GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Android, BYOD, and AirWatch MDM

GIAC (GSEC) Gold Certification

Author: Tim Collyer, tccollyer@gmail.com
Advisor: Rich Graves

Abstract

The proliferation of mobile devices presents a heightened risk to the security of accessing and storing sensitive information. Beyond this, additional risk is introduced in Bring-Your-Own-Device (BYOD) environments where many of the security decisions are made by the end users who own them and not the organization which owns the data. Specifically, the Android platform presents more challenges for corporate security posture due to the wide array of hardware platforms, operating system versions, and the variety of marketplaces for Android apps. A further indicator that Android devices are at a higher risk are the recurring reports of widespread malware campaigns targeting the Android operating system. Despite the growing number of attacks targeting Android systems, security teams have a limited number of tools to control the flow and use of corporate data on mobile devices. One such tool that is currently available for managing Android mobile devices is a platform called AirWatch Mobile Device Management. This paper examines the use of AirWatch as a Mobile Device Management Platform.

1. Introduction

1.1. Growth of mobile

Not surprisingly, mobile devices are an increasingly important part of the topology of how people access business data. According to the Pew Research Internet Project, as of January 2014 90% of American adults have a cell phone and 58% own a smartphone (Pew Research, 2014). The growth of adoption of phones and tablets as a means to access data has been coupled with an increasing demand to use those devices in places of business (Schadler, 2013).
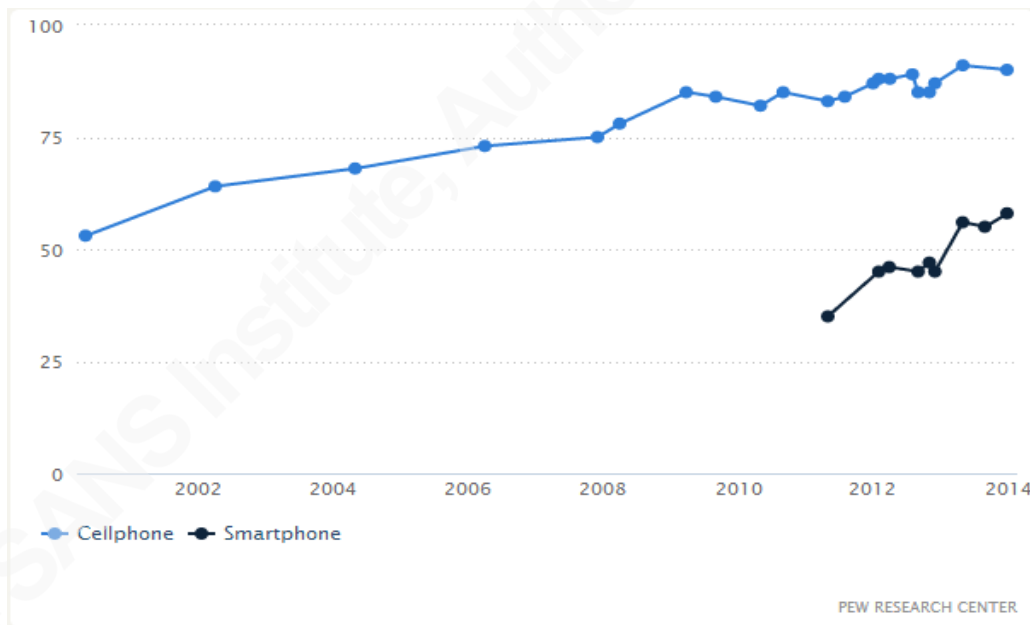

Figure 1 - the rise of smartphone adoption in the United States

This paper will follow the defining characteristics for a "mobile device" as described by the National Institute of Science and Technology (NIST) in Special Publication 800-124:

- A small form factor
- At least one wireless network interface for network access (data communication)

Tim Collyer, tccollyer@gmail.com

- Local built-in (non-removable) data storage
- An operating system that is not a full-fledged desktop or laptop operating system
- Applications available through multiple methods (provided with the mobile device, accessed through web a browser, acquired and installed from third parties)

This description singles out phones and tablets and specifically excludes laptops. This exclusion is important because mobile devices by this definition represent relatively new mechanisms for accessing and storing data, which require new mechanisms for control and management. Security programs have had to rethink the concept of defending a network perimeter and bring new tools and controls to bear in order to account for the sudden permeability of the network boundary (Gordon, 2013).

1.2. Concept of MDM

One of the more frequently discussed tools used to compensate for the network permeability created by mobile devices is mobile device management (MDM). MDM is a mechanism by which corporations can maintain some control over corporate data on a device which, by design, can be walked out of a network. NIST says of MDM capabilities, "In addition to managing the configuration and security of mobile devices, these technologies offer other features, such as providing secure access to enterprise computing resources," (NIST, 800-124).

One aspect of the threat landscape surrounding smartphones and tablets is a lack of physical controls that often protect much of the IT infrastructure of a company. When this absence degrades into complete loss of physical control due to theft or user carelessness, compensating mechanisms to reduce the risk to data are required, e.g. remotely wiping the device in question. Almost all mobile management tools offer such a capability including basic management via ActiveSync within Microsoft Exchange. This remote wipe is an acknowledgement that what is of prime importance in a mobile device is not the device itself, but the proprietary data contained in it.

Tim Collyer, tccollyer@gmail.com

Though this last-ditch effort to protect data might help security administrators sleep at night, it is often viewed differently by the end users who use their devices to store personal information alongside business data. "Users are less likely to report a lost device to IT when it's their own for fear of losing their personal data, along with any company information, when the device is wiped," (Deloitte CIO Journal, 2014). A delay in reporting represents a serious risk to proprietary data by giving malicious parties time to sever network connections to the device to prevent the remote wipe command from being received. Even if non-malicious parties discover the device, the risk of data leakage is high. The Symantec Honey Stick project was an innovative study of what happens to a misplaced mobile device. They placed 50 devices in various locations in major metropolitan areas and tracked what was done with the phones after discovery by passersby. 83 percent of devices in the study showed attempts to access corporate-related apps or data (Wright, 2012), indicating that concerns regarding the physical security of mobile devices need to be taken seriously.

The likelihood of user reluctance to risk personal data loss is further increased when companies maintain a Bring Your Own Device (BYOD) policy in which employees may use a personal device to access business resources. In such cases a full device wipe can be more challenging, "the organization may not have the freedom to define policies and requirements for a device they do not own" (Wright, 2013). Yet there had been increasing demand for BYOD for convenience and perceived cost savings

For BYOD, MDM solutions must therefore fall back on other methods of controlling data which includes the concept of containerization. In the context of mobile device management, containerization represents a sandbox which holds business data and separates it from end user data and applications. This concept seems a good compromise between the security requirement of maintaining control over business data and the end user's desire to maintain control over personal data. For instance, the contents of the container can be remotely destroyed without necessarily affecting the rest of the data on the phone. Thus an end user can be encouraged to report a lost or stolen phone promptly in order to minimize the exposure window for proprietary information.

Tim Collyer, tccollyer@gmail.com

1.3. Threat model

Before one can define the ways in which MDM solutions can protect devices, it is important to understand the nature of the threats. They may be simplified to the following categories:

1. Data leakage
   a. Insider threats
      i. Terminated employees
      ii. Malicious/disgruntled employees
   b. Device loss
   c. Untrusted software
2. Malicious software (malware)
   a. Malware which targets the device owner (for identity theft or direct revenue through mechanisms such as causing a device to send premium SMS messages)
   b. Targeted malware with intent to steal organizational information stored locally
3. Mobile devices as pivot point for attackers to gain entrance to an internal network

Within the context of protecting sensitive information on devices enrolled in a BYOD program, the malware which targets the device owner (2a in the list above) is not a direct concern to an organization's security posture. Indirectly, a device compromised via one vector may represent additional risk later on. However, the greatest focus and resources for a business security program should be devoted to the higher risk areas such as various avenues for data leakage or targeted attacks against mobile devices as an entry point into a network.

Tim Collyer, tccollyer@gmail.com

1.4. What MDM protects against

Mobile device management suites offer an array of potential controls, most of which are designed to assist with asset and policy management. These include options such as pushing VPN connection settings, providing Wi-Fi credentials or certificates, and installing selected apps for use within the environment. The security controls available allow a fully managed (usually company-owned) device to be locked down significantly - with a few caveats which will be discussed later. These options include mechanisms such as application whitelisting, limitations on camera use, prohibiting screenshots, and preventing access to device settings. Management of a BYOD environment, however, typically needs to compromise with the requirements of the device owner, as has been noted above.

The pressures of BYOD force the use of other more indirect management techniques, such as virtualization or containerization. In the virtualization methodology the data is never stored on the end device and instead the mobile unit is just a terminal to access the virtual environment. This mechanism mirrors the already well-established model of using virtual desktops and thin clients, but is now applied to the mobile world. XenMobile by Citrix is an example of a product which makes use of this technique (http://www.citrix.com/products/xenmobile/overview.html).

AirWatch uses a more common approach among MDM solutions to protect sensitive data - containerization. The containerization approach helps to directly protect against some threats and indirectly against others. The very nature of the container - encrypted storage of company data - is designed to prevent data leakage resulting from physical loss of the device. For devices compromised logically (not physically), AirWatch employs additional controls beyond encryption of data at rest within the container.

In order to help define how an MDM platform can help to manage risk on Android BYOD devices, this paper will provide a brief look at some of the available policies and settings within the software which might be used in such a circumstance. In addition to

Tim Collyer, tccollyer@gmail.com

examining the application layer of MDM, it is also worth exploring some of the technical underpinnings of how the platform functions on the Android operating system.

2. AirWatch Policy Configuration

AirWatch is a complex and multi-featured suite of controls and this paper is not intended to be an administration guide. It will, however, cover a few policies which might be used to help reduce the risk of allowing untrusted Android devices (via a BYOD program) into a company network. Not all of the recommendations below may be appropriate for a given organization and will depend upon the balance of risk appetite and end user resistance. Policy settings and screenshots are taken from the AirWatch management console version 6.5.1.4. Most of the referenced settings can be found under the Restrictions section within the Policy editor.

Each policy below has a description of the functionality, a subjective estimate of the priority, i.e. how effective that setting is at reducing risk, as well a suggestion as to how much impact such a setting might have upon an end user.

Policy recommendation: Deny USB debugging
Purpose: Prevent file system level access to the device, also prevent one method for root access.
Priority: High
User impact: Low - Few end users need to make use of this capability and those that do are most likely not a good fit for BYOD.
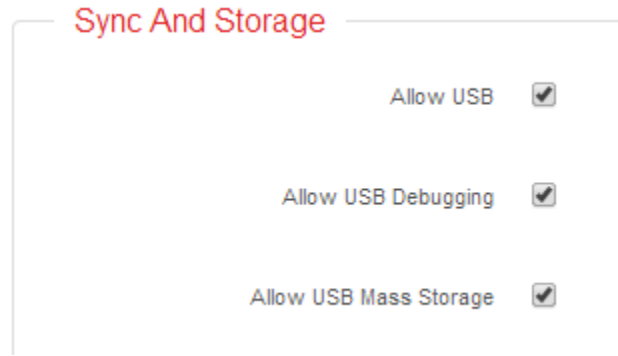
Tim Collyer, tccollyer@gmail.com

Figure 2 - AirWatch setting to enable/disable USB Debugging

Policy recommendation: Android version current - 2 (i.e. current is 4.4, minimum allowed would be 4.2)

Purpose: Many vulnerabilities and security fixes are patched with each new version of Android - according to Juniper 77% of Android threats could be largely eliminated by running the latest OS (Juniper, 2013). A moving window for supported OS versions can help to mitigate the risk from old vulnerabilities. The size of the window can be adjusted to match risk appetite - i.e. from n-2 to n-4 for a less risk averse environment. Though this 'moving window' is a bit arbitrary, it provides a stated policy which end users can understand and which allows a business to enforce a type of patch policy in an environment which isn't otherwise conducive to normal patching.

Priority: High

User impact: High - Many Android phone manufacturers offer limited support in terms of OS upgrades, preferring to steer users toward new hardware instead. Therefore this policy places the burden of upgrading hardware onto end users, requiring them to stay current in order to maintain access to company data from a BYOD device.

Policy recommendation: Prevent non-Market app installation

Purpose: The Google Play store has anti-malware measures which helps to remove malicious apps. However third-party app markets do not have similar prevention measures and are therefore rife with malicious applications. Though Android accounted for 97% of mobile malware in 2013, only 0.1% were found on the Google Play store (F-Secure, 2013). Blocking access to the third-party markets can therefore substantially

Tim Collyer, tccollyer@gmail.com

reduce the risk to the device owners, and therefore to some extent to the organization as well.

Priority: High

User impact: Low - Though this policy constrains user freedom slightly, the majority of apps are available in the Google Play store and the reduction of risk is substantial.

Policy recommendation: Limit mail and calendar sync duration

Purpose: Limiting the duration of synchronized data helps to limit the data exposure should a compromise or loss of device occur. For example, an organization may choose to limit the sync window to 2 weeks.

Priority: Moderate

User impact: Low - Users should still be able to search and access older data, it just won't be stored locally on the phone.

Policy recommendation: No Wi-Fi auto-connect

Purpose: Reduce the risk of man-in-the-middle (MiTM) attacks from mobile devices which are set to auto-connect to commonly named wireless networks. Though this will not mitigate the risk of a MiTM attack directly, the auto-connect feature of phones makes them easy targets in public places for attackers looking to harvest data and credentials which are sent during the attack process. Requesting that users connect to a network at least helps to reduce the risk of this type of attack happening without any user interaction.

Priority: Moderately Low

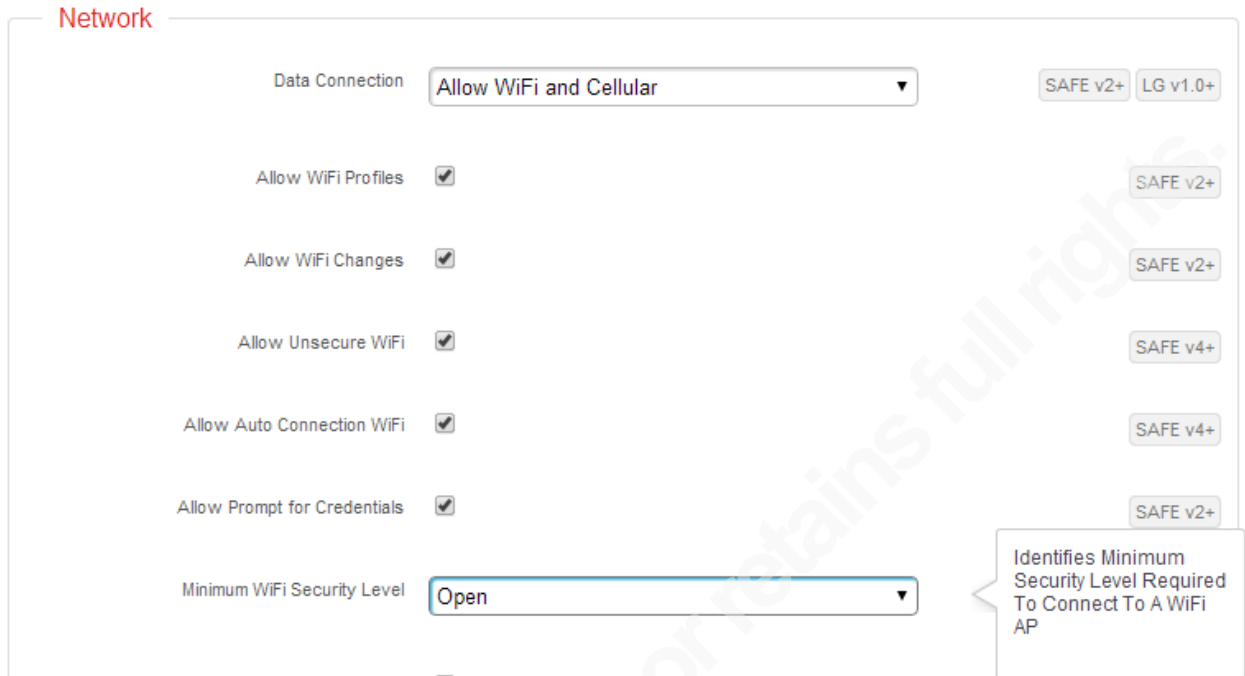**User impact**: High - Asking users to manually connect to any wireless network will likely be fairly disruptive.

Tim Collyer, tccollyer@gmail.com

Figure 3 – Some of the available Wi-Fi restrictions including Auto Connection

Policy recommendation: Device and container passcode

Purpose: Prevent unauthorized access to the device and the container. The device passcode provides an additional layer of defense which also helps to protect the end user. The container passcode is the final layer of defense against unauthorized access and should accordingly be relatively robust (recommended 6 characters minimum)

Priority: Moderately high

User impact: Moderate – Many users prefer the ease of accessing a device without a passcode. The double code required to access corporate email also adds some additional inconvenience.

Policy recommendation: Deny cut/paste out of email

Purpose: Prevent data leakage from insider threats

Priority: Low – Though this does stifle an easy path to leak data, there are many other methods of doing so and this therefore has only the benefit of "keeping the honest people honest." Still, for particularly risk averse environments, raising the difficulty to leak data as much as possible may be worthwhile.

User impact: Low

Tim Collyer, tccollyer@gmail.com

Policy recommendation: No screen capture

Purpose: Prevent data leakage from insider threats

Priority: Low

User impact: Moderately high – There are numerous legitimate uses for screen capture.

The additive combination of all of the policies above could provide a fairly restrictive environment for a mobile device. From mandating a recent version of the operating system to preventing such basics as screenshots the resulting managed device would be relatively secure and also concomitantly inconvenient to the end user. The number of options available allows an AirWatch administrator to tune the settings to match the security posture of the organization. However, the number of settings also highlights how challenging it is to exercise any real measure of control over the data once it has been released to a mobile device, particularly in the face of a malicious insider determined to steal information.

3. AirWatch Container

When asked to manage mobile devices that are a part of a BYOD program, the answer from many MDM solutions, including AirWatch, is to use a 'container'. The concept here is that all company data is stored in an encrypted fashion which can be wiped separately from rest of the OS. The simplest mechanism to 'wipe' the data is to destroy the encryption key so that, though the data may still remain on the device, it is rendered effectively useless. This mechanism naturally rests upon the use of a strong encryption algorithm to resist long term offline cracking attempts.

Most of the AirWatch (and other MDM) controls are based upon a foundation of preventing and detecting root-level access to the file system. A user with root-level privileges might be able to bypass the MDM policies and possibly access the data stored on the device. But what if someone is able to get around the AirWatch compromise

Tim Collyer, tccollyer@gmail.com

detection to access the full file system? How does the container work and what data might be exposed?

AirWatch offers a number of different products and apps for use on a mobile device. For simplicity's sake, only the AirWatch Email app will be discussed here with regards to containerization. All data were gathered from a Nexus 7 device running Android 4.4.2. The device was rooted and the AirWatch policies were relaxed so that the device could be enrolled in a QA testing environment even though AirWatch still considered it to be compromised.

The Android permission model is designed so that each application has its own account. This means that only the root account and the application itself can access the app files. Many applications store information in a public location – the camera app for example, stores photos in a location that other apps like the email client can access. The first line of defense in an MDM container is to keep all files inside the app's own directory, thereby limiting what else can get access to them. This is why solutions like AirWatch require an email client app and do not use any of the prepackaged Android email solutions.

Specifically, email protected by AirWatch are stored in the /data/data/com.airwatch.email/ directory. Within the app directory are several sub-directories, including /databases and /files/Attachments. Email that is processed by the app is stored in SQLite databases in the /data/data/com.airwatch.email/databases/ directory:

Tim Collyer, tccollyer@gmail.com

Figure 4 – Contents of the AirWatch Inbox app databases directory

As seen in the screenshot above, there are contacts and calendar items stored here as well as email information. However, if one attempts to peek inside the database, even with root permissions (which are required to get to the file in the first place), it can be seen that the contents are encrypted and provide no usable information in the current state:

Figure 5 – encrypted contents of the EmailProvider.db database

The encrypted state of the data was verified by attempting to base64 decode it as well as viewing the data in a SQLite database viewer. Neither method revealed any plaintext which reinforced the conclusion that encryption mechanisms are being implemented within the databases.

The /data/data/com.airwatch.email/files/Attachments/ directory is similar in that it is possible to see the downloaded attachments but not the contents:

Tim Collyer, tccollyer@gmail.com

Figures 6 and 7 – A listing of the Attachments directory and a display of the encrypted
contents of the airwatch_test.csv attachment

Another line of defense beyond just storing data within the app directory, is to encrypt the

data while it is stored. In this fashion both root level access and the means to decrypt the

information are required to view the contents. Both the presence of libcrypto.so.0.0.1 in

the /data/data/com.airwatch.email/ directory as well as other indications, suggest that

AirWatch is making use of the OpenSSL cryptographic libraries to encrypt the contents

of files stored in its directories. These cryptographic libraries are part of the Android

operating system and this convenience may be why AirWatch chooses to use them. The

precise mechanism for use of encryption as well as the algorithm used in this process is

beyond the scope of this paper.

The use of built-in Android encryption libraries has both positive and negative

ramifications. By using the mechanisms already present within the operating system,

AirWatch is able to offload the development and troubleshooting of that mechanism to

other parties - namely the developers of OpenSSL and the divisions at Google which

implement the code on Android. In this way AirWatch can tap into greater resources than

might be available internally and benefiting from the work down by those other

resources. However, it also means that AirWatch has little control or ability to respond if

a flaw in the encryption process should be discovered.

Tim Collyer, tccollyer@gmail.com

Some elements of the encryption process however, are stored in the shared preference file, an XML file called "com.airwatch.email.preferences_shared.xml" and located in /data/data/com.airwatch.email/shared_prefs. The shared preference file is created in such a way as to once again leverage the app sandbox security - limiting access to the file to only the source app (Makan, 2013).


```
127|root@flo:/ # cat /data/data/com.airwatch.email/shared_prefs/com.airwatch.email_preferences.xml
il/shared_prefs/com.airwatch.email_preferences.xml                              <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <set name="attachmentFileDetails">
        <string>airwatch_test.csv,2143,1,                          </string>
        <string>airwatch_test.pdf,2145,1,                          </string>
        <string>            ,339,1</string>
    </set>
    <string name="db_enc_key">-~~i2kPORwCwJvkPZgw68GRinHYuB/l897iFlYi2oeILJG9AmWmQP4EvDyajPopbOnV
    </string>
    <boolean name="isSSOEnable" value="false" />
    <boolean name="accountDeleted" value="false" />
    <boolean name="firstRefresh" value="false" />
    <string name="locale">en_US</string>
    <string name="previousFileConfiguration"></string>
    <string name="master_encryption_key">CC+fPVN+iEIgmQQPaVcQ5+b8mxMIh+1S79MAOSpfPYN8gkjOFwpFDmoZKsgAygvc
    </string>
    <string name="hachmacToken">59634287-3786-427d-80cf-2866acfd6e74</string>
    <string name="emailConfiguration"></string>
    <string name="master_key_hash">K1kdNQBxk6nAXDIo1rUCUdVo2Yx+5OHjzCkd5OPbOTs=
    </string>
    <string name="hashed_dk">wDOsnluQ+QLAB8+7LkLKkxDjGRrnBvgUsxV+Vln1GNc=
    </string>
    <long name="selectedAccountId" value="1" />
    <boolean name="profileConfiguration" value="true" />
    <int name="internalAppVersion" value="1" />
    <string name="Vector">4Y676Kun65Ol4qqr4YKL7Ket168/5Ie35oKV7oef64+n4quP6auH6Lep6JuJ
    </string>
    <string name="deviceUID">685b4decbd955b1eda1f706fd7e7b5c5</string>
    <int name="configurationMode" value="1" />
</map>
root@flo:/ #
```

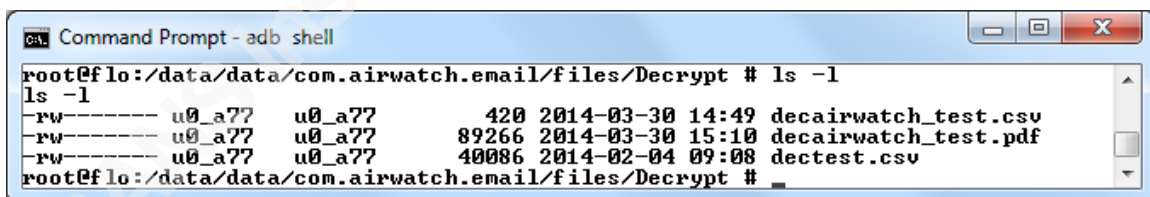*Figure 8 – contents of com.airwatch.email_preferences.xml file*

The sandbox security does not protect the file on a rooted device, so additional measures have been implemented. Though this file does have some suggestive fields in it, such as "master_encryption_key", the field values are encrypted and/or encoded to protect them. No attempt was made to determine the method of encryption, but it is likely something along the lines of Secure-Preferences (https://github.com/scottyab/secure-preferences) which is a wrapper designed to do just that: encrypt the Shared Preferences keys using 256-bit AES. Additionally, the values are generated with each instance of the app installed on a device. This means that information from one device cannot help to decrypt information on another device.

Tim Collyer, tccollyer@gmail.com

3.1. Container Vulnerability

During the course of research about how the AirWatch email container functions, we discovered a vulnerability which could expose the data. AirWatch was notified through a responsible disclosure process and fixed the vulnerability in a subsequent version of the AirWatch Inbox app. We independently verified that the issue was addressed.

The vulnerability will be described here not as criticism of the product, but to highlight the challenge of securing data on a mobile device and to illustrate some of the methods in which data leakage could occur.

As described above, AirWatch stores attachments with a plaintext title, but with the contents encrypted. They are stored in the /data/data/com.airwatch.email/files/Attachments/ location. However, when a user needs to view an attachment on the mobile device, it must be decrypted. The email client accomplishes this by storing a decrypted copy of the attachment in /data/data/com.airwatch.email/files/Decrypt.

Figure 9 – contents of the Decrypt folder

The screenshot above shows that the decrypted attachments are pre-pended with 'dec' to indicate their status as decrypted. The contents can now be clearly seen:
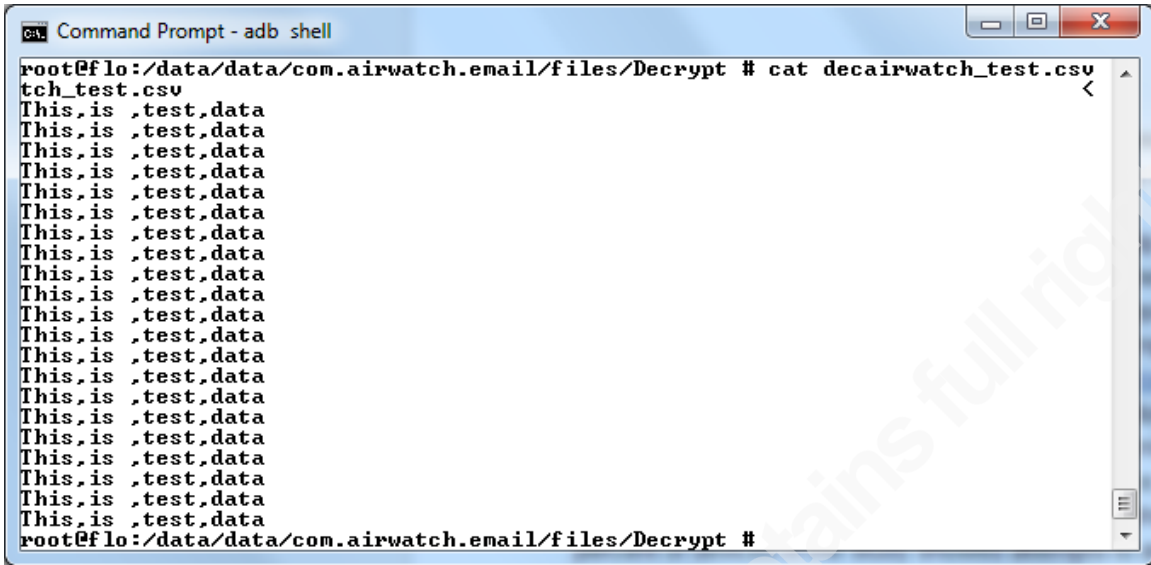
Tim Collyer, tccollyer@gmail.com

```
Command Prompt - adb shell
root@flo:/data/data/com.airwatch.email/files/Decrypt # cat decairwatch_test.csv
tch_test.csv
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
This,is ,test,data
root@flo:/data/data/com.airwatch.email/files/Decrypt #
```

Figure 10 – contents of the decrypted attachment airwatch_test.csv (prepended with 'dec' by the app)

So far, the container is operating as might be expected and as is necessitated for functionality. There are two factors which rest on top of the attachment decryption process which lead to the data leakage vulnerability.

The first factor has already been mentioned – the mechanism for data wipe. When a mobile device needs to have its data wiped, one consideration that MDM suites must take into account is that speed of the process can be important. If the data wipe took a long time, an attacker might be able to subvert or stop the process and thereby gain access to the sensitive data. The simplest and most expedient method to destroy the data quickly is to delete the encryption key. Without the encryption key, the data is rendered effectively useless. It should be noted that the data is still present on the device however, so the term "device wipe" is a bit of a misnomer.

The presence of the data after a device wipe becomes important in light of the contents of the Decrypt directory. As has been demonstrated, the files stored in Decrypt are an unencrypted state. The data leakage vulnerability comes into play because the contents of that folder are not removed after the attachment has been read. The screenshot above with the contents of the Decrypt folder was take several months after the attachments were

Tim Collyer, tccollyer@gmail.com

downloaded and viewed, and more importantly, after the device wipe command was issued and executed.

When a remote wipe is issued, the end user is provided the option to remove the AirWatch email application from the device. By selecting 'No', the application, and all of its contents, remain. The device appears in the AirWatch management console to have been successfully wiped. Unfortunately, any attachments which had been opened on the device remain, and in an unencrypted state.

The purpose of describing the vulnerability is to highlight some of the challenges surrounding containerized data stored on an Android device that is owned by the end user. Because of the hostile nature of the environment on a BYOD Android phone, minor vulnerabilities could lead to loss of data, as illustrated here. This should be a consideration when evaluating what devices will be permitted access to organizational data as well as what type of data (how sensitive) should be allowed on the device.

4. Conclusion

The challenge of securing organizational information on a mobile device is complex. Bring Your Own Device programs, which introduce devices owned by end users, significantly increase the difficulty of the challenge. A BYOD device runs untrusted applications, browses to untrusted websites, uses untrusted networks, and operates in physically insecure environments. Android specifically has had numerous problems with security in addition to the untrusted nature of a BYOD environment. Those security issues are related to the openness of the platform, the proliferation of different handset manufacturers and hardware profiles, the largely untended and un-reviewed nature of app stores, and other factors. In short, the BYOD environment for Android devices could not be more hostile from a security perspective.

Additionally, the uneasy relationship between end user ownership and use of the device, and business desires to secure company data results in limited control over devices'

Tim Collyer, tccollyer@gmail.com

configurations. Therefore many policies which might apply to a fully managed device are not appropriate in a BYOD environment. An example might be full device encryption which would protect data on a lost or stolen device. However, implementing full device encryption may require deleting all data on the device to initiate the encrypting process. This would be a particular hassle for end users who may already have personal information and device customization in place.

Mobile device management products provide asset management for mobile devices in a business environment but the capabilities also bleed over into the realm of security. MDM policies for BYOD devices must pick and choose which device restrictions provide the most reduction in risk to sensitive company information. Many of the policies options, such as disabling Non-Market (Google Play) app installation or disabling Wi-Fi auto connect, are intended to limit or remove vectors for likely device compromise. They do not however, directly prevent the installation of malware, or provide a way to detect and respond should malware be present.

Faced with the task of securing information in a hostile environment, many MDM providers implement a "container" in which they attempt to carve out a safe space for sensitive data in addition to the policy-level management for the device. Examination of the AirWatch email container, by way of example, shows that the security of the container hinges on two factors - permissions restrictions and encryption. Application data such as emails and attachments are stored in a folder that is accessible only to the email app and the root account. Additionally, the data is encrypted while stored by using the built-in encryption capabilities of the Android operating system.

Mobile Device Management suites attempt to protect data on a BYOD Android through this multi-layered approach. It should be noted that the end result is a managed device which protects data from being accessed by anything but the MDM app account and the root account. This means that co-opting either of those accounts, especially root, becomes the primary goal for an attacker (external or malicious insider) looking to bypass the restrictions of MDM. Therefore the security focus of Mobile Device Management suites

Tim Collyer, tccollyer@gmail.com

which store data on the phone is on preventing root-level access and detecting when such access has been gained. An attacker with root permissions can exercise full control over a device, including lying to the MDM server about whether the device is compromised or not. Root detection is the lynchpin on which the entire plan for protection hinges.

MDM root detection mechanisms often look for specific files and configurations or for the presence of specific packages and directory permissions (Gruber, 2013). Essentially these come down to a method of signature-based detection for root access. Static detection based upon signatures is a good place to start, but unfortunately it is relatively easy to avoid detection (Gruber, 2013) by changing small details - a fact to which host-based anti-virus applications can attest. Anti-virus is still an important layer of defense, but cannot be relied upon to prevent the installation of malware. Similarly, static root detection is important but should not be counted on for complete effectiveness. This results in a situation in which a determined attacker can still compromise a managed device, bypass the restrictions of MDM, and potentially access the corporate data stored there.

Container-based MDM solutions on BYOD Android devices are therefore best at helping normal users avoid accidental data leakage as well as providing some measure of asset control to administrators. Organizations with a risk averse security posture should therefore approach Android, and specifically end user owned devices, with caution. The multi-layered controls within many containerized MDM products may not yet be sufficient to meet stringent security requirements.

Tim Collyer, tccollyer@gmail.com

5. References

F-Secure (2013) F-Secure Mobile Threat report Q3 2013. Retrieved March 31, 2014
from: http://www.f-
secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q3_2013.pdf

F-Secure (2013) Threat Report H2 2013 Retrieved April 8, 2014 from: http://www.f-
secure.com/static/doc/labs_global/Research/Threat_Report_H2_2013.pdf

Federal Trade Commission (2014, March 28) Fandango, Credit Karma Settle FTC
Charges that They Deceived Consumers By Failing to Securely Transmit
Sensitive Personal Information. Retrieved March 28, 2014 from:
http://www.ftc.gov/news-events/press-releases/2014/03/fandango-credit-karma-
settle-ftc-charges-they-deceived-consumers

Fleming, N. (2014, March 28) How secure is that mobile app? Retrieved March 28, 2014
from: http://www.onguardonline.gov/blog/how-secure-mobile-app

Gordon, S. (2013, February 27) Perimeter Security: Evolved, Not Dead Retrieved March
28, 2014 from: http://www.infosecurity-magazine.com/view/30983/perimeter-
security-evolved-not-dead/

Gruber, E. (2013, August 15) Bypassing AirWatch Root Restriction. Retrieved January 4,
2014 from: https://www.netspi.com/blog/entryid/192/bypassing-airwatch-root-
restriction

Gruber, E. (2013, December 2) Android Root Detection Techniques. Retrieved April 22,
2014 from: https://www.netspi.com/blog/entryid/209/android-root-detection-
techniques

Juniper Networks, Inc (2013). Juniper Networks Third Annual Mobile Threat Report.
Retrieved February 14, 2014 from:
http://www.juniper.net/us/en/local/pdf/additional-resources/jnpr-2012-mobile-
threats-report.pdf

Lehr, J. (2012, October 31) Android Pin/Password Cracking: Halloween isn't the Only
Scary Thing in October. Retrieved March 25, 2014 from:

Tim Collyer, tccollyer@gmail.com

http://linuxsleuthing.blogspot.co.uk/2012/10/android-pinpassword-cracking-halloween.html

Leopando, J. (2014, January 20) Looking Forward Into 2014: What 2013's Mobile Threats Mean Moving Forward. Retrieved February 26 2014, from: http://blog.trendmicro.com/trendlabs-security-intelligence/looking-forward-into-2014-what-2013s-mobile-threats-mean-moving-forward/

Makan, K., & Alexander-Bown, S. (2013). Android Security Cookbook. Birmingham, UK: Packt Publishing LTD

Mathias, C. (2013, March 01), Minimizing BYOD security risks through policy and technology. Retrieved March 18, 2014 from: http://searchconsumerization.techtarget.com/tip/Minimizing-BYOD-security-risks-through-policy-and-technology

National Institute of Standards and Technology. (2013). Guidelines for Managing the Security of Mobile Devices in the Enterprise (NIST Special Publication 800-124 Rev. 1). Retrieved from: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-124r1.pdf

Norton, K. (2012, October 24) Mobile Security: 6 Reasons Devices Remain Vulnerable. Deloitte CIO Journal, The Wall Street Journal. Retrieved March 18, 2014 from: http://deloitte.wsj.com/cio/2012/10/24/mobile-security-6-reasons-devices-remain-vulnerable/

Pellegrino, E., Elyashiv, E., Murtagh, & Avraham, Z. (2013) Mobile Intrusion Prevention Landscape. Retrieved March 25, 2014 from: https://www.zimperium.com/whitepapers/Mobile%20Intrusion%20Prevention-Threat-Landscape-Whitepaper.pdf

Pew Research Center (2014) Device Ownership Over Time. Retrieved March 25, 2014 from: http://www.pewinternet.org/data-trend/mobile/device-ownership/

Pew Research Center (2014) Mobile Technology Fact Sheet. Retrieved March 25, 2014 from: http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/

Tim Collyer, tccollyer@gmail.com

Phifer, L. (2013, Jan 28) BYOD security strategies: Balancing BYOD risks and rewards. Retrieved March 18, 2014 from: http://searchsecurity.techtarget.com/feature/BYOD-security-strategies-Balancing-BYOD-risks-and-rewards

Schadler, T. (2013, February 4) 2013 Mobile Workforce Adoption Trends. Retrieved February 26, 2014 from: http://www.vmware.com/files/pdf/Forrester_2013_Mobile_Workforce_Adoption_Trends_Feb2013.pdf

SecretKeyFactory (n.d.). In Android Developer Reference. Retrieved from: http://developer.android.com/reference/javax/crypto/SecretKeyFactory.html

Sreenivas, G. (2013) Securing BYOD: Mitigating Risk, not Forcing Control! [Powerpoint Slides]. Retrieved March 25, 2014 from: http://www.rsaconference.com/writable/presentations/file_upload/mbs-w07-securing-byod-mitigating-risk_-not-forcing-control_.pdf

State of Mobile Security (2014, February 20) Deloitte CIO Journal, The Wall Street Journal. Retrieved March 18, 2014 from: http://deloitte.wsj.com/cio/2014/02/20/the-state-of-mobile-security/

Su, J., Wei, T., & Zhai, J. (2014, March 27) A Little Bird Told Me: Personal Information Sharing in Angry Birds and its Ad Libraries. Retrieved March 28, 2014 from: http://www.fireeye.com/blog/technical/2014/03/a-little-bird-told-me-personal-information-sharing-in-angry-birds-and-its-ad-libraries.html

Wright, J. (2013, December) Fear and Loathing in BYOD. Retrieved February 18, 2014 from: http://www.sans.org/reading-room/analysts-program/fear-loathing-byod-survey

Wright, S. (2012) The Symantec Smartphone Honey Stick Project. Retrieved March 18, 2014 from: http://www.symantec.com/content/en/us/about/presskits/b-symantec-smartphone-honey-stick-project.en-us.pdf

Tim Collyer, tccollyer@gmail.com

Wu, L., Grace, M., Zhou, Y. Wu, C., & Jiang, X. (2013) The Impact of Vendor
    Customizations on Android Security. Retrieved March 31, 2014 from:
    http://www.csc.ncsu.edu/faculty/jiang/pubs/CCS13.pdf

Zhou, Y., & Jiang, X. (2012) Dissecting Android Malware: Characterization and
    Evolution. Retrieved February 26th, 2014 from:
    http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf

Tim Collyer, tccollyer@gmail.com