Blackbox iOS App Testing Using idb

Daniel A. Mayer

@DanlAMayer
http://www.idbtool.com
Who we are…

**Me: Daniel Mayer**

Senior Security Consultant with NCC Group in Chicago

Formerly Matasano Security

Ph.D. in Computer Science (Security and Privacy)

**NCC Group**

UK Headquarters, Worldwide Offices

Application Security Consultancy

Software Escrow, Testing, Domain Services
Mobile Dominates Internet Use...
... and Apps Dominate Mobile

Apps Continue to Dominate the Mobile Web

<table>
<thead>
<tr>
<th>Year</th>
<th>Mobile Web</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>20%</td>
<td>80%</td>
</tr>
<tr>
<td>2014</td>
<td>14%</td>
<td>86%</td>
</tr>
</tbody>
</table>

Source: Flurry Analytics
Anyone Lost or Got Their Phone Stolen?
Well, you are not alone...

**Physical cell phone privacy experiences by age group**

% of cell owners in each age group who have experienced the following

![Bar chart showing privacy experiences by age group](chart.png)

**Source:** Pew Research Center’s Internet & American Life Project, March 15-April 3, 2012 Tracking survey.

N=2,254 adults ages 18 and older, including 903 interviews conducted on respondent’s cell phone. Margin of error is +/-2.6 percentage points based on cell phone owners (n=1,954).
1. Introduction

2. (Reasonably) New Tool: idb

3. Common iOS Vulnerabilities
   1. Binary
   2. Local Storage
   3. Information Disclosure
   4. Inter-Process Communication
   5. Network Communication

4. Conclusion
Introduction
iOS Platform Security

Apps are sandboxed (‘seatbelt’)
  All apps share same UNIX user ‘mobile’

App code has to be signed
  Bypassed when jailbroken

Raising the bar
  Data Execution Prevention (DEP)
  Address Space Layout Randomization (ASLR)

Passcode / TouchID
iOS Apps

Native applications
  Objective-C(++), superset of C(++)
  Cocoa touch for GUI

Web view applications
  Display mobile websites in a UIWebView
iOS App Attack Surface

Vulnerabilities typical arise at trust boundaries

Daniel A. Mayer - Blackbox iOS App Testing Using idb
Pentest Setup

Jail-broken iDevice
SSH access!
  Full UNIX-like environment
  Full file system access

Unauthorized modification of iOS can cause security vulnerabilities, instability, shortened battery life, and other issues

Mobile (Cydia) Substrate
Patch system functions at runtime
http://www.cydiasubstrate.com/

Intercepting Proxy
Monitor app communication
Introducing idb
Existing Tool Landscape

Many great tools [1]
  Scattered
  Static and dynamic

Fully understand app’s behavior in assessment

My background is in dynamic testing
  No “click and done” solution
  Tool that automates analyses

Introducing idb

Ruby and Qt (4,500+ loc)
Demo: Pentesting Setup

Connecting to device
- SSH directly
- SSH via USB

Port forwarding
- Remote
- Local
Common iOS Vulnerabilities
OWASP Mobile Top 10 - 2014

1. Weak Server Side Controls
2. Insecure Data Storage
3. Insufficient Transport Layer Security
4. Unintended Data Leakage
5. Poor Authentication and Authorization
6. Broken Cryptography
7. Client Side Injection
8. Security Decision via Untrusted Input
9. Improper Session Handling
10. Lack of Binary Protections
1. Weak Server Side Controls

2. Insecure Data Storage

3. Insufficient Transport Layer Security

4. Unintended Data Leakage

5. Poor Authentication and Authorization

6. Broken Cryptography

7. Client Side Injection

8. Security Decision via Untrusted Input

9. Improper Session Handling

10. Lack of Binary Protections

Daniel A. Mayer - Blackbox iOS App Testing Using idb
The App Binary

Native Code!

Buffer overflows
Format string flaws

WithFormat - don’t let user specify the format! [1]
User after frees

Used as storage space

API keys
Credentials
Crypto Keys


Amazon is decompiling our apps in security gaffe hunt, says dev
Putting secret AWS keys in software is a big no-no
Exploit Mitigation

Take advantage of OS protections

Compile as Position Independent Executable (PIE)

Enable stack canaries

Use Automatic Reference Counting

Do not store credentials in the binary
Demo: Poor-Man’s Reversing

Basic binary information using otool

```
RunKeeper.app git:(gui) x otool -Vh RunKeeper
  flags
  NOUNDEFS DYLDLINK TWOLEVEL WEAK_DEFINES BINDS_TO_WEAK PIE
- RunKeeper.app git:(gui) x otool -I -v RunKeeperlegrep 'stack_chk_(fail|guard)'
  0x003d3dc8  748 ___stack_chk_fail
  0x004e0044  749 ___stack_chk_guard
  0x004e22e4  748 ___stack_chk_fail
- LumosityiPad.app git:(gui) x otool -I -v LumosityiPadgrep _objc_release
  0x006f7aa8  9934 _objc_release
  0x0085af0  9934 _objc_release
```

Strings

Weak Class Dump

https://github.com/limneos/weak_classdump

Uses cycript (http://www.cycript.org/)
Local Storage

Apps are sandboxed to
/private/var/mobile/Applications/[guid]/

Sandbox accessible to app
Stored in backups

If stolen
Jailbreak
File system access
File System Encryption

All files encrypted

One key per File

Passcode!

Attacks
  PIN cracking
  Backups
  Jail-break not enough!

Device UID

User Passcode

File System Key

Protection Class Key

File Meta Data

File Key

File Data
Using the Data Protection API

Enforce a strong passcode
Set a NSFileProtection when storing files

<table>
<thead>
<tr>
<th>NSFileProtection</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>Protected when device is locked</td>
</tr>
<tr>
<td>CompleteUnlessOpen</td>
<td>If open, file can be read when locked</td>
</tr>
<tr>
<td>CompleteUntilFirstUserAuthentication (iOS 8)</td>
<td>Protected from boot until user unlocks</td>
</tr>
<tr>
<td>None (iOS &lt; 8)</td>
<td>No protection</td>
</tr>
</tbody>
</table>

Example

```
[[[NSFileManager defaultManager] createFileAtPath:@"filename"
    contents:[@"super_secret" dataUsingEncoding:NSUTF8StringEncoding] 
    attributes:[NSDictionary dictionaryWithObject:NSFileProtectionComplete forKey:NSFileProtectionKey]]];
```
Don’t do your own crypto

Existing frameworks make it hard to get crypto right!

Look into libsodium-ios

General problem on mobile

Where does the key come from?
Have to use some Key Derivation Function (KDF)

Shameless plug

Do the Matasano crypto challenges!

http://cryptopals.com/
SQLite

A small relational database API

Popular to persist data

Data stored unencrypted in a file
SQLite Mitigation

Use Data Protection to encrypt sqlite file

Third-Party solutions
e.g., http://sqlcipher.net/

Journal may leak deleted data
Use VACUUM to rebuild DB
Property List Files

Structured storage (NSUserDefaults)
Stored unencrypted in XML files or binary plist

plutil -convert xml1

Often used for crypto keys, credentials, etc.
Property List Files: Mitigation

Don’t use for sensitive data!

File storage for binary data

NSProtectionComplete!

Use keychain for structured data

http://software-security.sans.org/blog/2011/01/05/using-keychain-to-store-passwords-ios-iphone-ipad/
Keychain

Key-Value store

/private/var/Keychains/keychain-2.db
Encryption similar to Data Protection

<table>
<thead>
<tr>
<th>Protection Class</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>kSecAttrAccessibleWhenUnlocked</td>
<td>Protected when device is locked.</td>
</tr>
<tr>
<td>kSecAttrAccessibleAfterFirstUnlock</td>
<td>Protected from boot until user unlocks.</td>
</tr>
<tr>
<td>kSecAttrAccessibleAlways</td>
<td>No protection.</td>
</tr>
<tr>
<td>kSecAttrAccessibleWhenPasscodeSet</td>
<td>Only store if passcode is set.</td>
</tr>
</tbody>
</table>

ThisDeviceOnly variants: no migration
Share Data Securely Between Apps

Keychain Access Group

app_id = [bundle_seed] || [bundle_id]

BEEF1337 || com.corp.myapp
[bundle_seed] generated by Apple.
Apps with same [bundle_seed] can share access.

kSecAttrAccessGroup

Access through search dictionary

[searchDictionary
setObject:@"BEEF1337.com.app.family"
forKey:(id)kSecAttrAccessGroup];
Demo: idb Local Storage Functions

Use SSH connection to analyze sandbox

Determine FileProtection using NSFileManager

https://github.com/dmayer/protectionclassviewer

```objective-c
NSString *fileProtectionValue = [[[NSFileManager defaultManager] attributesOfItemAtPath:@"filename" error:NULL] valueForKey:NSFileProtectionKey];
```

Keychain viewer using keychain_dump

https://code.google.com/p/iphone-dataprotectionn
Use Crypto and done, right?

http://xkcd.com/538/
Example: Remote File Read

```javascript
var xhttp = new XMLHttpRequest();
xhttp.open("GET","file:///var/mobile/Applications/[..]/file.pdf",false);
xhttp.send();
alert(xhttp.responseText);
// Don't use alert unless you want entire PDF in alert box :)
```
Information Disclosure: Screenshot

iOS takes screenshot when app backgrounds

 Stored unencrypted at

/var/mobile/Applications/
 [guid]/Library/Caches/
 Snapshots/[bundle_id]/

./Main subfolder
Mitigation: Screenshot

Hide sensitive information from screen

- Implement `applicationDidEnterBackground`
- Popular: Place launch image in foreground

```swift
User switches to a different app

Foreground
- Deactivate this app

Background
- Enter background

Your code
- `applicationWillResignActive:`
- `applicationDidEnterBackground:`

ignoreSnapshotOnNextApplicationLaunch

Does NOT prevent screenshot from being taken
Data Leakage: Cache.db

iOS caches requests and responses

```sql
sqlite> select * from cfurl_cache_response where request_key like '%password%';
801259328918101https://api._.v3/register/email/?client_id=1431599&timestamp=1389571536&oauth_signature=b34f5da43330c58cc7cc59b0e33662c9ba1c35db27ee9dd223a0d5ff2b2f9fifemail=xxx%40ccc.com&first_name=A&gender=maile&last_name=B&locale=en_US&password=testtest3username=2014-01-13 00:05:37
```

Disable caching

Send no store headers from server

```swift
- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
  willCacheResponse:(NSCachedURLResponse *)cachedResponse
  {
    return nil;
  }
```
40 % of 40 tested banking apps disclose data [1]
Log files accessible by other apps

```
<loginWithRole id="0" c:root="1" xmlns:n0="http://mobile.services.xxxxxxxxxxx.com/">
  <n0:loginWithRole>
    <n0:loginWithRole>
      <n0:loginWithRole>
        <n0:loginWithRole>
          USER-ID
        </n0:loginWithRole>
        XRS
        PASSWORD
        xxxxxxxx
      </n0:loginWithRole>
    </n0:loginWithRole>
  </n0:loginWithRole>
</loginWithRole>
```

Wrap your NSLog statements, e.g.

```
#ifndef DEBUG
  NSLog(@"password");
#endif
```

Demo: idb Information Disclosure

Screenshot Tool
Walks through steps that create screenshot.
Displays screenshot in idb.

iOS console available in
Xcode or iPhone Configuration Utility.

idb uses idevicesyslog [1]

Inter-Process Communication

There is no proper IPC

Poor-man’s IPC
  UIPasteboard

Custom URL schemes
  Apple’s approved solution

Consider using the keychain with access group
Pasteboard

Any app can read it

Private Pasteboards are not private
There seems to be no API to find all Pasteboards

```objectivec
[UIPasteboard generalPasteboard];
[UIPasteboard pasteboardWithName:@"super_secret" create:NO ];
```

Don’t use the Pasteboard for IPC

To prevent Copy/Paste, subclass UITextView
```objectivec
canPerformAction should return “NO” for copy
```
URL Schemes

Register in Info.plist
Handle in:

```c
- (BOOL) application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation: (id)annotation
{ // Handle request }
```

Security Considerations

Malicious input
Trust
Hijacking

Note: If more than one third-party app registers to handle the same URL scheme, there is currently no process for determining which app will be given that scheme.
URL Schemes

Exploiting Trust

my_app://configure?server=..&port=..

Inject attacker controlled server.

bank://redirect?page=http%3A%2F%2Fphish.me

Phishing —> Credentials.

Verify the caller of the URL handler

sourceApplication parameter

Perform strict input validation
Demo: idb IPC Functions

Pasteboard monitor

- Runs binary on device which pulls content
- Supports custom pasteboards
- https://github.com/dmayer/pbwatcher

URL Schemes

- List
- Invoke
- Basic fuzzer
Network Communication

Communication with Network Services
  HTTP/S
  Socket connections
  Push Notifications

Challenge similar to browsers
  Protect data in transit

Typically done through SSL/TLS
iOS Certificate Validation

Default: Accept if signed by CA in trust store
Check when using 3rd party libs

iOS offers great flexibility in cert. validation
the good: can make cert. validation stronger
the bad: cert. check often overridden in dev
the ugly: easy to accept any cert

- (void)connection:(NSURLConnection *)connection
  willSendRequestForAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge {
    NSURLProtectionSpace * pSpace = [challenge protectionSpace];
    NSURLCredential * cred = [NSURLCredential credentialForTrust:[pSpace serverTrust]];
    [[challenge sender] useCredential:cred forAuthenticationChallenge:challenge];
  }
Certificate Validation

Don’t bypass certificate validation

In dev, use free certificates (e.g. startssl.com)
Install server cert explicitly on device.

Implement certificate pinning!

https://github.com/iSECPartners/ssl-conservatory
https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning#iOS
iOS CA Cert Management

Simulator: [sim]/Library/Keychains/TrustStore.sqlite3
Fiddly: ASN.1 anyone?

Device: /private/var/Keychains/TrustStore.sqlite3
Adding entry not sufficient
Fell back to ‘MDM’-based install

Pentest Pinning bypass
https://github.com/iSECPartners/ios-ssl-kill-switch
Planned idb Features

Improvements

- Grep for the log view
- Search/upload for the FS Browser
- Copy data to Pasteboard
- Analysis of used privacy-invasive APIs

Integration of more awesome tools

- iOS SSL Kill Switch

Send me bug reports, feature / pull requests!
Thanks!

Questions?

http://www.idbtool.com

github.com/dmayer/idb

gem install idb

Email / XMPP: mayer@cysec.org

@DanIAMayer
Image Attributions

**iPhone icon, unchanged:**
By Adrian Dediu, https://www.iconfinder.com/iphone5cunlock
License: https://creativecommons.org/licenses/by/3.0/us/

**CA certificate icon, unchanged:**
By http://snipicons.com/
License: https://creativecommons.org/licenses/by-nc/3.0/

**Storage icon, unchanged:**
By Barrymieny, http://barrymieny.deviantart.com
License: https://creativecommons.org/licenses/by-nc-sa/3.0/

**Key, unchanged:**
Double-J designs, http://www.doublejdesign.co.uk/
License: https://creativecommons.org/licenses/by/3.0/us/

**Slide 21, cropped:**

**Slide 35, cropped:**
Europe
Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Munich
Amsterdam
Zurich

North America
Atlanta
Chicago
New York
San Francisco
Seattle
Austin

Australia
Sydney