
- 2 system UIPasteboard access methods:
  - UIPasteboardNameGeneral & UIPasteboardNameFind
- Pasteboards marked “persistent” will be kept in local storage
Copy/Paste

Pasteboards

- Also “private” application pasteboards, which (in true Objective-C form) are not in any way “private”
- Occasionally used as IPC hack
  - Migrating data from free → paid app
  - I saw one suggestion to transfer private keys with the pasteboard 😐

- Bottom line: avoid sensitive data here & clean up after yourself
  - Clear pasteboard on `applicationWillTerminate`
  - `pasteBoard.items = nil`
Copy/Paste

Example Abuse

How not to pasteboard: Twitter OAuth library

- (void) pasteboardChanged: (NSNotification *) note {
  UIPasteboard *pb = [UIPasteboard generalPasteboard];

  if ([note.userInfo objectForKey:UIPasteboardChangedTypesAddedKey] == nil)
    return;
  NSString *copied = pb.string;

  if (copied.length != 7 || !copied.oauthtwitter_isNumeric) return;
  [self gotPin:copied];
}

7 3rd-party library, not by Twitter
Copy/Paste
Disabling it

Possible mitigation: For fields with sensitive data, disable copy/paste menu

```swift
-(BOOL)canPerformAction:(SEL)action withSender:(id)sender {
    UIMenuController *menuController = [UIMenuController sharedMenuController];
    if (menuController) {
        [UIMenuController sharedMenuController].menuVisible = NO;
    }
    return NO;
}
```

- Can also disable menu items individually

---

8http://stackoverflow.com/questions/1426731/
UDIDs

Use and Abuse

- Unique identifier derived from hardware information
- Often abused as a user tracking mechanism
- Occasionally abused as an authenticator
  - See: Tapulous
- Contrary to popular belief, this is mutable

---

UDIDs

UDID Faker available on Cydia
UDIDs

Don’t use them.

Summary:

- Don’t rely on UDID for anything ever
- Don’t use it for tracking, it gets you bad press
- If you really need to track users, use hash of UDID + salt
- Check code for use of [[UIDevice currentDevice] uniqueIdentifier]
Classic C Attacks

Nothing new here

- Still has the same classic issues
- Buffer overflows
- Integer issues, especially with `malloc()`
  - Why are you malloc’ing, grandpa? We are in the future here
  - Sanitize int calculations with `checkint(3)`
- Double-frees
- Format strings
Exploitable! Under some circumstances.\endnote{10}

Procedure:

- Release object
- Release some other object
- Allocate space of same size as first object
- Write your code to the new buffer
- ...
- Send message or release to original object

\endnote{10} http://felinemenace.org/~nemo/slides/eusecwest-STOP-objc-runtime-nmo.pdf
iOS & Format Strings

- `withFormat/appendingFormat` family
- `%x` works — `%n` does not 😞
- `%n` does still work with regular C code…
Format Strings
Format string confusion

- Found on pentest:

```objc
NSString myStuff = @"Here is my stuff.";
myStuff = [myStuff stringByAppendingFormat:[UtilityClass formatStuff:unformattedStuff.text]];
```

- Bzzt. NSString objects aren’t magically safe.

```objc
NSString myStuff = @"Here is my stuff.";
myStuff = [myStuff stringByAppendingFormat:@"%@", [UtilityClass formatStuff:unformattedStuff.text]];
```
Format Strings
Likely culprits

- [NSString *WithFormat]
- [NSString stringByAppendingFormat]
- [NSMutableString appendFormat]
- [NSAlert alertWithMessageText]
- [NSEException]
- [NSLog]
Secure coding checklist

Or penetration tester’s hit list

- HTTPS used and correctly configured (i.e. not bypassed by delegation or setAllowsAnyHTTPSCertificate)
- All format strings properly declared
- General C issues (malloc(), str*, etc.)
  - Any third-party C/C++ code is suspect
- Entropy gathered correctly
- Secure backgrounding
Secure coding checklist

Continued

- UIPasteBoards not leaking sensitive data
- Correct object deallocation, no use-after-release
- URL handler parameters sanitized
- Secure keychain usage
- No inappropriate data stored on local filesystem
- CFStream, NSStream, NSURL inputs sanitized/encoded
- No direct use of UDID
QUESTIONS?
HTTPS://WWW.ISECPARTNERS.COM
For Further Reading I

H. Dwivedi, C. Clark, D. Thiel
*Mobile Application Security.*
McGraw Hill, 2010

Neil Archibald
STOP!!! Objective-C Run-TIME.

Apple, Inc.
iOS Application Programming Guide
Other resources

http://culater.net/wiki/moin.cgi/CocoaReverseEngineering
http://www.musicalgeometry.com/archives/872
iPhone-Applications-Privacy-Issues.pdf