

Report
On
“Mobile Compliance”

Prepared By:
Meghna Baijal (2008B3A7620G)

At
VMware, Bangalore

A Practice School Station Of
Birla Institute of Technology and Science – Pilani

A
Report
On
"Mobile Compliance"

Prepared By:
Meghna Baijal (2008B3A7620G)

Prepared in fulfillment of the
Practice School 2 Course

At
VMware, Bangalore

A Practice School Station Of
Birla Institute of Technology and Science – Pilani
January-June, 2013

Birla Institute of Technology and Science – Pilani

Practise School Division

Station: **VMware**

Centre: **Bangalore**

Duration: **6 months**

Date of Start: **7th January 2013**

Date of Submission: **6th June 2013**

Title of the project: **Mobile Compliance**

Details of Student:-

Name: **Meghna Baijal**

ID No: 2008B3A7620G

Discipline: BE Computer Science, MSc Economics

Details of Experts:-

Mr. Sanjay Swami Sr. Manager, VCM

Mr. Arunvijai Sridharan Manager, CP&C

Mr. Pravin Goyal Mentor

Name of the PS Faculty: **Ms Rita Gopalan, Ms Ritu Arora**

Key Words: **VCM, CP&C, Security, Compliance, Android, CIS Benchmark, Eclipse, Rooting, ADB, SDK, ADT plugin, app, emulator, iOS, public API, Private API**

Project Area: **Mobile Compliance Checker based on the CIS Security Benchmark for Google Android 2.3 and for iOS 5.0. Event Driven Compliance**

BIRLA INSTITUTE OF SCIENCE AND TECHNOLOGY, PILANI (RAJASTHAN)

PRACTICE SCHOOL DIVISION

Response option sheet

PS Station: VMware Centre Bangalore

ID No. & Name(s) 2008B3A7620G, MEGHNABAIJAL

Title of the Project: MOBILE COMPLIANCE CHECKER

Usefulness of the project to the on-campus of study in various disciplines. Project should be scrutinized keeping in view the following response options. Write course No. and Course Name against the option under which the project comes.

Refer Bulletin for Course No. and Course Name.

Code No	Response Options	Course No. (s) & Name
1.	A new course can be designed out of the Project	NA
2.	The project can help modification of the course content of some of the existing courses.	NA
3.	The project can be used directly in some of the existing compulsory Discipline Courses (CDC)/Discipline Courses Other than Compulsory (DCOC) /Emerging Area (EA) etc. Courses.	NA
4.	The project can be used in preparatory courses like analysis and Application Oriented Courses (AAOC)/ Engineering Science (ES)/ Technical; Art (TA) and Core Courses	NA
5.	This project cannot come under any of the above mentioned options as it relates to the professional work of the host organization	Yes

Signature of Student

Signature of Faculty

Abstract

With the advent of low-cost Internet and the massive proliferation of mobile devices, there is an exponential rise in security attack vectors. The increased usage and dependence on these mobile devices has led to new challenges - keeping them managed and secured.

How do organizations resolve the problem of efficiently and effectively managing these devices and to what level? There are many Mobile Device Management (MDM) solutions available, but their capabilities are limited. The existing solutions miss the core capabilities of this solution, such as automated device detection, centralized micro-level management, and compliance.

At present, there is no sure or proven way to automatically determine the presence of Android devices on the network. Also, the level of information captured from the mobile devices by existing solutions is limited.

This project is about developing a mobile compliance management solution. Some of the key deliverables (areas to work on) for the project are (for both Android and iOS):

- 1) Automated network discovery of a mobile device in the corporate network
- 2) Be able to push an app remotely on the mobile device using exchange or similar technologies
- 3) Collect device specific information to facilitate compliance management
- 4) Extending compliance management to mobile device management as far as possible

Signature of student

Signature of PS Faculty

Acknowledgements

I would like to thank my mentor **Mr. Pravin Goyal** for his constant support, patient guidance and encouragement. I would also like to express gratitude to our CP&C manager **Mr. Arunvijai Sridharan** and our VCM team manager **Mr. Sanjay Swami** for all their help.

I would also like to thank our BITS PS instructors **Ms. Rita Gopalan** and **Ms. Ritu Arora** for all their guidance.

I am also, extremely thankful to all fellow interns at VMware for making this period fun and comfortable and to **BITS** for providing me with this wonderful opportunity.

Meghna Baijal

Table of Contents

Response option sheet	4
Abstract	5
Acknowledgements	6
Table of Figures	10
Chapter 1: About VMware.....	11
1.1 Introduction	11
1.2 History	11
Chapter2: Project Description	12
2.1 Android Mobile Compliance Checker	12
2.2 Android Network Discovery	12
2.3 Android Device Information Collection	12
2.4 Android Event Driven Compliance.....	12
2.5 Android Secure File Transfer	13
2.6 iOS Mobile Compliance Checker	13
2.7 iOS Event Driven Compliance	13
Chapter 3: Project Significance	14
3.1 Android Security	14
3.2 Extending the project to iOS	15
Chapter 4: Background and Environment.....	16
4.1 Android Architecture	16
4.2 iOS Architecture.....	17
Chapter 5: Tools Used	19
5.1 Android Tools	19
5.2 iOS Tools.....	20
Chapter 6: Design	22
6.1 Android Architecture Diagrams	22
6.1.1 Component Diagram.....	22
6.1.2 Deployment Diagram	23
6.1.3 Class Diagram.....	24
6.1.4 Sequence Diagram	25

6.1.5 Use Case Diagram	26
6.1.6 State Diagram	27
6.2 iOS Architecture Diagrams	28
6.2.1 Component Diagram.....	28
6.2.2 Deployment Diagram	28
6.2.3 Class Diagram.....	29
6.2.4 State Diagram	30
Chapter 7: Methodology & Development	31
7.1 Android Compliance Checker App	31
7.1.1 STEP 1: The CIS Security Benchmark	31
7.1.2 STEP 2: Implementation Issues.....	31
7.1.3 STEP 3: Pulling the Data.....	33
7.1.4 The Compliance Checker App	33
7.2 Android Network Discovery	36
7.2.1 NMAP.....	36
7.2.2 AngryIP	37
7.2.3 Ike-scan.....	38
7.2.4 Android Apps	38
7.2.5 Android Apps: Apps that require a Rooted Device.....	39
7.2.6 Android Apps: to run these apps on an emulator	40
7.2.7 MAC Address	41
7.2.8 MDM- Existing Technology	41
7.2.9 MDM- APIs.....	42
7.2.10 MDM- Google's Take	43
7.2.11 Google Play	43
7.2.12 Microsoft Exchange.....	43
7.2.13 WAP Push.....	44
7.2.14 C2DM & GCM.....	44
7.2.15 Avahi Daemon/ Zeroconf/ Apple-Bonjour/ mDNS.....	45
3.2.16 The Net Admin	46
7.3 Android - Device Information Collection	47

7.4 Android - Event Driven Compliance	48
App Screenshots	49
7.5 Android - Secure File Transfer.....	50
7.6 Android - Secure File Transfer.....	51
7.6.1 Step 1: Private vs public API.....	51
7.6.2 Step 2: iOS app deployment options	51
7.6.3 Step 3: Data collection using Public vs Private API	53
7.6.4 Step 4: App Status	54
7.7 iOS – Event Driven Compliance	55
Chapter 8: Testing	56
8.1 Android Testing.....	56
8.1.1 Testing on the emulator/simulator	56
8.1.2 Device Testing.....	56
8.2 iOS Testing.....	57
8.2.1 Testing on the emulator/simulator	57
8.2.2 Device Testing.....	57
Chapter 9: Scope Of The Project	58
Chapter 10: Summary And Conclusions.....	60
10.1 Android Mobile Compliance Checker	60
10.2 Android Network Discovery	60
10.3 App Installation	60
10.4 Android Device Information Collection	60
10.5 iOS Mobile Compliance Checker	61
10.6 Android & iOS Event Driven Compliance:	61
10.7 iOS Mobile Compliance Checker	61
10.8 Android Secure File Transfer.....	61
References	62

Table of Figures

Figure 1 - iOS Market Share (2013).....	15
Figure 2 - Android Architecture	17
Figure 3 - iOS Architecture	18
Figure 4 - Android Component Diagram	22
Figure 5 - Android Deployment Diagram	23
Figure 6 - Android Class Diagram	24
Figure 7 - Android Sequence Diagram	25
Figure 8 - Android Use Case Diagram	26
Figure 9 - Android State Diagram.....	27
Figure 10 - iOS Component diagram.....	28
Figure 11 - iOS Deployment Diagram.....	29
Figure 12 - iOS Class Diagram.....	29
Figure 13 - iOS State Diagram	30
Figure 14 - The CC App.....	33
Figure 16 - App Help Screen.....	34
Figure 15 - App Compliance Report	34
Figure 17 - App Save Screen.....	35
Figure 18 - App Final Screen	35
Figure 19 - App Generated Report.....	36
Figure 20 - NMAP	37
Figure 21 - AngryIP	37
Figure 22 - Ike-scan	38
Figure 23 - Android Apps for discovery.....	39
Figure 24 - WAP Push.....	44
Figure 25 - The XML	47
Figure 26 - Event Driven On Android	48
Figure 27 – Event Driven on Android App.....	49
Figure 28 - File Transfer	50
Figure 29 - iOS App.....	54
Figure 30 - Android Emulator.....	56
Figure 31 - iOS simulator.....	57

Chapter 1: About VMware

1.1 Introduction

VMware, the global leader in virtualization and cloud infrastructure, delivers customer-proven solutions that accelerate IT by reducing complexity and enabling more flexible, agile service delivery. VMware enables enterprises to adopt a cloud model that addresses their unique business challenges. VMware's approach accelerates the transition to cloud computing while preserving existing investments and improving security and control. With more than 400,000 customers and 55,000 partners, VMware solutions help organizations of all sizes lower costs, increase business agility and ensure freedom of choice.

1.2 History

In 1998 VMware was founded by Diane Greene, Mendel Rosenblum, Scott Devine, Edward Wang and Edouard Bugnion.

The company has its headquarters in Palo Alto, California, United States, and established an R&D Center in Cambridge, Massachusetts, as well as one at the Time Warner Center in New York City, in 2005.

VMware operated throughout 1998 in stealth mode with roughly 20 employees by the end of that year. The company was launched officially in February 1999 at the DEMO Conference organized by Chris Shipley.

VMware delivered its first product, VMware Workstation, in May 1999 and entered the server market in 2001 with VMware GSX Server (hosted) and VMware ESX Server (hostless). In 2003 VMware launched VMware Virtual Center, the VMotion and Virtual SMP technology. 64-bit support appeared in 2004. The company was acquired by EMC Corporation in 2004.

Chapter2: Project Description

2.1 Android Mobile Compliance Checker

This project is to create the similar or a lighter version of vCM with at least major functionalities of compliance, patching and reporting. VMware vCenter Configuration Manager's policy-driven automation detects deep system changes and identifies whether that change is within policy - an expected and acceptable behaviour based on industry, regulatory, or your own self-defined best practices - or whether that change has created a compliance violation or security vulnerability. The idea is to make sure that vCM can be run as an app on the mobile devices like ipad, android devices which IT admins of today/tomorrow would use to manage their infrastructure. This version of vCM must have the same level of security features, if not more than the existing full version like, there must be a mechanism that the user is authenticated to use the vCM and it must be usable only when the device is on the corporate network, either directly or remotely (through VPN)

2.2 Android Network Discovery

To be able to detect all android mobile devices on the company's private network and/or to push the app remotely onto compatible devices.

2.3 Android Device Information Collection

To create an android agent for VCM that collects all possible device information and then make an extensive xml out of it to display all of it in an easily readable and usable format

2.4 Android Event Driven Compliance

The data collection should happen each time a certain event occurs. The event could be a network connection, an app installation/removal, a change in some device setting etc

2.5 Android Secure File Transfer

The XML with the extracted device information stored on the phone sdcard. The need is to upload the file from the device onto a server to analyze and use it

2.6 iOS Mobile Compliance Checker

To extend the mobile compliance checker project to include iOS. iOS covers a vast market in the smartphone world and it was realized that no Mobile app research project could be completed without taking iOS into consideration.

2.7 iOS Event Driven Compliance

The data collection should happen each time a certain event occurs. The event could be a network connection, an app installation/removal, a change in some device setting etc.

Chapter 3: Project Significance

3.1 Android Security

Android is a modern mobile platform that was designed to be truly open. Android applications make use of advanced hardware and software, as well as local and served data, exposed through the platform to bring innovation and value to consumers. To protect that value, the platform must offer an application environment that ensures the security of users, data, applications, the device, and the network.

Securing an open platform requires robust security architecture and rigorous security programs. Android was designed with multi-layered security that provides the flexibility required for an open platform, while providing protection for all users of the platform.

Android was designed with developers in mind. Security controls were designed to reduce the burden on developers. Security-savvy developers can easily work with and rely on flexible security controls. Developers less familiar with security will be protected by safe defaults.

Android was designed with device users in mind. Users are provided visibility into how applications work, and control over those applications. This design includes the expectation that attackers would attempt to perform common attacks, such as social engineering attacks to convince device users to install malware, and attacks on third-party applications on Android. Android was designed to both reduce the probability of these attacks and greatly limit the impact of the attack in the event it was successful.

3.2 Extending the project to iOS

The two biggest players in the Smartphone game are Apple's iOS and Google's Android. Just when it was thought that Android was on the rise, it appears that the iOS Smartphone market share is actually growing, breaking the 50% mark in the United States. Recently, a report was issued saying that iOS had a 48.1% market share when it came to smartphones, compared to the combined 46.7% enjoyed by Android devices. The lead was slight, but it was there. The assumption is that this has to do with the launch of the iPhone 5, despite the many troubles that it has faced. Android's market share shrank to 42% over the same period.

The outlook is that the iPhone's Smartphone market share will continue to rise in the US now that it's broken the 50% mark.

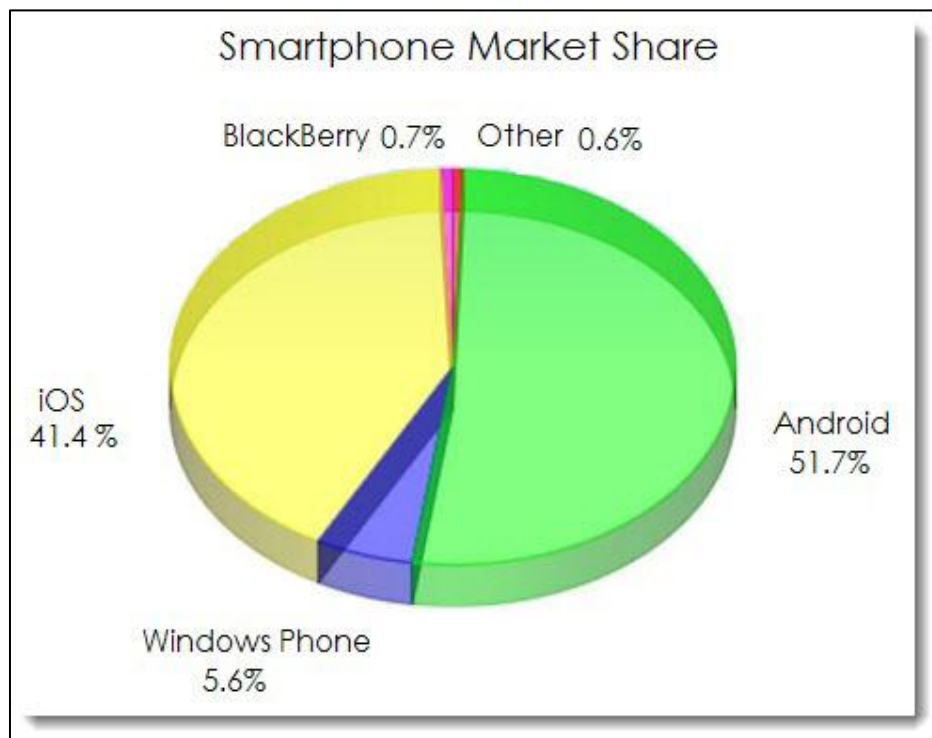


Figure 1 - iOS Market Share (2013)

Chapter 4: Background and Environment

4.1 Android Architecture

The main Android platform building blocks are:

- **Device Hardware:** Android runs on a wide range of hardware configurations including smart phones, tablets, and set-top-boxes. Android is processor-agnostic, but it does take advantage of some hardware-specific security capabilities such as ARM v6 eXecute-Never.

- **Android Operating System:** The core operating system is built on top of the Linux kernel. All device resources, like camera functions, GPS data, Bluetooth functions, telephony functions, network connections, etc. are accessed through the operating system.

- **Android Application Runtime:** Android applications are most often written in the Java programming language and run in the Dalvik virtual machine. However, many applications, including core Android services and applications are native applications or include native libraries. Both Dalvik and native applications run within the same security environment, contained within the Application Sandbox. Applications get a dedicated part of the filesystem in which they can write private data, including databases and raw files.

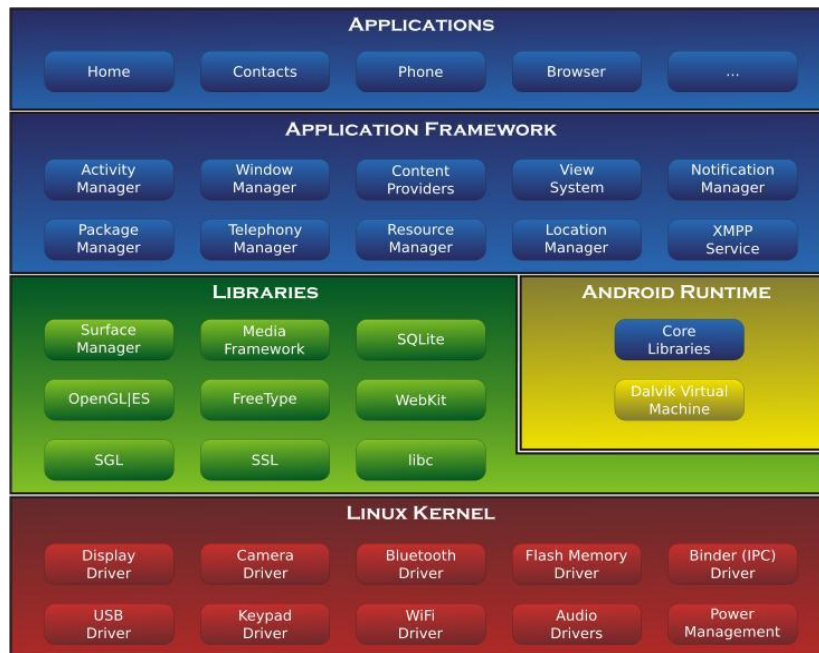


Figure 2 - Android Architecture

4.2 iOS Architecture

- **Application:** This is the currently-running iPhone application, purchased through the app store (unless jailbroken firmware is used). This application was compiled to native code by the Apple-distributed iPhone compiler, and linked with the Objective-C runtime and C library by the linker. This application also runs entirely within the userspace environment set up by the iPhone OS.
- **Frameworks/API:** Cocoa Touch, upper-level OpenGL calls, its all in here. These API calls are simply Apple-distributed headers with the iPhone SDK, with some dynamic linking occurring at runtime. These reside on top of the Objective-C runtime, as many of these are written in Objective-C.
- **Objective-C runtime:** This layer is comprised of both the Objective-C dynamically-linked runtime libraries, as well as the underlying C libraries. The C library sets up the environment for the Objective-C runtime so much that I simply included them both within the same layer (although I should have probably called it “runtime libraries” instead).

- **iPhone OS:** This is the kernel, drivers, and services that comprise of the iPhone Operating System. This is sometimes called iPhone OS, iPhone OS X, or just OS X, but it all refers to the same deal: it sits between the userspace and hardware.
- **Processor:** Not so much the physical ARM chip (that's contained within the hardware layer), but instead referring to the ARM instruction set and the interrupt descriptor table as set up by the iPhone OS during boot and driver initialization.
- **Firmware:** Although we refer to the entire OS as “firmware” sometimes (especially in respect to jailbreaking), this layer instead references the chip-specific code that is either contained with memory in/around the peripheral itself, or within the driver for said peripheral (example: touch screen or gyroscope)
- **Hardware:** Pretty obvious, but refers to the physical chips soldered to the iPhone's circuitry. If you can feel/see it, it's under this layer. The actual processor falls under this layer, but the instruction set and in-memory descriptor tables are contained within the “processor” layer.

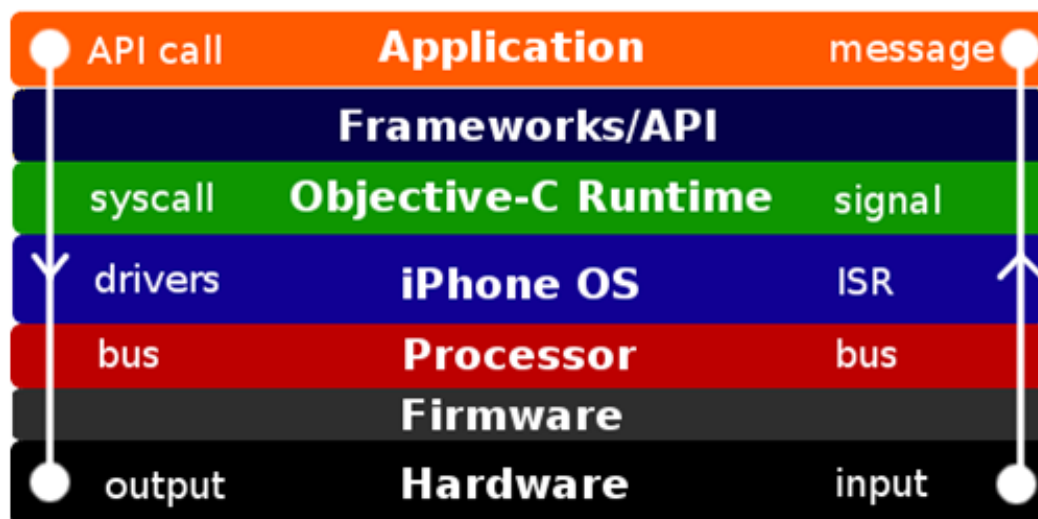


Figure 3 - iOS Architecture

Chapter 5: Tools Used

5.1 Android Tools

1. Eclipse IDE for Java

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Haskell, Perl, PHP, Python etc. Android requires the Eclipse Java development tools (JDT) for Java.

2. The Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plugin.

3. The ADT plugin

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files

in order to distribute your application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications.

4. **ADB shell**

Android Debug Bridge (ADB) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program.

5. **DDMS**

Android ships with a debugging tool called the Dalvik Debug Monitor Server (DDMS), which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more. This page provides a modest discussion of DDMS features; it is not an exhaustive exploration of all the features and capabilities.

5.2 iOS Tools

1. **Xcode**

Xcode provides tools to manage your entire development workflow—from creating your app and designing your user interface, to testing, optimizing, and shipping your app to the App Store.

2. Objective-C

Objective-C is an elegant object-oriented language that powers all iOS apps. You write Objective-C code to create your app, and you need to understand this language in order to use most application frameworks. Although you can use other languages for development, you can't build an iOS app without Objective-C.

3. The iOS SDK

An app consists of code that you write and frameworks provided by Apple. A framework contains a library of methods that your app can call. Apple writes frameworks that anticipate the basic features you might want to implement. You should use frameworks both to save time and effort and to make sure your code is efficient and secure. The system frameworks are the only way to access the underlying hardware.

Chapter 6: Design

6.1 Android Architecture Diagrams

6.1.1 Component Diagram

The Component Diagram depicts the three different parts of the app.

1. **Device Information Collection:** This is the main component of the app that scans the device for information. It receives event notifications from the Event Driven Compliance component and acts on it. After this component finishes its task, it provides the XML file to the third component Secure File transfer.
2. **Event driven Compliance:** this component receives notification from the android system as soon as any event occurs out of these three: wifi connection change, app install or app removal. It then fires an intent to the second component, Device information collection that and prompts it to recollect device data
3. **Secure File Transfer:** receives the XML file from the second component, device information collection. It also receives server credentials that are either hardcoded into the app or else entered by the user. It Then uploads the file to the server via HTTPS POST.

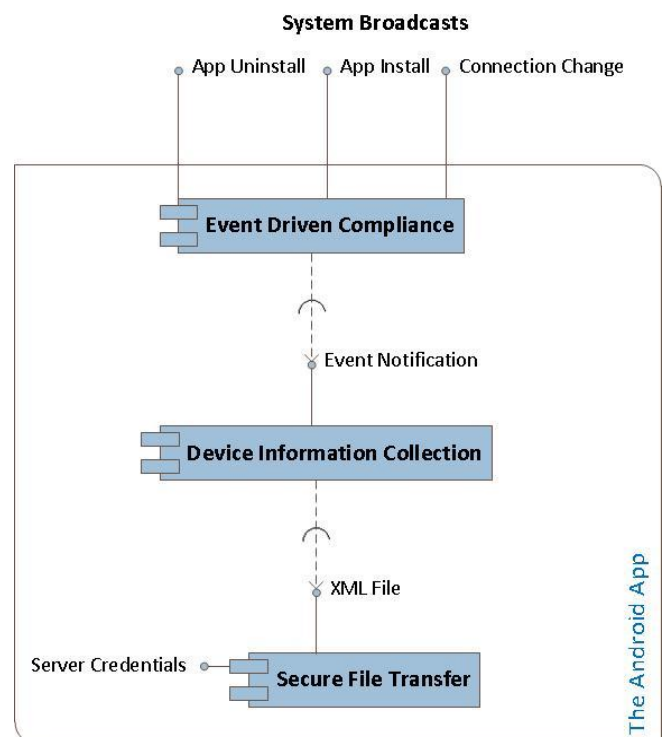


Figure 4 - Android Component Diagram

6.1.2 Deployment Diagram

The deployment diagram lists the hardware and software components from the actual deployment and runtime point of view.

1. **The Android device:** The user phone onto which the app's .apk installed. After install the android system is responsible for sending system broadcasts to the app when events occur.
2. **The Android App:** This app does the three main tasks as listed as a part of the component diagram: collect data, listen for events, upload XML file to server.
3. **The WAMP server:** The XML file is uploaded from the android device to the Apache HTTPS server.
4. **Database:** The data from all android devices in the enterprise is stored in a database and used as and when required.

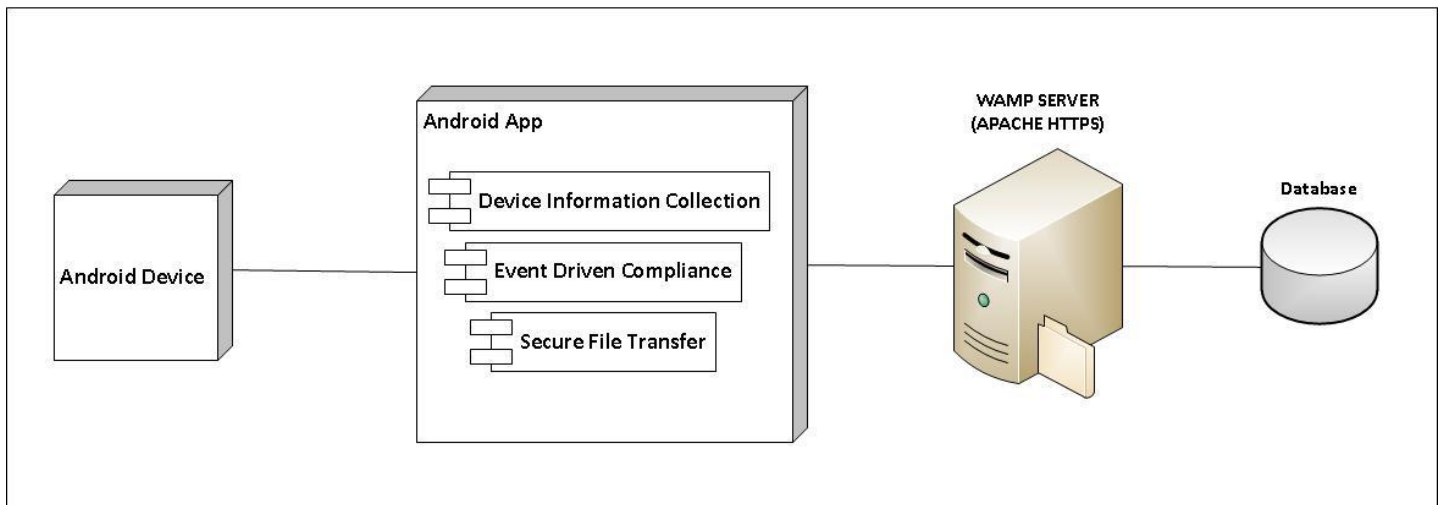


Figure 5 - Android Deployment Diagram

6.1.3 Class Diagram

The UML Class diagram depicts all the classes used and created in the app:

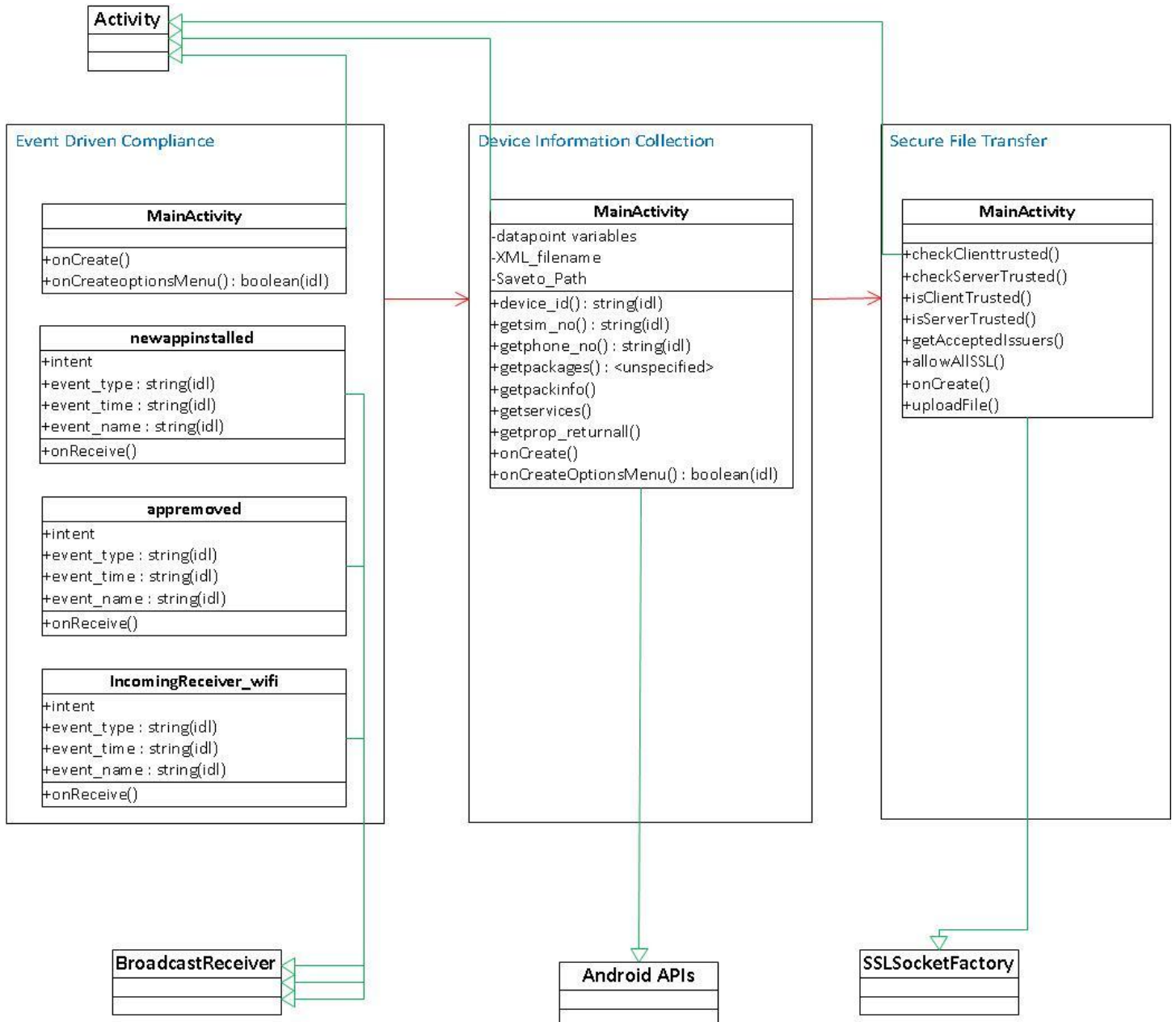


Figure 6 - Android Class Diagram

6.1.4 Sequence Diagram

The sequence diagram details the events as they occur in time:

1. The Android agent is installed onto the device and prompts data collection.
2. XML created and stored to phone memory
3. The agent has now registered to listen to certain events.
4. Event occurs -> system broadcasts message to all apps on device -> agent collects it -> notifies collector class -> recollection of data happens -> XML stored to phone memory
5. User wants to upload to server -> enters credentials -> connect to server -> file uploaded.
6. Infinite loop to wait for events...

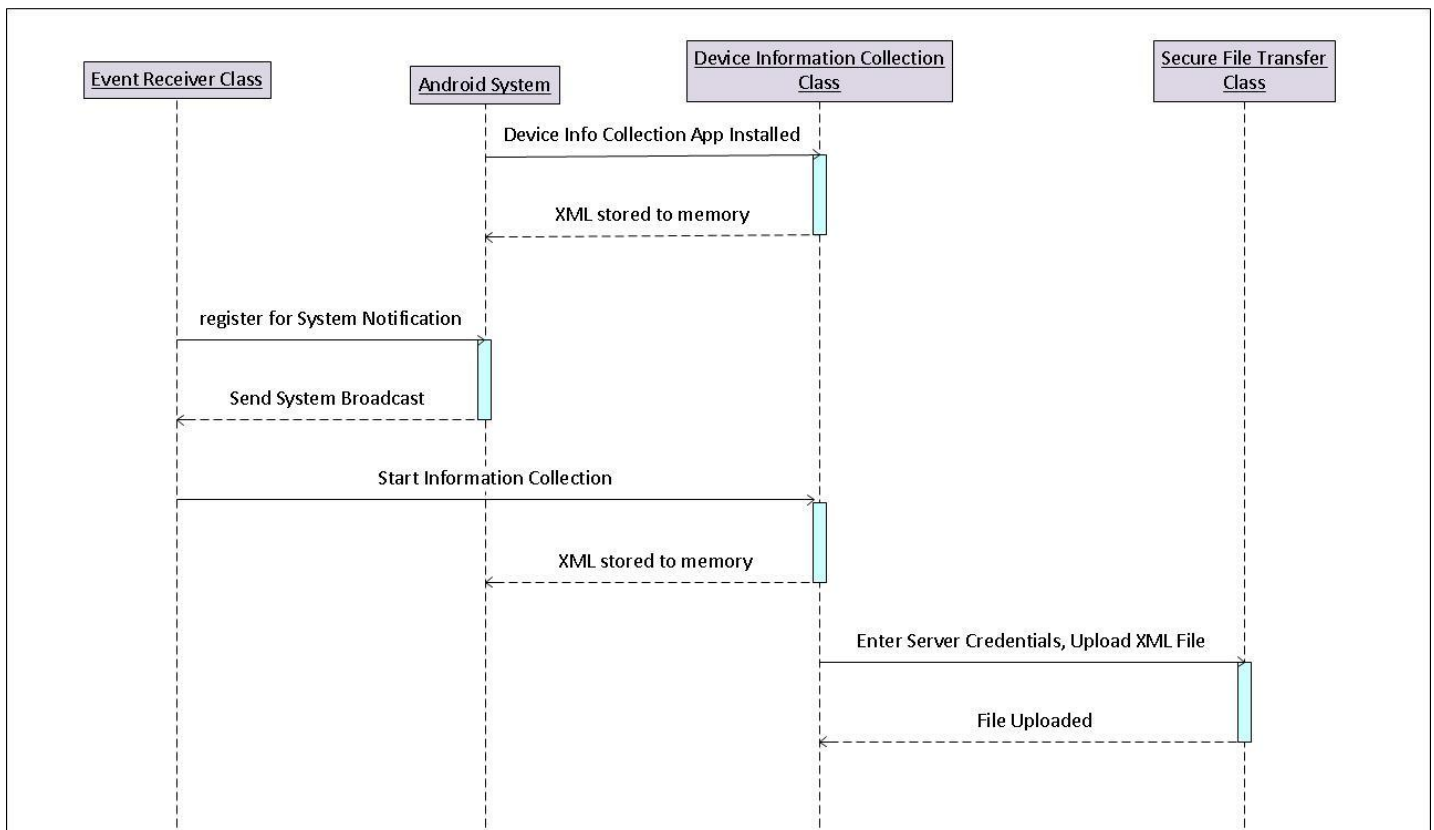


Figure 7 - Android Sequence Diagram

6.1.5 Use Case Diagram

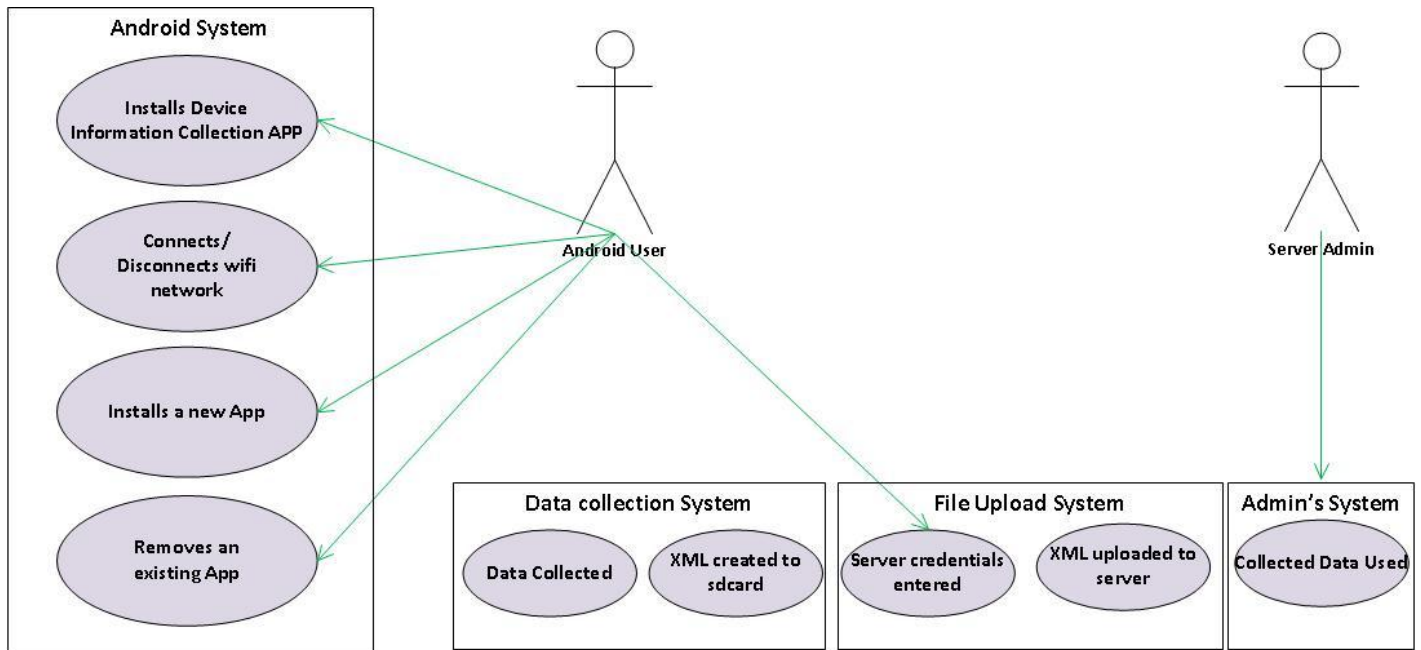


Figure 8 - Android Use Case Diagram

Use Case Diagram shows the app from the user and the admin point of view.

1. The User:

- Installs app and hence prompts data collection
- Changes wifi connection
- Installs a new app
- Removes an existing app
- After the XML is saved to the memory, the user prompts file upload
- The user enters server credentials

2. The Admin:

- Gets the uploaded file from all android devices
- Puts data into a database
- Uses the data as and when required for compliance checking or analysis

6.1.6 State Diagram

Demonstrates the different states the android mobile device goes through as the app is installed, run and removed from it.

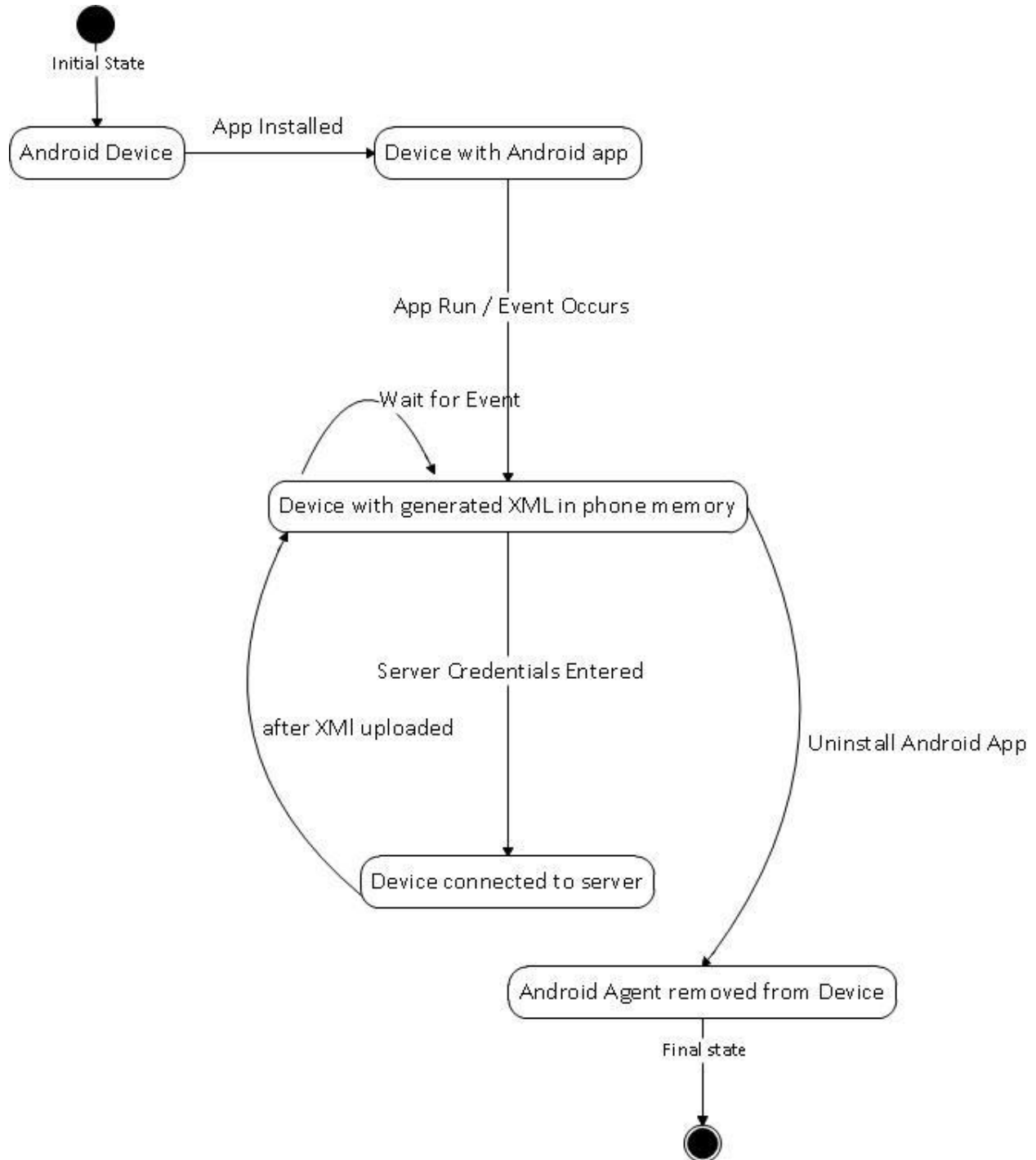


Figure 9 - Android State Diagram

6.2 iOS Architecture Diagrams

6.2.1 Component Diagram

The Component Diagram depicts the two different parts of the app:

1. **Device Information Collection:** This is the main component of the app that scans the device for information. It receives event notifications from the Event Driven Compliance component and acts on it. After this component finishes its task, the data can be saved in an XML format to the device memory and be transferred to a server.
2. **Event driven Compliance:** this component receives notification from the system as soon as an event occurs ie a wifi connection change. It then calls a method that prompts recollection of the device data.

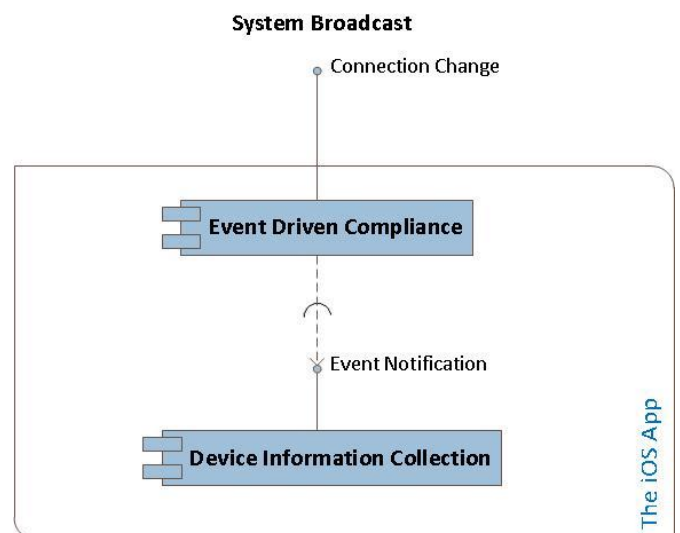


Figure 10 - iOS Component diagram

6.2.2 Deployment Diagram

The deployment diagram lists the hardware and software components from the actual deployment and runtime point of view.

1. **The iOS device:** The user phone onto which the app is installed. After install the iOS system is responsible for sending system broadcasts to the app when events occur.
2. **The iOS App:** This app does the main tasks as listed as a part of the component diagram: collect data and listen for events.

3. **The server and Database:** The data can be saved and transferred to the server and included in a database for use at enterprise level.

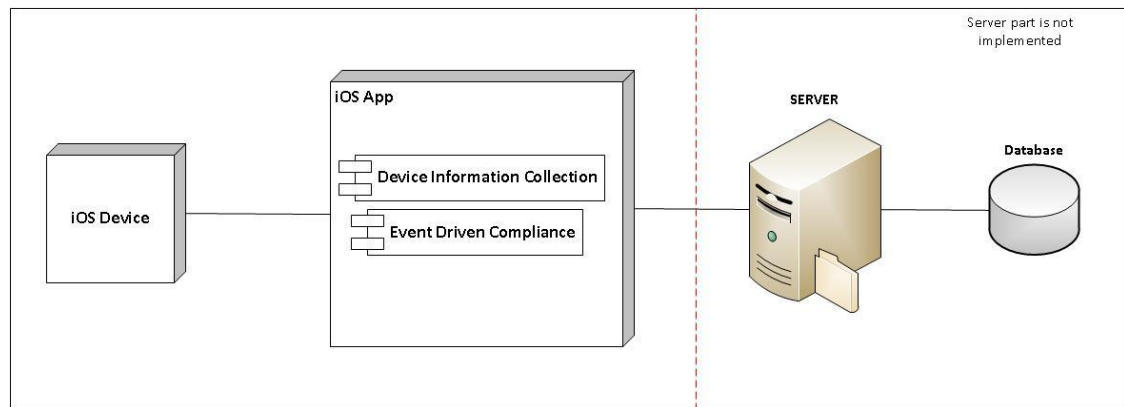


Figure 11 - iOS Deployment Diagram

6.2.3 Class Diagram

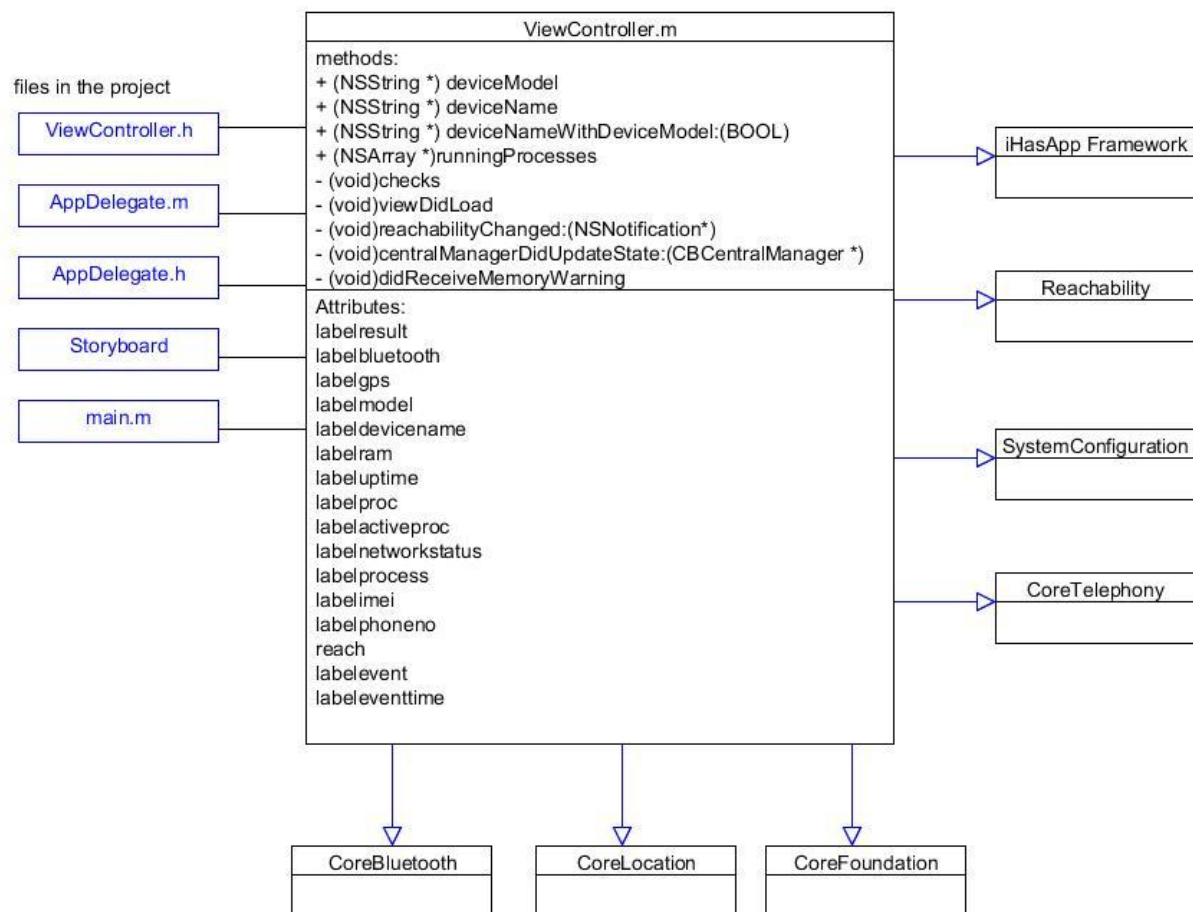


Figure 12 - iOS Class Diagram

6.2.4 State Diagram

Demonstrates the different states the iOS mobile device goes through as the app is installed, run and removed from it.

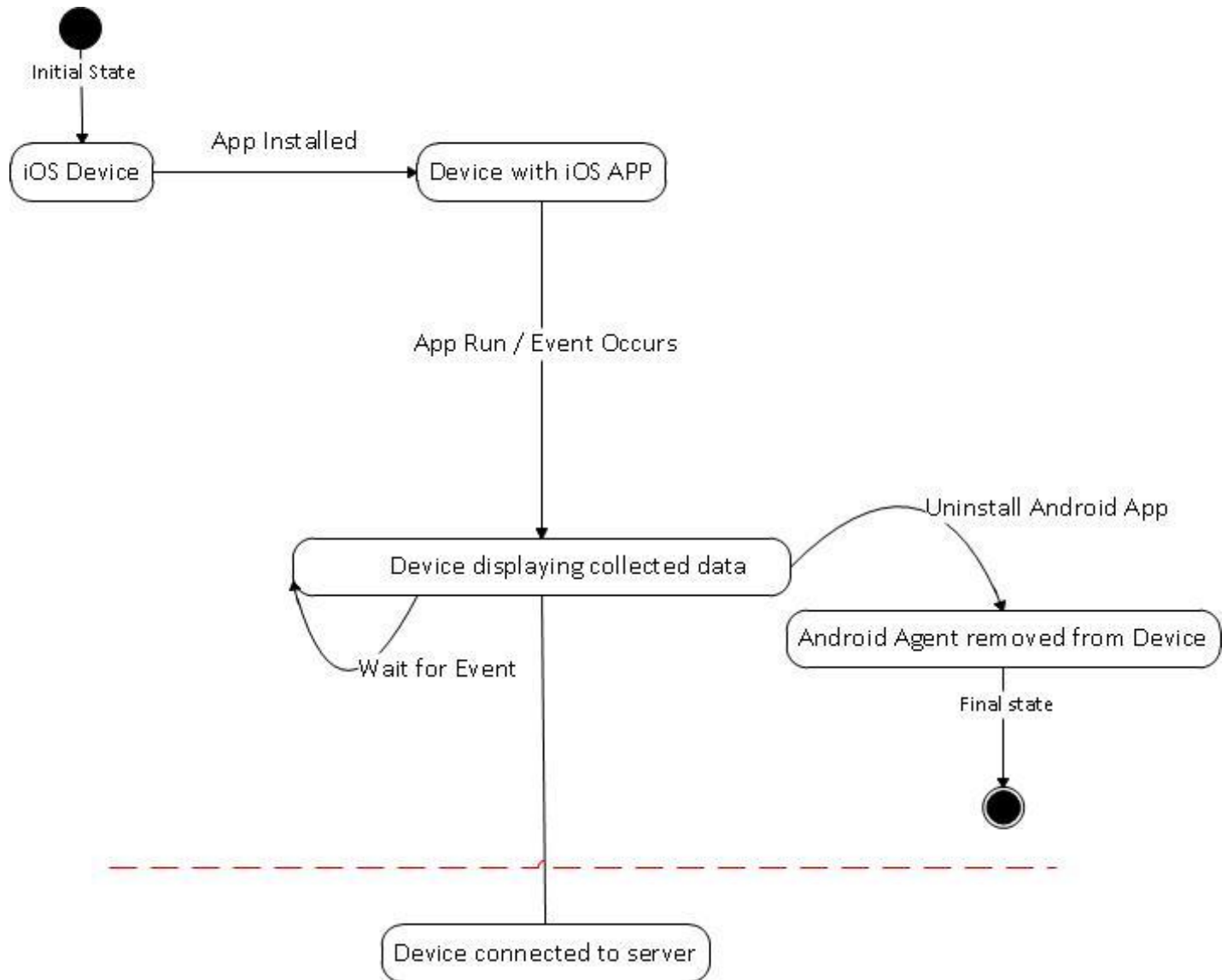


Figure 13 - iOS State Diagram

Chapter 7: Methodology & Development

7.1 Android Compliance Checker App

7.1.1 STEP 1: The CIS Security Benchmark

Various benchmark developing organizations such as Centre for Internet Security and Defence Information Systems Agency have pro-actively developed and released security benchmarks for various mobile operating systems. With onset of such benchmark development work, the enterprises that are now complying against benchmarks from such organizations for their computer systems and servers would have to extend their compliance metrics to include mobile devices soon. The Security Configuration Benchmark released by the **Center for Internet Security for Google Android 2.3 (Gingerbread)** on December 1, 2011 is chosen as the security guideline.

7.1.2 STEP 2: Implementation Issues

After the rules that were to be checked had been chosen, the actual implementation had to be done of those rule checks. There were a number of questions that needed answering.

➤ How to connect to the device?

Options included a USB cable vs wifi. It was realized that USB cables are rarely used anymore. Hence, that option was eliminated. Using the VPN network over wifi seemed most appropriate. Hence this was chosen in the beginning. However eventually it was realized that this wasn't a possible option wither as most android devices are shipped without the ADB server required for ADB over wifi. ADB is the core requirement of the project. Therefore the current status is thus: app distributed via some android app market, installed on phone, performs checks, generates reports, and report pulled from the phone over the web.

➤ **Require Rooting?**

Since the ADB could not pull all data without root permission it was considered whether rooting will be required to perform the checks. However rooting voids the device warranty and therefore was out of question. Without rooting there were two options. Either create an app and install on phone and generate a report. Or without rooting perform as many checks as possible. Finally an app was chosen because eventual updates as when new android versions would be released would be easier. Details are given in later subsections.

➤ **Require a client on the device? App or Non-App.**

This was an issue of major debate. Once rooting was out of the picture, this was the biggest question. Two options existed. One was to install an app, which through APIs directly accessed phone settings and reported them. Other was to use the ADB shell DUMPSYS command and dump system files and read them to search for the setting information. Currently a POC has been made for the first option. And comparisons are on to determine the better approach. Option two will allow the use of the already existing framework and will be more in sync with the existing compliance checkers.

Hence finally based on the above factors, three options were available for data access these are:

1. Without app: with root access
2. Without app: without root access
3. With app: without root access

➤ **How to report and use the accessed data?**

Now that we have some ways to generate the data, the important next step is how to pull and use this data. Either we could use ADB over wifi to do this or else the file needs to be uploaded by the user onto a server. Either way, there a lot of adjustments to be made to the

existing framework and server side software.

7.1.3 STEP 3: Pulling the Data

Once this file exists it's the next step to figure out how to put it onto the server and how exactly the server side of the project will look. The original idea was to pull it using ADB over wifi but since that is not an option anymore, the text/xml file will be pushed onto a server from the device itself.

7.1.4 The Compliance Checker App

After considering and working on all three approaches, the third one was implemented. The first was rejected because it required rooting and was totally out of question. The second and third provided conflict. Both performed an almost equal number of checks but the app approach ensured easier updates and also the pull to the server would not require connection using a USB cable. A POC was prepared for the app and is ready. The app was distributed to android users within the company for review. Feedback was taken and bugs were fixed.



Figure 14 - The CC App

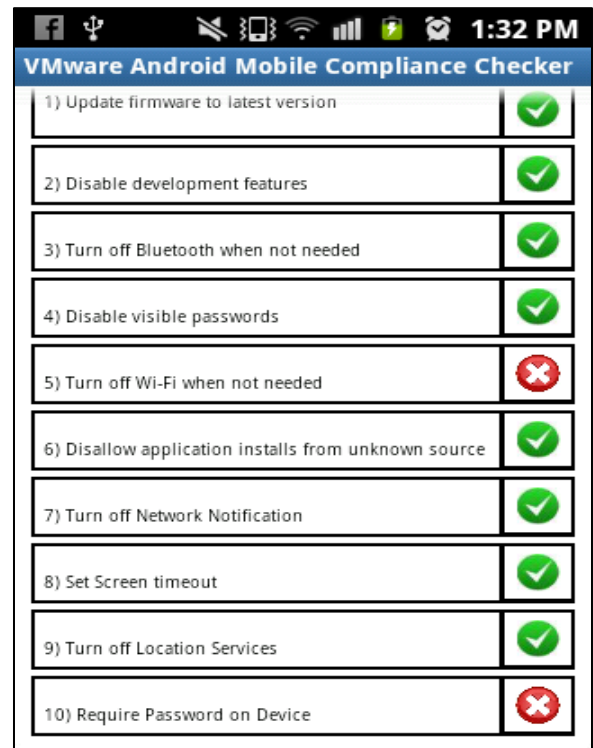
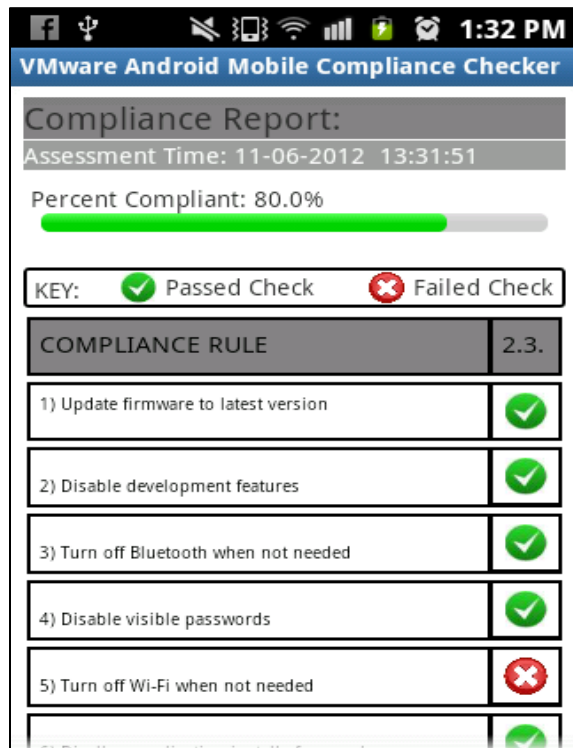


Figure 15 - App Compliance Report

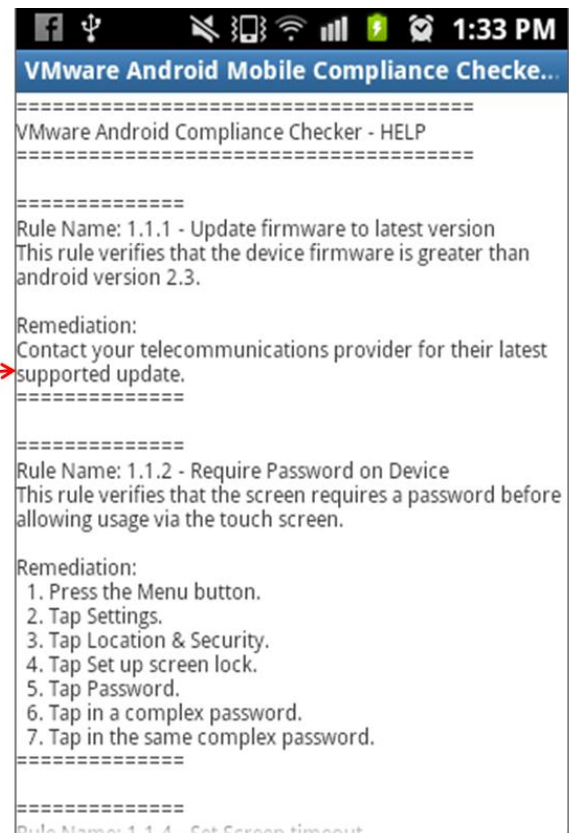
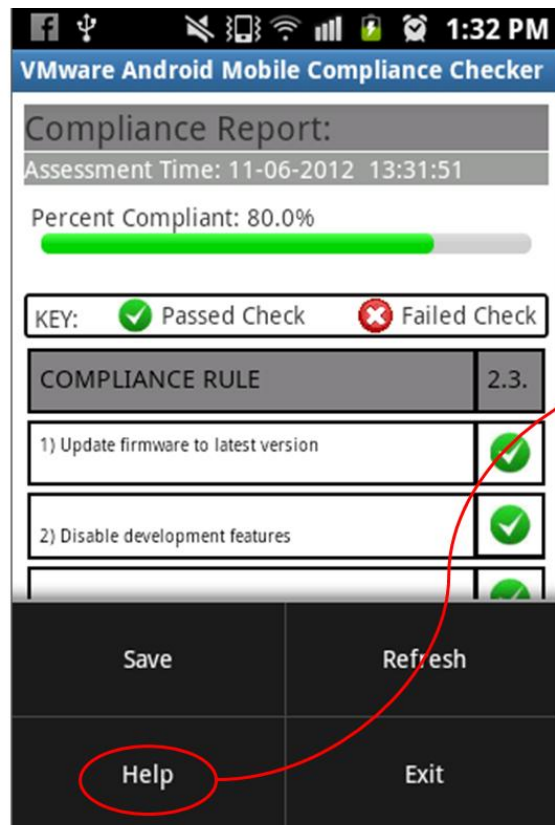


Figure 16 - App Help Screen

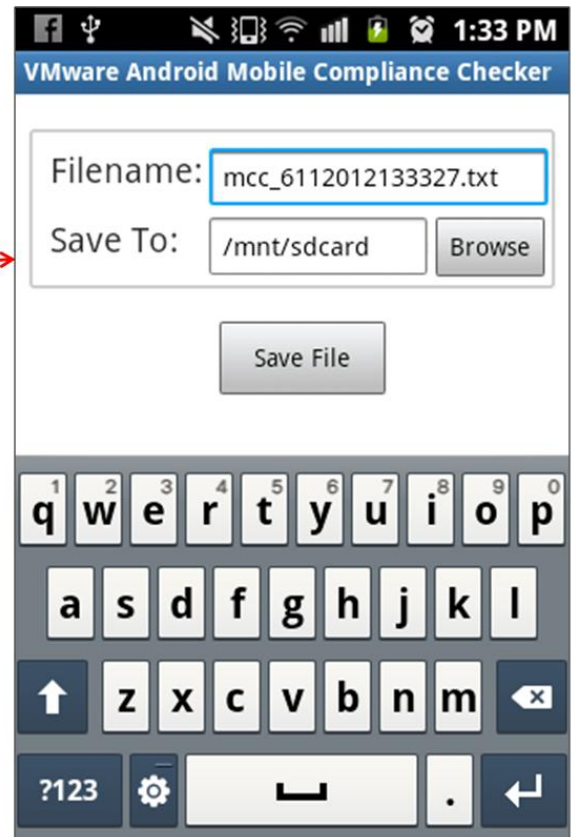
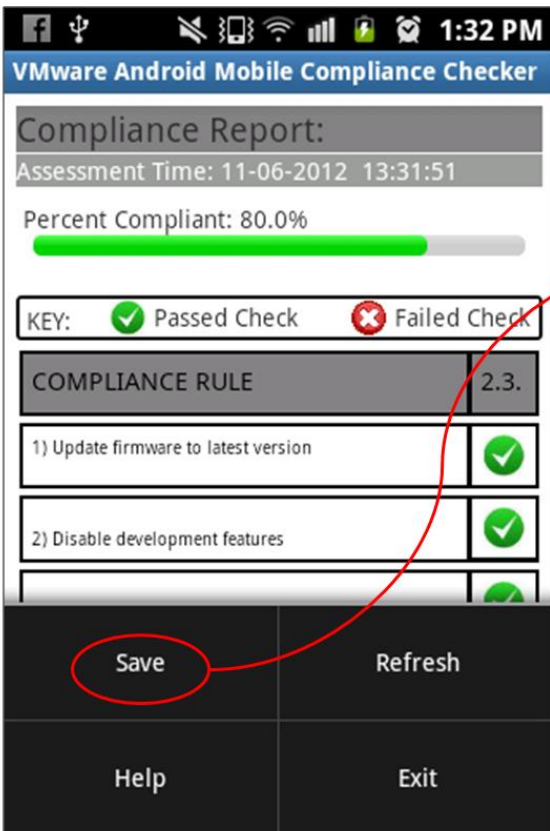


Figure 17 - App Save Screen

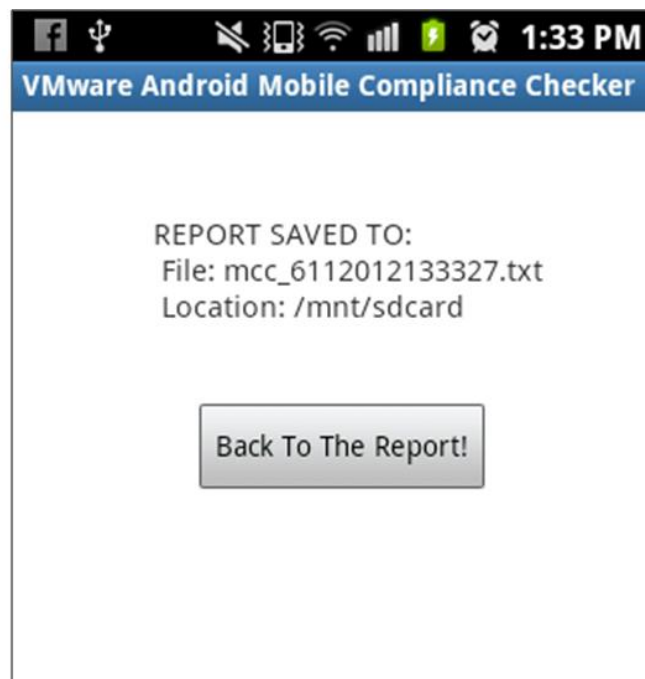


Figure 18 - App Final Screen

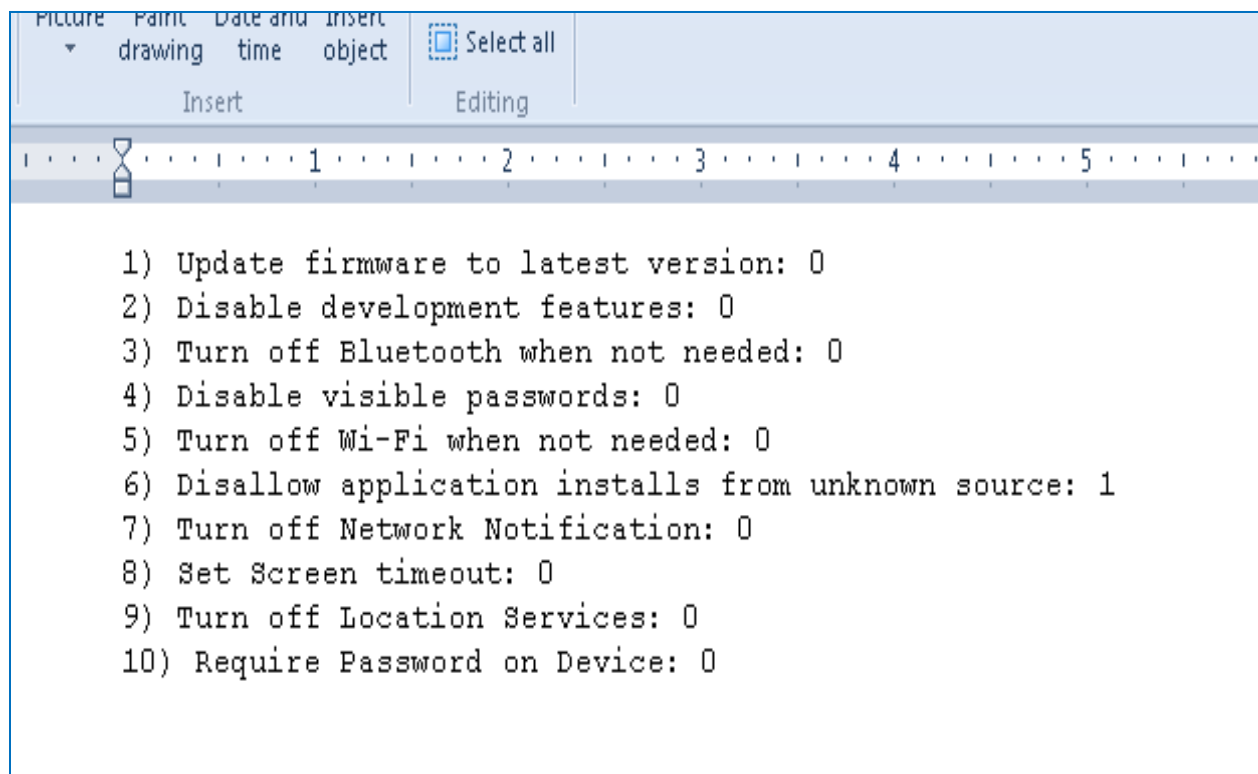


Figure 19 - App Generated Report

7.2 Android Network Discovery

TOOLS AND TECHNIQUES ANALYZED

7.2.1 NMAP

Nmap (Network Mapper) is a security scanner used to discover hosts and services on a computer network. RESULTS:

- All ports are detected as closed on any android device
- Hence OS fingerprinting not possible. However it detects the underlying Linux kernel.
- But no way to distinguish between a normal Linux machine and android device
- Solutions possible:
 - The nmap-os-db could be updated to include the fingerprints of android devices
 - The results returned could be analyzed for some given patterns or keywords to

identify the android devices

- Maybe find a way to open some port on the given device like ssh or http and then check results.

```
root@bt:~# nmap -O --fuzzy 10.112.244.124

Starting Nmap 6.01 ( http://nmap.org ) at 2011-08-23 11:09 EDT
Nmap scan report for 10.112.244.124
Host is up (0.072s latency).
Not shown: 984 filtered ports
PORT      STATE SERVICE
22/tcp    closed ssh
80/tcp    closed http
443/tcp   closed https
1494/tcp  closed citrix-ica
3128/tcp  closed squid-http
3389/tcp  closed ms-wbt-server
7001/tcp  closed afs3-callback
8000/tcp  closed http-alt
8001/tcp  closed vcom-tunnel
8080/tcp  closed http-proxy
8081/tcp  closed blackice-icecap
8100/tcp  closed xprint-server
8200/tcp  closed trivnet1
8888/tcp  closed sun-answerbook
9080/tcp  closed glrpc
10000/tcp closed snet-sensor-mgmt
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: remote management|general purpose
Running: Dell embedded, Linux 2.6.X, Raritan embedded
OS CPE: cpe:/o:linux:kernel:2.6.18 cpe:/h:raritan:dominion kx-ii
OS details: Dell Remote Access Controller (DRAC 5), Linux 2.6.18, Raritan Dominion KX II KVM switch

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.26 seconds
```

Figure 20 - NMAP

7.2.2 AngryIP

NO information. Only alive hosts detected

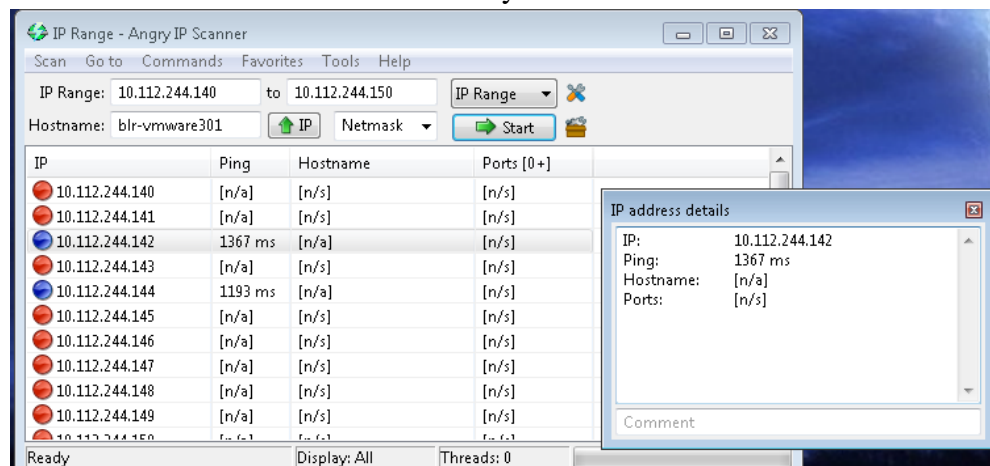


Figure 21 - AngryIP

7.2.3 Ike-scan

Ike-scan is a VPN scanning tool. No results returned



```
root@bt:~# ike-scan 10.112.244.142
Starting ike-scan 1.9 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)

Ending ike-scan 1.9: 1 hosts scanned in 2.449 seconds (0.41 hosts/sec). 0 returned handshake; 0 returned notify
```

Figure 22 - Ike-scan

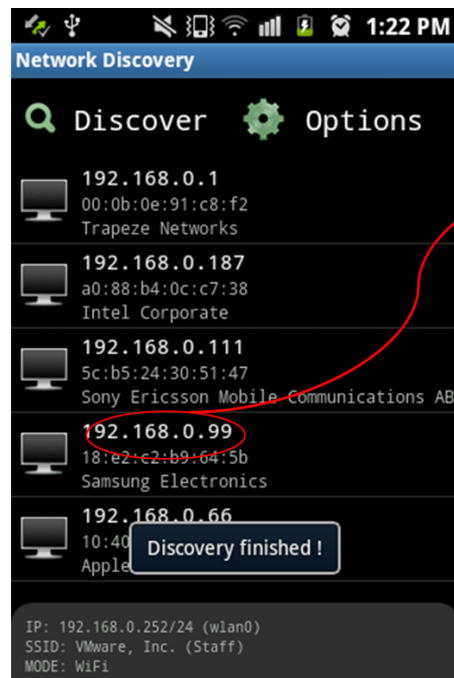
7.2.4 Android Apps

PROS

- Provides more info than nmap
- Provides the MAC address, ip and detects devices as Samsung, Motorola, Intel, Apple etc.
- Atleast helps identify device manufacturers
- So maybe a combination of linux (from NMAP) + (Samsung / Motorola / Sony etc) could lead to android.

CONS

- needs to run on the server not android phone
- Final detection still not possible. There must be some similar application that runs on windows and gives this result.



■ App 1: Network Discovery

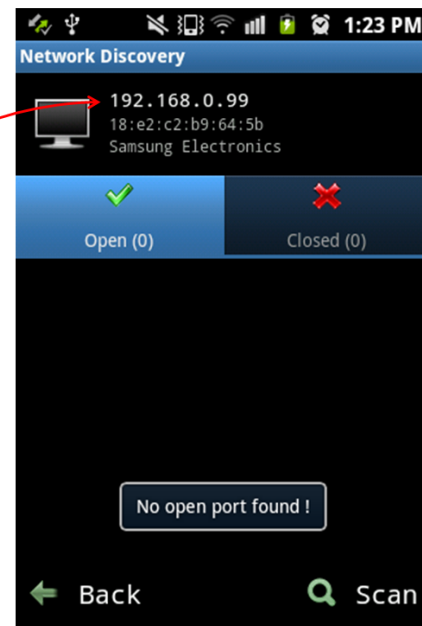
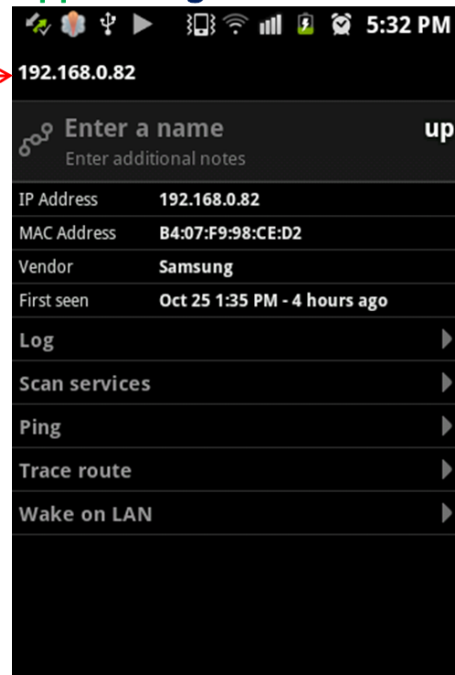


Figure 23 - Android Apps for discovery



■ App 2: Fing



7.2.5 Android Apps: Apps that require a Rooted Device

- **Wifi kill:** can remove other devices from the wifi network

- **Network spoofer:** the list of devices returned are all other android devices!
- These apps require a rooted android device. They perform admin level tasks on the network like removing other devices from the LAN network and changing URLs.
- Working: basically use hacking methodologies to fool the devices into thinking this android phone is the router and hence perform high level tasks on them.
- Would require creating something much higher level that hacks into the network to read device data.
- Not a viable option.
- However, according to a video network spoofer detects only android devices on the network by default. So this app could directly deliver a list! (need a rooted phone to test)

7.2.6 Android Apps: to run these apps on an emulator

Points to consider:

- 1) Internet on the emulator
- 2) VPN on the emulator
 - Market App on emulator – download the .apk for install
 - Access the VPN connection from the app not by giving server address
 - Pros: the emulator is rooted hence no issues with the apps that require root permission

- 3) Idea: have app run the emulator. So basically the enterprise admin will not need to use a separate android phone to run the app. Generate result and store on the server. Then u get the complete list of ip addresses of all android devices on network.

7.2.7 MAC Address

OUI: organizational unique identifier (IEEE std)

- The first three octets of the MAC address are supposed to be supplied by the organization (OUI). Hence are specific to certain companies. like 00-1a-11 is for Google.
- Android Apps allow us to check the MAC address of all devices on network.
 - Is mapping possible?
 - Android does not have its own OUI
 - So its either by manufacturer like Samsung or could be Google.
 - However results for android devices were not consistent. (more checking required)
 - Manufacturer anyway we know. So this could be useful only if its by Google.

7.2.8 MDM- Existing Technology

I have Read through the documentation and watched videos closely of all the MDM websites.

Even chatted with the online support group of Fiberlink. There is no MDM software that performs network discovery. There always has to be an initial app download or enrollment by the client. Or else the syncing is done with the AD/ Exchange server(Fiberlink).MDM Providers studied in detail:

- 1) MobileIron
- 2) AirWatch
- 3) Zenprise

- 4) FiberLink
- 5) Good technology
- 6) Centrify

7.2.9 MDM- APIs

- I did extensive search for any existing APIs.
- Each android OEM is coming up with its own set of MDM APIs like 'SAFE' by Samsung. Then it is working with third party MDM providers like Samsung with MobileIron.
- There are no open source APIs for android. Developers as of now have no access to these APIs.
- The MDM providers share these APIs only with their clients. HTC, Motorola are working on their own APIs.
- In any case since none of them actually perform network discovery or remote install, not of much use to us
- Motorola is apparently working on remote app push for android devices as said by a random online forum. No confirmations.
- What can be done for remote push is as is done by all MDM providers ie to develop an enterprise server that controls apps that can be pushed and removed from the enterprise devices. However requires an initial client on the device.
- Better support available for iOS in this regard. But none as such for android at the moment

7.2.10 MDM- Google's Take

- According to google android is not enterprise ready as yet and they are working on it.
- The only existing android API as of now, is the Device Admin API.
- It basically allows u to set certain policies such as password qualities etc that the device must comply with.

As for app install, google provides three options.

- 1) Sideloaded – enterprise app server
- 2) Internal App Directory
- 3) The official app market- google play

7.2.11 Google Play

Developers created google play such that remote app install via it was possible. However this feature isn't used. Not by Google itself, so anybody else doing it is out of question. As online forums say, remote app install on android is totally not possible. Apps have to be either downloaded or distributed via email.

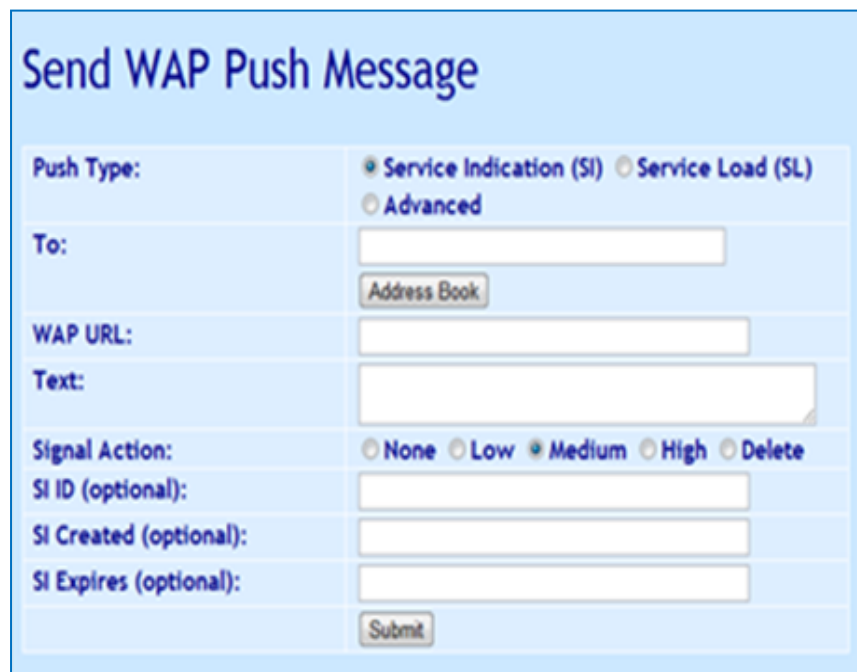
7.2.12 Microsoft Exchange

- Should be the best option.
- Requires the server admin who has the required permissions
- identify which mobile devices are synced with each account
- Hence it should be possible to identify the android devices
- the app can then be sent via mail or users can be notified to download it.

7.2.13 WAP Push

Mentioned on one of the MDM provider websites as being an option to push the client onto the device. Basically sends an SMS that contains a URL. Can directly open up the URL in the device browser.

- Pros: App can directly be downloaded online.
- Cons: This is again only a distribution mechanism. Requires prior identification and phone numbers of devices. All devices may not support WAP Push.



The screenshot shows a web form titled "Send WAP Push Message". The form has several fields and radio buttons. The "Push Type:" field has three radio buttons: "Service Indication (SI)" (selected), "Service Load (SL)", and "Advanced". The "To:" field is a text input with an "Address Book" button below it. The "WAP URL:" field is a text input. The "Text:" field is a larger text input. The "Signal Action:" field has five radio buttons: "None", "Low", "Medium" (selected), "High", and "Delete". Below these are three optional fields: "SI ID (optional)", "SI Created (optional)", and "SI Expires (optional)", each with a text input. A "Submit" button is at the bottom right.

Figure 24 - WAP Push

7.2.14 C2DM & GCM

- **Cloud to device messaging** is a push notification service
- Used for communication between a third party application server and an app client already installed on the device

- Basically used for services such as facebook, twitter instant updates.
- Now deprecated and replaced by **Google Cloud Messaging**
- Not much use until app client already deployed onto the android device

7.2.15 Avahi Daemon/ Zeroconf/ Apple-Bonjour/ mDNS

- **Zeroconf:** Zero configuration networking is a set of techniques that automatically creates a usable IP network without manual operator intervention or special configuration servers.
- **Avahi:** Avahi is a free zerconf implementation, including a system for mDNS/DNS-SD service discovery. Avahi allows programs to publish and discover services and hosts running on a local network with no specific configuration.
- **Apple-Bonjour:** Bonjour is Apple's implementation of Zero configuration networking (Zeroconf), a group of technologies that includes service discovery, address assignment, and hostname resolution. Bonjour locates devices such as printers, other computers, and the services that those devices offer on a local network using multicast Domain Name System(mDNS) service records.
- **mDNS:** Multicast DNS (mDNS) is a way of using familiar Domain Name System (DNS) programming interfaces, packet formats and operating, without configuring a conventional DNS server. It is useful in small networks without a DNS server, but can also work in environments beside a DNS server. mDNS functionality is provided using IP multicast over User Datagram Protocol (UDP). The mDNS protocol is used by Apple's Bonjour, GNU/Linux Avahi

How it works: Discovers all the devices and services on the network that broadcast themselves.

Why useful? You do not need to know the device ip. Each device and service assigned a separate ip and u can use that to access or identify the device directly.

Why not?

- However, only devices that have an avahi daemon/ bonjour running broadcast themselves and hence can be discovered.
- As of now it is not supported by default on android.
- However few have managed to run the avahi daemon on android as well
- Still does not solve our purpose, as installing an app on each individual device would be an easier option.

3.2.16 The Net Admin


- I spoke to the VMware net admin in tech-ops
- He has a complete database of device information. Can list the android devices on the VPN network
- Chances are that the app or atleast a notification can be sent to all the devices
- However if that is not possible the via the admin, we can get the list of android device IPs and they can be pinged to notify

7.3 Android - Device Information Collection

Create an Android Client that collects all possible information from the device such as:

- All Hardware Information – CPU, RAM, Memory Card, Slots, IMEI no., SIM Card details, etc.
- OS details – Android version, etc.
- Software Inventory – ALL installed Apps and programs on the device, Etc.
- All Settings – Settings for password, network, messaging, email, internet, etc.
- Device performance data – Uptime, last reboot, etc.
- Logs & Any other information relevant

And then make an extensive xml out of it to display all of it in an easily readable and usable format. The idea is to have a VCM agent that can help assess/ monitor android devices.



```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<root>
  <HARDWARE_DETAILS>
    <property name="brand" value="samsung" />
    <property name="model" value="GT-S7500" />
    <property name="cpu_abi" value="armeabi-v7a" />
    <property name="hardware" value="qcom" />
    <property name="device_id" value="352569050385028" />
    <property name="ANDROID_ID" value="e288f95e4da30be9" />
    <property name="sim_no" value="89914500030198018937" />
    <property name="phone_no" value="Not Available" />
    <MEMORY_DETAILS>...
    <CPU_DETAILS>...
  </HARDWARE_DETAILS>
  <OS_DETAILS>...
  <APP_DETAILS>...
  <SERVICES_API>...
  <SERVICES_GETPROP>...
  <SERVICES_DUMPSYS>...
  <ACCOUNT_DETAILS>...
  <SETTINGS>
    <NETWORK_SETTINGS>...
    <SECURITY_SETTINGS>...
    <SYSTEM_SETTINGS>...
    <GETPROP>...
  </SETTINGS>
</root>
```

Figure 25 - The XML

7.4 Android - Event Driven Compliance

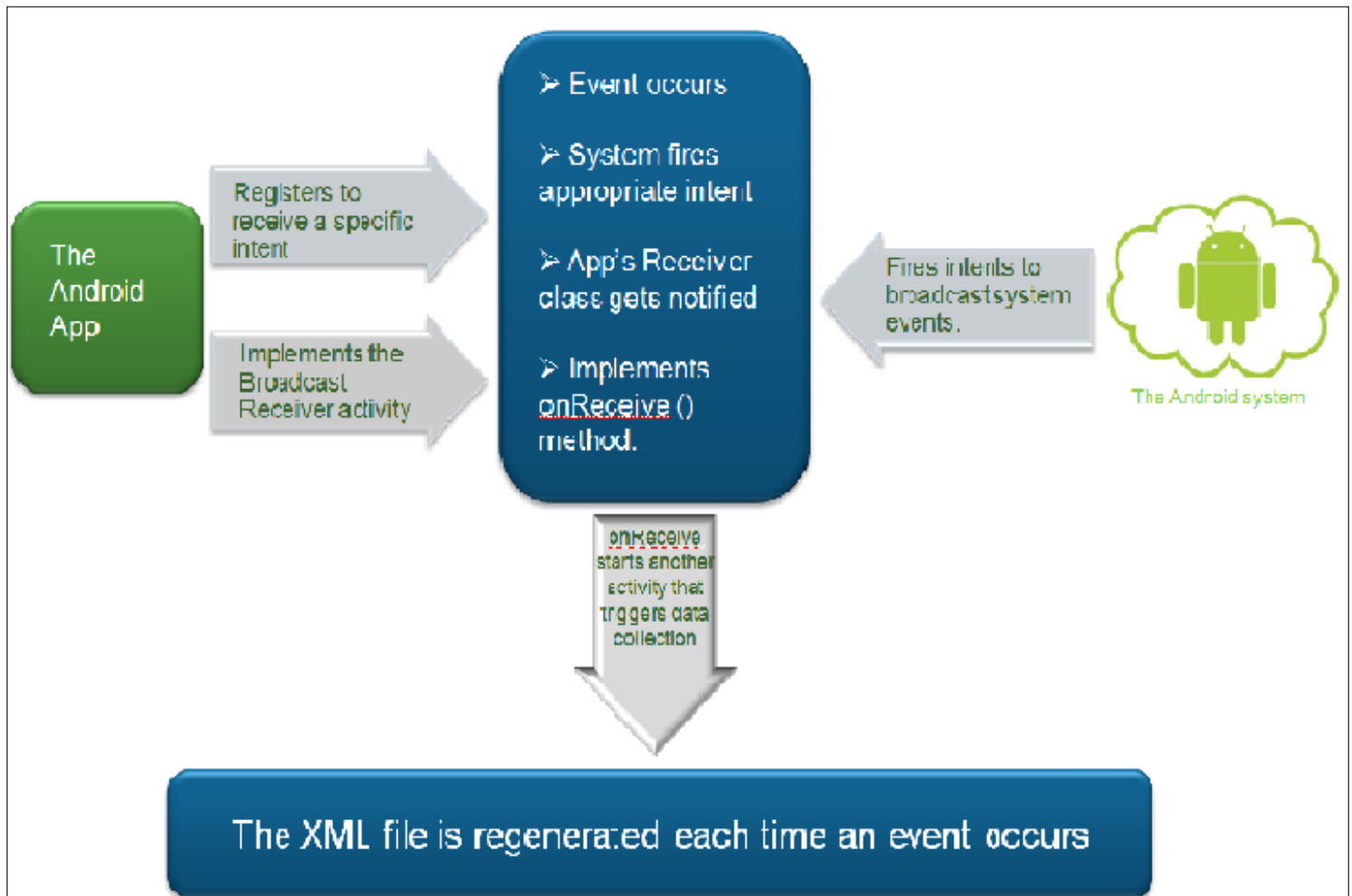
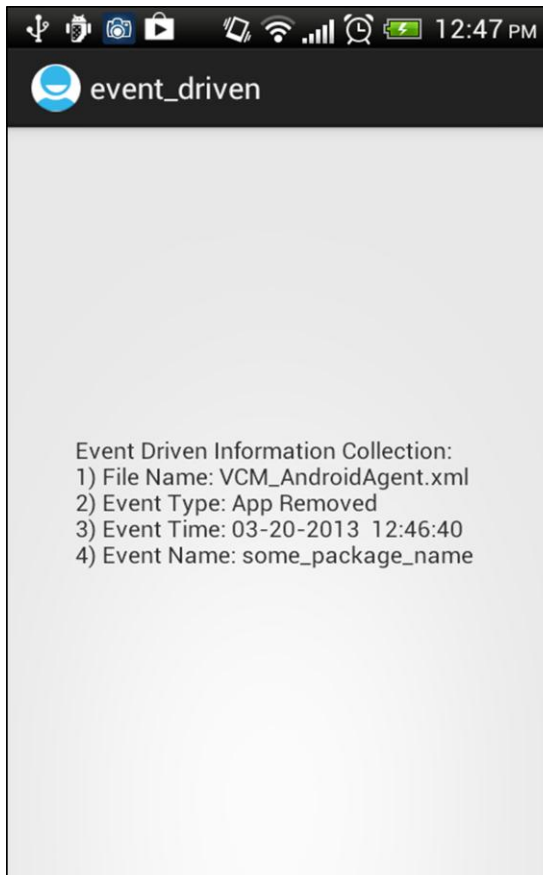


Figure 26 - Event Driven On Android

The data collection should happen each time a certain event occurs. The event could be a network connection, an app installation/removal, a change in some device setting etc

App Screenshots



App screen
pops up as
soon as event
occurs

Figure 27 – Event Driven on Android
App



The XML file is
updated after
each event

7.5 Android - Secure File Transfer



Figure 28 - File Transfer

Options:

- Line by line streaming via TCP sockets
- upload from device to server via HTTP POST
- Upload from device to server using HTTPS



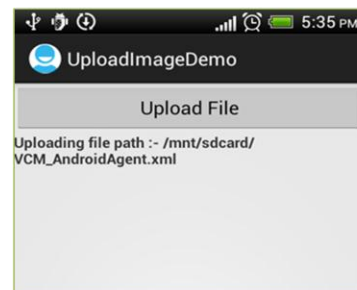
WAMP with Apache Server

- Server IP address: `http://10.112.72.xxx/`
- Runs a php script that handles the receipt of the HTTP POST from the Android client
- receives the file and stores it in the target folder on the server(`C:\wamp\www`) as specified in the [script](#)

Android Client

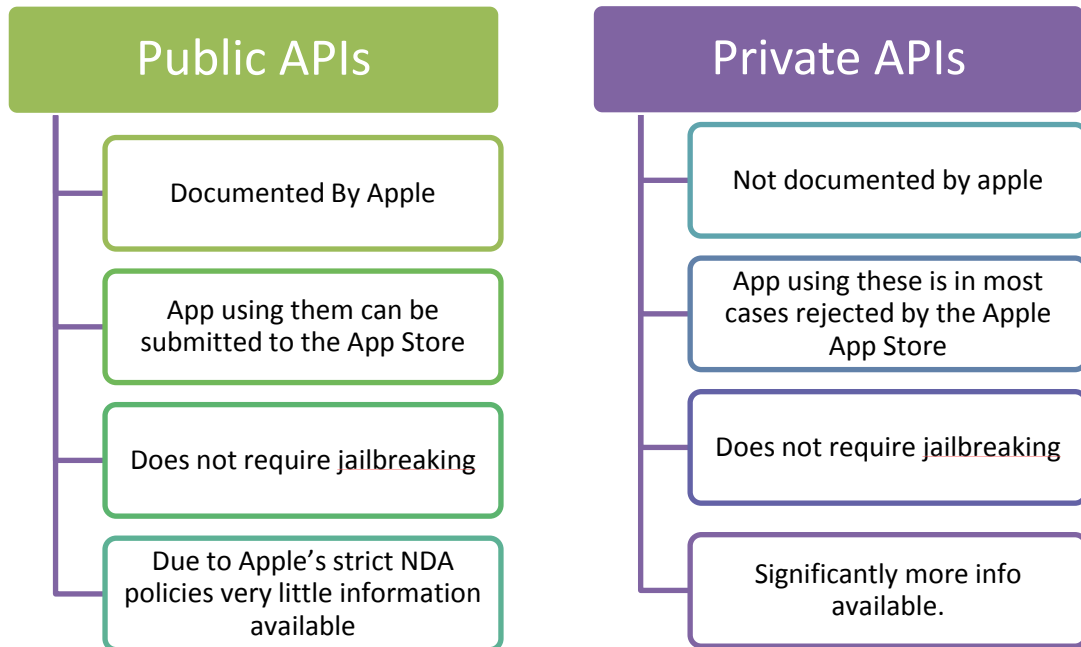


- Client on VPN via Junos Pulse.
- phone IP address: `10.112.244.xxx`
- App connects to the Apache HTTP server and transfers file via HTTP POST
- The app currently hard codes the server IP address (could be taken as runtime user input as well)
- location of the XML file specified in app
- as soon as user presses 'upload', file copied to the specified server location!



7.6 Android - Secure File Transfer

7.6.1 Step 1: Private vs public API



7.6.2 Step 2: iOS app deployment options

1) The Public App Store

- Highly available
- Well understood
- Requires Apple's approval for apps
- Hence only public APIs documented by apple are allowed.
- Access to only limited data

2) iOS Developer Enterprise Program

- Allows a company to distribute apps in-house

- Company must have a DUNS number
- Apps need not be approved by Apple
- Hence almost any API can be used
- But app cannot be distributed to clients outside the company.

3) **Custom B2B Apps Program**

- Provides some features of an MDM
- Bulk app purchase by businesses
- For large distributions
- Requires app store approval

4) **Ad Hoc Distribution**

- For Testing
- Limited to 100 devices
- Apple approval not required
- Not visible to public
- Only to devices whose UDID is associated with the provisioning profile
- Expires after a year

5) **TestFlight**

- Used for testing within the team
- A dashboard to manage testing and receive feedback

7.6.3 Step 3: Data collection using Public vs Private API

S.NO	SETTING / DATA POINT	PUBLIC API	PRIVATE API
1	MAC address	YES	YES
2	Model	YES	YES
3	Processor	NO	YES
4	IMEI	NO	YES
5	sim details	YES	YES
6	Phone number	NO	YES
7	Unique Device Identifier (UDID)	YES	YES
8	Total Memory	PARTIAL	YES
9	RAM	NO	CHECK
10	CPU details	NO	YES
11	iOS version	YES	YES
12	Timesinceboot	YES	YES
13	Uptime	YES	YES
14	apps installed	PARTIAL	CHECK
15	processes running	YES	YES
16	account details	CHECK	CHECK
17	WIFI on	YES	YES
18	Bluetooth on	YES	YES

19	Location Services	YES	YES
20	data roaming	NO	NO
21	Screen lock enabled	NO	NO
22	developer settings		
23	Airplane mode	NO	CHECK
24	Screen timeout	NO	

7.6.4 Step 4: App Status

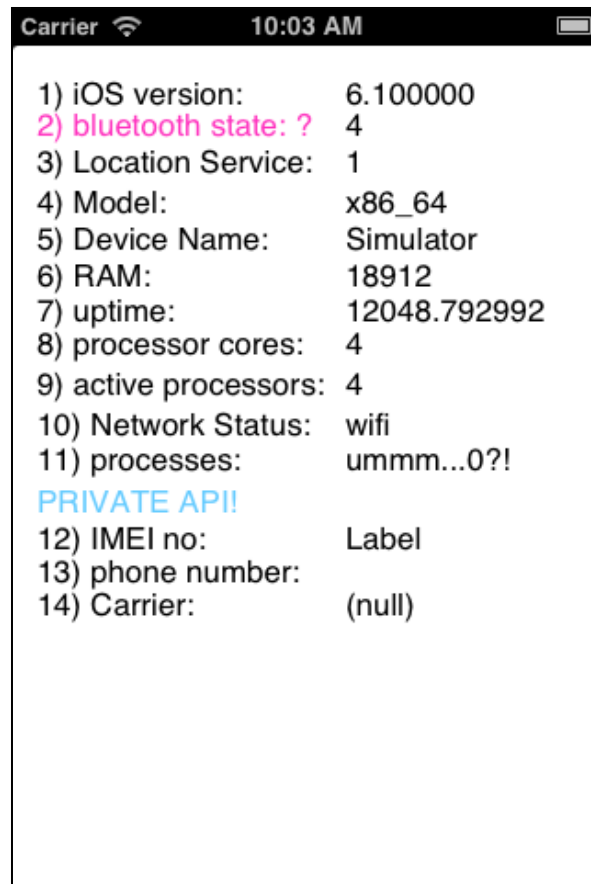


Figure 29 - iOS App

7.7 iOS – Event Driven Compliance

This is taken care of by the one function ‘reachabilityChanged’ which registers to be notified every time there is a change in network connection. It then calls the ‘checks’ method each time to recollect data.

Chapter 8: Testing

8.1 Android Testing

8.1.1 Testing on the emulator/simulator

The Initial testing was done on the emulator. After that once the final app was ready, the emulator was used again to check that the app was compatible with all android versions after 2.3. The app has been extensively tested on all possible devices including tablets and phones from android version 2.3 to android version 4.2.

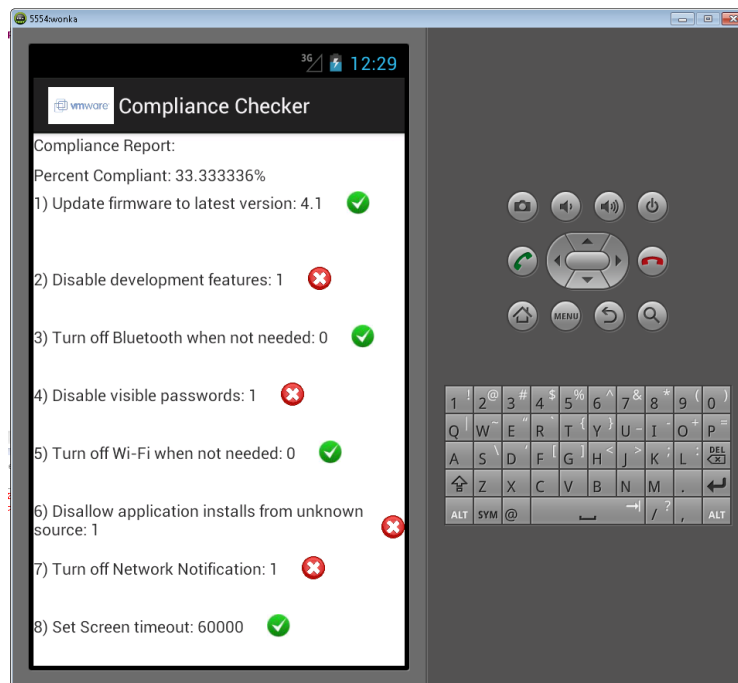


Figure 30 - Android Emulator

8.1.2 Device Testing

In android, eclipse generates a file in the .apk format that can directly be installed and run on any device. Hence testing was extremely easy. However, since only limited devices were available, it could not be tested on all possible real devices. The app/agent was tested by me on:

1. Samsung Galaxy ace plus (Android 2.3.6)

2. HTC Desire X (Android 4.0)
3. LG Nexus Tab (Android 4.2)

Besides this the Compliance checker app was distributed via the local site, Socialcast and feedback was received regarding bugs and usability

8.2 iOS Testing

8.2.1 Testing on the emulator/simulator

The complete testing was done on the simulator. The iOS simulator emulates the real time device better than android and hence makes up for the lack of testing on the iPhone itself.



Figure 31 - iOS simulator

8.2.2 Device Testing

In case of iOS, the app could only be tested on the simulator as it is not allowed to test the apps developed on Xcode on real devices without first registering with the iOS developer program for which you need to pay a fee.

Chapter 9: Scope Of The Project

Today every organization in any vertical has a mandate to adhere to standards in one form or the other. These can be quality standards, regulatory standards, service level agreements, and many more. There are numerous security configuration settings or security controls available in various products – Operating Systems, Applications, Browsers, etc. Organizations needing to adhere to such standards have a hard time managing and reporting compliance status due to large-sized data centers, unknown changes, lost blueprints, changing roles, etc. A compliance management product is intended to solve this.

Typically a compliance management product involves below steps:

- 1) **Collect** – In this step, various security attributes or configurations parameters are queried and their setting or value is fetched.
- 2) **Analyze** – The values or settings found on the target product are compared with the expected value.
- 3) **Report** – Based on comparisons done in analysis step, a compliance assessment report is generated and is presented to the user.
- 4) **Manage** – In this step, reports are reviewed and various needed actions are planned. There might be some non-compliant results which need to be fixed by administrators or there might be some unapproved changes which must be rolled back.
- 5) **Act** – In the final step, actions planned are carried out. Once the actions are completed, we move again to step 1 to ensure that everything is in place meaning we are compliant!

Such data collection, report generation, compliance testing and patching at enterprise level is done today by VMware's product called VCM. The idea here is to extend this functionality to include all possible mobile devices. With the sudden explosion in usage of smartphones, tablets and other devices the concept of BYOD(Bring Your Own Device) is fast gaining popularity. Enterprise admins will soon need tools to manage these devices. Our Mobile Compliance Checker might just be the answer.

Chapter 10: Summary And Conclusions

10.1 Android Mobile Compliance Checker : At the end there would be a complete health report for all mobile devices, reporting whether all devices on the network are compliant to the rules and security guidelines specified in any standard benchmark for the specific device. Having a consolidated compliance report would help the security officer get an in- depth view of compliance status for the enterprise.

10.2 Android Network Discovery : There is no sure shot way of determining that an Android device is present in the network. We would probably require a combination of things to make sure the detected device is surely an Android device.

- NMAP + Android Apps
- Microsoft Exchange
- Via the Net Admin

10.3 App Installation : there is no way to silently install the app on the mobile devices. The user intervention would be required in all mechanisms – in some more intervention is needed and in some cases only little effort from the user is required.

10.4 Android Device Information Collection: Currently there is an Android agent that can collect upto 800 data points on the device and store it as an XML on the device

sdcard, which as a next step should be transferred to the server via https. (currently via streaming)

10.5 iOS Mobile Compliance Checker : POC is ready to be presented soon.

However it appears that the same level of information as android will not be available.

10.6 Android & iOS Event Driven Compliance: A POC is ready which runs the device information collection app each time a certain event occurs. Currently checking for any network changes and apps added or removed on android and only for connection change on iOS.

10.7 iOS Mobile Compliance Checker : POC is ready. However it appears that the same level of information as android will not be available.

10.8 Android Secure File Transfer: transfers the generated XML file securely to the windows server using HTTPS POST.

References

Android – Getting Started

- <http://developer.android.com/sdk/installing/index.html>
- <http://www.wikihow.com/Set-up-an-Android-Development-Environment>
- <https://developer.amazon.com/sdk/fire/setup.html>

Android – Data Collection

- <http://developer.android.com/reference/android/provider/package-summary.html>
- <http://developer.android.com/guide/topics/manifest/provider-element.html>
- <http://developer.android.com/guide/topics/manifest/manifest-intro.html>
- <http://stackoverflow.com/questions/9944747/parsing-android-getprop>
- <http://developer.android.com/tools/help/adb.html>
- <http://adbshell.com/>
- <http://developer.android.com/reference/android/telephony/TelephonyManager.html>
- <http://stackoverflow.com/questions/7514163/using-telephony-manager-in-android-to-find-imei-number>

File Transfer

- <http://reecon.wordpress.com/2010/04/25/uploading-files-to-http-server-using-post-android-sdk/>
- <http://stackoverflow.com/questions/3204476/android-file-uploader-with-server-side-php>

- <http://vikaskanani.wordpress.com/2011/01/11/android-upload-image-or-file-using-http-post-multi-part/>
- http://www.anddev.org/upload_files_to_web_server-t443-s15.html
- <http://www.coderzheaven.com/2012/04/26/upload-image-android-device-server-method-4/>

Secure File Transfer

- <http://www.jscape.com/blog/bid/82339/What-is-an-SSL-File-Transfer>
- <http://www.javaworld.com/javatips/jw-javatip96.html>
- <http://stackoverflow.com/questions/6914266/https-in-android>
- <http://www.cs.sunysb.edu/documentation/jsse/jssefaq.html#6>
- <http://www.mkyong.com/java/java-https-client-httpsURLConnection-example/>
- <http://docs.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html#KeyClasses>

Server Configuration

- <http://www.debian-administration.org/articles/349>
- <http://fash7y.wordpress.com/2011/12/03/solved-how-to-set-up-https-with-openssl-in-wamp/>

Certificates

- <http://blog.callistaenterprise.se/2011/11/24/creating-self-signed-certificates-for-use-on-android/>
- <http://stackoverflow.com/questions/2642777/trusting-all-certificates-using-httpclient-over-https/6378872#6378872>
- <http://nelenkov.blogspot.com/2011/12/using-custom-certificate-trust-store-on.html>
- <http://www.bouncycastle.org/wiki/display/JA1/Provider+Installation>

PHP

- http://www.atksolutions.com/articles/how_to_create_upload_file_php_script.html
- <http://php.net/manual/en/features.file-upload.errors.php>
- <http://www.learnbycode.com/content/how-run-php-code-local-computer>

Event Driven Compliance

- <http://developer.android.com/reference/android/content/BroadcastReceiver.html>
- <http://stackoverflow.com/questions/5888502/android-wifi-how-to-detect-when-wifi-connection-has-been-established>
- <http://stackoverflow.com/questions/5888502/android-wifi-how-to-detect-when-wifi-connection-has-been-established>

<http://stackoverflow.com/questions/7470314/receiving-package-install-and-uninstall-events>

iOS – Getting Started

- <http://mobiledan.net/2012/03/02/5-options-for-distributing-ios-apps-to-a-limited-audience-legally/>
- <http://developer.apple.com/library/ios/#featuredarticles/ViewControllerPGforiPhoneOS/Introduction/Introduction.html>
- <http://mobile.smashingmagazine.com/2009/08/11/how-to-create-your-first-iphone-application/>
- <https://developer.apple.com/programs/ios/enterprise/>
- <http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Introduction.html>

Objective-C

- <https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>
- <https://developer.apple.com/devcenter/ios/index.action>
- <https://en.wikipedia.org/wiki/Objective-C>
- https://developer.apple.com/library/mac/#referencelibrary/GettingStarted/Learning_Objective-C_A_Primer/
- http://cocoadevcentral.com/d/learn_objectivec/

Data Collection

- <http://stackoverflow.com/questions/3339722/check-iphone-ios-version>

- <http://stackoverflow.com/questions/7328545/access-ios-settings-from-code>
- <http://stackoverflow.com/questions/13460804/is-it-possible-to-detect-a-phone-number-of-the-device-in-ios>
- <http://iphonesdkdev.blogspot.com/2009/01/source-code-get-hardware-info-of-iphone.html>
- <http://stackoverflow.com/questions/4841958/programmatically-check-whether-gps-is-enabled-or-not-on-iphone>
- <http://stackoverflow.com/questions/8032635/responding-to-ram-availability-in-ios/8072278#8072278>
- <http://stackoverflow.com/questions/7241936/how-do-i-detect-a-dual-core-cpu-on-ios>
- <http://stackoverflow.com/questions/7049127/no-provisioning-ios-device-connected>
- <http://stackoverflow.com/questions/3407556/iphone-mobile-number-using-core-telephony>

BluetoothManager

- <http://stackoverflow.com/questions/4955007/how-to-get-the-status-of-bluetooth-on-off-in-iphone-programatically>
- http://developer.apple.com/library/ios/documentation/CoreBluetooth/Reference/CoreBluetooth_Framework/CoreBluetooth_Framework.pdf
- <http://weblog.invasivecode.com/post/39707371281/core-bluetooth-for-ios-6-core-bluetooth-was>
- <http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/coreoslayer/coreoslayer.html>

LocationManager

- <http://stackoverflow.com/questions/4700987/how-to-check-if-location-services-are-enabled-for-a-particular-app-prior-to-ios>
- <http://stackoverflow.com/questions/4318708/checking-for-ios-location-services>
- https://developer.apple.com/library/mac/#documentation/CoreLocation/Reference/CLLocationManager_Class/CLLocationManager/CLLocationManager.html

Network Notification (Event Driven)

- <http://stackoverflow.com/questions/8264053/detect-if-iphone-is-in-airplane-mode>
- <http://stackoverflow.com/questions/13317776/check-and-toggle-airplane-mode-with-private-apis-on-iphone>
- <http://developer.apple.com/library/ios/#samplecode/Reachability/Introduction/Intro.html>
- <http://stackoverflow.com/questions/11177066/how-to-use-ios-reachability>
- <http://stackoverflow.com/questions/3790957/reachability-guide-for-ios-4>
- <http://stackoverflow.com/questions/15372005/working-with-ios-reachability-to-display-a-uialertview>