Unwrapping the Truth:
Analysis of Mobile App Wrapping

Ron Gutierrez
Outline

Major BYOD Risks & Threat Scenarios

MDM vs MAM Application Wrapping

MAM Solution Test Cases

Vulnerability Patterns in MAM Solutions

Conclusions and Testing Checklist
About Our Research

• Current State of MAM BYOD Solutions
  – Cutting Edge, Emerging Technology

• Based on GDS’ 2013 AppSecUSA Research on “Secure Containers”

• Goal is to share Common Vulnerability Patterns & Considerations

• Vendor Agnostic
Outline

- Major BYOD Risks & Threat Scenarios
- MDM vs MAM Application Wrapping
- MAM Solution Test Cases
- Vulnerability Patterns in MAM Solutions
- Conclusions and Testing Checklist
Major BYOD Risks & Threat Scenarios

• Lost or Stolen Device
• Stolen Device Backup Data
• Disgruntled Former Employees
• Malware / Malicious Apps
• Unattended Device
• Bypassing Client Restrictions
• Malicious User on Network
• Targeted Attacks Against Organization Endpoint
Major BYOD Risks & Threat Scenarios

- Lost or Stolen Device
- Stolen Device Backup Data
- Disgruntled Former Employees
- Malware / Malicious Apps
- Unattended Device
- Bypassing Client Restrictions
- Malicious User on Network
- Targeted Attacks Against Organization Endpoint

Out of Scope For This Talk
BYOD Goal – Protect The Data

• It's Easy To Say – Don’t Store Sensitive Data
  – In Real Life. That’s Not Going To Fly

• Primarily Two Approaches
  – Mobile Device Management (MDM)
  – Mobile Application Management (MAM)
  – Sometimes A Hybrid Of Both

• Many BYOD Vendors
Who Are The Major BYOD Players?

“Leaders” According to “Magic Quadrant for Enterprise Mobility Management Suites 2014” (Gartner)

http://www.gartner.com/technology/reprints.do?id=1-1UURNKA&ct=140603
Mobile Device Management (MDM)

- Device Enrolls to MDM Server
- Allows MDM Server to
  - Set Device Level Policies
  - Push Security Commands (Wipe, Locks, etc)
  - Query Information (Device Info, Installed Apps, etc)
  - Install Applications
- There is Existing Research On This Topic
  - MDM Research from David Shuetz (Intrepidus Group)
  - NTT Security Presented Yesterday

# MDM Feature Breakdown

<table>
<thead>
<tr>
<th>Category</th>
<th>Mobile Device Management (MDM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Commands</td>
<td>Quickly pushed and invoked by the OS</td>
</tr>
<tr>
<td>App User Experience</td>
<td>Organization data can be accessed using native OS applications (Mail, Contacts, Calendar, etc).</td>
</tr>
<tr>
<td>Device User Experience</td>
<td>Strict device level policies may impede user’s personal device experience</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>Device level policies ensure usage of system wide data protection (DP) capabilities. However, DP implementation may be opt-in for apps</td>
</tr>
<tr>
<td>Device Privacy</td>
<td>MDM server can query potentially personal data from employee devices</td>
</tr>
<tr>
<td>Other Limitations</td>
<td>Relies available MDM APIs on the OS</td>
</tr>
</tbody>
</table>
# MDM Feature Breakdown

<table>
<thead>
<tr>
<th>Category</th>
<th>Mobile Device Management (MDM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Commands</td>
<td>Quickly pushed and invoked by the OS</td>
</tr>
<tr>
<td>App User Experience</td>
<td>Organization data can be accessed using native OS applications (Mail, Contacts, Calendar, etc).</td>
</tr>
<tr>
<td>Device User Experience</td>
<td>Strict device level policies may impede user’s personal device experience</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>Device level policies ensure usage of system wide data protection (DP) capabilities. However, DP implementation may be opt-in for apps</td>
</tr>
<tr>
<td>Device Privacy</td>
<td>MDM Management server can query potentially sensitive information from employee devices</td>
</tr>
<tr>
<td>Other Limitations</td>
<td>Relies on support OS level MDM APIs exposed</td>
</tr>
</tbody>
</table>
MDM Drawbacks

- Strict Policies Ruin Personal Device Experience
- Implementation is OS Dependent
- Privacy Concerns
  - Device Wipes
  - Querying of Installed Applications
- Data Protection Dependent on Application
  - Opt-in Data Protection APIs
Mobile Application Management (MAM)

- Policy Enforcement & Data Protection At App Layer
- Requires Development of “Secure Containers”
- Application Wrapping Used To “Secure” Org Apps

Mobile Application Management (MAM)
Provision and manage apps on mobile devices to optimize employees' work while making sure company data is secured

Quick Intro to Application Wrapping

- Custom Mail App
- Custom Org App
- Custom Calendar App
MAM Overview

1. Signed Custom Mobile App

2. Inject of Wrapping Code (Dynamic Libraries, DEX Modification, etc)

3. Resigned and Wrapped Custom Mobile App

4. Wrapped Application Installed On Employees Device through MAM Agent

MAM Web Server

Wrapping Utility

Resign Modified Application w/ Enterprise’s App Signing Key

Mobile Device

MAM Agent

Secure Container
iOS App Wrapping Analysis

Differing a pre-wrapped and post-wrapped iOS binary with HexFiend

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001AA0</td>
<td>00000000C</td>
<td>Command</td>
<td>LC_LOAD_DYLIB</td>
</tr>
<tr>
<td>0001AA4</td>
<td>00000050</td>
<td>Command Size</td>
<td>80</td>
</tr>
<tr>
<td>0001AA8</td>
<td>00000018</td>
<td>Str Offset</td>
<td>24</td>
</tr>
<tr>
<td>0001AAC</td>
<td>00000000</td>
<td>Time Stamp</td>
<td>Wed Dec 31 19:00:00 1969</td>
</tr>
<tr>
<td>0001AB0</td>
<td>00000001</td>
<td>Current Version</td>
<td>0.0.1</td>
</tr>
<tr>
<td>0001AB4</td>
<td>00000001</td>
<td>Compatibility Version</td>
<td>0.0.1</td>
</tr>
</tbody>
</table>
| 0001ABB | 4065706563757461625655... | Name | @executable_path/lib.dylib

A LC_LOAD_DYLIB is added to the App’s Mach-O Load Commands
iOS App Wrapping Analysis

Differing a pre-wrapped and post-wrapped iOS binary with HexFiend

View address offset with MachOView tool to see what was changed.

A LC_LOAD_DYLIB is added to the App's Mach-O Load Commands.
iOS App Wrapping Analysis

Differing a pre-wrapped and post-wrapped iOS binary with HexFiend

View address offset with MachOView tool to see what was changed.

A LC_LOAD_DYLIB is added to the App’s Mach-O Load Commands
iOS App Wrapping Analysis

Updates to the Code Signature of the Binary

GOTHAM DIGITAL SCIENCE
New URL Scheme App Entry Point Added to Info.plist
Android Wrapping Analysis

Additions of various NDK libraries

DEX Bytecode Modification
Android Wrapping Analysis

Common Android APIs Are Replaced Throughout App (ASMDEX Library)
Added Content Providers, Services, Activities and Broadcast Receivers

```xml
<service android:name="com.MAM.Android.ManagedApp.TMManager" android:exported="true" />
<service android:name="com.MAM.Android.ManagedApp.TMService" android:process=":mitm" />
<receiver android:name="com.MAM.Android.ManagedApp.PackageReceiver">
  <intent-filter>
    <action android:name="android.intent.action_PACKAGE_REMOVED" />
    <data android:scheme="package" />
  </intent-filter>
</receiver>
<activity android:name="com.MAM.Android.ManagedApp.Locked" android:enabled="true" />
```
MAM Overview

Mobile Device

Native Mail
Native Calendar
Native Browser

MAM Agent
Secure Container

Wrapped Mail App
Secure Container

Inter Process Communication (IPC)

Authentication, Application Policies, Security Commands, Client Certs, etc

MAM Web Server
MAM Security Checks

✓ Allows Employees Keep Device Policies As They Like
✓ Less Privacy Issues
✓ Secure Container Does Not Rely On OS DP Support

- Custom Crypto Implementations
- Custom IPC Implementations
- Wrapped App Experience May Not Be As Good
- Security Commands May Not Be Invoked Immediately
## MAM Feature Breakdown

<table>
<thead>
<tr>
<th>Category</th>
<th>Mobile Device Management (MDM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Commands</td>
<td>Likely not pushed to the device. Implementations may vary across vendors. OS limitations may prevent commands to be pushed and invoked immediately.</td>
</tr>
<tr>
<td>App User Experience</td>
<td>Third party apps used to access organization data. May not provide as good a user experience as the bundled native OS applications.</td>
</tr>
<tr>
<td>Device User Experience</td>
<td>Allows employees keep the device level policies as they choose</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>Custom crypto implementations may lead to security issues</td>
</tr>
<tr>
<td>Device Privacy</td>
<td>MAM Management server can only query data accessible by normal mobile applications on the OS</td>
</tr>
<tr>
<td>Other Limitations</td>
<td>Heavy reliance on IPC between wrapped apps in order to push policies and security commands to wrapped applications</td>
</tr>
</tbody>
</table>
## MAM Feature Breakdown

<table>
<thead>
<tr>
<th>Category</th>
<th>Mobile Device Management (MDM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Commands</td>
<td>Likely not pushed to the device. Implementations may vary across vendors. OS limitations may prevent commands to be pushed and invoked immediately.</td>
</tr>
<tr>
<td>App User Experience</td>
<td>Third party apps used to access organization data. May not provide as good a user experience as the bundled native OS applications.</td>
</tr>
<tr>
<td>Device User Experience</td>
<td>Allows employees keep the device level policies as they choose</td>
</tr>
<tr>
<td>Data Encryption</td>
<td>Custom crypto implementations may lead to security issues</td>
</tr>
<tr>
<td>Device Privacy</td>
<td>MAM Management server can only query data accessible by normal mobile applications on the OS</td>
</tr>
<tr>
<td>Other Limitations</td>
<td>Heavy reliance on IPC between wrapped apps in order to push policies and security commands to wrapped applications</td>
</tr>
</tbody>
</table>
Outline

- Major BYOD Risks & Threat Scenarios
- MDM vs MAM Application Wrapping
- MAM Solution Test Cases
- Vulnerability Patterns in MAM Solutions
- Conclusions and Testing Checklist
MAM Security Checks

- MAM Secure Container Authentication
- MAM Secure Container Cryptography
- Completeness of MAM Secure Container
- Inter Process Communication (IPC)
- Effectiveness of Security Commands
- Policy Configuration Features
Outline

Major BYOD Risks & Threat Scenarios
MDM vs MAM Application Wrapping
MAM Solution Test Cases
Vulnerability Patterns in MAM Solutions
Conclusions and Testing Checklist
MAM CONTAINER AUTHENTICATION
Principles To Live By

1. All data stored by app must be encrypted seamlessly.

2. Strength of crypto cannot rely on any device policies.

3. Crypto keys must be retrieved upon successful authentication.
Vulnerability Pattern #1

- After reverse engineering the key derivation process
  - All Key Material Stored on Device
  - Offline Authentication is Only Client Logic

**Violated Principles**

2. The strength of the cryptography cannot rely on any device policies

3. The cryptographic keys protecting app data must not be available pre-authentication

Might as well start encrypting with **ROT13+1**

@YOLOCrypto approved algorithm
Acquire Employee’s iDevice

Jailbreak the Device

Reverse Engineer the Key Derivation Logic

Use Runtime Hacking to Modify the Passcode Verification Logic To Always Return True

Use Runtime Injection and Leverage the Application Wrapping In Order To Decrypt At Will

Reverse Engineer Decryption Logic and Decrypt Organization Data

Decrypted Organization Data
Vulnerability Pattern #1

Test Application That Will Be Wrapped

```objective-c
//make a file name to write the data to using the documents directory:
NSString *fileName = [NSString stringWithFormat:@"%@/writeToFileTest.txt",
documentsDirectory];

//create content - four lines of text
NSString *content = @"This is just some plaintext data";

//save content to the documents directory
[content writeToFile:fileName
atomically:NO
encoding:NSUTF8StringEncoding
allowLossy:NO
error:nil];
```
Vulnerability Pattern #1

Confirming The File Is No Longer Plaintext
Vulnerability Pattern #1

- Use Cycript in order to hook running wrapped app
- Use iOS File APIs in order to read arbitrary file
- Since app is wrapped, file decryption happens seamlessly

```swift
var filename = @"Documents/writeToFileTest.txt";

var bundlePath = [[[NSBundle mainBundle] bundlePath]
    stringByDeletingLastPathComponent];

var fullPath = [NSString stringWithFormat:@"%@/%@
    ,
    bundlePath, filename];

var fh = [NSFileHandle fileHandleForReadingAtPath:fullPath];

var inputbuf = [fh readDataToEndOfFile];

contentsStr = [[NSString alloc] initWithData:inputbuf encoding:NSUTF8StringEncoding];
```

Proof of Concept Cycript Script
Vulnerability Pattern #1

Application Enters Unauthenticated State

Doesn’t Matter… Still Owned
Vulnerability Pattern #2

- Decoupling of the Passcode Verification & Key Derivation
- Key Derivation uses PBKDF2
- Offline Passcode Derivation uses Unsalted SHA-256
Vulnerability Pattern #2

1. Acquire Employee’s iDevice
2. Retrieve Passcode Hash
3. Brute Force Passcode
   - Device Not Revoked
     - Authenticate to MAM Agent or Wrapped Application
       - Access Internal Data
     - Device Revoked
6. Device Revoked
   - Authenticate to MAM Agent or Wrapped Application w/o network connection
     - Decrypted Offline Data
8. Reverse Engineer Decryption Logic
Vulnerability Pattern #2

Location of Shared Preferences File
/data/data/com.[Omitted]/shared_prefs/com.[Omitted].[Omitted]_preferences.xml

<string name="seedHash">NEHfC6vCot21UdfNOfsjW8TgnNHkVWvyYbtJGI9Ug0g=
</string>

Proof of Concept
>>> import base64
>>> import hashlib
>>> output = base64.b64encode(hashlib.sha256("testing1234").digest())
>>> print(output)
'NEHfC6vCot21UdfNOfsjW8TgnNHkVWvyYbtJGI9Ug0g='
DO YOU EVEN CRYPTO

BRO

MAM CONTAINER CRYPTOGRAPHY
Cryptography Implementation

- Cryptography is Hard
- I repeat, Cryptography is Hard
- Common to see OpenSSL as primary crypto library
  - FIPS Compliant (Nice little checkbox to have)
  - Not a Very High Level
  - High Potential For Implementation Flaws
- We Will Not Be Going In Depth In This Presentation
Vulnerability Pattern #3

- Master Key Stored As String Object
- Keep A Lookout Use Of SQLCipher for DB Encryption
  - Creds/Master Key Passed As String

```
SQLiteDatabase.openOrCreateDatabase(dbFile, "test123", null);
```

- What Is The Risk?
  - Key Data Might Persistent For Long Period Of Time
  - Charset Encoding May Reduce Entropy
private String e(String paramString1, String paramString2) {
  char[] arrayOfChar = paramString1.toCharArray();
  String str1 = this.d.getString("Vector", "");
  String str2;
  ..snip..
  byte[] arrayOfByte;
  while (true) {
    str2 = new String(Base64.decode(str1, 0));
    PBEKeySpec localPBEKeySpec =
      new PBEKeySpec(arrayOfChar,
                   a(paramString2, str2, "[Omitted]"), 20000, 256);
    arrayOfByte =
      SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1")
                    .generateSecret(localPBEKeySpec).getEncoded();
    if (arrayOfByte != null)
      break;
    return null;
  } 
  return new String(arrayOfByte);
}
Tracing Obfuscated Code

```
04-13 14:23:56.096: I/TheHook(6740): com.[Omitted].crypto.a.e(2 args)(a.e(String,String)) is hit
04-13 14:23:56.096: I/TheHook(6740): param1(PLAIN): testing1234
04-13 14:23:56.096: I/TheHook(6740): param2(PLAIN): 01c0d1e63656057ac6720eb688b988d1
..snip..
$MethodPointer.invoke(Native Method)
$MethodPointer.invoke(MS.java:58)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].crypto.a.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].g.c.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].u.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].u.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].ui.fragment.bt.doInBackground(Unknown Source)
..snip..
04-13 14:23:57.938: I/TheHook(6740): returnvalue(PLAIN) a.e(String,String): <I[G&\v\\b\\G\\h\\O3-[^v\\](Y8
04-13 14:23:57.938: I/TheHook(6740): returnvalue(HEX) a.e(String,String):
1B3C495B47EFBFBD3426EFBFBD76EFBFDEB8794EFBFDEFBFBD47EFBFDEBFBD68EFBFBD0E4F332D5B76EFBFDEBFBD28
5938
```
Tracing Obfuscated Code

04-13 14:23:56.096: I/TheHook(6740): com.[Omitted].crypto.a.e(2 args)(a.e(String,String)) is hit
04-13 14:23:56.096: I/TheHook(6740): param1(PLAIN): testing1234
04-13 14:23:56.096: I/TheHook(6740): param2(PLAIN): 01c0d1e63656057ac6720eb688b988d1
..snip..
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].crypto.a.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].u.a(Unknown Source)
04-13 14:23:57.938: I/TheHook(6740): returnvalue(PLAIN) a.e(String,String): \[G\&\{Y8
04-13 14:23:57.938: I/TheHook(6740): returnvalue(HEX) a.e(String,String):
1B3C495B47EFBFBD3426EFBFBD76EFBFDEBFBD47EFBFDEBFBD68EFBFBD0E4F332D5B76EFBFDEBFBD28
5938
Tracing Obfuscated Code

04-13 14:23:56.096: I/TheHook(6740): com.[Omitted].crypto.a.e(2 args)(a.e(String,String)) is hit
04-13 14:23:56.096: I/TheHook(6740): param1(PLAIN): testing1234
04-13 14:23:56.096: I/TheHook(6740): param2(PLAIN): 01c0d1e63656057ac6720eb688b988d1

..snip..

..snip..
04-13 14:23:56.116: I/TheHook(6740): u.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): ui.fragment.bt.doInBackground(Unknown Source)

..snip..
Tracing Obfuscated Code

04-13 14:23:56.096: I/TheHook(6740): com.[Omitted].crypto.a.e(2 args)(a.e(String,String)) is hit
04-13 14:23:56.096: I/TheHook(6740): param1(PLAIN): testing1234
04-13 14:23:56.096: I/TheHook(6740): param2(PLAIN): 01c0d1e63656057ac6720eb688b988d1

..snip..

$MethodPointer.invoke(Native Method)
$MethodPointer.invoke(MS.java:58)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].crypto.a.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].g.c.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].u.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].u.a(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740): at com.[Omitted].ui.fragment.bt.doInBackground(Unknown Source)

..snip..


returnValue(HEX) a.e(String,String):
1B3C495B47EFBFBD3426EFBFBD76EFBFDEBFBD8794EFBFDEBFBD47EFBFDEBFBD68EFBFBD0E4F332D5B76EFBFDEBFBD285938
Vulnerability Pattern #3

Return Value (Derived Symmetric Key):
1B3C495B47EFBFBD3426EFBFBD76EFBFBDEB8794EFBF
BDEFBFBD47EFBFBDEFBFBD68EFBFBD0E4F332D5B76EF
BFBD3EFBFBD285938
Vulnerability Pattern #3

• Default Charset in Android is UTF-8

• Symmetric Key Utilizes Full Byte Range [0-255]
  – Might Not Be Supported By UTF-8

• Invalid UTF-8 is Converted to EF BF BD (hex)
  – Unicode U+FFFD ‘REPLACEMENT CHARACTER’

• Entropy Loss Depends On Output of PBKDF2
  – In This Case Reduced to 22 Bytes from 32 Bytes
INCOMPLETE SECURE CONTAINER
Incomplete Secure Container

- Is Everything That Is Supposed To Be Encrypted, Actually Encrypted?

- Develop Test Harness Application
  - Open Up API Documentation and Start Coding!

- Lets Cover Some Of The Common Issues Observed
iOS Common Missed APIs

Identified in iOS MAM Solutions

- iOS Keychain
- NSUserDefaults
- iCloud APIs
- C/C++ APIs (e.g. fwrite)
- Data stored by WebViews
- Persistent HTTP Cookies
- HTTP(S) Request Caches
- Document Caching (Open-in)
- Filenames
Android Common Missed APIs

Identified in Android MAM Solutions

- NDK File system writes
- File system paths with symbolic links (e.g. /sdcard)
- Data stored by WebViews
- Runtime Execs, Reflection
- Filenames
INTER PROCESS COMMUNICATION (IPC)
Inter Process Communication

- MAM Relies Heavily on IPC
  - Between Agent and All Wrapped Apps

- Lots of Sensitive Data May Be Passed Around
  - Security Policies
  - Security Commands
  - Offline Authentication Data
  - Crypto Keys
iOS IPC Considerations

- **Keychain Access Groups**
  - Require Being Signed by Same Developer
  - Not Feasible for MAM Deployments

- **URL Schemes**
  - Authorization Based on Bundle IDS
  - Tricky but Somewhat Effective If Not Jailbroken
  - Bad User Experience (Application UI Switches)
  - Data Size Limitations
iOS IPC Considerations

- **UIPasteboard**
  - Most common form of IPC implementation for MAM
  - Allows Large Data To Be Passed
  - Better User Experience

- **Security Considerations**
  - Data can be read/modified by third party apps
  - Data must be encrypted to prevent unauthorized access
Android IPC Considerations

• Intents
  – Signature Based Authorization Controls in Manifest File

• Not Feasible for MAM Deployments
  – MAM Agent and Wrapped Apps Not Signed By Same Developer

• Programmatic Source App Validation
  – Binder.getCallingUid(), Binder.getCallingPid()
  – PackageManager Object Can Then Retrieve App Name
  – Agent Must Track App Installs/Uninstalls
Enforcing Authentication On IPC

Identify Entry Points Via AndroidManifest.xml

```xml
<activity android:name="com.[Omitted By GDS].ui.SearchActivity"
  android:launchMode="singleTop" android:windowSoftInputMode="adjustPan">
  <intent-filter>
    <action android:name="android.intent.action.SEARCH" />
  </intent-filter>
  <meta-data android:name="android.app.searchable" android:resource="@xml/searchable" />
</activity>
```

Invoke It To Confirm It Is Enforcing Authentication On IPC

```
rgutierrez@rav-2:~$ adb shell am start -a android.intent.action.SEARCH -n com.[Omitted By GDS].ui.SearchActivity -e "query" "t"
Starting: Intent { act=android.intent.action.SEARCH cmp=com.[Omitted By GDS].ui.SearchActivity (has extras) }
rgutierrez@rav-2:~$  
```
Enforcing Authentication On IPC

Access File Metadata Without Offline Authentication.
Relies on Metadata Not Being Encrypted
Effectiveness of Security Commands

• Commands Should Ideally Execute Immediately
  – Not Always What Happens in MAM..

• Wipes Should Delete ALL Data
  – Key Material, Encrypted Data, Passcode Validation Data

• Wipe Should Apply to Agent and Wrapped Apps
Conclusions

• Initial Research Uncovered Common Vulnerability Patterns in MAM Solutions

• Security Posture Has Matured Over The Past Year

• To Defend Against Evolving Threat Landscape and Mobile Attack Techniques, More Work Needed
MAM Testing Checklist

• MAM Solution Security Checklist

• Covers The Topics In The Presentation
  – And Many More!

• Over 50 Security Checks To Assess MAM Solutions
  – Organizations – Ask Your Vendors!
  – Vendors – Ask and Test Yourself!
  – Security Testers – Help These Vendors!

• We hope this checklist will create a security baseline for these solutions
Thanks For Coming!

• **Shouts outs**
  – Stephen Komal for helping with the research and paper
  – GDS Research Team (Joe Hemler and Oliver Lavery) for all their feedback

• White paper almost done and coming very soon!

• Pay attention to our Blog and Twitter (@gdssecurity) for details

My Contact Info:
email: rgutierrez@gdssecurity.com
twitter: @rgutie01
github: https://github.com/rongutierrez