Who we are...

- **Me:** Daniel A. Mayer
  - Appsec consultant with Matasano Security.
  - Ph.D. in Computer Science (Security and Privacy).
  - Twitter: @DanIAMayer
  - Website: http://cysec.org

- **Matasano Security**
  - Application Security Consultancy.
  - Offices in New York, Chicago, Mountain View.
  - We are hiring! :-)
  - Part of nccgroup

Daniel A. Mayer » idb - iOS Blackbox Pentesting
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

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).
Agenda

1. Introduction
2. 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
iOS Apps

1. Native applications
   - Objective-C(++)
   - Cocoa touch for GUI

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

- Vulnerabilities typical arise at trust boundaries

Visual representation of attack surface with components such as User Input, IPC, Network, Backend Service, iOS Interaction, Data Storage, and Physical Theft.
Pentest Setup

- Jail-broken iDevice
  - SSH access!
  - Full UNIX-like environment
  - Full file system access
- Mobile (Cydia) Substrate
  - Patch system functions at runtime
  - http://www.cydiasubstrate.com/

- Intercepting Proxy
  - Monitor app communication

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

---Apple
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)
- New tools
- Integrates existing tools

Goal:
- Easier setup and access
- Work in progress
Demo: Pentesting Setup

- Connecting to device
  - SSH directly
  - SSH via USB

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

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

# The OWASP Mobile Top 10

1. **Insecure Data Storage**
2. **Weak Server Side Controls**
3. **Insufficient Transport Layer Security**
4. **Client Side Injection**
5. **Poor Authentication and Authorization**
6. **Improper Session Handling**
7. **Security Decision via Untrusted Input**
8. **Side Channel Data Leakage**
9. **Broken Cryptography**
10. **Sensitive Information Disclosure**


Daniel A. Mayer » idb - iOS Blackbox Pentesting
The App Binary

- Native Code!
  - Buffer overflows
  - Format string flaws
    - WithoutFormat - don’t let user specify the format! [1]
  - User after frees

- Used as storage space:
  - API keys
  - Credentials
  - Crypto Keys

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 RunKeeper|egrep '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 LumosityiPad|grep _objc_release
0x006f7aa8 9934 _objc_release
0x0085aaf0 9934 _objc_release
```

- Strings

- Cycrypt
  - [http://www.cycrypt.org/](http://www.cycrypt.org/)
  - JS to Objective-C
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!

### Flowchart

```
Device UID ➔ File System Key ➔ File Metadata ➔ File Key ➔ File Data
Layer 1

PBKDF2 ➔ Protection Class Key ➔ File System Key ➔ File Metadata ➔ File Key ➔ File Data
Layer 2

User Passcode ➔ Protection Class Key ➔ File System Key ➔ File Metadata ➔ File Key ➔ File Data
Layer 3
```
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</td>
<td>Protected from boot until user unlocks.</td>
</tr>
<tr>
<td>None (Default!)</td>
<td>No protection.</td>
</tr>
</tbody>
</table>

Example:

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

- Getting crypto right is hard!

- 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!
  - Email: cryptopals@matasano.com
SQLite

- 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/

Daniel A. Mayer » idb - iOS Blackbox Pentesting
### Keychain

- Key-Value store
- `/private/var/Keychains/keychain-2.db`
- Encryption similar to Data Protection
  - Device + Passcode
  - `secItemAdd, secItemDelete, etc.`

<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 (default)</td>
<td>No protection.</td>
</tr>
</tbody>
</table>

- **ThisDeviceOnly variants:** no migration
Share Data Securely Between Your 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](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-dataprotection](https://code.google.com/p/iphone-dataprotection)
- Beta-level integration with snoop-it
  - [https://code.google.com/p/snoop-it/](https://code.google.com/p/snoop-it/)
Use Crypto and done, right?

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

- App locally caches documents (inc. HTML)

```javascript
var xhttp = new XMLHttpRequest();
xhttp.open("GET","file:///var/mobile/Applications/[guid]/../evil.html",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`

```
total 492
-rw-r--r-- 1 mobile mobile 500529 Jan 11 14:31 UIApplicationAutomaticSnapshotDefault-Portrait@2x.png
```
Mitigation: Screenshot

- Hide sensitive information from screen
- Implement `applicationDidEnterBackground`
- Popular: Place launch image in foreground

```objective-c
- ignoreSnapshotOnNextApplicationLaunch
  - Prevents screenshot from being taken
```

Daniel A. Mayer » idb - iOS Blackbox Pentesting
Data Leakage: Cache.db

- iOS caches requests and responses

```
sqlite> select * from cfurl_cache_response where request_key like '%password%';
![](https://api.s3.amazonaws.com/v3/register/email/?client_id=1431599&timestamp=1389571536&oauth_signature=b34f5da48330c58cc7cc59bc033662cbaf1c5d27ee9dd223a0d5ff2b2f29f7fe78Il=xxx%40ccc.com&first_name=A&gender=male&last_name=B&locale=en_US&password=testtest&username=12014-01-13 00:05:37
```

- Disable caching
  - Send no store headers from server

```swift
- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
  willCacheResponse:(NSCachedURLResponse *)cachedResponse
  {
    return nil;
  }
```
Information Disclosure: Log Files

- 40% of 40 tested banking apps disclose data [1]
- Log files accessible by other apps.

Wrap your NSLog statements, e.g.:

```c
#ifdef 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.

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

- Don’t use the Pasteboard for IPC.
- Delete content with `items = nil`.
- To prevent Copy/Paste, subclass `UITextView`.
  - `canPerformAction` should return “NO” for `copy`:
URL Schemes

- Register in Info.plist
- Handle in:

```swift
-(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
    - Display attacker controlled site in trusted app.
    - Intercept 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
An SSL Certificate Primer

CA

Intermediate 1

Intermediate 2

company.com

I vouch that this key belongs to...

I trust this!

Verify
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
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

My server’s cert was signed by:

I trust this!

I don’t trust this!

Verify
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](https://github.com/iSECPartners/ios-ssl-kill-switch)
Planned idb Features

- Hosts file editor

Improvements
- Grep for the log view
- Search for the FS Browser
- Robustness improvements

Integration of more awesome tools.
- class-dump-z, iOS SSL Kill Switch

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

Questions?

Email+XMPP: mayer@cysec.org
Twitter: @DanlAMayer
Github: https://github.com/dmayer/idb

THIS JUST IN!
Square + Matasano CTF
https://microcorruption.com

Thanks to
• Jeff Jarmoc, Mike Tracy, Andy Schmitz, David Goldsmith
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:
    ManagingYourApplicationsFlow.html

- Slide 35, cropped:
    FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html