

iOS Mobile Application Security (iMAS)

FY13 -14 MITRE Research

January 2014



Research Team: Gregg Ganley(PI) and Gavin Black

MITRE

About MITRE

The screenshot shows the official website for MITRE. At the top, there's a navigation bar with links for "ABOUT US", "OUR WORK", "EMPLOYMENT", and "NEWS & EVENTS". Below the navigation is a search bar and social media links for Facebook, Twitter, LinkedIn, YouTube, and RSS. The main banner features a globe and the text "Applying Systems Engineering and Advanced Technology to Critical National Problems." A sidebar on the left lists "MISSION AREAS" including Acquisition and Systems Analysis, Airspace Development, Safety, Security, Command and Control, Cybersecurity, Emerging and Disruptive Technologies, Global Networking, Health Transformation, Homeland Security, Intelligence, Surveillance, Reconnaissance, and Large-Scale Enterprise Transformation. The "WHAT IS AN FFRDC?" section shows four circular icons representing different sectors. The "LATEST NEWS" section highlights three articles: "By Design: Integrating Privacy into Information Systems" (October 2011), "EyesFirst Aims to Automatically Detect Chronic Diseases Through Retinal Scans" (October 2011), and "Developing a Healthy "Cyber Ecosystem"" (October 2011). Each news item includes a thumbnail image, the author's name, the date, and a brief summary.

- **Not for profit, working in the public interest**
- **MITRE does not manufacture products, enabling us to provide *conflict-free guidance***
- **MITRE operates several FFRDCs**
 - DoD, FAA, IRS, VA, DHS, U.S. Courts
- **MITRE invests in an independent R&D (IR&D) program to 'look ahead' for our sponsors**

iOS Mobile App Security (iMAS) Elevator Pitch



iMAS – iOS Application Defense

iMAS – iOS secure, open source application framework to reduce iOS application vulnerabilities and information loss

iOS Security Model

■ iOS four layers

- Device - prevents unauthorized access to device
- Data - protects data stored on the device

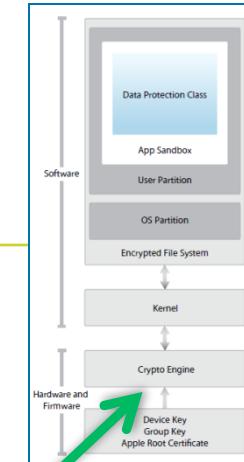
iOS 6 granted FIPS 140-2, approved for U.S. government use

Summary: Move over, BlackBerry. iPhones and iPads running iOS 6 are now certified for low-level secure government use.



By Zack Whittaker for Between the Lines | May 7, 2013 -- 17:30 GMT (10:30 PDT)

[Follow @zackwhittaker](#)



Isolates applications while they are

Limited flash support, reduced PDF/.MOV features, stripped

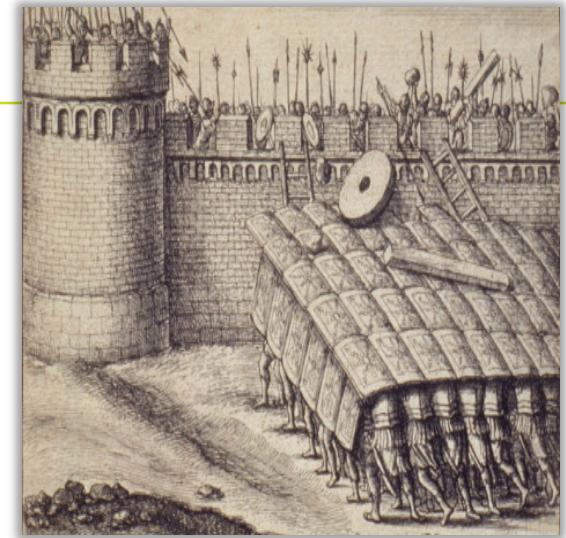
Crypto Officer Role Guide for FIPS 140-2 Compliance

iOS 7

FIPS compliance means custom enterprise apps, and the need for iMAS app security

Research Scope

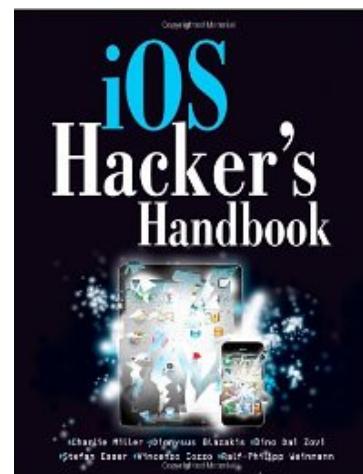
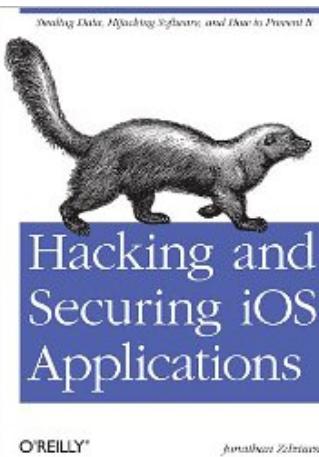
- **iOS attack surface is multi-faceted**
- **Threat vectors**
 - Lost/stolen device, physical access attacks
 - Malware inserted via
 - Browser JavaScript engine
 - Apps downloaded from *Enterprise App store*
- **Attacker purpose**
 - Steal application data, find enterprise credentials
 - Eavesdropping on video, microphone, and phone calls
 - C2/Recon into Enterprise networks



- **iMAS focus - *Application use and its data as the target***
 - Enterprise data, tactical operational data, patient health info
 - Protect against: malware and physical access attacks
 - Secure Static and Dynamic application controls

Hacking and Jailbreaking iOS

- Attacks and weaknesses are well documented:



- Recent Jailbreak: <http://evasi0n.com/>



Worst-Case Passcode
Guessing Time (iPhone4)

Passcode Length	Complexity	Time
4	Numeric	18 minutes
4	Alphanumeric	51 hours
6	Alphanumeric	8 years
8	Alphanumeric	13 thousand years
8	Alphanumeric, Complex	2 million years

Assuming 26 lowercase letters + 10 digits + 34 complex characters = 70 chars

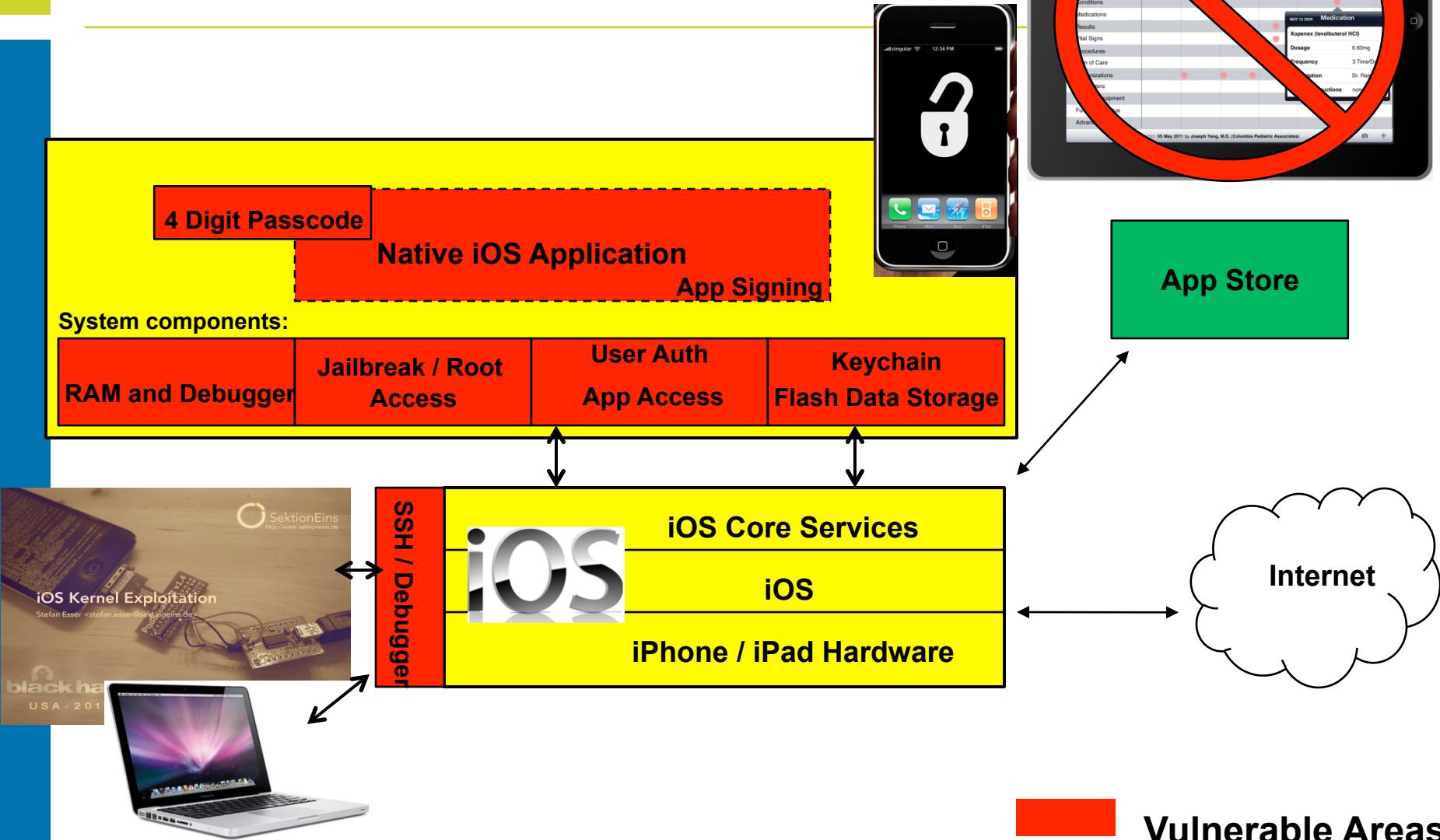


iPhone data protection in depth

Jean-Baptiste Bédruine
Jean Sigwald
Sogeti / ESEC
jean-baptiste.bedruine(at)sogeti.com
jean.sigwald(at)sogeti.com

```
[pad1:~ root# ./bruteforce
Writing results to 38e42cc50899d7d1.plist
keybag id=1
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
Found passcode : 0011
Keybag version : 3
Keybag keys : 10
Class      Wrap      Key
11          0         671f575ddf114bd6895b0d4d31c4b2
```

Problem: Standard iOS Application Today

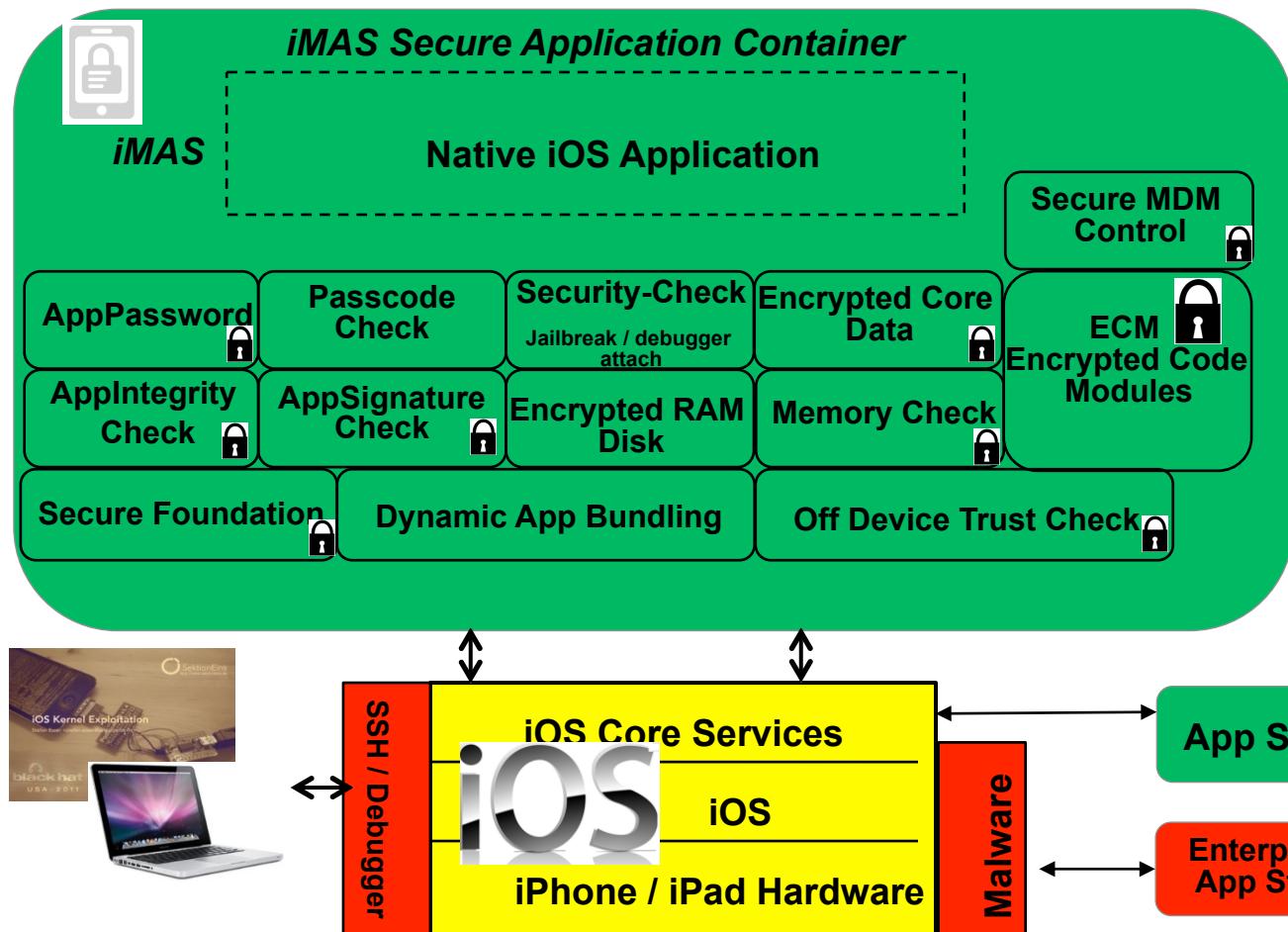


Research Idea: *iMAS Secure Application Framework*



Security Areas:

- App authentication
- Data at rest protection
- App at rest security
- Device Passcode check
- Jailbreak detection
- Debugger attach detect
- Encrypted SQLite
- Dynamic application security bundle
- Secure Keychain
- Memory scrub after use
- Dynamic memory usage check
- Remote App Wipe
- Lighting Connector Off Device Trust

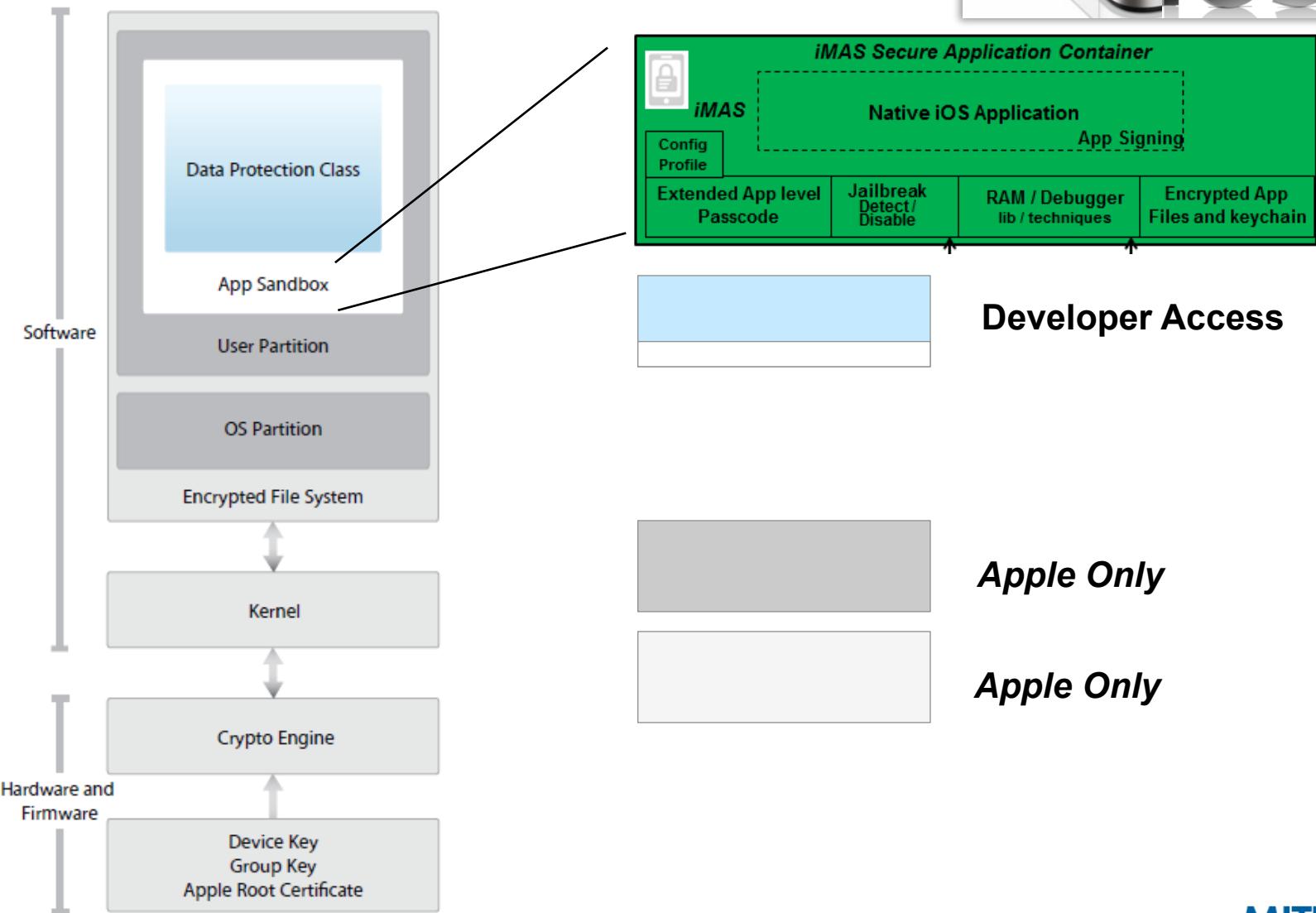


- Security Controls beyond Apple iOS
- Reduces iOS app attack surface
- Vetted, prioritized security control set
- Open source, grow community

█ Tolerable Security Risk



iOS Security Architecture



Mobile Threats

- 24 threats most relevant to the mobile environment

Software-Based Threats

- Malware
- Exploitation of Vulnerable App
- Exploitation of Vulnerable OS

Web-Based Threats

- Mobile Code
- Drive-By-Downloads
- Exploitation of Vulnerable Browser

Network-Based Threats

- Data/Voice Collection over the air
 - Bluetooth
 - WiFi
 - Cellular
- Data/Voice Collection over the network
- Manipulation of data-in-transit
- Connection to Masqueraded service
- Jamming
 - GPS
 - Mobile Carrier
 - WiFi
- Flooding

Physical-Based Threats

- Loss of Device
 - Data recovered by unauthorized party
 - Use of authorized Credentials
- Physical Tamper
- Supply Chain

Enterprise-Based Threats

- Access to Enterprise Resources

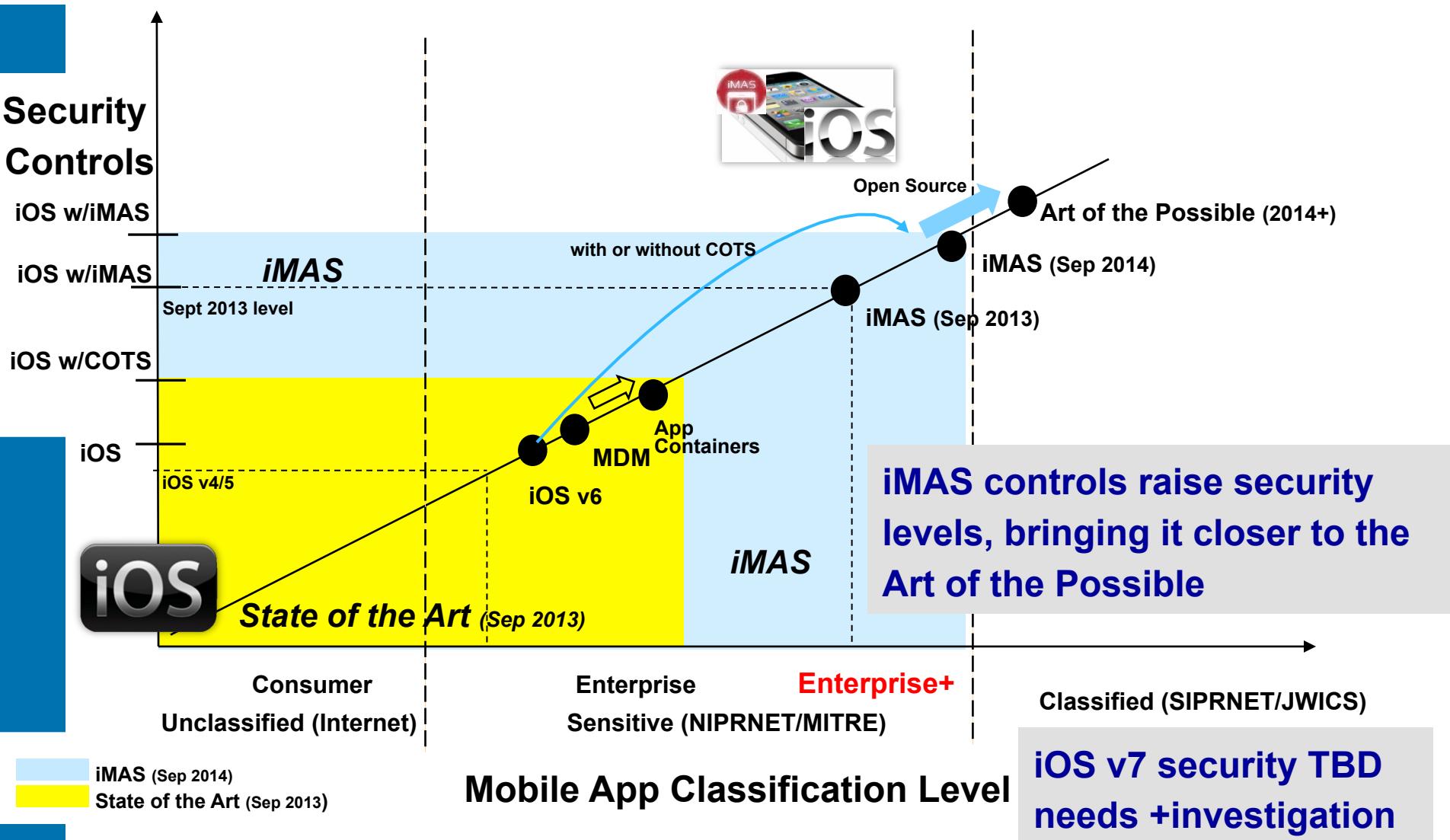
User-Based Threats

- Social Engineering
 - Phishing / Spear Phishing
 - Theft/Misuse of Services
 - Malicious Insider
- Tracking
 - Geolocation
 - Tower/Signal Traingulation
 - Geotagging
 - Mining of Call Records / Billing
 - Usage Patterns

 50% (12) iMAS Applicable

iMAS App Security “trade-space” Comparison

Sep 2013



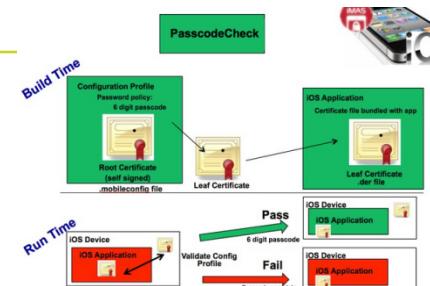
iMAS Security Controls

Security Controls

Breakout

Passcode Check

- Programmatic library and method to enforce complex device passcode use
- When in place, extends jailbreak brute-force passcode guessing duration to years from minutes



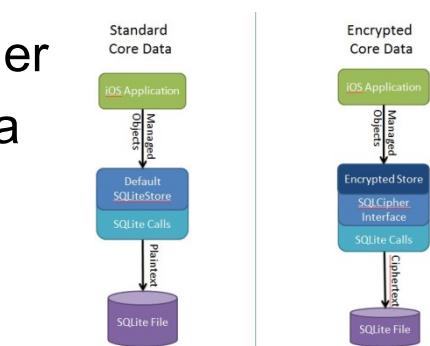
App-Password

- Secure, simple, user resettable password to protect access to an application and its data
- App level authentication enhances defense in depth
- Rotating keypad



Encrypted-core-data

- iOS CoreData encrypted SQLite store using SQLCipher
- iMAS has made it easy for developers to incorporate a secure SQL DB



Security Controls

Breakout (cont.)

Security-Check

- Application level, attached debug detect and jailbreak testing
- iMAS added extensive anti-forensic techniques to this library including; C macro functions, nameless function blocks, callback based, easy thread spawn for continuous checking.

Secure Foundation

- OpenSSL based secure components enabling application authentication, secure file storage, and app level file-based keychain
- iMAS alternative to Apple provided crypto support

iOS OpenSSL FIPS

- FIPS compliant OpenSSL drop-in ready for iOS

Jailbreak



Debugger

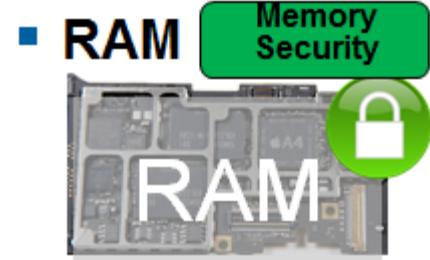


Security Controls

Breakout (cont.)

■ Memory-Security

- Provides a set of tools for memory regions or individual variables
- Securing through encryption
- Clearing and zeroing
 - Data sections overwritten either with an encrypted version or null bytes
 - Automatic on App exit
- Anti-tamper with checksum
 - For validating secure memory



■ Mobile Device Management

- Stood up an open source MDM test server
 - Open sourced method to create and register Certs with Apple
- Research security associated with MDM and remote app control
- Single sign on app prototype, remote app lock, remote app password reset, remote JB reporting, and iMAS MDM API

Forced Inlining

- Inlining considered an optimization technique
- Using it for security requires compiler specific knowledge
 - Both Apple's LLVM compiler and GNU's GCC LLVM compiler honor the same set of directives and flags
 - GCC has the benefit of broader community that has documented quirks
- always_inline
- Directive will force functions to be inlined in binary output, provided the compiler flags are appropriately set
 - Example: `inline void debug_check() __attribute__((always_inline));`
 - Only needed on function declarations, and not implementations
- Inline functions have a compiler limited size, must explicitly set the size as a compiler flag (`-finline-limit`) to ensure `always_inline` is honored
- Safest to use optimization at the lowest level (-O1)
 - Not setting optimization will remove inline functions
 - Xcode Project Build Settings:
 - Other C Flags: `-finline-limit=5000`
 - Compiler for C/C++: LLVM GCC
 - Optimization Level: Fast [-O, O1]

iMAS – Security Controls

Device Access:

4 Digit Passcode
No Passcode

Passcode
Check



App Access:

None

AppPassword

Data At Rest:

Keychain
CoreData

Encrypted Core
Data

Secure Foundation

Lightning Connector

MDM Remote Control

Run-time:

RAM and Debugger

Jailbreak / Root
Access

Security-Check

Jailbreak / debugger
attach

Memory
Security

Encrypted RAM
Disk

AppStore / Malware:

App Tampering

Forced-inlining

ApplIntegrity
Check

Encrypted Code
Modules (ECM)

Data in Transit:

iSECpartners



ssl-conservatory

forked from iSECPartners/ssl-conservatory

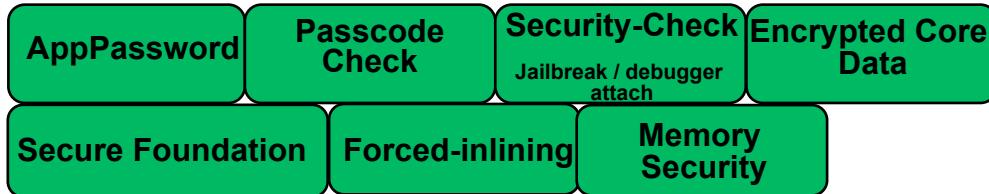
SSL certificate pinning implementation

iMAS

Vulnerable Areas
Future Research

iMAS Availability

- Available Today on github



- Jan - Mar 2014: Active Research



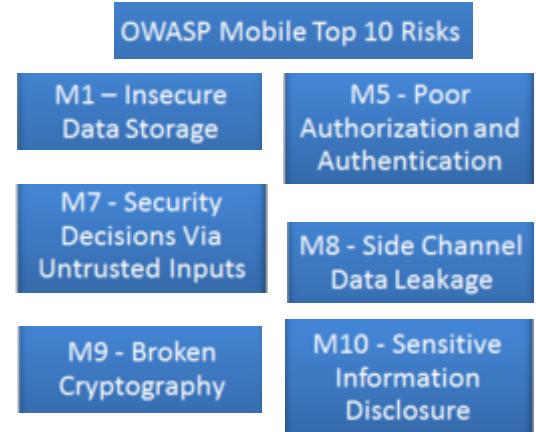
- Future



Security Areas:

- App authentication
- Encrypted SQLite
- Data at rest protection
- App at rest security
- Device Passcode check
- Jailbreak detection
- Debugger attach detect
- FIPS compliance openSSL

- Applicable OWASP Mobile Risks (6 out of 10)



iMAS Deep Dive

Security Check Deep Dive

- Jailbreak and debug attach detection
- Two Interesting iMAS techniques

1. Sysctl

- Using sysctl call to detect if debugger is attached
- Forensically attacker can “catch” this and simply override to return false
- Instead, iMAS is making direct assembly call to read process flag directly

2. LW Threads

- Making calls to JB and debugger detect repeatedly
- Both techniques are enormously more difficult to work around

```
//-----
// Assembly Level interface to sysctl
//-----

#define sysCtlSz(nm,cnt,sz)    readSys((int *)nm,cnt,NULL,sz)
#define sysCtl(nm,cnt,lst,sz)   readSys((int *)nm,cnt,lst, sz)
```

```
BEGIN_FUNCTION readSys
// -----
// prolog
// -----
push    {r4,r5,r6, r7,lr}      // Save registers r4-r7
add     r7, sp,#12              // Adjust FP to point
push    {r8,r10,r11,r14}        // Save any general reg
vstmda  sp!, {d8-d15}          // Save any VFP or NEON
sub     sp, sp,#4               // Allocate space for local vars

mov    r4, #0
mov    r5, #0
mov    r8, #0
mov    r9, #0

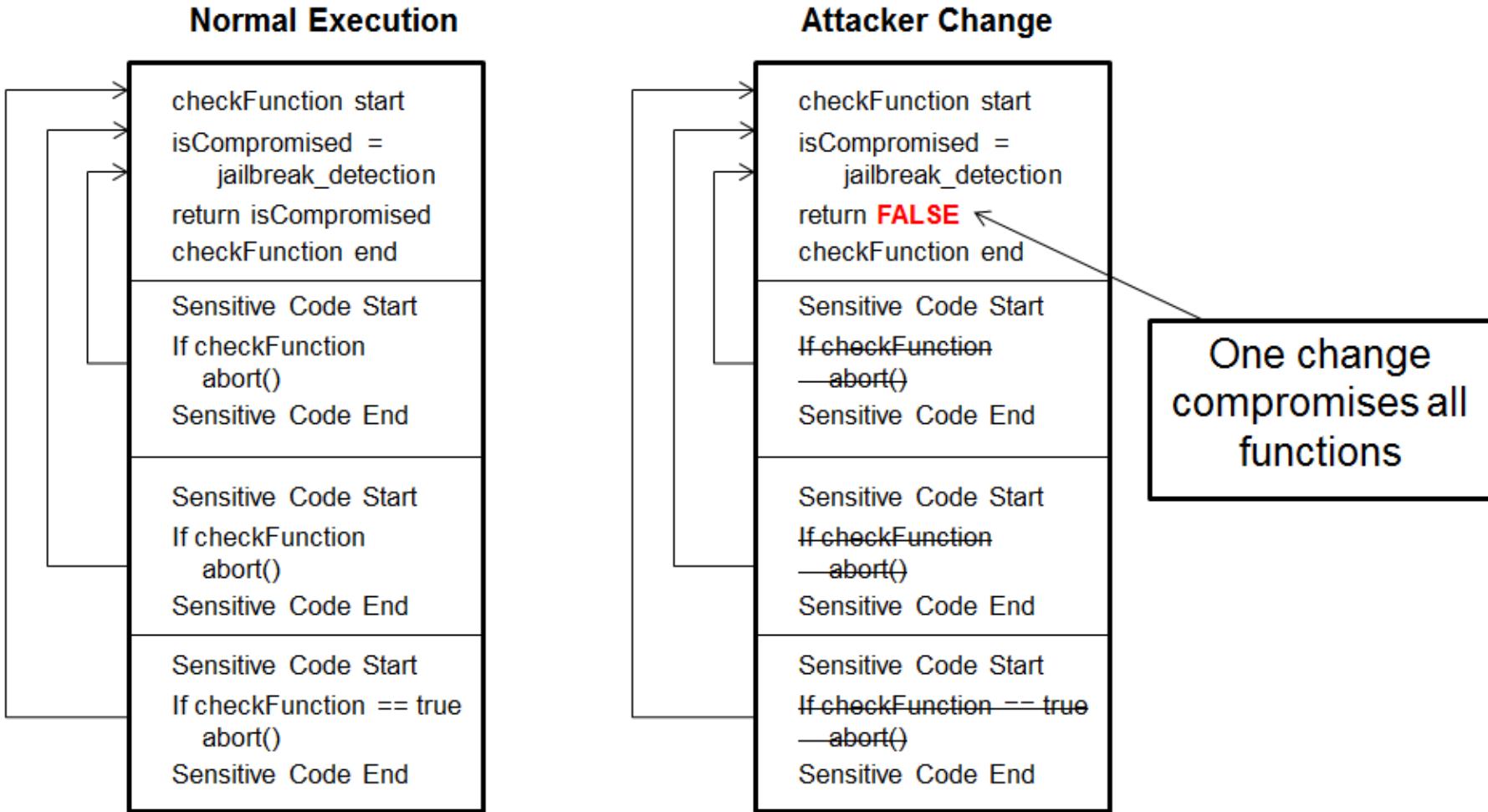
mov    r12, #202
svc    #128

// -----
// epilog
// -----
add    sp, sp,#4               // Deallocate space for local vars
vldmia sp!, {d8-d15}          // Restore any VFP or NEON
pop    {r8,r10,r11,r14}        // Restore any general register
pop    {r4,r5,r6, r7,pc}       // Restore saved r4-r7 and PC

END_FUNCTION
```

Forced Inlining

Compromising Security Checks



Forced Inlining

Inlining Compiled in

Compiled Version With Forced Inline Changes

```
Sensitive Code Start
checkFunctionVar =
jailbreak_detection
If checkFunctionVar
    abort()
Sensitive Code End
```

```
Sensitive Code Start
checkFunctionVar =
jailbreak_detection
If checkFunctionVar
    abort()
Sensitive Code End
```

```
Sensitive Code Start
checkFunctionVar =
jailbreak_detection
If checkFunctionVar
    abort()
Sensitive Code End
```

Attacker Changes

```
Sensitive Code Start
checkFunctionVar =
jailbreak_detection
If FALSE
    abort()
Sensitive Code End
```

```
Sensitive Code Start
checkFunctionVar =
jailbreak_detection
If FALSE
    abort()
Sensitive Code End
```

```
Sensitive Code Start
checkFunctionVar =
jailbreak_detection
If FALSE
    abort()
Sensitive Code End
```

Attacker must
now replace
each individual
check

Forced Inlining

Increases Exploit Time

Assembly Code Comparison (Red indicates security check flag)

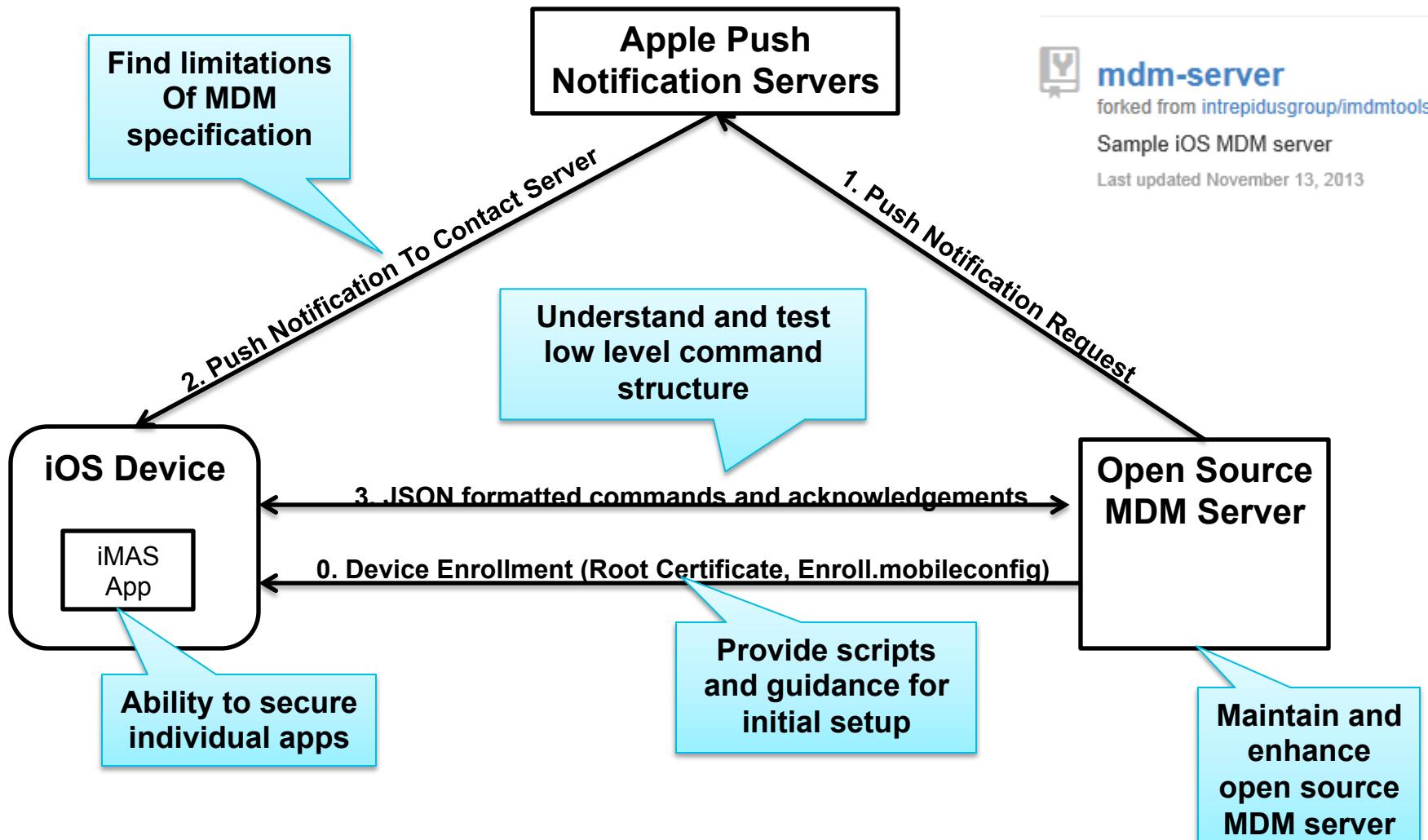
Without Forced Inlining

(Small haystack, one needle)
(232 instructions, 1 patch needed)

Same Program with Forced Inlining

(Large haystack, multiple needles)
(530 instructions, 13 patches needed)

Mobile Device Management (MDM) Research

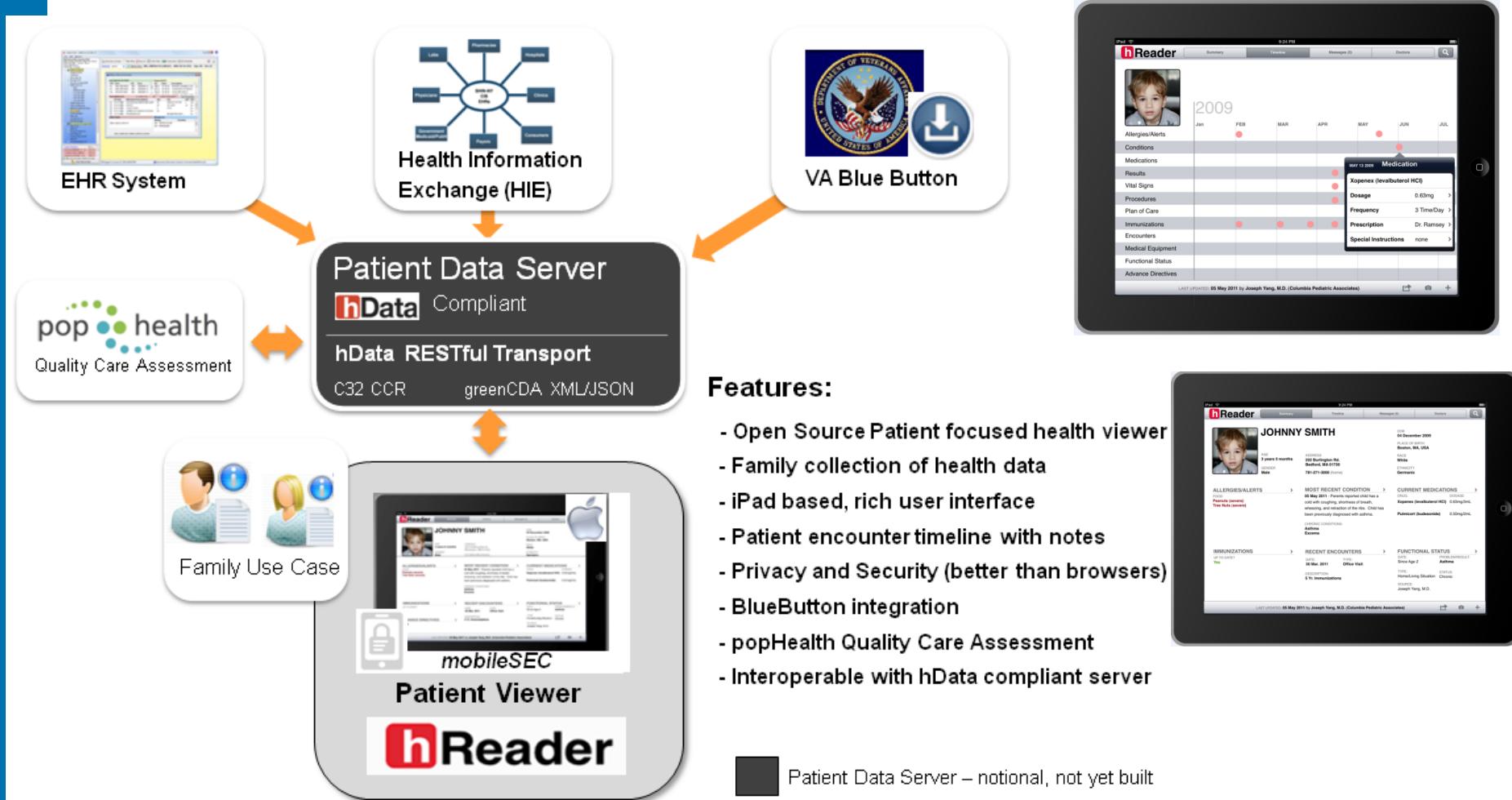


iMAS Applied



mobile Patient Health Reader

MITRE Internal Research - MIP



Audit Summary

August 2012

- **hReader team targeting Open Source space for hReader application**
 - hReader would be used mostly offline
 - Application security needs to be bolstered
 - Production hReader app would contain sensitive patient health information
- **hReader requested a red-team to security audit the application**
 - Test app contains “synthetic” patient data; NOT real patient data
- **A DISA App security STIG checklist was used to measure the security compliance**
- **Iterative process over several months**
 - First pass, many areas were found to be exposed
 - Third pass, considerably more secure

STIG Compliance

Security Technical Implementation Guide

- DISA / DoD Application STIG
- 255 total criteria
- 78 Applicable to hReader as a mobile prototype
- Initial assessment
 - 27 pass (16%), 10 fail, 129 unsure (166 Total), 89 NA
- After several audits
 - 72 pass (90%), 8 fail (80 total), 175 NA

184	(APP3060:CATII)	The Designer will ensure the application does not store configuration and control files in the same directory as user data.	NO	See "Audit v2 finder - Patient Data"	NO
185	(APP3210.1:CATII)	The Designer will ensure NIST-certified cryptography is used to generate/store sensitive information if required by the information system.	NO	See "Audit v2 finder - Keyboard cache data leak"	NO
186	(APP3220.1:CATII)	The Designer will ensure sensitive data held in memory is cryptographically protected when not in use if required by the information system.	NO	See "Audit v2 finder - Keyboard cache data leak"	NO
187	(APP3320.2:CATII)	The Designer will ensure the application has the capability to require password strength requirements that contain a mix of uppercase letters, lowercase letters, numbers, and special characters.	NO	The password used in testing was *123456*	NO
188	(APP3320.8:CATII)	The IAO will configure the application to ensure account passwords conform to DOD security policy.	NO	Passwords are freely chosen by user.	NO
189	(APP3320.1:CATII)	The Designer will ensure threat models are documented and reviewed for each application release and updated as required by design and functionality changes or new threats are discovered.	UNKNOWN	See above.	NO
190	(APP3100:CATII)	The Designer will ensure the application removes temporary storage of files and cookies when the application is terminated.	UNKNOWN	hReader's source code and file organization are unknown.	NO
191	(APP3230.1:CATII)	The Designer will ensure the application properly clears or encrypts all sensitive data before it is transmitted to a consecutive host required by the information system.	UNKNOWN	hReader manages memory at run time is unknown.	NO
192	(APP3390:CATI)	The Designer will ensure user accounts are locked after three consecutive unsuccessful login attempts within one hour.	UNKNOWN	Whether or not hReader implements this is unknown.	NO
193	(APP2060.1:CATII)	The Program Manager will ensure the development team follows a set of coding standards.	UNKNOWN	Establishing coding standards for developers is good practice, and is recommended.	NO
194	(APP2060.2:CATII)	The Program Manager will ensure the development team creates a list of unsafe functions to avoid and document this list in the coding standards.	UNKNOWN	User function calls are a potential security vulnerability and should not be used. Documentation for unsafe functions in any language is readily available, or are automated checkers, both of which are recommended.	NO
195	(APP3010:CATII)	The Designer will create and update the Design Document for each role of the application identifying the following: - all external interfaces (from the threat model) - the nature of information being exchanged - categories of sensitive information processed or stored and their specific protection plan - the protection mechanism associated with each interface - user roles required for access control - access privilege assigned to each role - unique application security requirements - categories of sensitive information processed or stored and specific protection plan (e.g., Privacy Act, Health Insurance Portability and Accountability Act).	UNKNOWN	Keeping a detailed and up-to-date design document like this is recommended.	NO
196	(APP3130:CATI)	The Designer will ensure the application follows the secure failure design principle.	UNKNOWN	hReader's source code is unknown, but the secure failure design principle states that all possible exceptions should be checked for, and in the event of an exception, the most error-prone code branch should be executed. This is the most reliable way to avoid error handling vulnerabilities, and should be implemented.	UNKNOWN
197	(APP3590.1:CATI)	The Designer will ensure the application does not have buffer overflow.	UNKNOWN	At the very least, no canonical representations of vulnerabilities were found in our tests.	UNKNOWN
198	(APP3590.2:CATII)	The Designer will ensure the application does not functions known to be vulnerable to buffer overflow.	UNKNOWN	hReader handles memory allocation internally is unknown.	UNKNOWN
199	(APP3590.3:CATII)	The Designer will ensure the application has no canonical representations of vulnerabilities.	UNKNOWN	The source code of hReader is unknown.	TEST
200	(APP3630.1:CATII)	The Designer will ensure the application does not write beyond the boundaries of memory allocated to it.	UNKNOWN	At the very least, no canonical representations of vulnerabilities were found in our tests.	TEST
201	(APP3630.2:CATIII)	The Designer will ensure the application does not vulnerable to race conditions.	UNKNOWN	hReader handles memory allocation internally is unknown.	TEST
202	(APP3630.3:CATII)	The Designer will ensure the application does not use global variables when local variables could be used.	UNKNOWN	The source code of hReader is unknown.	TEST
203	(APP3630.3:CATII)	The Designer will ensure a multi-threaded application uses thread-safe functions when threads are accessing the same global data.	UNKNOWN	The source code of hReader is unknown.	TEST
204	(APP3630.4:CATII)	The Designer will ensure global resources are locked before being accessed by the application.	UNKNOWN	The source code of hReader is unknown.	TEST
205	(APP3050:CATII)	The Designer will ensure the application does not contain source code that is never invoked during operation, except for software components and libraries from approved third-party products, which may include un-validated code.	UNKNOWN	The source code of hReader is unknown.	TEST

■ hReader application security compliance increased to 90% criteria met – about a 5X increase from original design

- STIG – A specification that defines the configuration standards for DOD IA and IA-enabled devices/systems
- STIG used: Application Security and Development STIG Version 3, Release 3

Third Party Audit

- **Report completed on May 6, 2013**
- **Failed Network security testing**
 - High risk found: network cache.db file had server password and patient information present
 - Related to backend server access
- **Other medium and low security risks found**
 - SQLite DB key was found in memory
 - Jailbreak detection and counter-debugging worked; patched over
- **Feedback from viaForensics**
 - hReader with iMAS is much more secure than most
 - Encouraged us to make changes to above and re-test for their “appSecure” certification
- **Fixed cited issues and concluded exercise**

- **PASSED “FINAL EXAM”**

Mitre (hReader)

appSecure – iOS 5/6

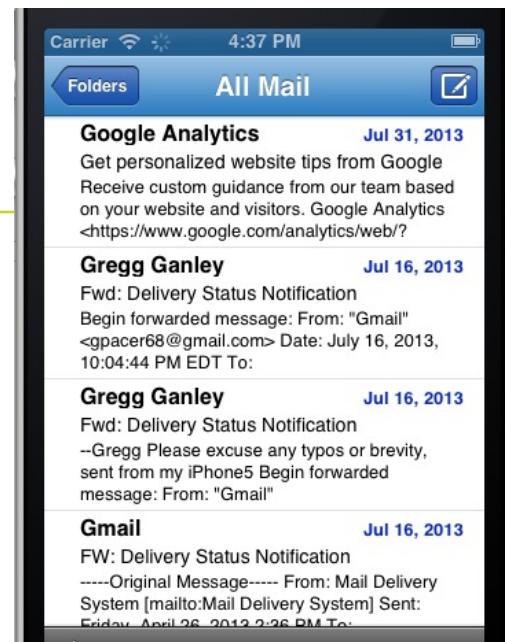
Mobile Forensic Security Assessment
Report of Findings

May 6, 2013

reMail

<http://code.google.com/p/remail-iphone/>

- FOSS project acquired by Google in 2010
- Email client for iOS
- Project site TODO lists “security needed”
- iMAS test app
 - Wrap app with iMAS security controls
 - [Github project started](#)



 **remail-iphone**
reMail for iPhone

[Project Home](#) [Wiki](#) [Issues](#) [Source](#)

reMail – iMAS Security Integration

- Securing
 - ✓ Access
 - ✓ Data at rest
 - ✓ Run-time
 - ✓ Anti-tampering
 - ✓ FIPS Compliance



- Data Storage
 - openSSL crypto
 - Keychain
- Jailbreak
 - Debugger
 - Security-Check
 - Jailbreak / debugger attach



Outreach

iMAS Now an OWASP Project



OWASP

The Open Web Application Security Project

[Log](#)
[Page](#)
[Discussion](#)
[Read](#)
[View source](#)
[View history](#)

Navigation

- [Home](#)
- [About OWASP](#)
- [AppSec Conferences](#)
- [Chapters](#)
- [Downloads](#)
- [Mailing Lists](#)
- [Membership](#)
- [News](#)
- [OWASP Books](#)
- [OWASP Gear](#)
- [OWASP Initiatives](#)
- [OWASP Projects](#)
- [Presentations](#)
- [Press](#)
- [Video](#)
- [Volunteer With OWASP](#)
- [Reference](#)

Projects/OWASP iMAS iOS Mobile Application Security Project

PROJECT INFO

What does this OWASP project offer you?

what

is this project?

Name: OWASP iMAS - iOS Mobile Application Security Project ([home page](#))

Purpose: iMAS – iOS secure application framework to reduce iOS application vulnerabilities and information loss. iMAS and its first open source static security controls for download and use in iOS applications. Visit and browse our project to find out more; download and give it a try. Once you do, tell us what you think or better yet, get involved and participate!

<https://github.com/project-imas/about>

iMAS is a collaborative research project from the MITRE Corporation focused on open source iOS security controls. Today, iOS meets the enterprise security needs of customers. however many security experts

RELEASE(S) INFO

What releases are available for this project?

current release

Not Yet Published

last reviewed release

Not Yet Reviewed

other releases

iMAS listed as OWASP Mobile Security Project – Mobile Tool

 Log in / create account

OWASP

The Open Web Application Security Project

Page Discussion Read View source View history Go Search

OWASP Mobile Security Project

(Redirected from [Mobile](#))

Project Overview Top Ten Mobile Risks Mobile Tools Mobile Security Testing Mobile Cheat Sheet Series Secure Mobile Development
Top Ten Mobile Controls OWASP Mobile Threat Model Project

iMAS

iMAS is a collaborative research project from the MITRE Corporation focused on open source iOS security controls. Today, iOS meets the enterprise security needs of customers, however many security experts cite critical vulnerabilities and have demonstrated exploits, which pushes enterprises to augment iOS deployments with commercial solutions. The iMAS intent is to protect iOS applications and data beyond the Apple provided security model and reduce the adversary's ability and efficiency to perform recon, exploitation, control and execution on iOS mobile applications. iMAS will transform the effectiveness of the existing iOS security model across major vulnerability areas including the System Passcode, jailbreak, debugger / run-time, flash storage, and the system keychain. Research outcomes include an open source secure application framework, including an application container, developer and validation tools/techniques.

[iMas Project Page](#)

The source code for iMAS is available on GitHub: [iMAS Source Code](#)

GoatDroid

OWASP GoatDroid is a fully functional and self-contained training environment for educating developers and testers on Android security. GoatDroid requires minimal dependencies and is ideal for both Android beginners as well as more advanced users. The project currently includes two applications: FourGoats, a location-based social network, and Herd Financial, a mobile banking application. There are also several feature that greatly simplify usage within a training environment or for absolute beginners who want a good introduction to working with the Android platform.

As the Android SDK introduces new features, the GoatDroid contributors will strive to implement up-to-date lessons that can educate developers and security testers on new security issues. The project currently provides coverage for most of the OWASP Top 10 Mobile Risks and also includes a bunch of other

Navigation

Home About OWASP AppSec Conferences Chapters Downloads Mailing Lists Membership News OWASP Books OWASP Gear OWASP Initiatives OWASP Projects Presentations Press Video Volunteer With OWASP

Reference Activities Attacks Code Snippets Controls

Web Site

project-imas.github.com



iOS Mobile Application Security

Home

Components

Contact



Defense for your iOS App

iMAS helps developers [encrypt app data](#), [prompt for passwords](#), [enforce password complexity](#) and [other security policies](#) on iOS devices.

[Learn More](#)

hReader

MITRE

CWE List | Full CWE View | Development View | CWE Identifier Reports | About | Contact | Process Documentation | FAQ | Community | Block On Report | Top 10 | Discussion List | Discussion Archives | Contact Us | Search | CWE ID | CWE-Index | CWE-Suite Top 25 | CWE Metrics | Requirements | CWE-Index Representation | CWE-ID Definition | Help & Feedback | Calendar | Search the Site | Related Efforts | Vulnerabilities (CVES) | Mitigation Patterns (CVMP) | Cyber Observables (CybOX) | NVD (National Vulnerability Database) | Reference ID | CWE-Index | Structured Threat Information (STIX) | Workforce Scoring System (CWSS) | MITRE ATT&CK Framework (CWRF) | Build Security In (BSI) | Making Security Happen (MSH)

Github.com site

Nine (9) iMAS security controls available



iMAS - iOS Mobile Application Security project-imas

project-imas-list@lists.mitre.org

<https://github.com/project-imas/about>

Joined on Dec 06, 2012

14

2

3

public repos private repos

members



encrypted-core-data

iOS Core Data encrypted SQLite store using SQLCipher

Objective-C ★ 175 ⚡ 35



app-password

Custom iOS user authentication mechanism (password with security questions for self reset)

Objective-C ★ 76 ⚡ 6



securefoundation

OpenSSL based secure components enabling application authentication, secure file storage, and app level file-based keychain

Objective-C ★ 22 ⚡ 6



security-check

Application level, attached debug detect and jailbreak checking

C ★ 27 ⚡ 5



memory-security

Tools for securely clearing and validating iOS application memory

Objective-C ★ 1 ⚡ 0



passcode-check

iMAS passcode-check, set passcode config profiles and check for conformance

Objective-C ★ 14 ⚡ 3



project-imas/single-sign-on

Simple MDM Single Sign On solution for application level logins

Objective-C ★ 0 ⚡ 0



forced-inlining

Inlining functions can be a very effective method of duplicating sensitive code for increased security

Assembly ★ 0 ⚡ 0



project-imas/mdm-server

forked from intrepidusgroup/imdmtools

Python ★ 1 ⚡ 29

Github Use and Value

Claim: value = visits and use

- **Open source statistics**

- Leveraging githalytics.com
- Adds a Google analytics link to each iMAS security control README file



- **Since March 4 (~ 10 months)**

- Over 25,000 unique visitors; 38,500 page views (89% new, 11% returning)
- Visits from 115 countries (mostly US, Germany, India, UK, Canada, China)

- **Site Numbers**



- **Community following:**

- Star count 340 (up from 331 1/10)
- 65 forks (up from 57 on 10/5)

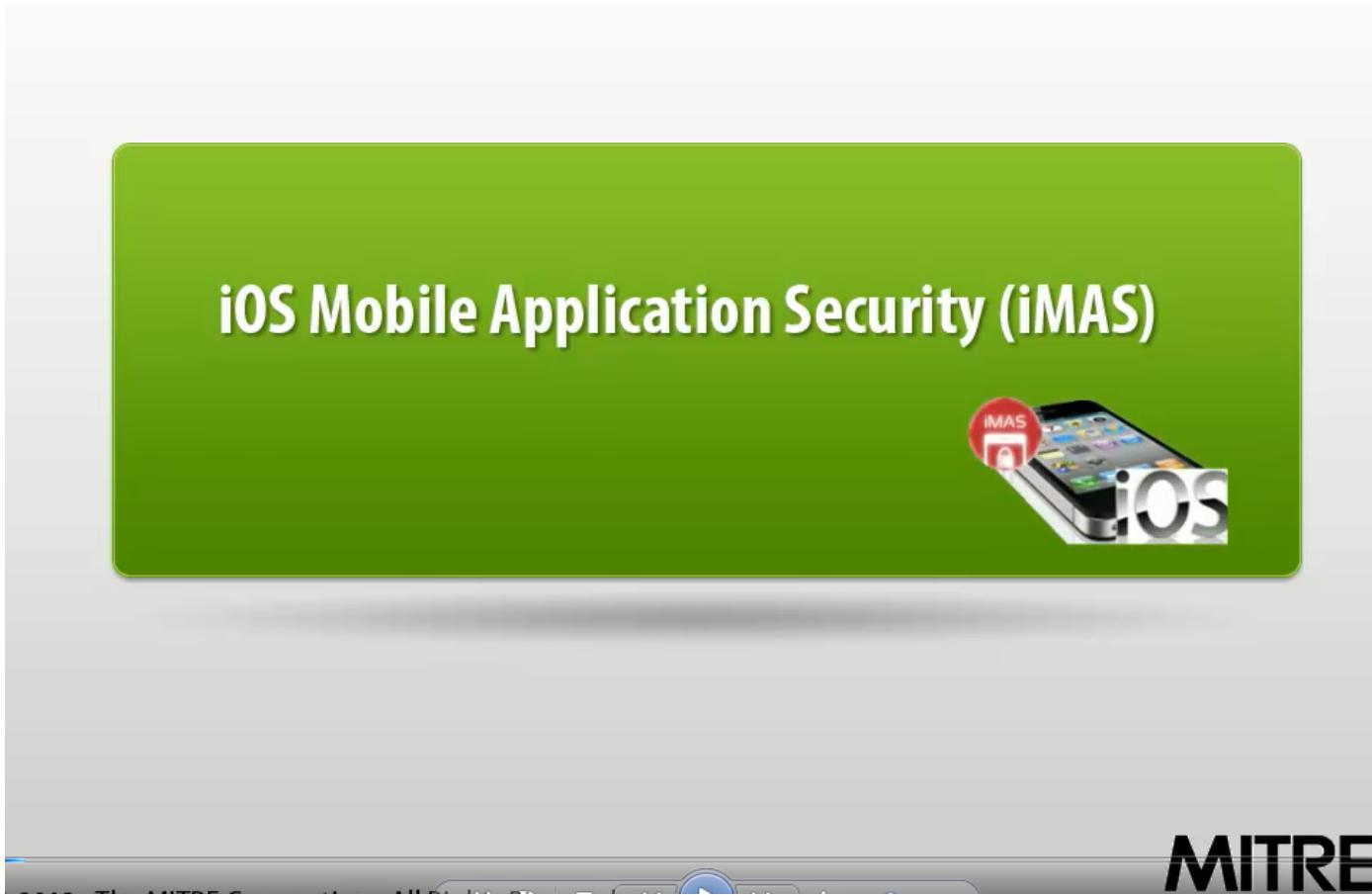
- **Value measure**

- Considerable, steadily increasing

Page	Pageviews	% Pageviews
1. /project-imas/encrypted-core-data	12,722	33.78%
2. /project-imas/app-password	6,742	17.90%
3. /project-imas/security-check	3,941	10.46%
4. /project-imas/securefoundation	3,862	10.25%
5. /project-imas/passcode-check	3,786	10.05%
6. /project-imas/about	1,788	4.75%
7. /	1,666	4.42%
8. /project-imas/iOS-openSSL-FIPS	1,242	3.30%
9. /project-imas/memory-security	1,045	2.77%
10. /project-imas/Rmail-iPhone	340	0.90%

iMAS Video

- Produced as part of MITRE's Mobile Computing Security Initiative (MOCSI) integration effort
- Short, 3.5 minutes
- http://youtu.be/92A3kg_kUSc



MITRE

MITRE

iMAS in the news ...

COMPUTERWORLD

Kenneth van Wyk: Making safer iOS apps

There still seem to be a lot of security flaws in iOS apps, but new tools could help fix that

Kenneth van Wyk

April 23, 2013 (Computerworld)

When it comes to developing secure apps for the iOS operating system, there's both good and bad news.

...

Relatively new is an open-source programming security framework from the folks at MITRE called iMAS. The iMAS library provides iOS developers with a set of easy-to-use tools to accomplish various security tasks in their apps.

Not sure how to use these things? Dive in and read the docs, and start trying them. And if you're

...

Without a doubt, the iOS app community needs more tools and frameworks like iMAS, but I think the new open-source developments give us good reason to believe more will be emerging. Keep

Transitions to date

- **MSR hReader FY12/13**
- **MSR Patient ToolKit FY13/14**
- **Demandware commercial product FY13**
- **Github Developers Dec/Jan FY14**
 - Gordon Tam
 - Lifestyle (think of Yelp but only for Food and Bars at this stage)
 - ECD
 - Oliver Woolgar
 - Tried using ECD, RestKit could not integrate with ECD
 - They will circle back on ECD later
 - Patrick Hartling and ECD
 - Mobile banking product. ECD as a replacement for our current mechanism of encrypting user financial details. I plan look at other IMAS security controls, as we are very interested in robust security controls for user data.
 - “This is a great library, and I hope to put it into production use soon. Thank you!”

Outreach

- **Emerging Technology TEM on Mobility Oct 2013**
 - iMAS presented as a turbo talk
- **OWASP AppSEC USA Nov 2013 speaker**
 - Great talk, demo'ed reMail and AppPassword; 250+ people attended a few questions at the end, several thanks for iMAS
- **ModDevEasy Conf 12/12**
 - Dave Keppler attended ModDevEast Conf on 12/12 and said “iMAS came up in at least three of the other talks I went to. Directly in the slides with the URL in too.”
- **CAMH Innovation Exchange 1/7/14**
 - Several leads, reached out to DHS Merv Leavy and Daniel Mobile Lab Team on 1/20
 - Briefed out iMAS Margie Zuk DH and Julie Connolly (J83B), Linda Koontz DH (K691)
- **Apple Federal Security Working Group 1/15**
 - Met Shawn Geddis, Apple Security Consulting Engineer
 - Very interested in iMAS, thought research was great
 - Wants to meet 1x1 in early Feb here at MITRE

Research Plan

Q1

- Inline code security control
- Open Source MDM research and controls (Single Sign on and iMAS MDM API)
- ~~Encrypted Code Modules (ECM)~~
- AppPassword and Encrypted CoreData (ECD) updates

Q2

- Encrypted CoreData (ECD)
 - Many to many updates, Cloud Storage security; iCloud interface
- Encrypted Code Modules (ECM) (Automated module wrap and iOS7 validation)
- OpenSSL removal (Apple is now FIPS compliant in iOS6 and iOS7)
 - Refresh secureFoundation, AppPassword, and ReMail

Q3

- ApplIntegrity check security control
- Encrypted RAM disk (TrueCrypt Integration)
- iOS7 Switchover (Change Dev environments to be iOS7 only)
- ReMail updated and third party vetted
- Encrypted Code Modules (ECM) integrated into iMAS security controls

Q4

- Dynamic App Bundling
- Multi-compiler research
- Off Device Trust (Smart charger, Lightning connector)

iMAS - iOS Mobile Application Security

Github:

<https://project-imas.github.com>

POC:

MITRE, Bedford MA

Gregg Ganley

781-271-2739

gganley@mitre.org

Gavin Black

781-271-4771

gblack@mitre.org



Questions?

Please !

- *Visit and Discover*
- *Download and Experiment*
- *Feedback and push requests*

Backup

iMAS

iOS Mobile Application Security



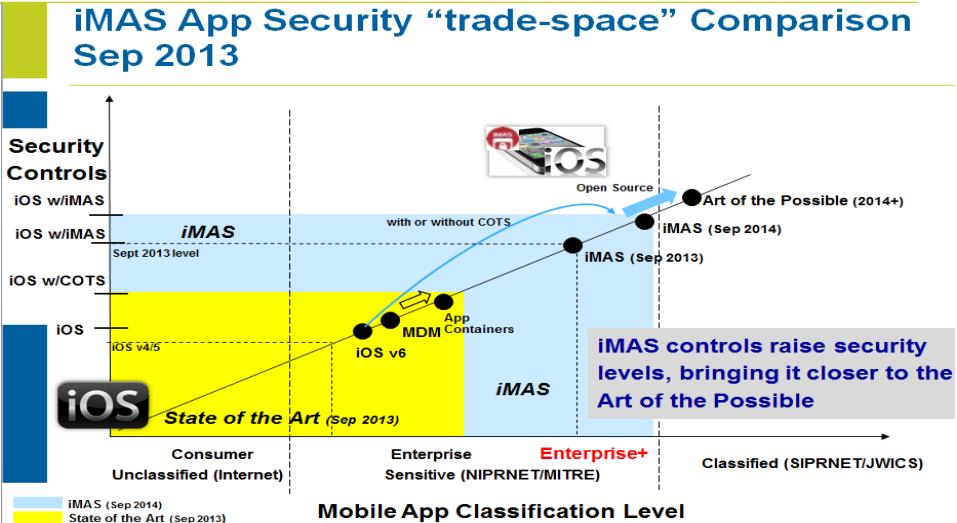
Gregg Ganley Gavin Black

Problem

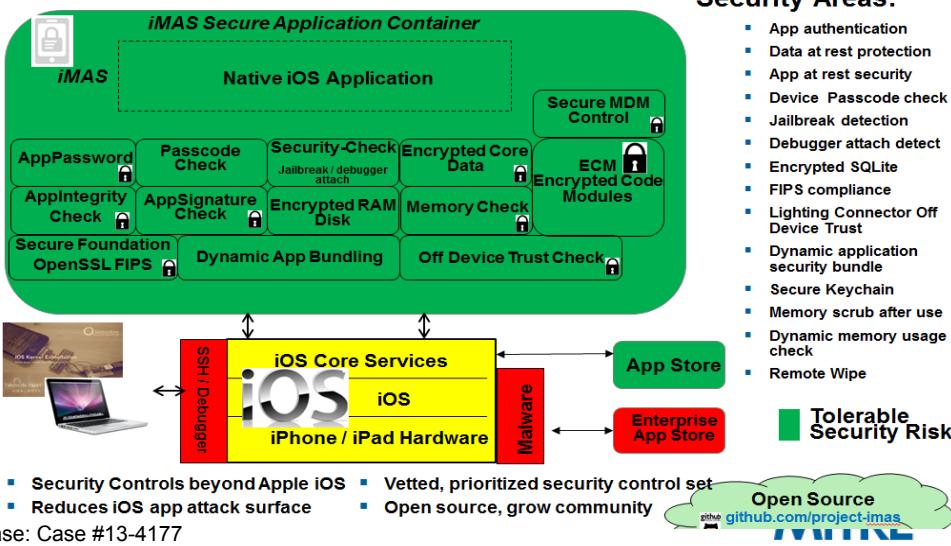
- iOS is considered secure, but out of the box security is not enough
- Simple device passcodes enable easy compromise of applications and data

Solution

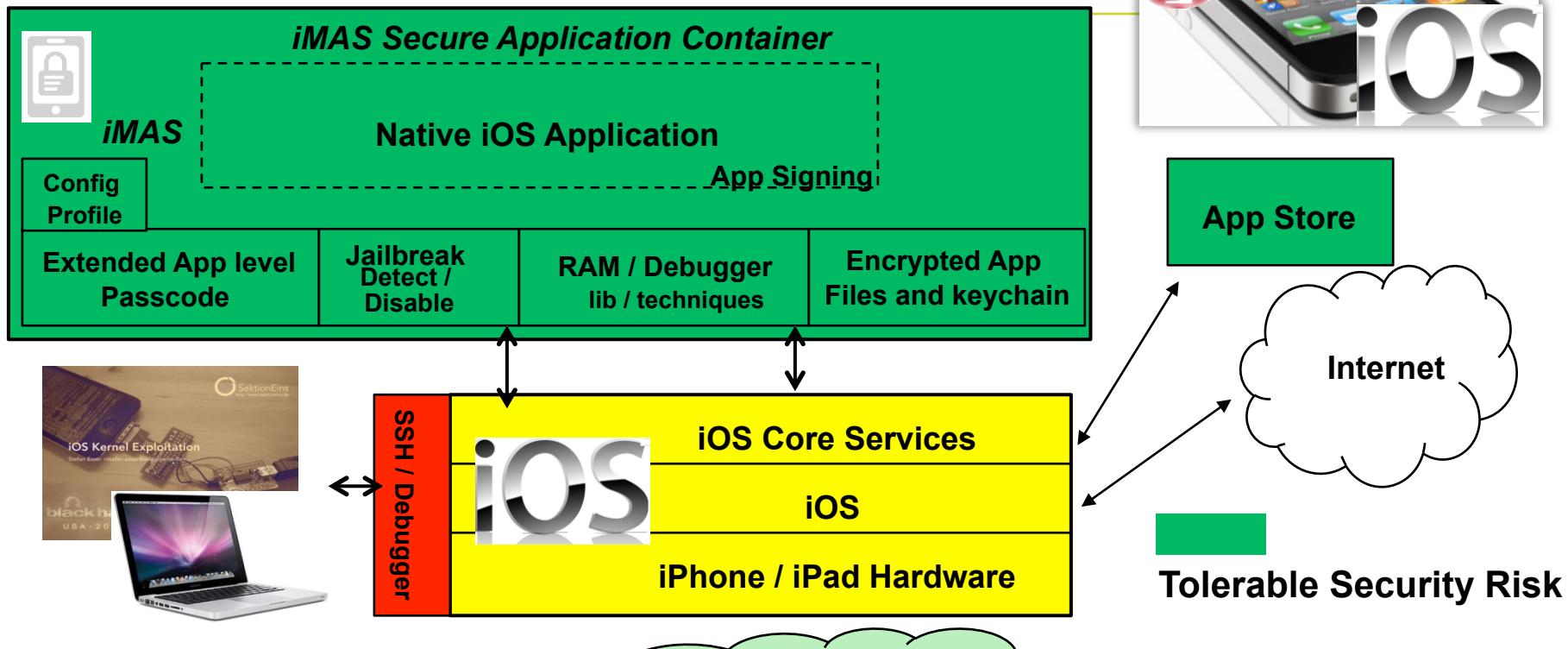
- Additional security controls beyond Apple
- Reduce iOS app attack surface
- Extends security with or without MDM and commercial solutions
- Measurable security (DISA Mobile App SRG)
- Open source available
project-imas.github.com
- Raise iOS app security levels - closer to the Art of the Possible



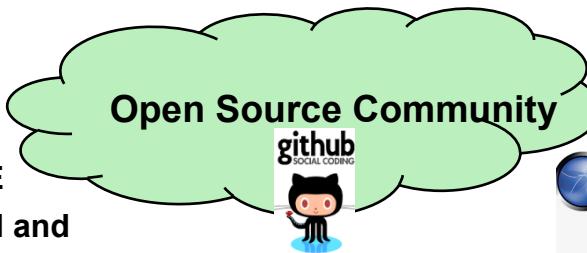
iMAS Secure Application Framework



iMAS Secure Application Framework FY13

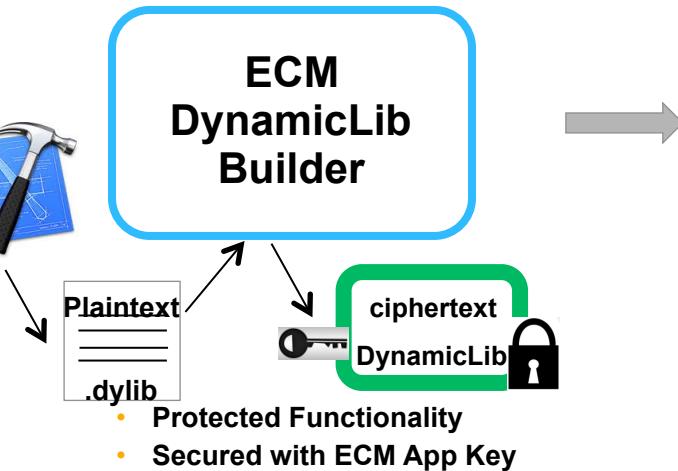


- Security Controls beyond Apple iOS
- Reduces iOS app attack surface
- Vetted security control set using CWE
- Extends security with or without MDM and COTS solutions
- Measurable security (DISA Mobile App SRG)
- Open source, grow community

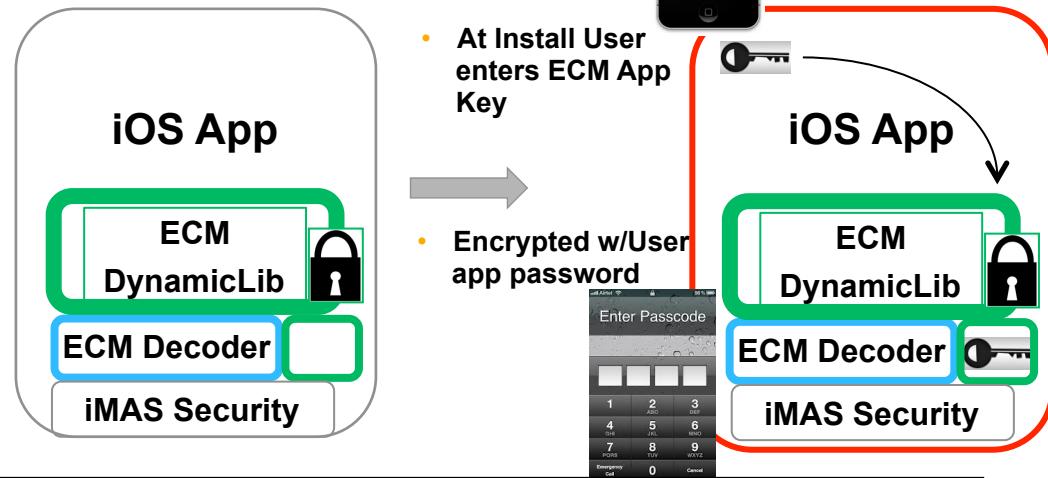


ECM – Encrypted Code Modules Concept

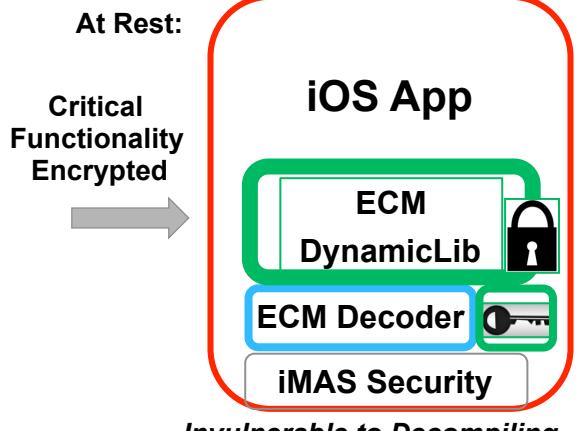
▪ Build Time



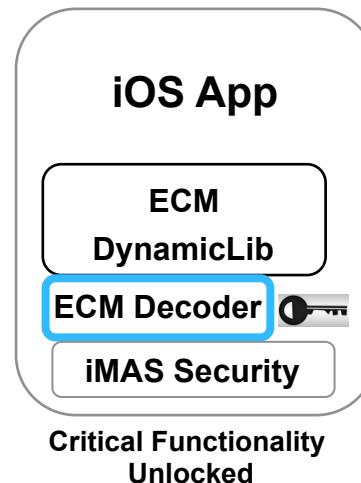
▪ ECM built into iOS App



▪ On Device



In Use:



iMAS Future

FY14 Technical Approach and Research

Deeper security controls

- Continue researching security controls (those not finished in FY13)

Advanced resilience and disruption techniques

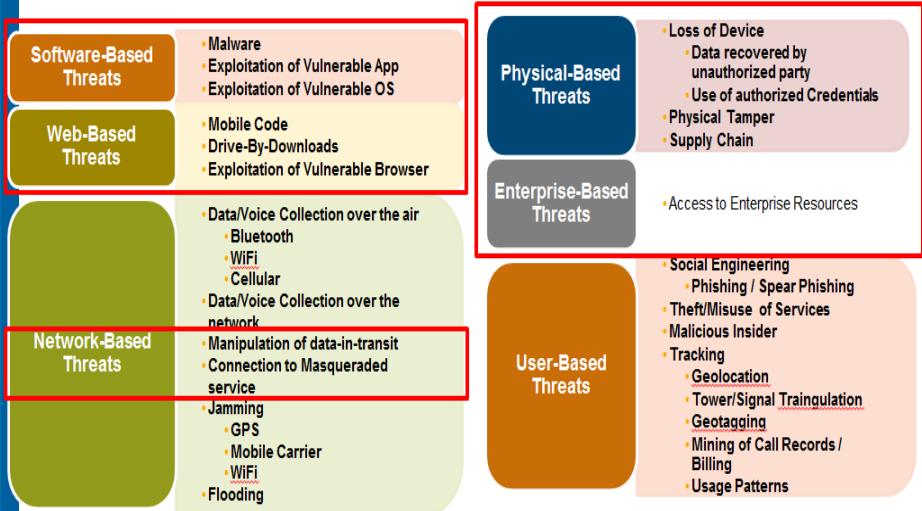
- MDM remote control
 - Explore securing MDM interface and remote C2 capabilities for iMAS controls (remote wipe etc.)
- iMAS Encrypted Code Modules research
 - Implement portions of security controls using iMAS ECM technique
- Off device trust
 - iOS Lightning Connector and trusted smart charger research
- Dynamic App Bundling research
 - Research app repackaging techniques; wrap app after build time
- Reference app bolstering
 - Implement iOS reference app (reMail) using iMAS ECM / SMIME
 - Third Party Audit reMail with viaForensics and Veracode

End Game

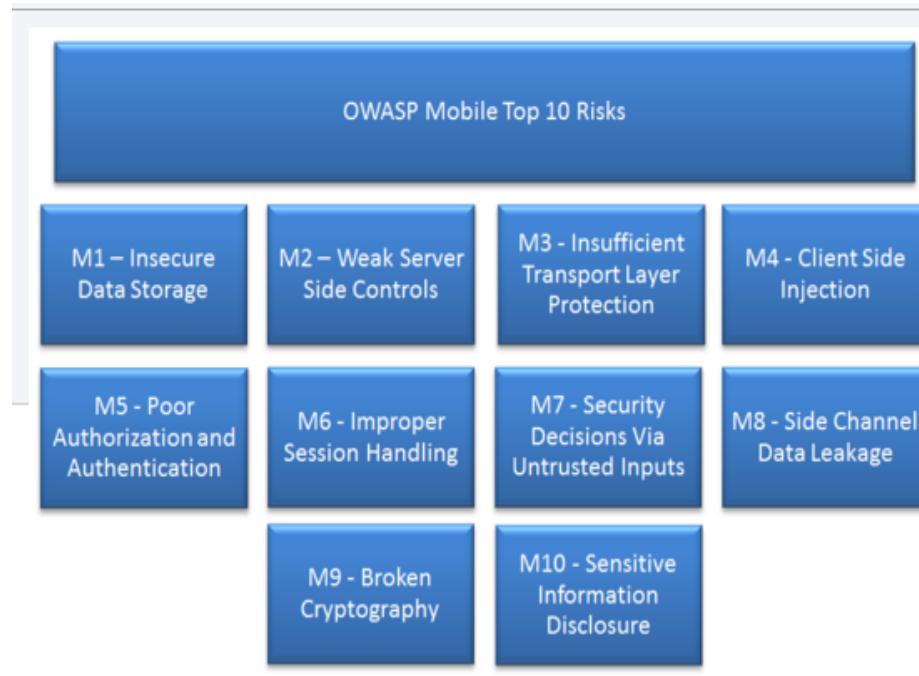
iMAS Goals

- Targeting “secure complete” when applicable OWASP mobile top 10 and 12/24 mobile threats are mitigated

- 24 threats most relevant to the mobile environment



■ 50% (12) iMAS Applicable



- Open Source is self sustaining
- Transition Research
 - Govt. and / or Commercial entity