GOTHAM
DIGITAL • SCIENCE

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

- Its 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)

CITRIX XenMobile

IBM

Good

airwatch

MobileIron

http://www.gartner.com/technology/reprints.do?id=1-1UURNKA&ct=140603

Outline

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

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

https://media.blackhat.com/bh-us-11/Schuetz/BH_US_11_Schuetz_InsideAppleMDM_Slides.pdf

MDM Feature Breakdown

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

MDM Feature Breakdown

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

MDM Drawbacks

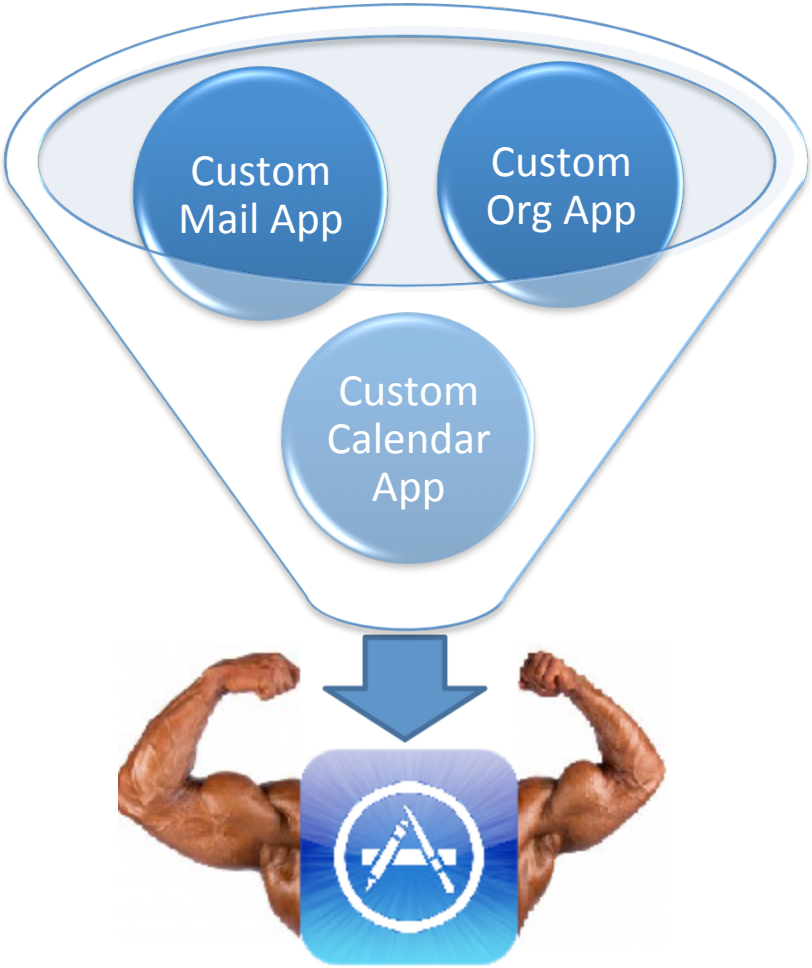
- Strict Policies Ruin Personal Device Experience
- Implementation is OS Dependent
- Privacy Concerns
  - Device Wipes
  - Querying of Installed Applications
- Data Protection Dependent on Application
  - Opt-in Data Protection APIs

Mobile Application Management (MAM)

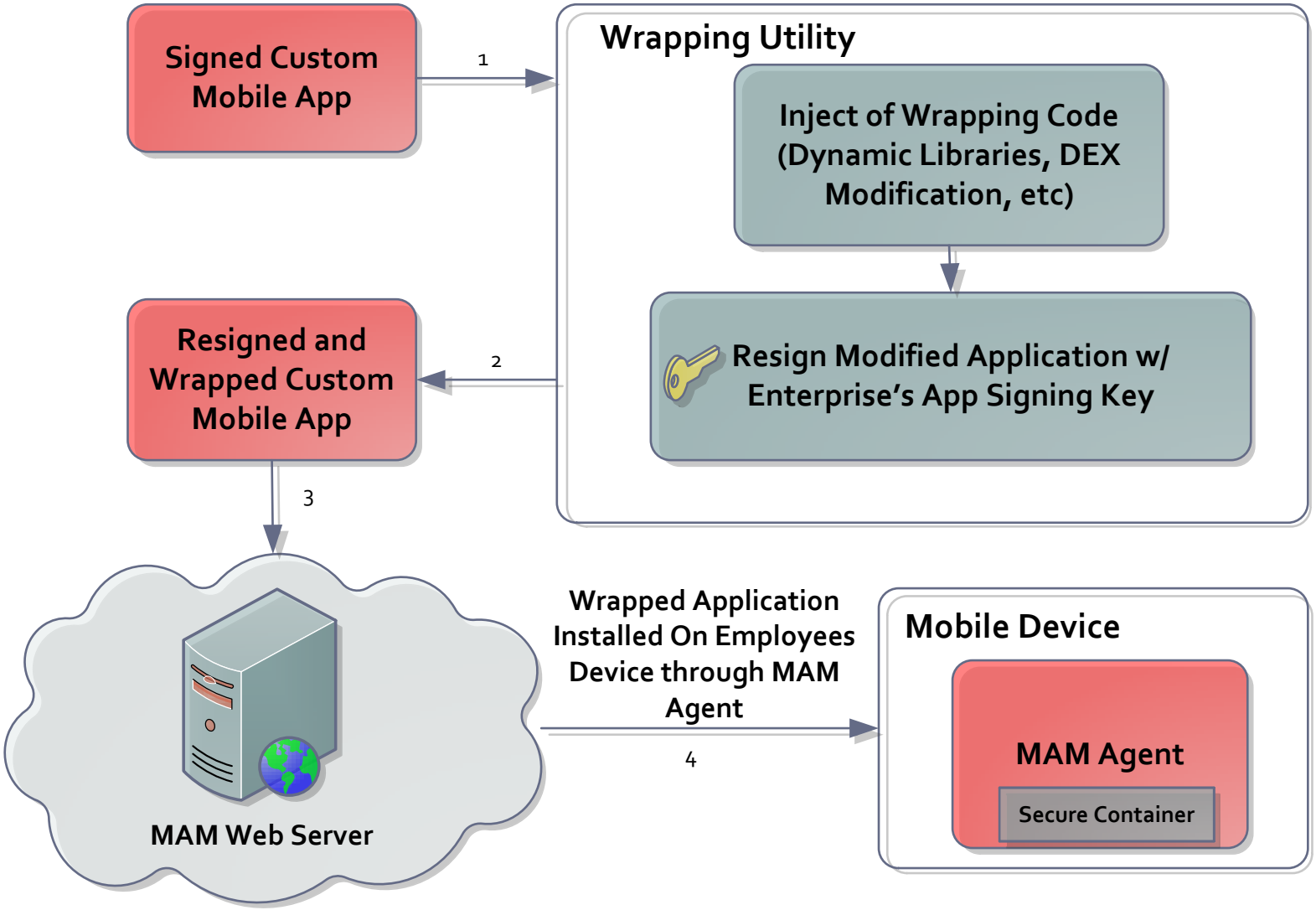
- Policy Enforcement & Data Protection At App Layer
- Requires Development of "Secure Containers"
- Application Wrapping Used To "Secure" Org Apps

http://www.endpointprotector.com/images/img/main/mobile-application-management-mam-en.png

Quick Intro to Application Wrapping



MAM Overview

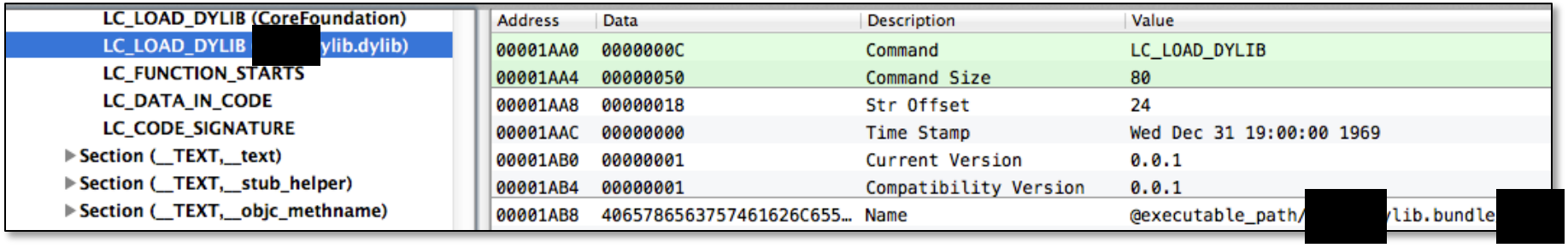


iOS App Wrapping Analysis

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

01A94	466F756E	64617469	6F6E0000	0C000000	50000000	18000000	00000000	6804	466F756E	64617469	6F6E0000	26000000	10000000	A4470000	30000000
01AB0	01000000	01000000	40657865	63757461	626C655F	70617468	2F436974	6832	29000000	10000000	D4470000	08000000	1D000000	10000000	A0500000
01ACC	72697844	796C6962	2E62756E	646C652F	43697472	69784479	6C69622E	6860	50270000	00000000	00000000	00000000	00000000	00000000	00000000
01AE8	64796C69	62000000	26000000	10000000	A4470000	30000000	29000000	6888	00000000	00000000	00000000	00000000	00000000	00000000	00000000
01B04	10000000	D4470000	08000000	1D000000	10000000	A0500000	50270000	6916	00000000	00000000	00000000	00000000	00000000	00000000	00000000

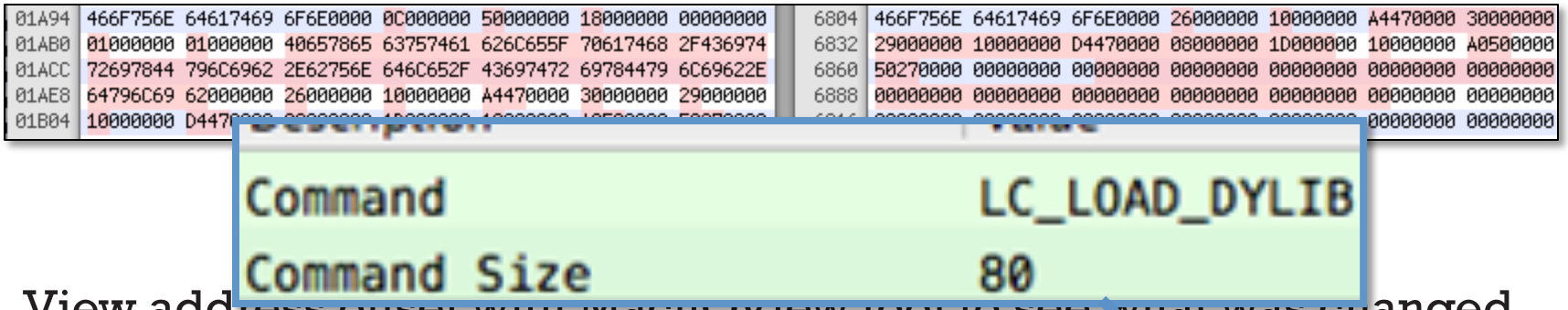
View address offset with MachOView tool to see what was changed



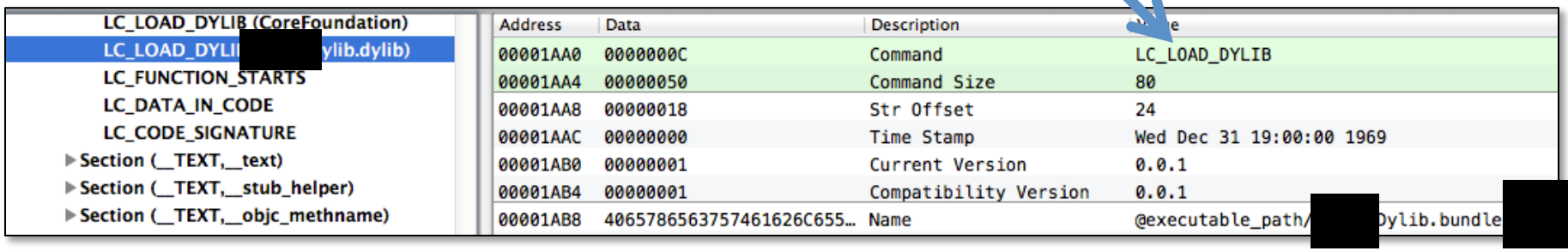
A LC_LOAD_DYLIB is added to the App's Mach-O Load Commands

iOS App Wrapping Analysis

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



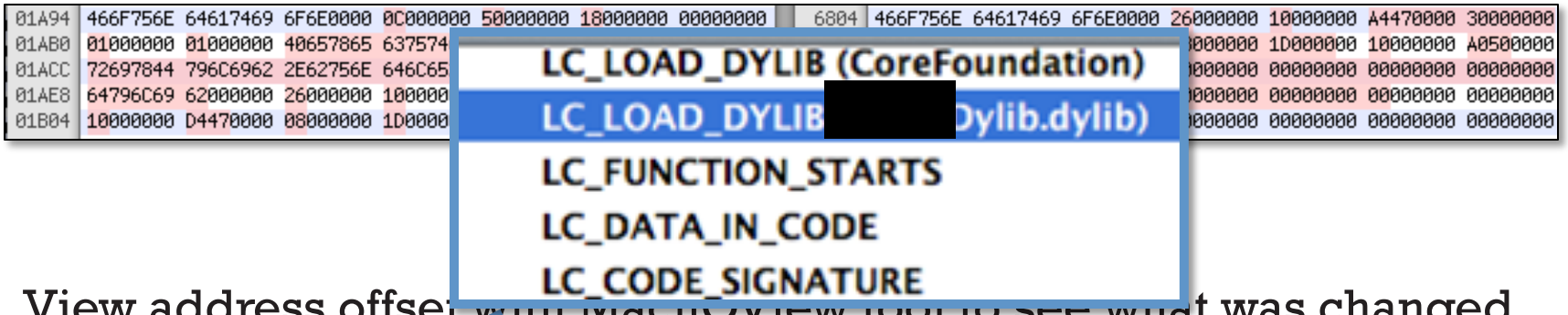
View address offset with MachOView tool to see what was changed



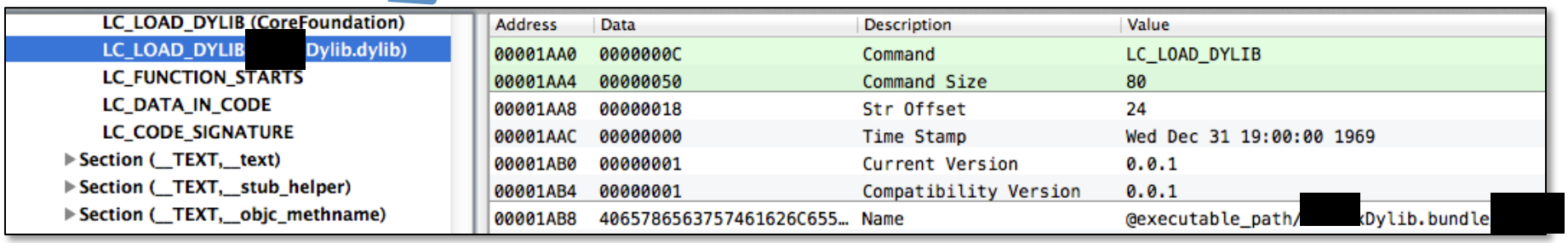
A LC_LOAD_DYLIB is added to the App's Mach-O Load Commands

iOS App Wrapping Analysis

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



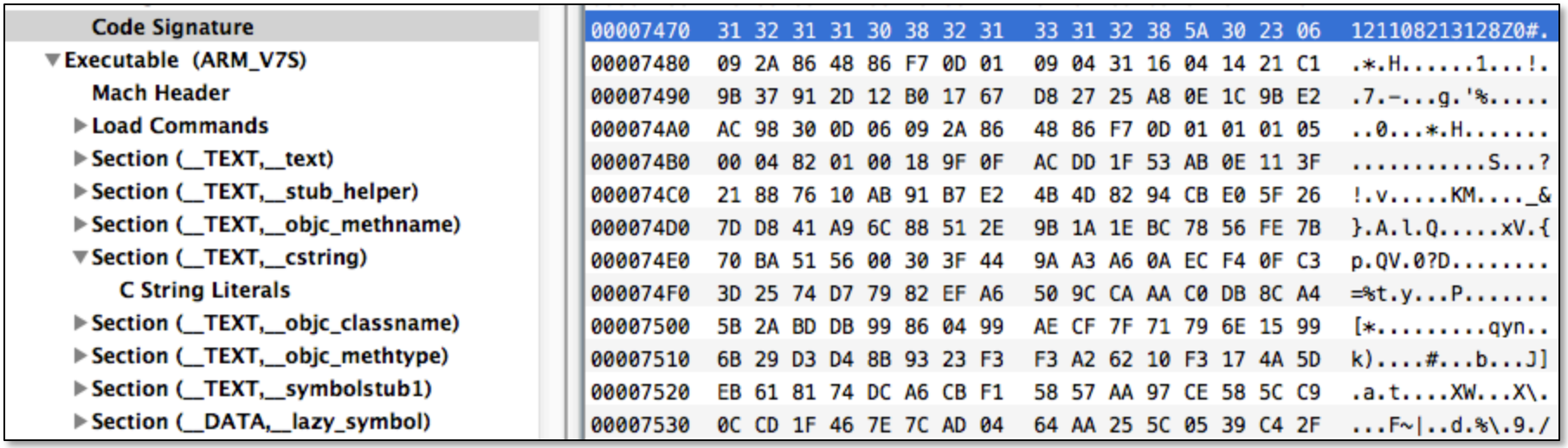
View address offset with MachOView tool to see what was changed



A LC_LOAD_DYLIB is added to the App's Mach-O Load Commands

iOS App Wrapping Analysis

0747C	5A302306	092A8648	86F70D01	09043116	041421C1	9B37912D	12B01767	0747C	5A302306	092A8648	86F70D01	09043116	0414B10E	3FC57B5C	74079A3D
07498	D82725A8	0E1C9BE2	AC98300D	06092A86	4886F70D	01010105	00048201	07498	CB61FF37	6304F139	06BF300D	06092A86	4886F70D	01010105	00048201
074B4	00189F0F	ACDD1F53	AB0E113F	21887610	AB91B7E2	4B4D8294	CBE05F26	074B4	0063488C	6892F23F	1C4C149B	16ED2B22	0403F068	1904FBA4	A2305B8C
074D0	7DD841A9	6C88512E	9B1A1EBC	7856FE7B	70BA5156	00303F44	9AA3A60A	074D0	0C9C51B2	9A5E8C40	78677EC5	74D6DA31	0D6B550A	2BA0461C	8CB0DFE6
074EC	ECF40FC3	3D2574D7	7982EFA6	509CCAAA	C0DB8CA4	5B2ABDDB	99860499	074EC	D89D26B7	AFE16C25	BFAAA579	83D118B2	49A35A2D	26434FF3	A8847179
07508	AECF7F71	796E1599	6B29D3D4	8B9323F3	F3A26210	F3174A5D	EB618174	07508	4131CE51	8326CF6F	A1356FD9	381C55B7	9FCCD5A6	EEED7673	46A2CFB2
07524	DCA6CBF1	5857AA97	CE585CC9	0CCD1F46	7E7CAD04	64AA255C	0539C42F	07524	890176F7	CEA2512A	0456510A	012E0DE5	D99D3A91	324A1FE5	08F0929F
07540	45F099EC	27001E95	7C21276D	CAA62A18	1C8607AA	F221BD65	65DC1BD6	07540	ABC468EA	783AE438	C112B634	DADBE9A2	540B415F	CCFE7639	AA21C590
0755C	39647992	94B70C9C	3798D2B6	B8FF18B6	9DB5E3E0	D8FF733A	A830261D	0755C	2EFBBFA5	9757EBC8	198F1D85	83B23F59	59255802	F62B92F4	B6F1C401
07578	313A0103	554099CB	C95970BF	5490B5EE	F710D8E9	BD677435	8E0B67CA	07578	F7F6CBA4	001E63AB	DC432C35	C5112CD8	36A3F2B7	5F604048	D6EF183D
07594	608224B3	553F5D11	787C4F22	25E355A0	43FEED23	51065954	E40DF2F0	07594	9ED241EB	5A3EEF3B	19B574F8	6396403F	4553EFD6	15B435B1	485C5A4E
075B0	D21299E4	E7000000	00000000	00000000	00000000	00000000	00000000	075B0	80DC58A5	4F000000	00000000	00000000	00000000	00000000	00000000

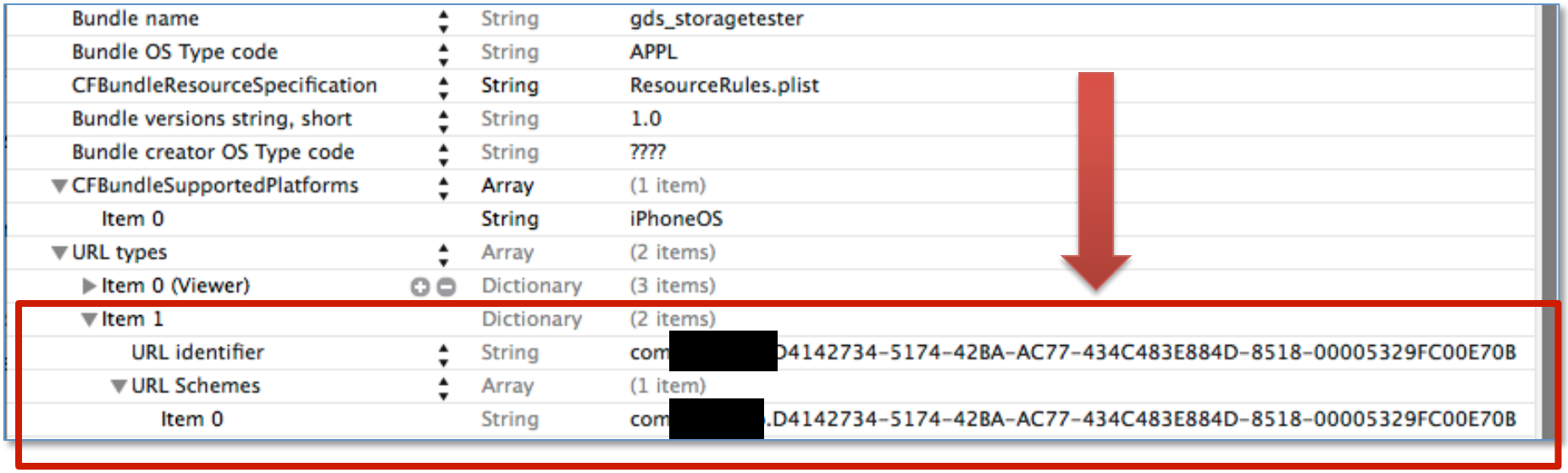


Updates to the Code Signature of the Binary

GOTHAM DIGITAL SCIENCE

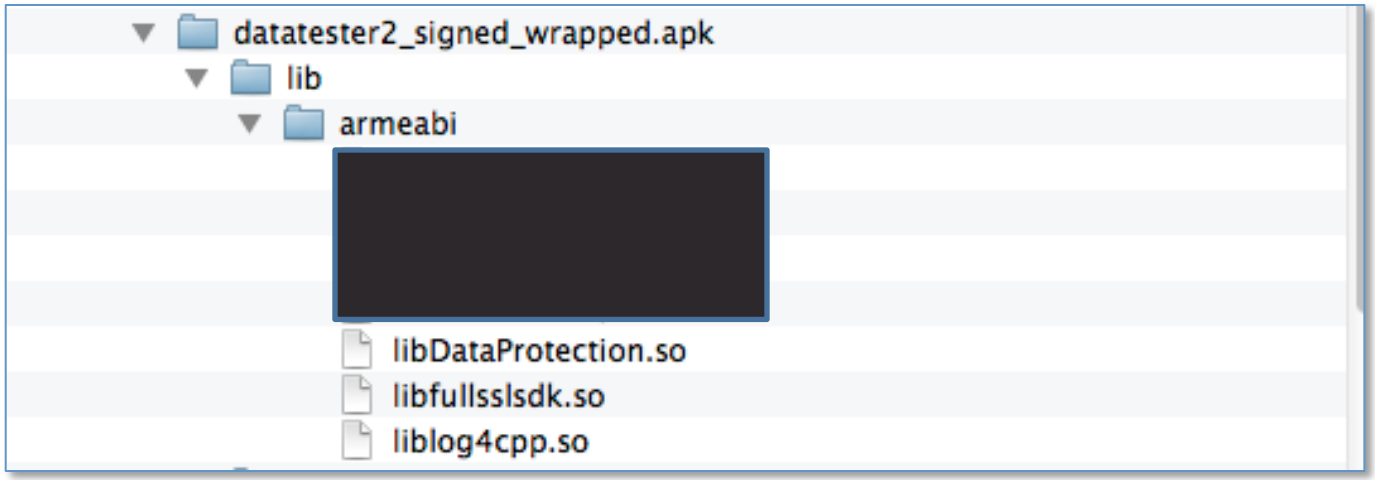
iOS App Wrapping Analysis

New URL Scheme App Entry Point Added to Info.plist

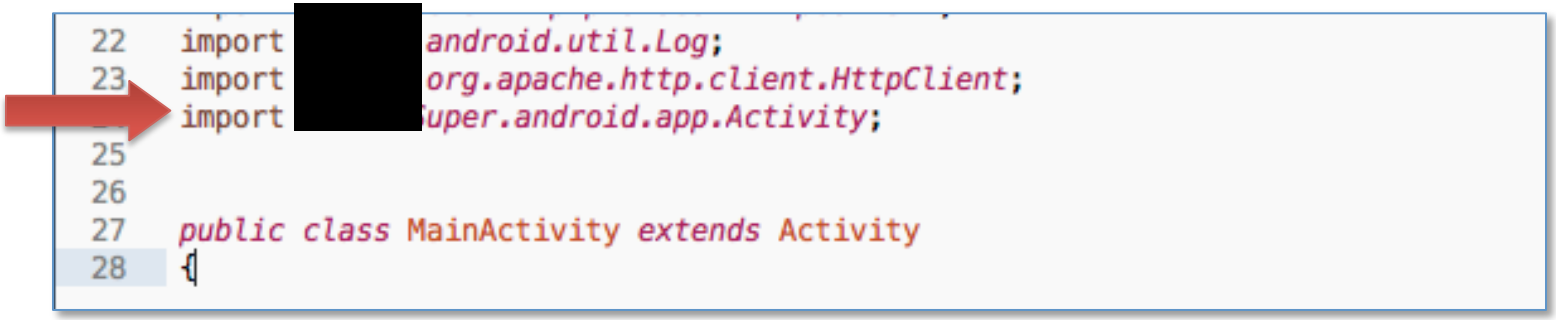


Android Wrapping Analysis

Additions of various NDK libraries



DEX Bytecode Modification



Android Wrapping Analysis

Common Android APIS Are Replaced Throughout App (ASMDEX Library)

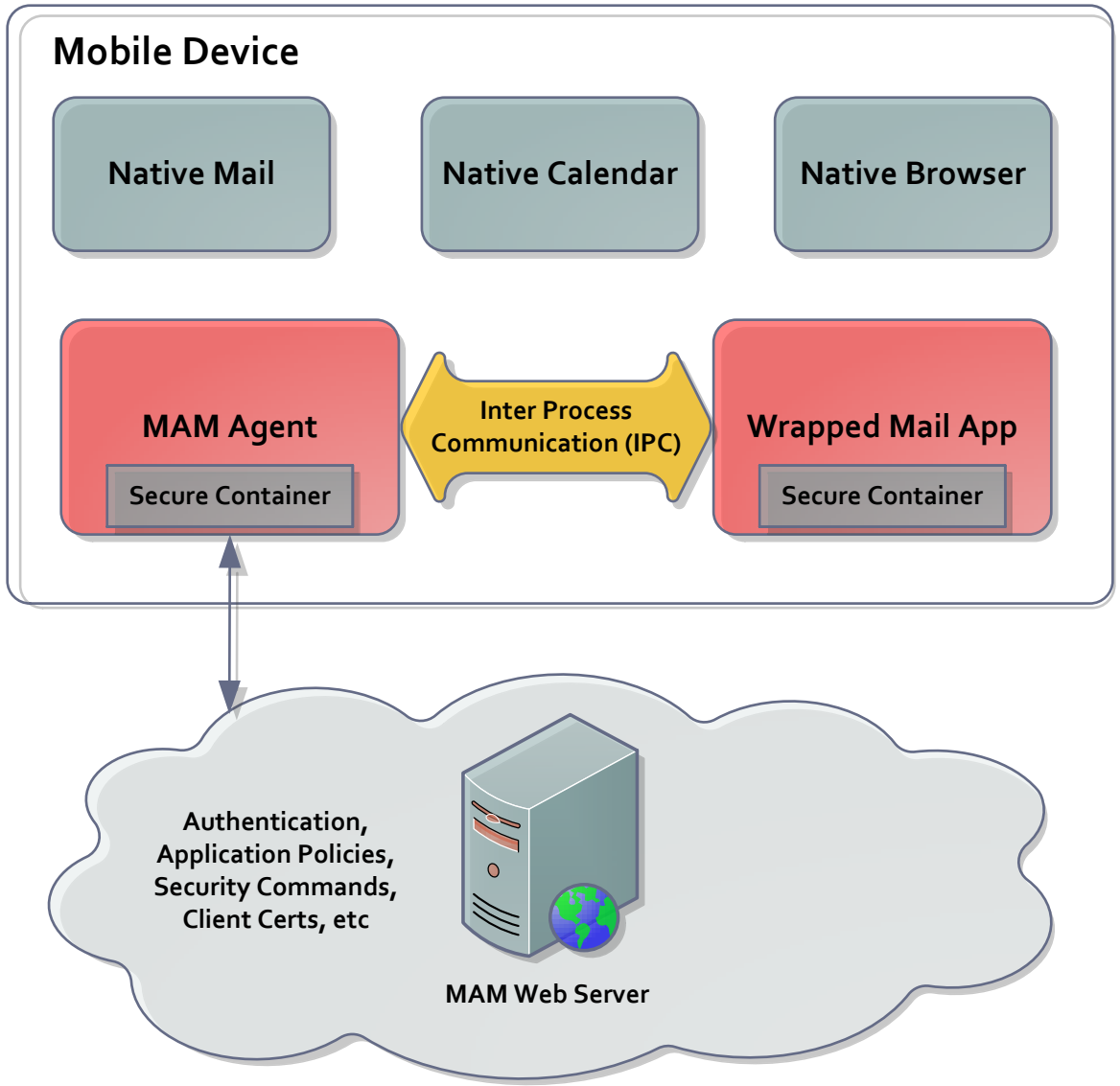
```
new File("");
    android.content.Context localContext = getApplicationContext();
    OutputStreamWriter localOutputStreamWriter =
      new OutputStreamWriter([redacted].android.content.Context.openFileOutput(localContext, "GDSFileOutputStreamTest.txt",
          ));
    localOutputStreamWriter.write("SensitiveData\n");
    localOutputStreamWriter.flush();
    localOutputStreamWriter.close();

    new org/apache/http/impl/client/DefaultHttpClient;
    [redacted].org.apache.http.impl.client.DefaultHttpClient.createObject();
    new BasicHttpContext().setAttribute("http.cookie-store", localBasicCookieStore);
    Log.i("GDSTest", "Performed Cookie Test");

    [redacted].android.text.ClipboardManager.setText((android.text.ClipboardManager)[redacted].android.content.Context.
        getSystemService(localContext, "clipboard"), "SensitiveData");
    Log.i("GDSTest", "Performed ClipBoardManager Test");
    Log.i("GDSTest", "Creating Webview");
    setContentView(2130903040);
    android.webkit.WebView localWebView = (android.webkit.WebView)findViewById(2131230720);
    localWebView.getSettings().setJavaScriptEnabled(true);
    [redacted].android.webkit.WebView.loadUrl(localWebView, "file:///android_asset/test.html");
```

Android Wrapping Analysis

Added Content Providers, Services, Activities and Broadcast Receivers

```
<provider android:name="com.[redacted]MAM.com.skomalzy.datahmacverifier.AppStateCPWrapper" android:exported
="false" android:authorities="com.skomalzy.datahmacverifier.com[redacted].managedApp.appState" />
<provider android:name="com.[redacted]AM.com.skomalzy.datahmacverifier.ManagedAppInfoCPWrapper" android:
exported="false" android:authorities="com.[redacted]MAM.Android.ManagedApp.ManagedAppInfoProvider.com.
skomalzy.datahmacverifier" />
<provider android:name="com.[redacted]MAM.Android.ManagedApp.AppQuitContentProvider" android:exported="
false" android:authorities="com.skomalzy.datahmacverifier.com[redacted].managedApp.quit" />
<provider android:name="com.[redacted]MAM.Android.ManagedApp.DiagContentProvider" android:exported="true"
android:authorities="com.skomalzy.datahmacverifier.com.[redacted].managedApp.Diag" />
<service android:name="com.[redacted]MAM.Android.ManagedApp[redacted]pManager" android:exported="true" />
<service android:name="com.[redacted]MAM.Android.ManagedApp[redacted]TMService" android:process=":mitm" />
<receiver android:name="com.[redacted].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.[redacted]MAM.Android.ManagedApp.[redacted]Locked" android:enabled="true" />
<activity android:theme="@*android:style/Theme.Translucent" android:name="com.[redacted].MAM.Android.
ManagedApp.DataContainmentActivity" />
```

MAM Overview



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

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

MAM Feature Breakdown

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

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

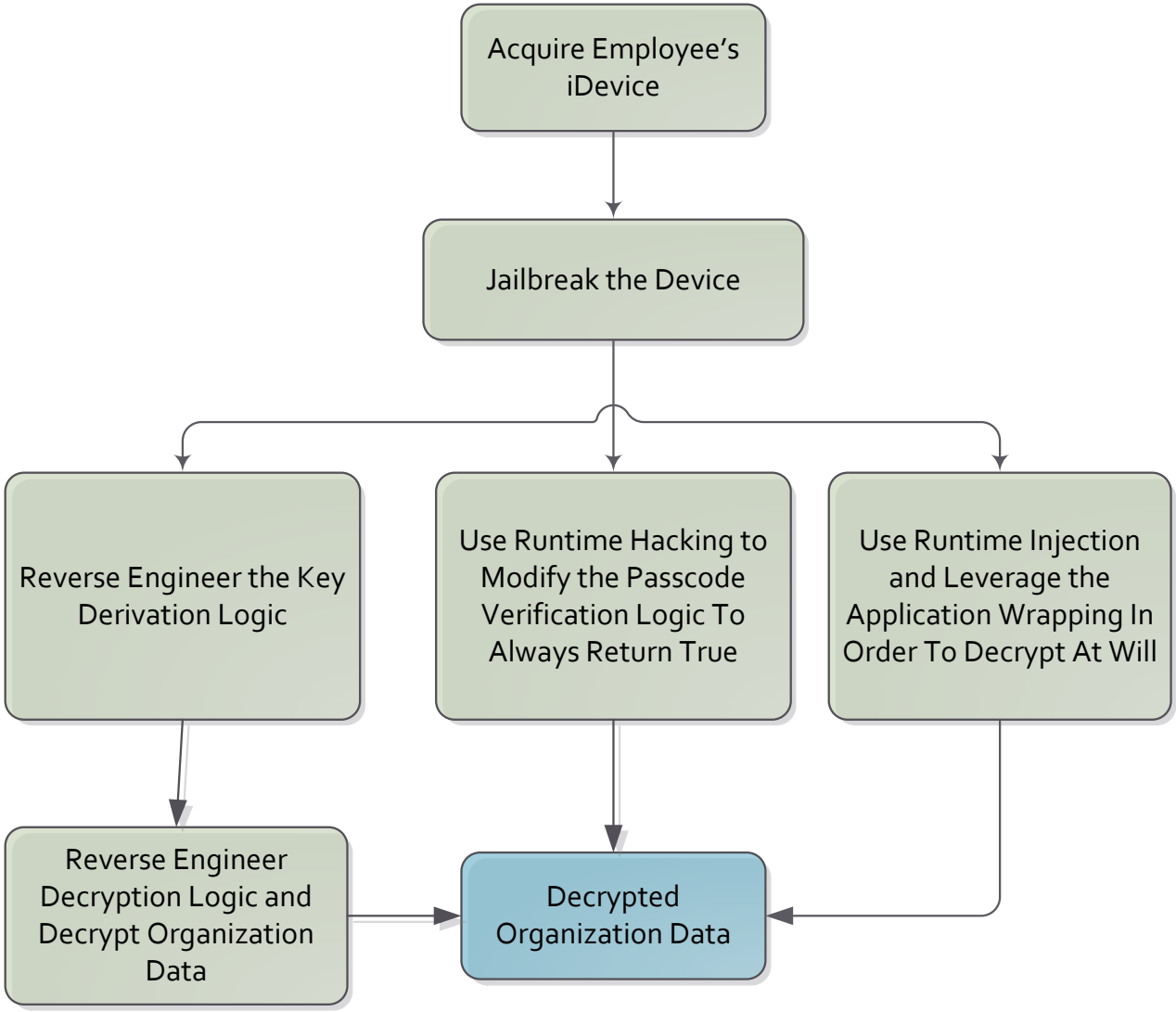
- After reverse engineering the key derivation process
  - All Key Material Stored on Device
  - Offline Authentication is Only Client Logic

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

Attack Tree


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

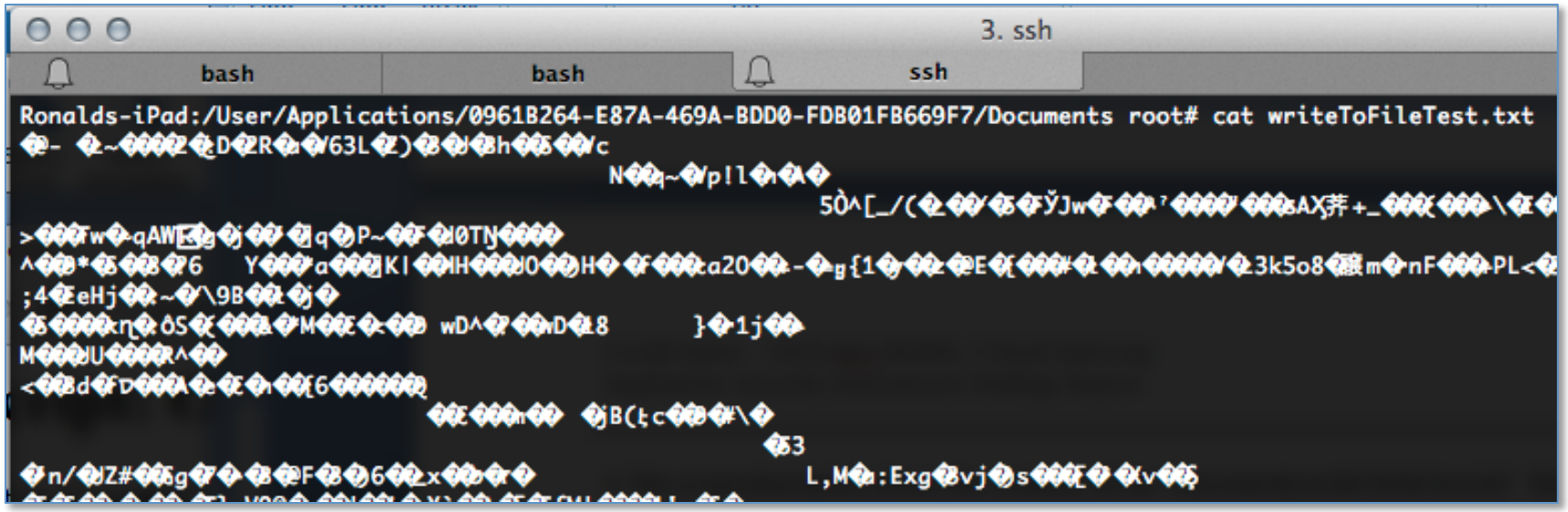
Vulnerability Pattern #1

Test Application That Will Be Wrapped

```
//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:NSStringEncodingConversionAllowLossy
               error:nil];
```

Vulnerability Pattern #1

Confirming The File Is No Longer Plaintext



Vulnerability Pattern #1

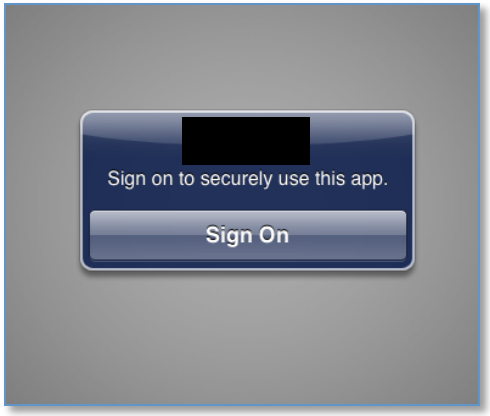
- Use Cycript in order to hook running wrapped app
- Use iOS File APIs in order to read arbitrary file
- Since app is wrapped, file decryption happens seamlessly

```
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:NULL];
```

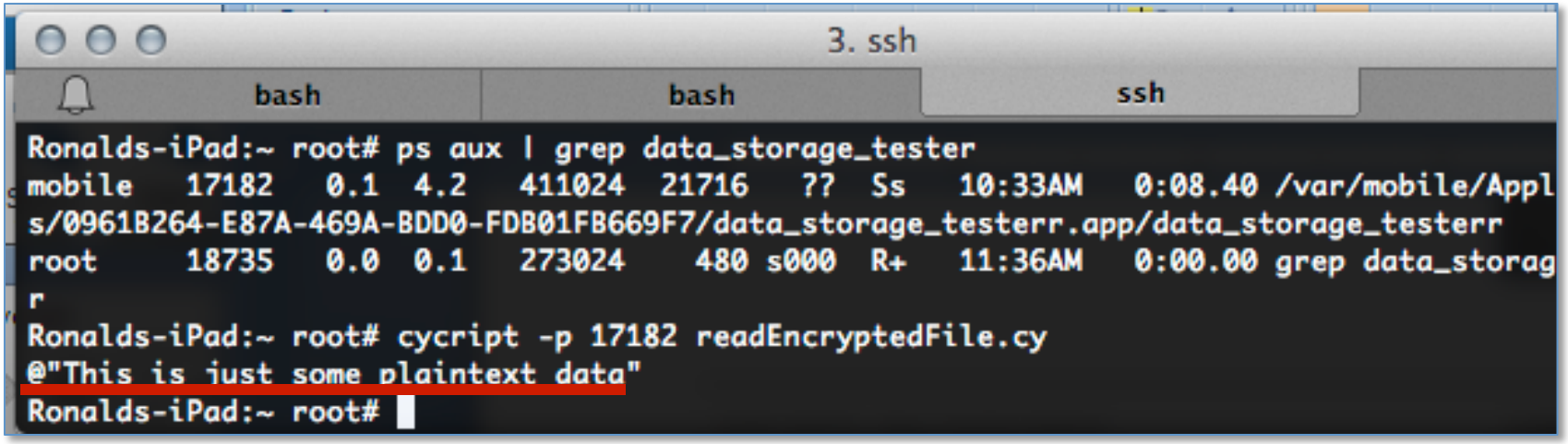
Proof of Concept Cycript Script

Vulnerability Pattern #1

Application Enters Unauthenticated State



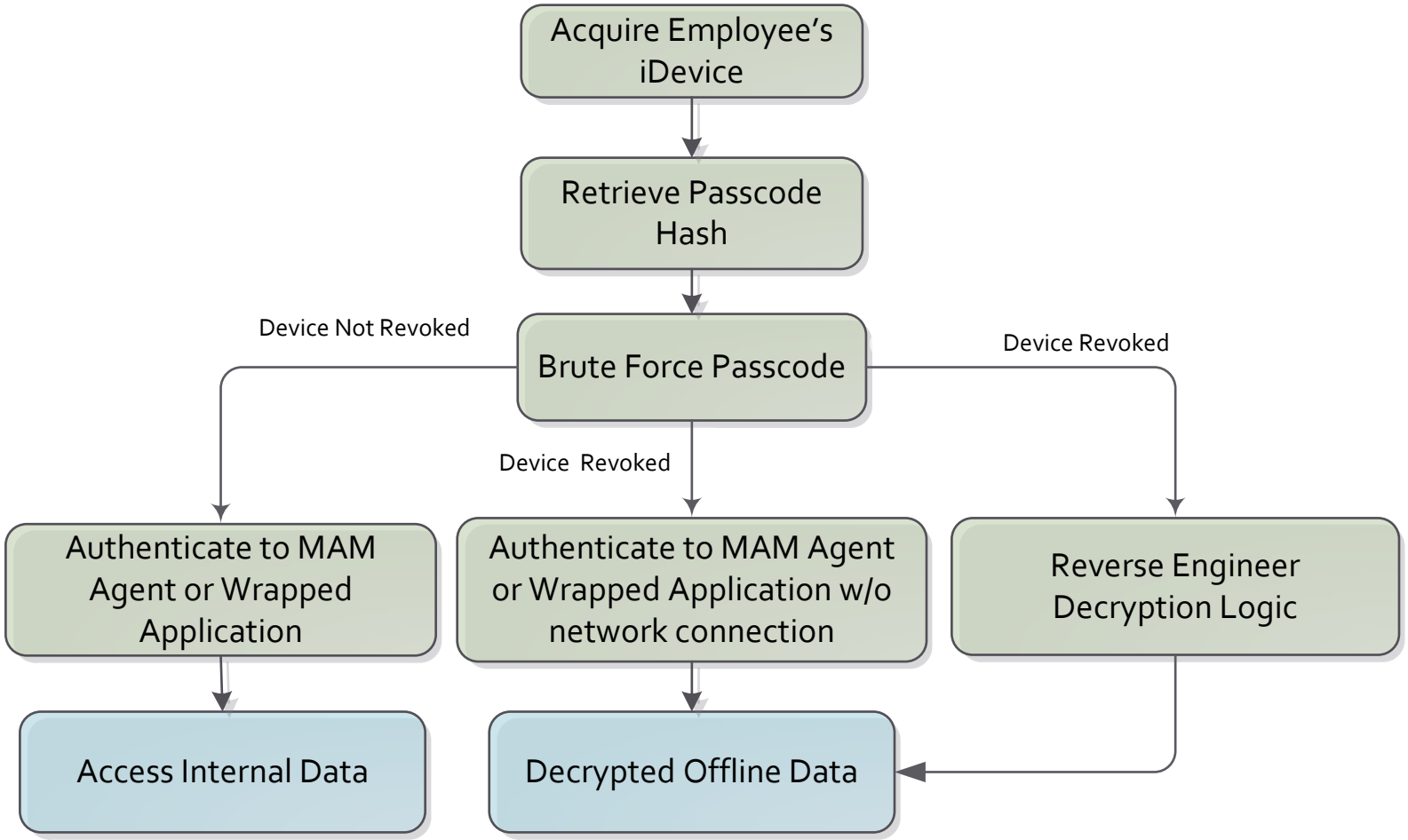
Doesn't Matter… Still Owned



Vulnerability Pattern #2

- Decoupling of the Passcode Verification & Key Derivation
- Key Derivation uses PBKDF2
- Offline Passcode Derivation uses **Unsalted** SHA-256

Vulnerability Pattern #2



Vulnerability Pattern #2

Location of Shared Preferences File

/data/data/com.[Omitted] /shared_prefs/com.[Omitted].[Omitted]_preferences.xml

```
<string
name="seedHash">NEHfC6vCot2lUdfNOfsjW8TgnNHkVWvyYbtJGI9Ug0g=
</string>
```

Proof of Concept

```
>>> import base64
>>> import hashlib
>>> output =
base64.b64encode(hashlib.sha256("testing1234").digest())
>>> print(output)
'NEHfC6vCot2lUdfNOfsjW8TgnNHkVWvyYbtJGI9Ug0g='
```

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

Vulnerability Pattern #3

```
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);
}
```

com.[Omitted].crypto.a.e

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
04-13 14:23:56.116: I/TheHook(6740): StackTrace(PBE):java.lang.Exception
..snip..
04-13 14:23:56.116: I/TheHook(6740):         at com.[Omitted].crypto.a.e(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740):         at com.saurik.substrate._MS
$MethodPointer.invoke(Native Method)
04-13 14:23:56.116: I/TheHook(6740):         at com.saurik.substrate.MS
$MethodPointer.invoke(MS.java:58)
04-13 14:23:56.116: I/TheHook(6740):         at com.android.TheHook.Hook$1$1.invoked(Hook.java:142)
04-13 14:23:56.116: I/TheHook(6740):         at com.saurik.substrate.MS$2.invoked(MS.java:68)
04-13 14:23:56.116: I/TheHook(6740):         at com.[Omitted].crypto.a.e(Native Method)
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].crypto.a.a(Native Method)
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[G4&v
GGhO3-[v(Y8
04-13 14:23:57.938: I/TheHook(6740): returnValue(HEX) a.e(String,String):
1B3C495B47EFBFBD3426EFBFBD76EFBFBDEB8794EFBFBDEFBFBD47EFBFBDEFBFBD68EFBFBD0E4F332D5B76EFBFBDEFBFBD28
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
04-13 14:23:56.116: I/TheHook(6740): StackTrace BE):java.lang.Exception
..snip..
04-13 14:23:56.116: I/T
04-13 14:23:56.116: I/T
$MethodPointer.invoke(M
04-13 14:23:56.116: I/T
$MethodPointer.invoke(M
04-13 14:23:56.116: I/TheHook(6740):          at com.android.TheHook.Hook$1$1.invoked(Hook.java:142)
04-13 14:23:56.116: I/TheHook(6740):          at com.saurik.substrate.MS$2.invoked(MS.java:68)
04-13 14:23:56.116: I/TheHook(6740):          at com.[Omitted].crypto.a.e(Native Method)
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].crypto.a.a(Native Method)
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[G4&v뇔
GhO3-[v(Y8
04-13 14:23:57.938: I/TheHook(6740): returnValue(HEX) a.e(String,String):
1B3C495B47EFBFBD3426EFBFBD76EFBFBDEB8794EFBFBDEFBFBD47EFBFBDEFBFBD68EFBFBD0E4F332D5B76EFBFBDEFBFBD28
5938
```

```
com.[Omitted].crypto.a.e(2 args)(a.e(String,String))
param1(PLAIN): testing1234
param2(PLAIN): 01c0d1e63656057ac6720eb688b988d1
```

GOTHAM DIGITAL SCIENCE

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
04-13 14:23:56.116: I/TheHook(6740): StackTrace(PBE):java.lang.Exception
..snip..
04-13 14:23:56.116: I/TheHook(6740):         com.[Omitted].crypto.a.e(Unknown Source)
04-13 14:2
$MethodPoi
04-13 14:2
$MethodPoi
04-13 14:2                                                                    ava:142)
04-13 14:2                                                                    8)
04-13 14:2
04-13 14:2
04-13 14:2
04-13 14:2
04-13 14:2
04-13 14:2                                                                    l(Unknown
Source)
..snip..
04-13 14:2
04-13 14:2
1B3C495B47                                                                    EFBFBD28
5938
```

```
StackTrace(PBE):java.lang.Exception

        at com.[Omitted].crypto.a.e(Unknown Source)
        at com.saurik.substrate._MS$MethodPointer.invoke(Native

        at
.invoke(MS.java:58)
        at com.android.TheHook.Hook$1$1.invoked(Hook.java:142)
        at com.saurik.substrate.MS$2.invoked(MS.java:68)
        at com.[Omitted].crypto.a.e(Native Method)
        at com.[Omitted].crypto.a.a(Unknown Source)
        at com.[Omitted].crypto.a.a(Native Method)
        at com.[Omitted].g.c.a(Unknown Source)
        at com.[Omitted].u.a(Unknown Source)
        at com.[Omitted].u.a(Unknown Source)
        at com.[Omitted].ui.fragment.bt.doInBackground(Unknown
```

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
04-13 14:23:56.116: I/TheHook(6740): StackTrace(PBE):java.lang.Exception
..snip..
04-13 14:23:56.116: I/TheHook(6740):         at com.[Omitted].crypto.a.e(Unknown Source)
04-13 14:23:56.116: I/TheHook(6740):         at com.saurik.substrate._MS
$MethodPointer.invoke(Native Method)
04-13 14:23:56.116: I/TheHook(6740):         at com.saurik.substrate.MS
$MethodPointer.invoke(MS.java:58)
04-13 14:23:56.116: I/TheHook(6740):         at com.android.TheHook.Hook$1$1.invoked(Hook.java:142)
04-13 14:23:56.116: I/TheHook(6740):         at com.saurik.substrate.MS$2.invoked(MS.java:68)
04-13 14:23:56.116: I/TheHook(6740):         at com.[Omitted].crypto.a.e(Native Method)
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].crypto.a.a(Native Method)
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[G4&v
GGh3-[v(Y8
04-13 14:23:57.938: I/TheHook(6740): returnValue(HEX) a.e(String,String):
1B3C495B47EFBFBD3426EFBFBD76EFBFBDEB8794EFBFBDEFBFBD47EFBFBDEFBFBD68EFBFBD0E4F332D5B76EFBFBDEFBFBD28
5938
```

Vulnerability Pattern #3

Return Value (Derived Symmetric Key):

1B3C495B47**EFBFBD**3426**EFBFBD**76**EFBFBD**EB8794**EFBFBDEFBFBD**47**EFBFBDEFBFBD**68**EFBFBD**0E4F332D5B76**EFBFBDEFBFBD**285938

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 AndroidManfest.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.[redacted]/
com.[redacted].ui.SearchActivity -e "query" "t"
Starting: Intent { act=android.intent.action.SEARCH cmp=com.[redacted]/.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

Outline

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

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