iOS Mobile Application Security (iMAS)

FY13 -14 MITRE Research

January 2014

Research Team: Gregg Ganley(PI) and Gavin Black

MITRE

Approved for Public Release: Case #13-4177

About MITRE

- **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**

Approved for Public Release: Case #13-4177
MITRE

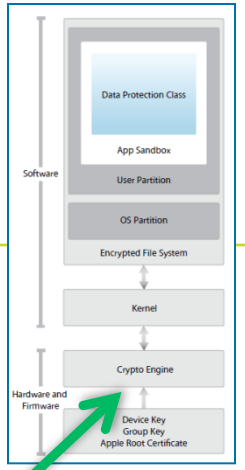
iOS Mobile App Security (iMAS) Elevator Pitch

iMAS – iOS Application Defense

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

Approved for Public Release: Case #13-4177

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.

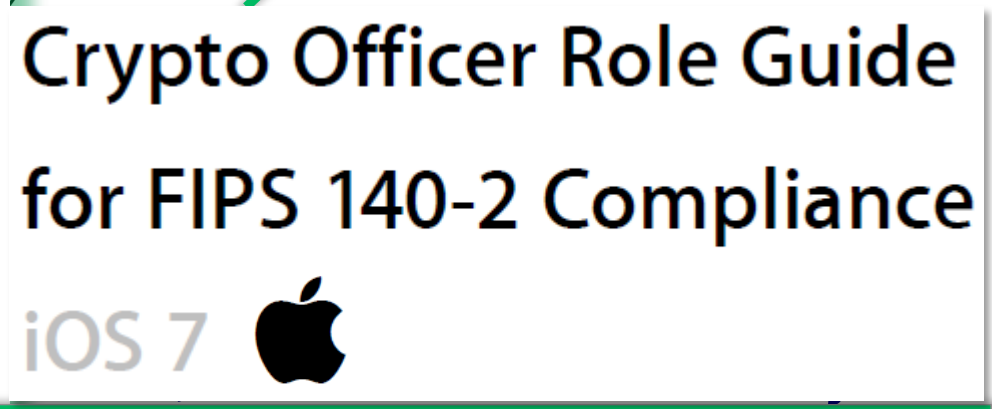
By Zack Whittaker for Between the Lines | May 7, 2013 -- 17:30 GMT (10:30 PDT)

Follow @zackwhittaker

Comments 52 Vote 1 Share more +

S and isolates applications while they are

lash support, reduced PDF/.MOV features, stripped

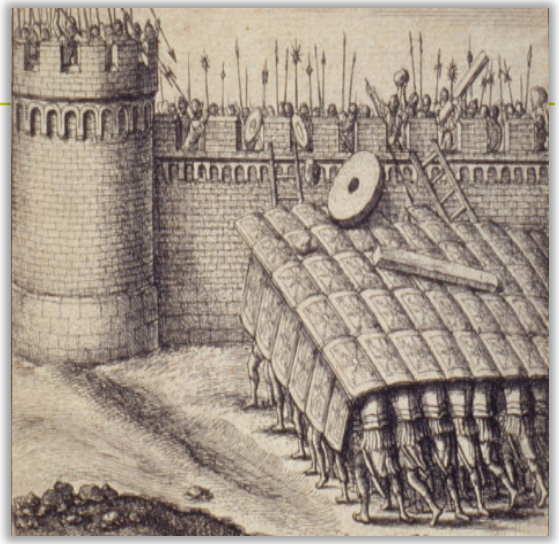


Crypto Officer Role Guide for FIPS 140-2 Compliance iOS 7

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

Approved for Public Release: Case #13-4177
MITRE

Research Scope



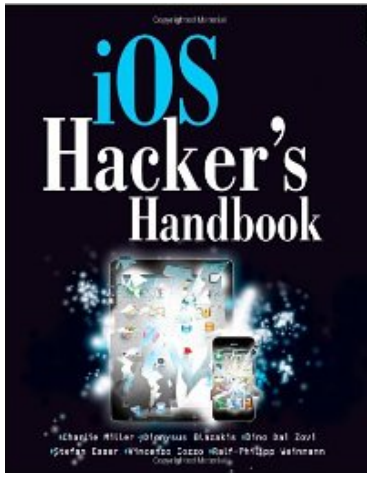
- **iOS attack surface is multi-faceted**
- **Threat vectors**
  - Lost/stolen device, physical access attacks
  - Malware inserted via
    - Browser JavaScript engine
    - Apps downloaded from *Enterprise App store*
- **Attacker purpose**
  - Steal application data, find enterprise credentials
  - 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

Approved for Public Release: Case #13-4177

Hacking and Jailbreaking iOS

- **Attacks and weaknesses are well documented:**

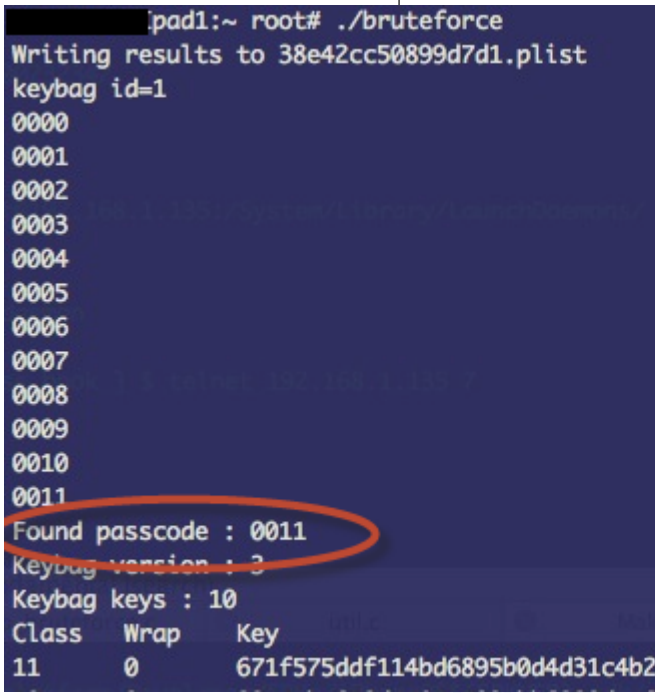


- **Recent Jailbreak:** http://evasi0n.com/

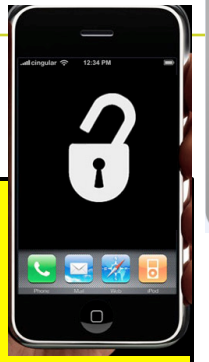
Worst-Case Passcode Guessing Time (iPhone4)

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



Problem:
Standard iOS Application Today



4 Digit Passcode

Native iOS Application

App Signing

System components:

RAM and Debugger

Jailbreak / Root Access

User Auth

App Access

Keychain

Flash Data Storage

App Store

SSH / Debugger

iOS Core Services

iOS

iPhone / iPad Hardware

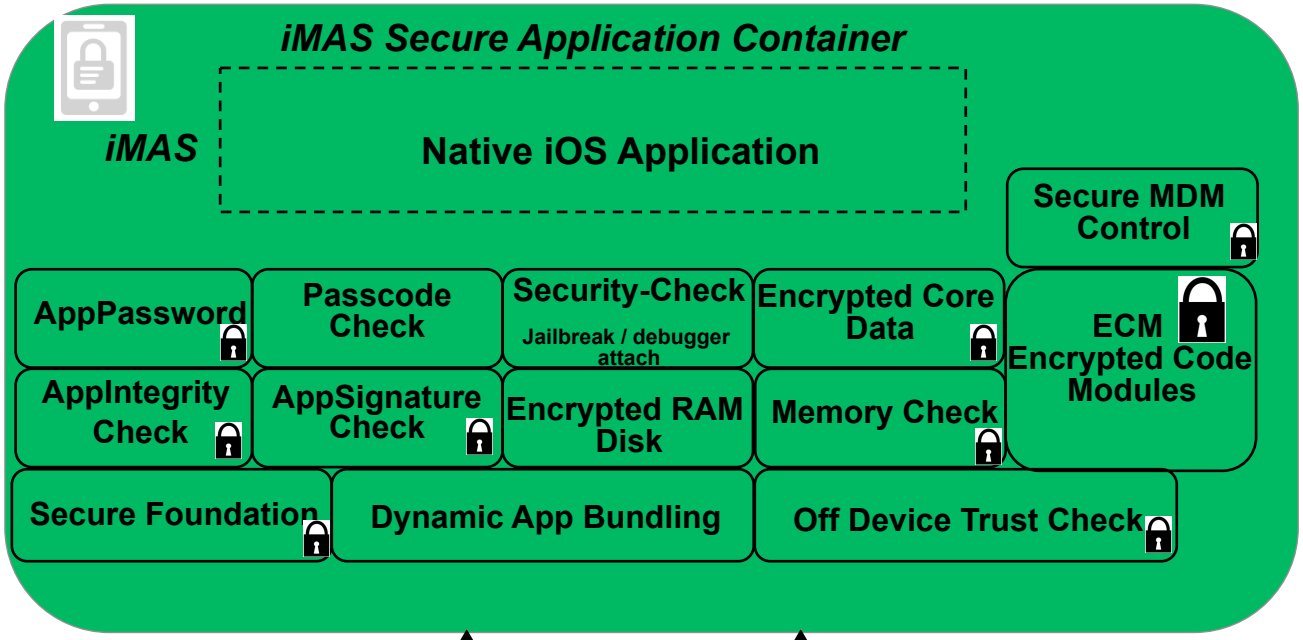
Internet

Vulnerable Areas

Approved for Public Release: Case #13-4177

MITRE

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

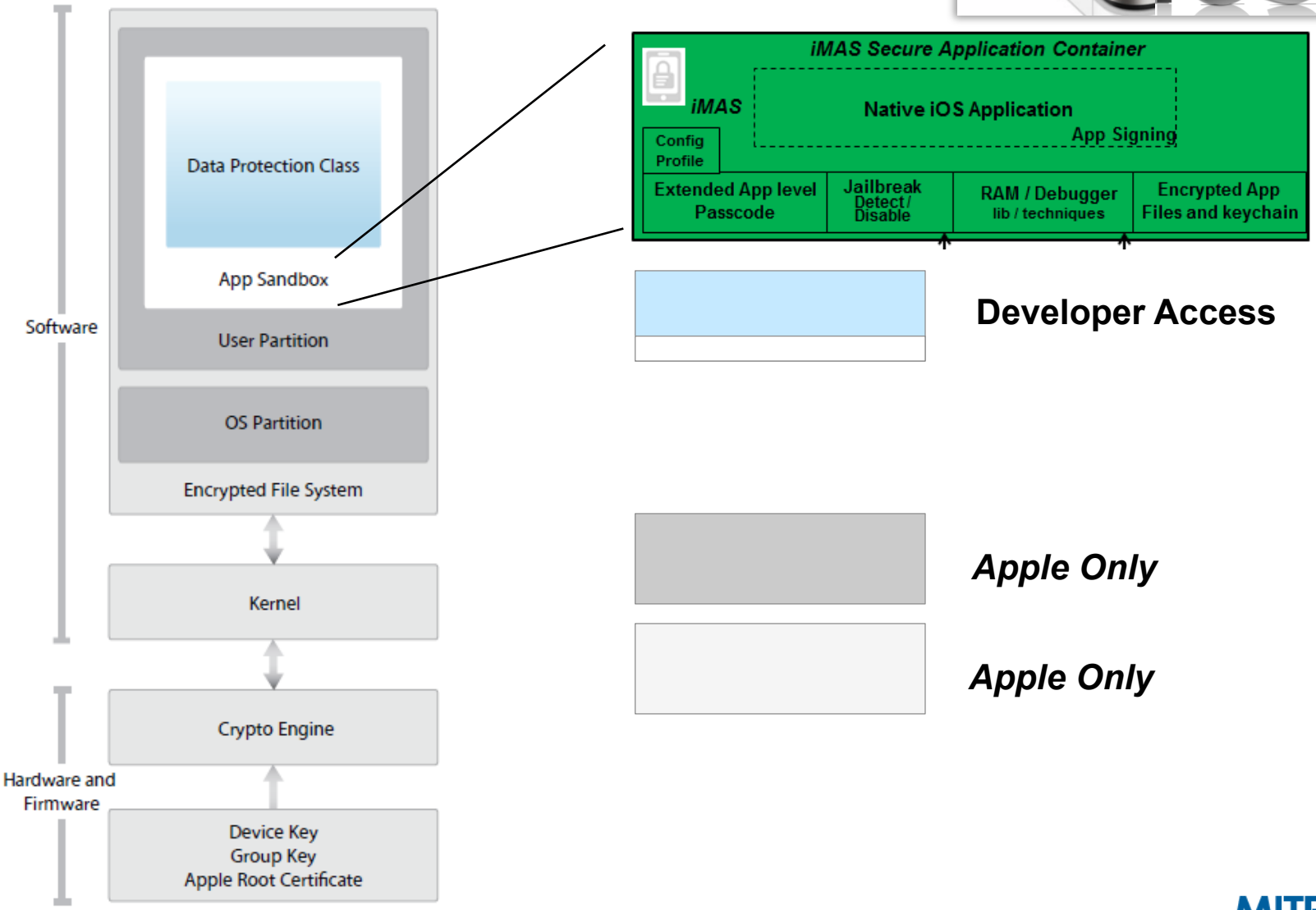
Tolerable Security Risk

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

Open Source
github.com/project-imas

MITRE

iOS Security Architecture



Approved for Public Release: Case #13-4177

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

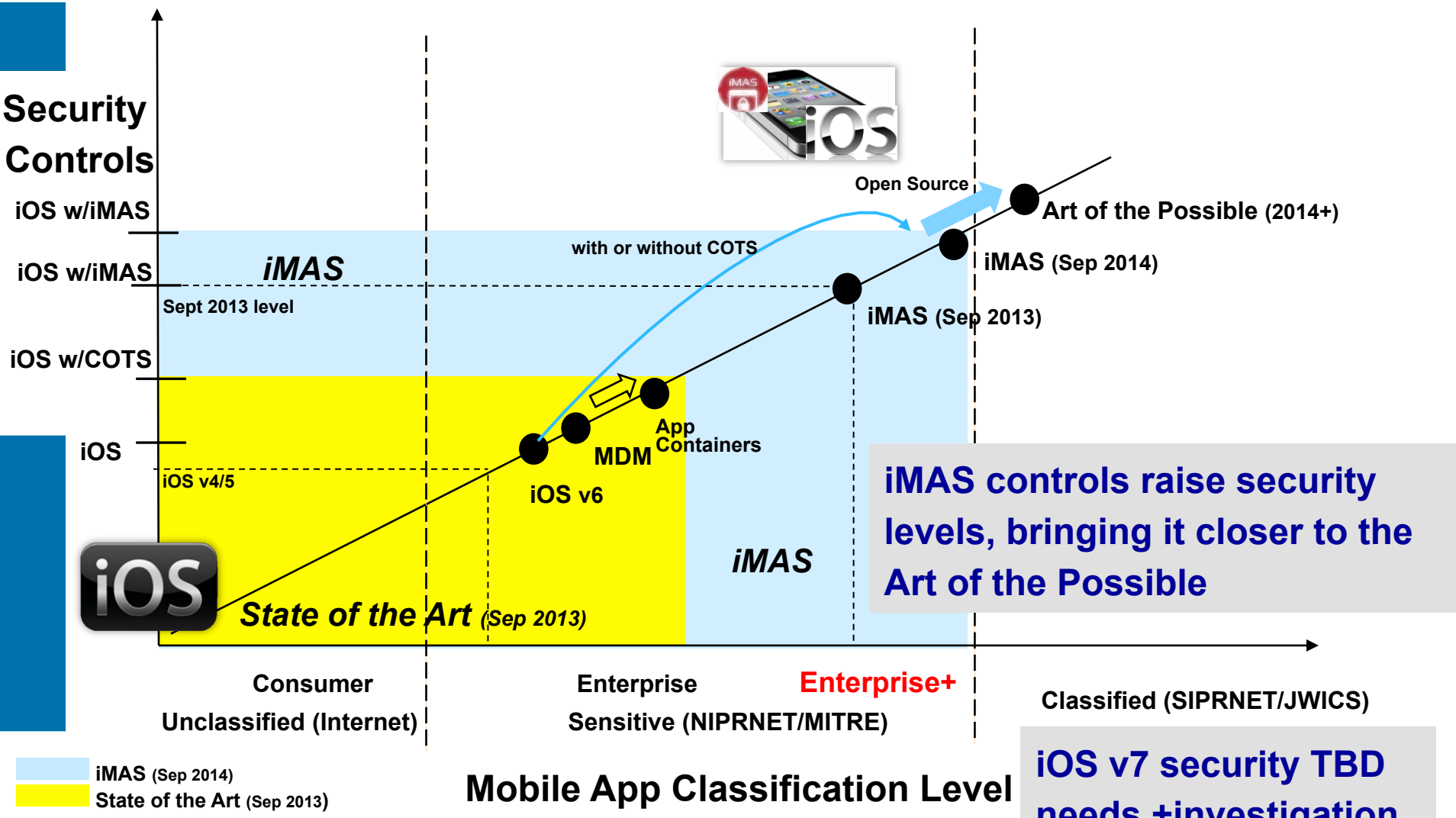
50% (12) iMAS Applicable

Approved for Public Release: Case #13-4177

MITRE

iMAS App Security "trade-space" Comparison

Sep 2013



iMAS controls raise security levels, bringing it closer to the Art of the Possible

iOS v7 security TBD needs +investigation

Approved for Public Release: Case #13-4177

MITRE

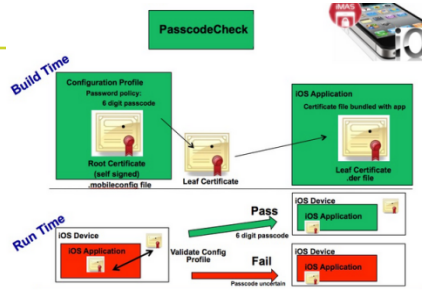
iMAS Security Controls

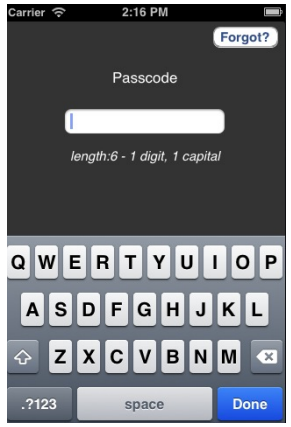
Approved for Public Release: Case #13-4177

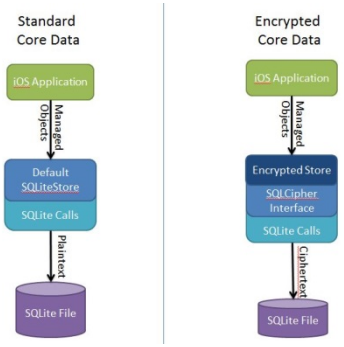
Security Controls

Breakout

- **Passcode Check**
  - Programmatic library and method to enforce complex device passcode use
  - When in place, extends jailbreak bruteforce 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





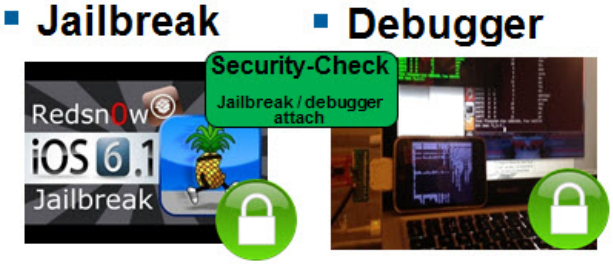


Approved for Public Release: Case #13-4177

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

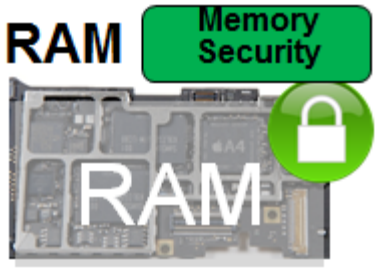
- **iOS OpenSSL FIPS**
  - FIPS compliant openSSL drop-in ready for iOS

Approved for Public Release: Case #13-4177

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



Approved for Public Release: Case #13-4177

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 ★
- Dynamic App Bundling ★

AppStore / Malware:

- App Tampering
- Forced-inlining
- AppIntegrity 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

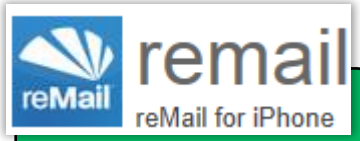
MITRE

iMAS Availability

- **Available Today on github**

AppPassword | Passcode Check | Security-Check (Jailbreak / debugger attach) | Encrypted Core Data

Secure Foundation | Forced-inlining | Memory Security



iMAS | MDM Remote Control

- **Jan - Mar 2014: Active Research**

AppIntegrity Check | Encrypted Code Modules (ECM) | Secure MDM Remote Control

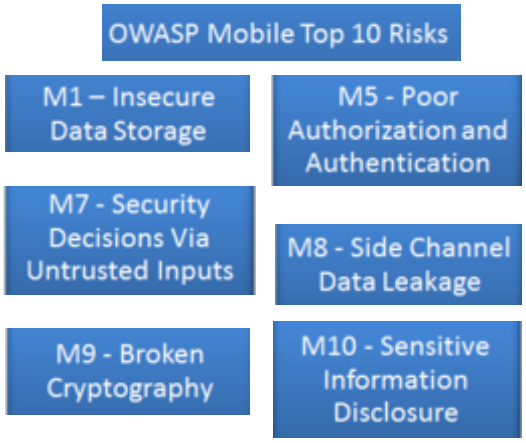
- **Future**

Encrypted RAM Disk | Dynamic App Bundling | Off-device Trust Check

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)



Approved for Public Release: Case #13-4177
MITRE

iMAS Deep Dive

Approved for Public Release: Case #13-4177

Security Check Deep Dive

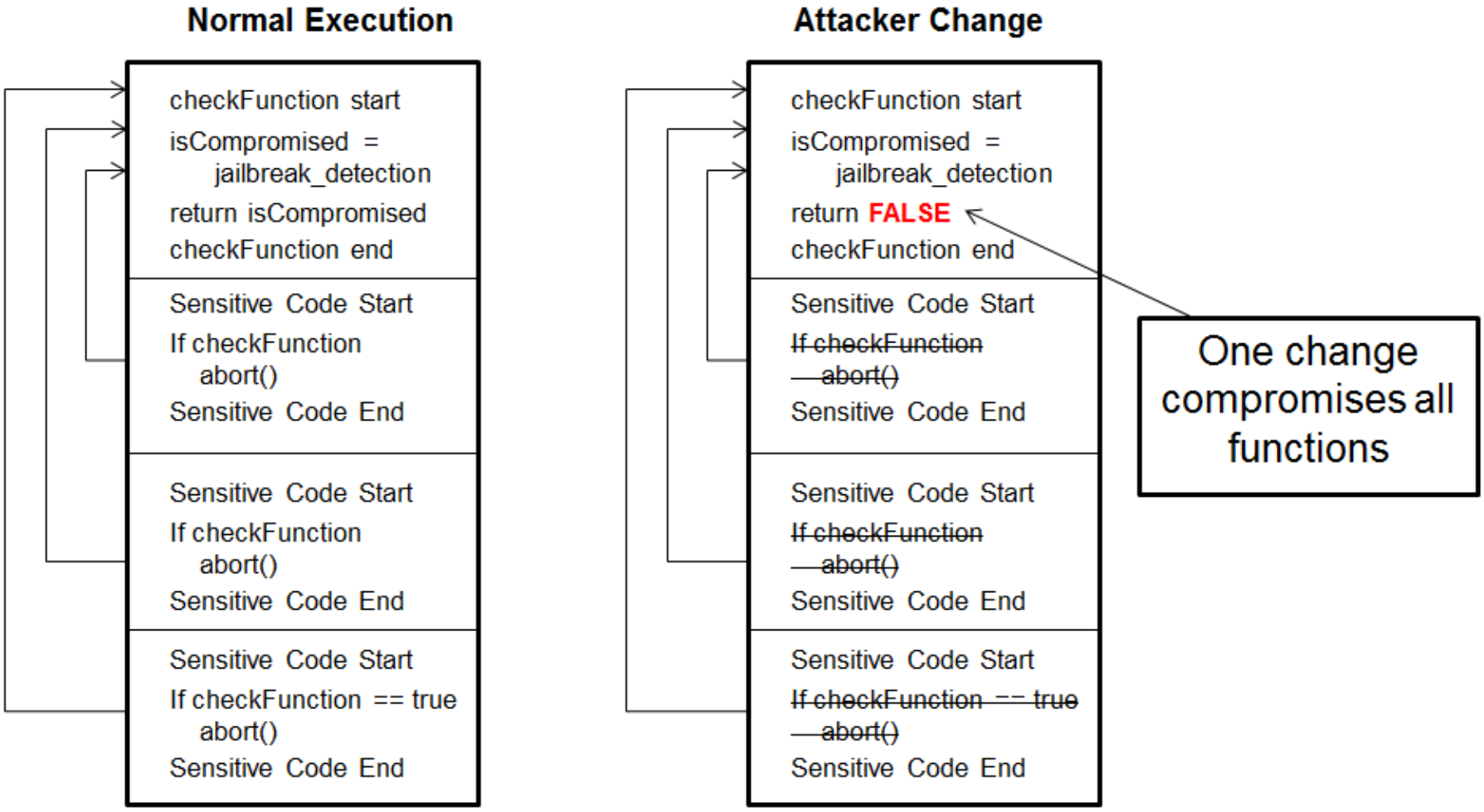
- **Jailbreak and debug attach detection**
- **Two Interesting iMAS techniques**
  1. Sysctl
     - Using sysctl call to detect if debugger is attached
     - Forensically attacker can “catch” this and simply override to return false
     - Instead, iMAS is making direct assembly call to read process 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-r
    add         r7, sp,#12          // Adjust FP to point
    push        {r8,r10,r11,r14}    // Save any general re
    vstmdb      sp!, {d8-d15}       // Save any VFP or NEO
    sub         sp, sp,#4           // Allocate space for

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

    mov     r12, #202
    svc     #128

    // ---------------
    // epilog
    // ---------------
    add         sp, sp,#4           // Deallocate space fo
    vldmia      sp!, {d8-d15}       // Restore any VFP or
    pop         {r8,r10,r11,r14}        // Restore any ger
    pop         {r4,r5,r6, r7,pc}       // Restore saved r

END_FUNCTION
```

Forced Inlining

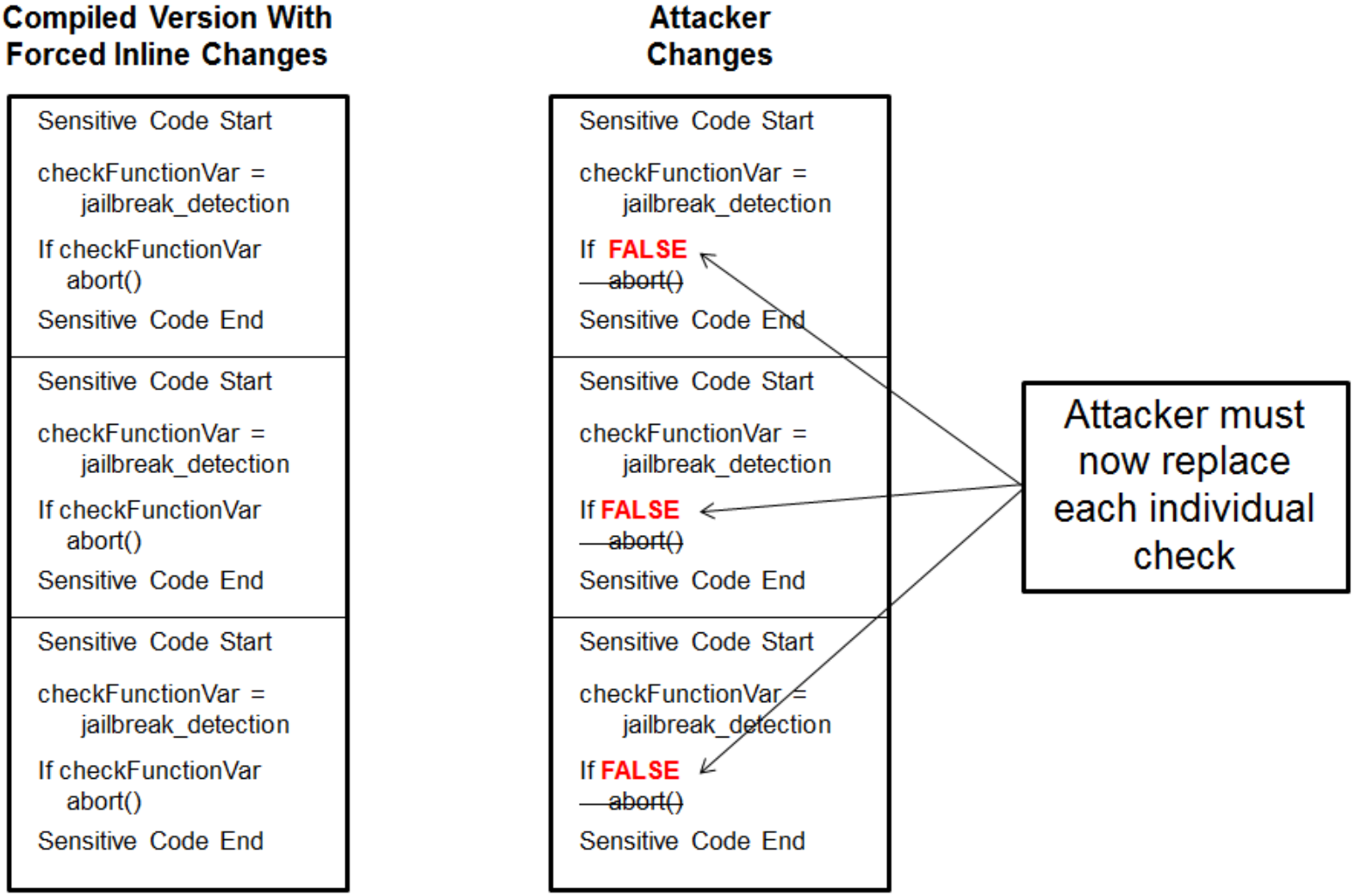
Compromising Security Checks



Approved for Public Release: Case #13-4177

Forced Inlining

Inlining Compiled in



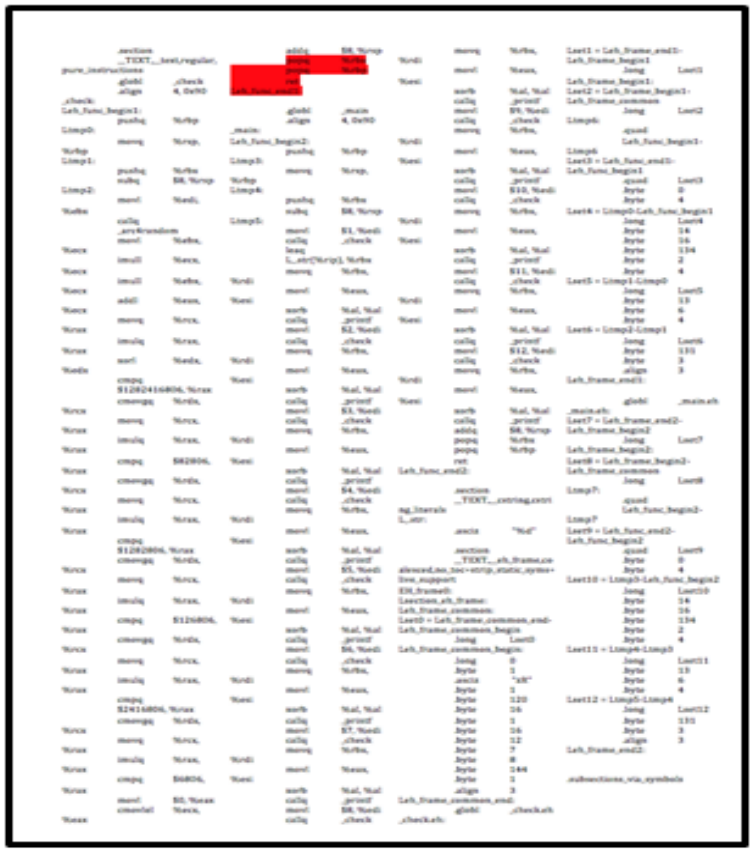
Approved for Public Release: Case #13-4177

Forced Inlining

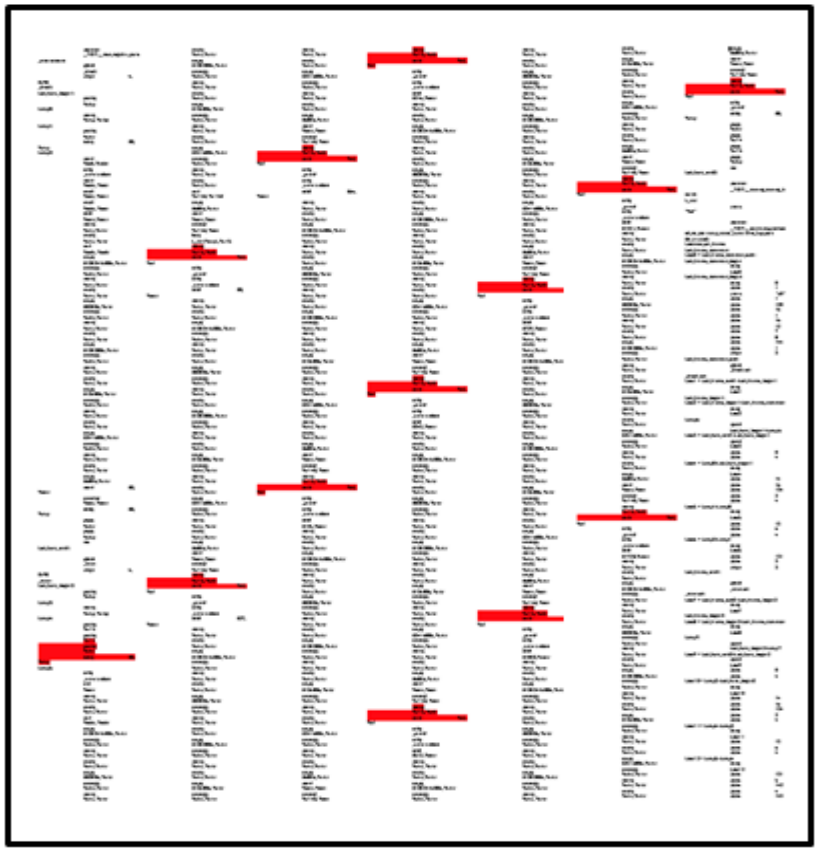
Increases Exploit Time

Assembly Code Comparison (Red indicates security check flag)

Without Forced Inlining
(Small haystack, one needle)
(232 instructions, 1 patch needed)



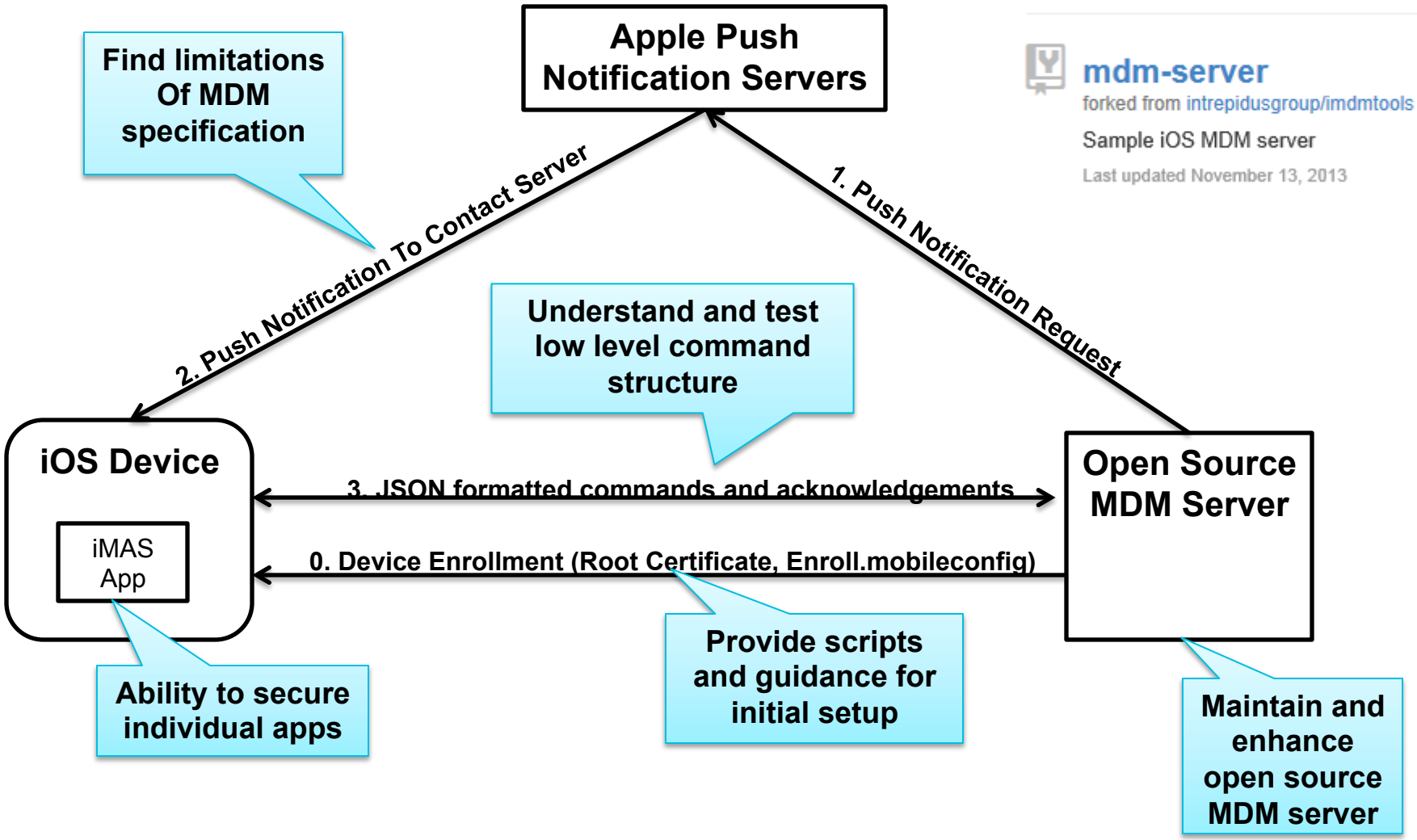
Same Program with Forced Inlining
(Large haystack, multiple needles)
(530 instructions, 13 patches needed)



Approved for Public Release: Case #13-4177

MITRE

Mobile Device Management (MDM) Research



mdm-server
forked from intrepidusgroup/imdmtools
Sample iOS MDM server
Last updated November 13, 2013

Approved for Public Release: Case #13-4177

iMAS Applied

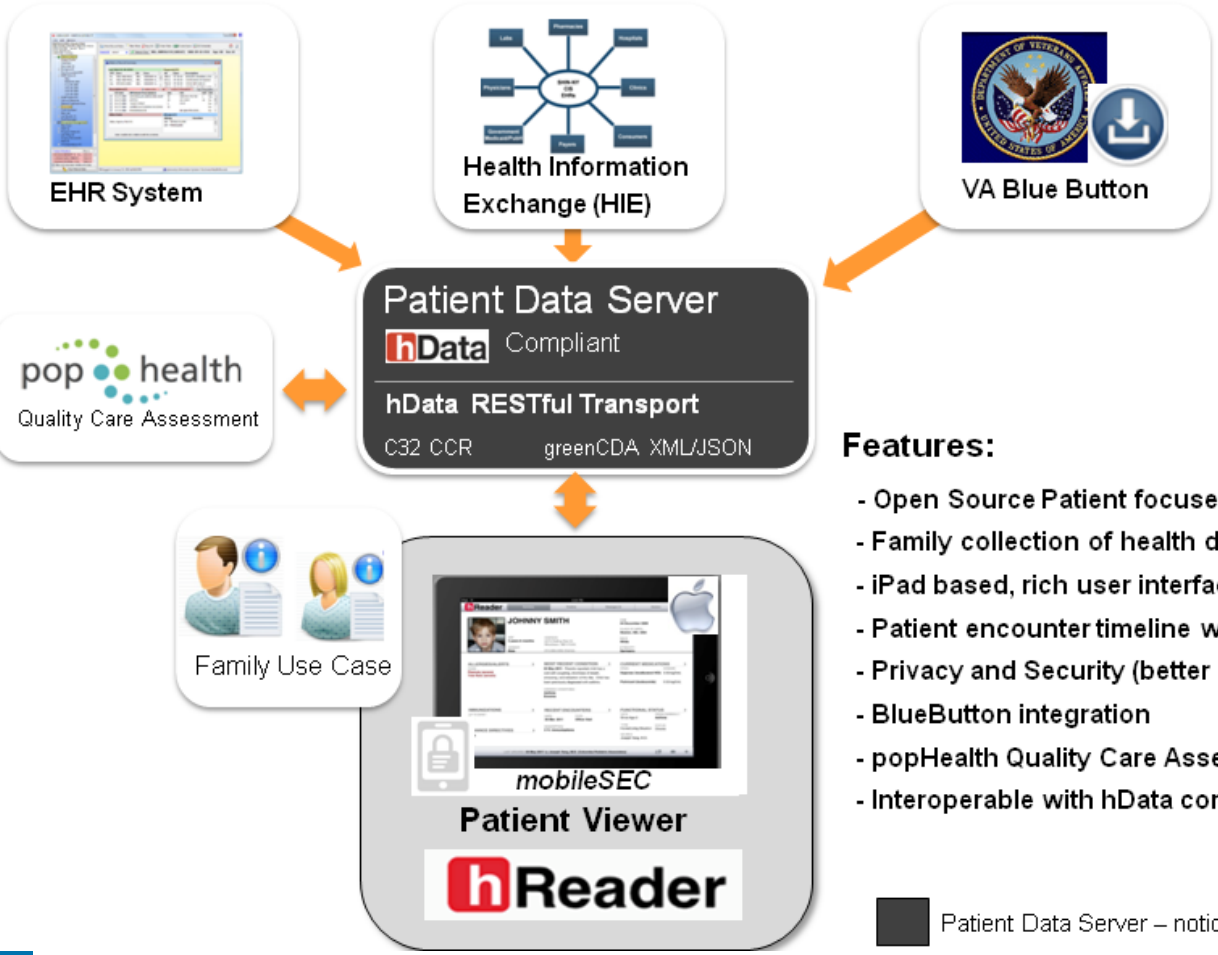
Approved for Public Release: Case #13-4177

FY12

MITRE Internal Research - MIP

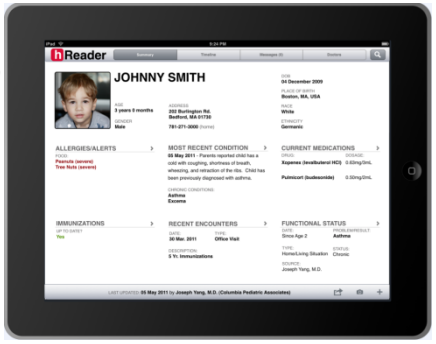
hReader

mobile Patient Health Reader



Features:

- Open Source Patient focused health viewer
- Family collection of health data
- iPad based, rich user interface
- Patient encounter timeline with notes
- Privacy and Security (better than browsers)
- BlueButton integration
- popHealth Quality Care Assessment
- Interoperable with hData compliant server



Patient Data Server – notional, not yet built

Approved for Public Release: Case #13-4177

MITRE

Audit Summary

August 2012

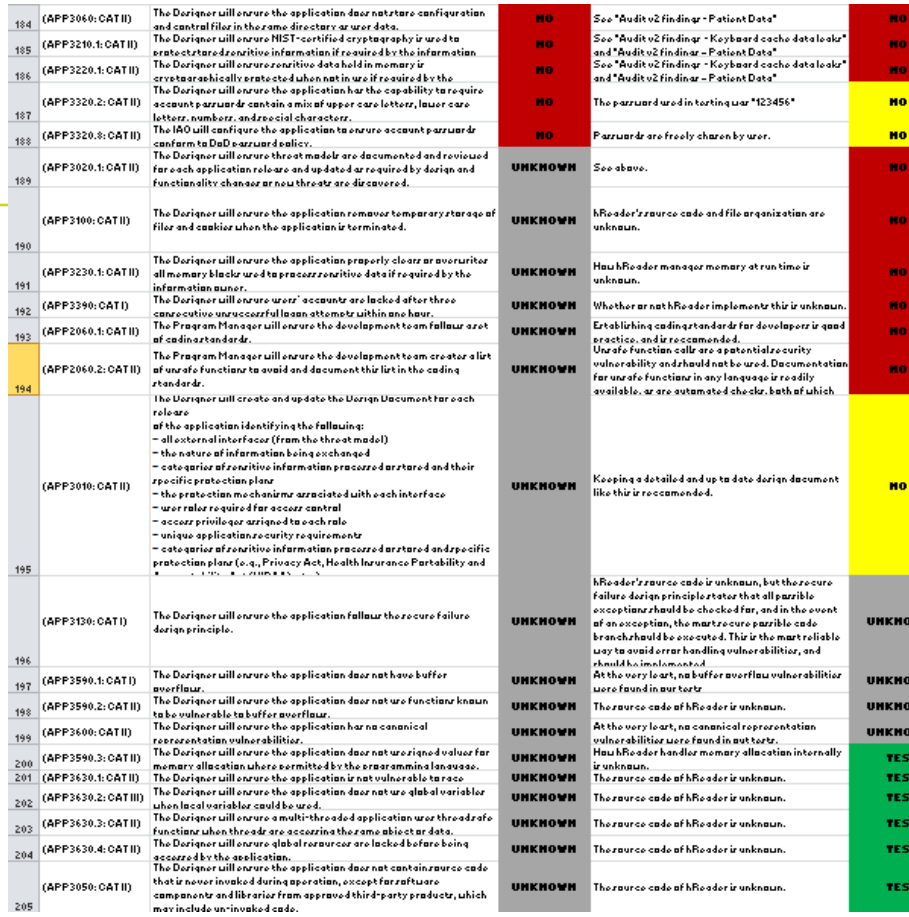
- **hReader team targeting Open Source space for hReader application**
  - hReader would be used mostly offline
  - Application security needs to be bolstered
  - Production hReader app would be contain sensitive patient health information
- **hReader requested a red-team to security audit the application**
  - 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

Approved for Public Release: Case #13-4177

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: CAT II)	The Designer will ensure the application does not store configuration and control files in the same directory as user data.	NO	See "Audit v2 findings - Patient Data"	NO
185	(APP3210.1: CAT II)	The Designer will ensure NIST-certified cryptography is used to protect stored sensitive information if required by the information	NO	See "Audit v2 findings - Keyboard cache data leaks" and "Audit v2 findings – Patient Data"	NO
186	(APP3220.1: CAT II)	The Designer will ensure sensitive data held in memory is cryptographically protected when not in use if required by the	NO	See "Audit v2 findings - Keyboard cache data leaks" and "Audit v2 findings – Patient Data"	NO
187	(APP3320.2: CAT II)	The Designer will ensure the application has the capability to require account passwords contain a mix of upper case letters, lower case letters, numbers, and special characters.	NO	The password used in testing was "123456"	NO
188	(APP3320.8: CAT II)	The IAO will configure the application to ensure account passwords conform to DoD password policy.	NO	Passwords are freely chosen by user.	NO
189	(APP3020.1: CAT II)	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: CAT II)	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: CAT II)	The Designer will ensure the application properly clears or overwrites all memory blocks used to process sensitive data if required by the information owner.	UNKNOWN	How hReader manages memory at run time is unknown.	NO
192	(APP3390: CAT I)	The Designer will ensure users' accounts are locked after three consecutive unsuccessful logon attempts within one hour.	UNKNOWN	Whether or not hReader implements this is unknown.	NO
193	(APP2060.1: CAT II)	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: CAT II)	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	Unsafe function calls are a potential security vulnerability and should not be used. Documentation for unsafe functions in any language is readily available, as are automated checkers, both of which	NO
195	(APP3010: CAT II)	The Designer will create and update the Design Document for each release 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 plans – the protection mechanisms associated with each interface – user roles required for access control – access privileges assigned to each role – unique application security requirements – categories of sensitive information processed or stored and specific protection plans (e.g., Privacy Act, Health Insurance Portability and	UNKNOWN	Keeping a detailed and up to date design document like this is recommended.	NO
196	(APP3130: CAT I)	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 secure possible code branch should be executed. This is the most reliable way to avoid error handling vulnerabilities, and should be implemented	UNKNO
197	(APP3590.1: CAT I)	The Designer will ensure the application does not have buffer overflows.	UNKNOWN	At the very least, no buffer overflow vulnerabilities were found in our tests	UNKNO
198	(APP3590.2: CAT II)	The Designer will ensure the application does not use functions known to be vulnerable to buffer overflows.	UNKNOWN	The source code of hReader is unknown.	UNKNO
199	(APP3600: CAT II)	The Designer will ensure the application has no canonical representation vulnerabilities.	UNKNOWN	At the very least, no canonical representation vulnerabilities were found in our tests.	UNKNO
200	(APP3590.3: CAT II)	The Designer will ensure the application does not use signed values for memory allocation where permitted by the programming language.	UNKNOWN	How hReader handles memory allocation internally is unknown.	YES
201	(APP3630.1: CAT II)	The Designer will ensure the application is not vulnerable to race	UNKNOWN	The source code of hReader is unknown.	YES
202	(APP3630.2: CAT III)	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.	YES
203	(APP3630.3: CAT II)	The Designer will ensure a multi-threaded application uses thread-safe functions when threads are accessing the same object or data.	UNKNOWN	The source code of hReader is unknown.	YES
204	(APP3630.4: CAT II)	The Designer will ensure global resources are locked before being accessed by the application.	UNKNOWN	The source code of hReader is unknown.	YES
205	(APP3050: CAT II)	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-invoked code.	UNKNOWN	The source code of hReader is unknown.	YES

- **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**

Approved for Public Release: Case #13-4177
MITRE

VIAFORENSICS
advancing mobile security

1000 Lake Street, Suite 203
Oak Park, IL 60301
Tel: +1 312-878-1100

Third Party Audit

Mitre (hReader)

appSecure – iOS 5/6

Mobile Forensic Security Assessment
Report of Findings

May 6, 2013

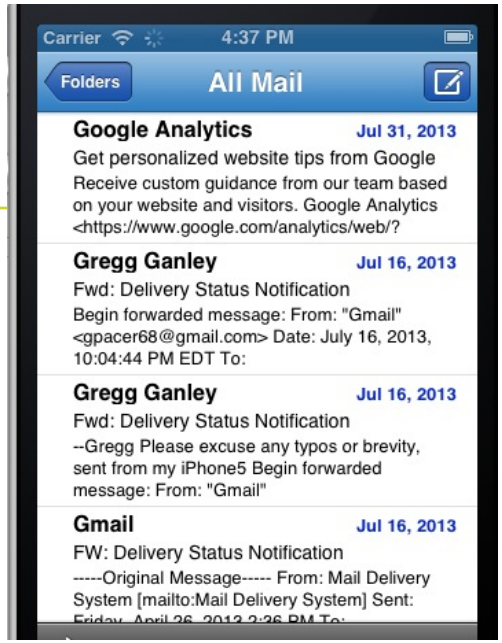
- **Report completed on May 6, 2013**
- **Failed Network security testing**
  - High risk found: network cache.db file had server password and patient information present
  - Related to backend server access
- **Other medium and low security risks found**
  - SQLlite DB key was found in memory
  - Jailbreak 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**

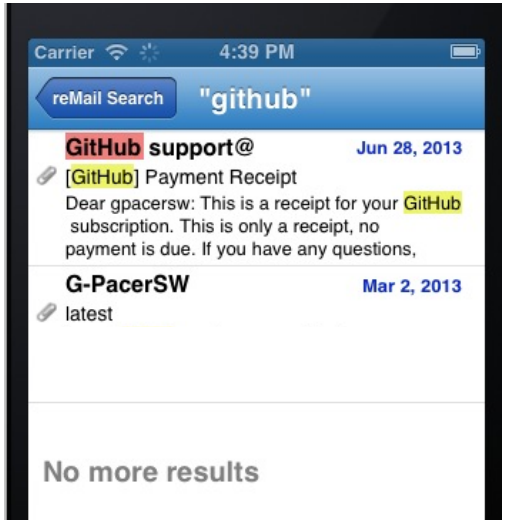
- **PASSED "FINAL EXAM"**

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

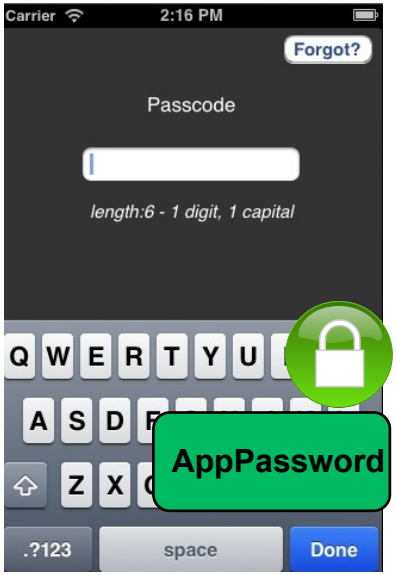




reMail – iMAS Security Integration

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

- **Access**
  - User resettable



- **Data Storage**
  - openSSL crypto
  - Keychain

Secure Foundation
OpenSSL / FIPS

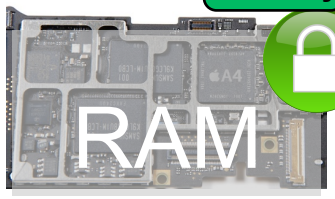
- **Jailbreak**

- **Debugger**

Security-Check
Jailbreak / debugger attach

- **RAM**

Memory Security



- **Database**

SQLCipher

SQLite

MITRE

Outreach

iMAS Now an OWASP Project

Approved for Public Release: Case #13-4177

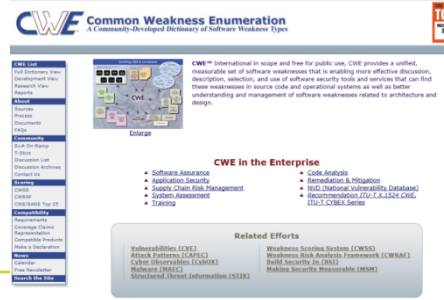
iMAS listed as OWASP Mobile Security Project – Mobile Tool

Web Site

project-imas.github.com

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 public repos | **2** private repos | **3** members

encrypted-core-data

Objective-C ★175 ⑂35

iOS Core Data encrypted SQLite store using SQLCipher

app-password

Objective-C ★76 ⑂6

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

securefoundation

Objective-C ★22 ⑂6

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

security-check

C ★27 ⑂5

Application level, attached debug detect and jailbreak checking

memory-security

Objective-C ★1 ⑂0

Tools for securely clearing and validating iOS application memory

passcode-check

Objective-C ★14 ⑂3

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

project-imas/single-sign-on

Objective-C ★0 ⑂0

Simple MDM Single Sign On solution for application level logins

forced-inlining

Assembly ★0 ⑂0

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

project-imas/mdm-server

Python ★1 ⑂29

forked from intrepidusgroup/imdmtools

Sample iOS MDM server

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/Remail-iPhone	340	0.90%

MITRE

iMAS Video

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

iMAS in the news …

COMPUTERWORLD

Print Article Close Window

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

Approved for Public Release: Case #13-4177

FY14 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
- AppIntegrity 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***

Approved for Public Release: Case #13-4177

MITRE

Backup

Approved for Public Release: Case #

iMAS
iOS Mobile Application Security

Gregg Ganley Gavin Black

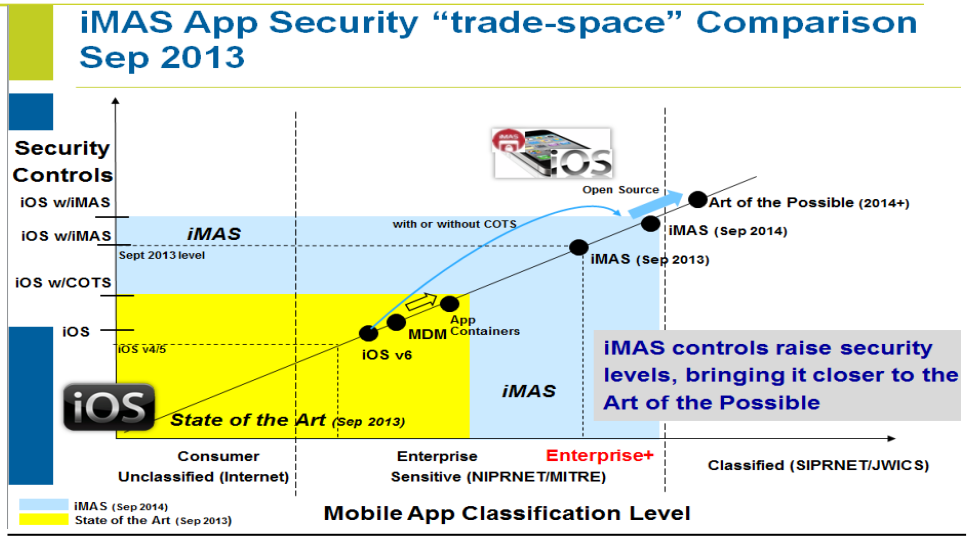
Problem

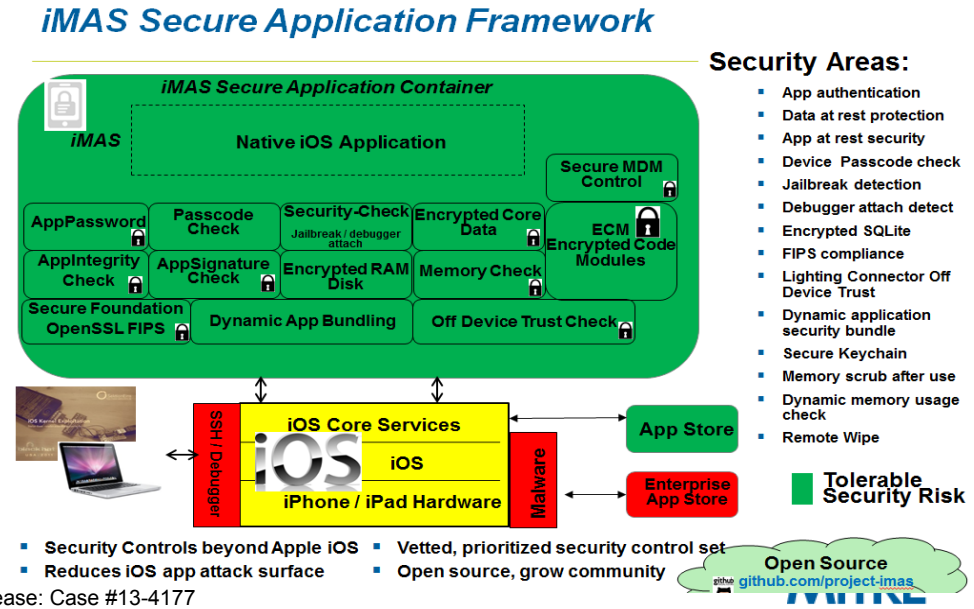
- iOS is considered secure, but out of the box security is not enough
- Simple device passcodes enable easy compromise of applications and data

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

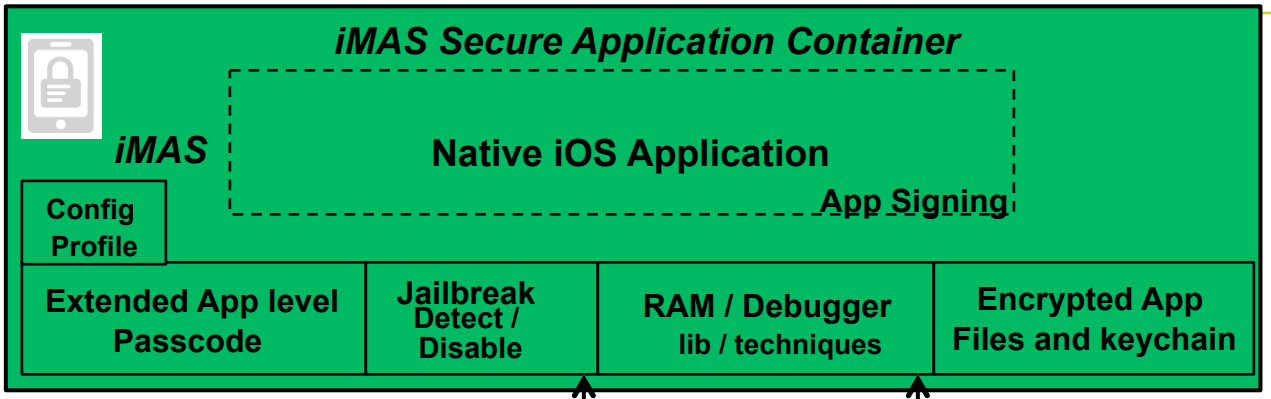
  project-imas.github.com
- Raise iOS app security levels - closer to the Art of the Possible





Approved for Public Release: Case #13-4177

iMAS Secure Application Framework FY13

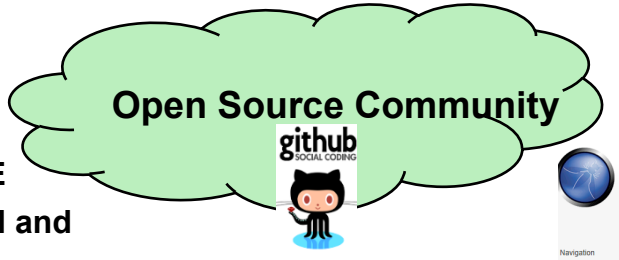


App Store

Internet

Tolerable Security Risk

- 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



CWE

Approved for Public Release: Case #13-4177

MITRE

iMAS Future

Approved for Public Release: Case #13-4177

FY14 Technical Approach and Research

Deeper security controls

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

Advanced resilience and disruption techniques

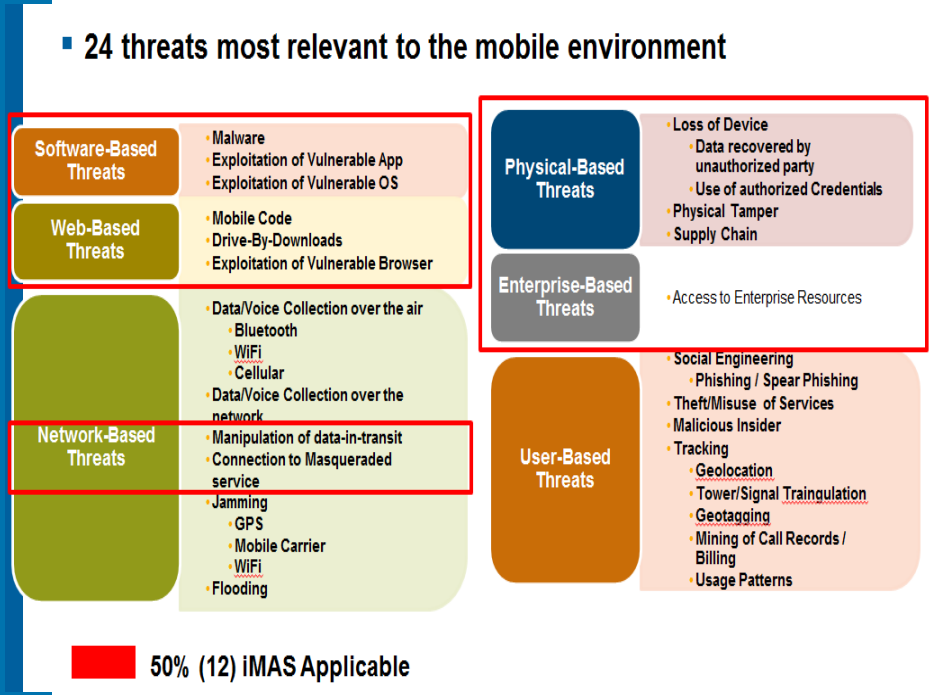
- **MDM remote control**
  - Explore securing MDM interface and remote C2 capabilities for iMAS controls (remote wipe etc.)
- **iMAS Encrypted Code Modules research**
  - Implement portions of security controls using iMAS ECM technique
- **Off device trust**
  - iOS Lightning Connector and trusted smart 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

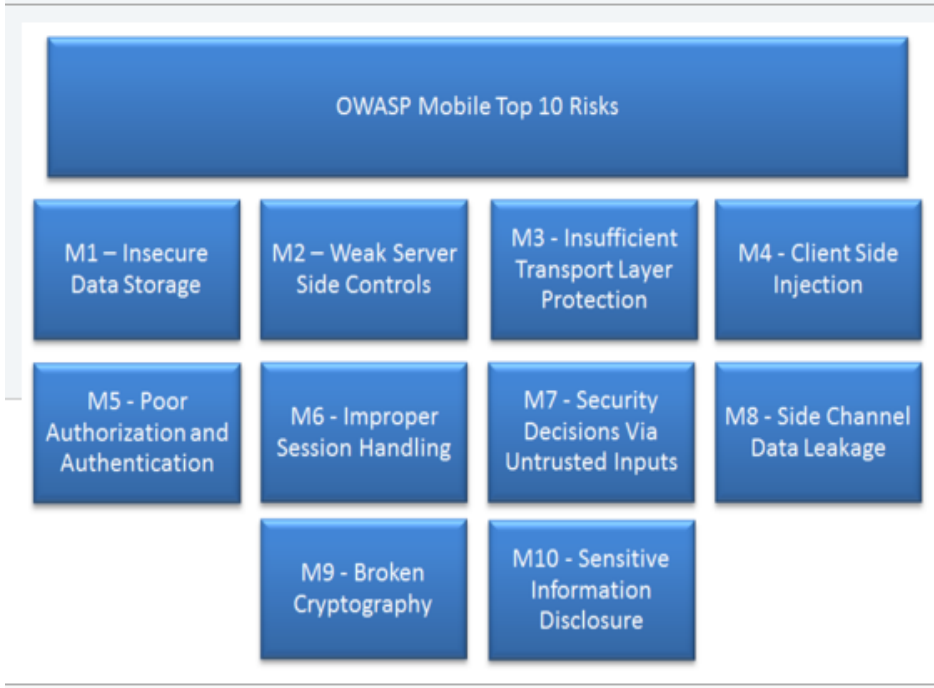
Approved for Public Release: Case #13-4177

End Game

iMAS Goals

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





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

Approved for Public Release: Case #13-4177