Protecting Android Devices Following BYOD Policy Against Data Security and Privacy Attacks

Student Name: Arun Kumar Jindal

IIIT-D-MTech-CS-IS-13-MT11003
July 10, 2013

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Dr. Vinayak Naik (Chair)
Dr. Gaurav Gupta
Dr. Sandip Bapat

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science,
with specialization in Information Security

This research was partially funded by Mobile and Ubiquitous Computing Group at Indraprastha Institute of Information Technology, Delhi.

Keywords: Bring Your Own Device (BYOD), Android, Mobile Devices, Security, Malware, Operating System (OS), Mobile Device Management (MDM), Network Access Control (NAC) and Rooting

Certificate

This is to certify that the thesis titled **"Protecting Android Devices Following BYOD Policy Against Data Security and Privacy Attacks"** submitted by **Arun Kumar Jindal** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering with specialization in Information Security* is a record of the bonafide work carried out by him under my guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Professor Vinayak Naik
Indraprastha Institute of Information Technology, New Delhi

Abstract

Bring Your Own Device (BYOD) is an IT policy being adopted by corporate organizations worldwide. It permits the employees to bring their own devices like smartphones, tablets, etc to their place of work and use them to access the privileged corporate information while being both inside and outside their place of work. Therefore, employees use the same device for their personal and office work. Such a corporate policy brings in a number of advantages like increased employee productivity, improved employee satisfaction, and reduction in corporate expenses. However, one of the major concerns in implementing such a policy is data security and privacy. Permitting employees to access the privileged corporate information on their personal device can lead to pertinent corporate data being compromised. On the other hand, employees are apprehensive that the corporate organizations may spy or track their personal cyber activities. Existing solutions for BYOD can be categorized into Mobile Device Management (MDM)-based and Network Access Control (NAC)-based. MDM-based solutions are comprised of a client software, which runs on the users' mobile devices 24/7 monitoring, securing, and managing the mobile device from a corporate-based server. Such a solution could lead to breach of employees' privacy and extensive battery drainage. NAC-based solutions assumes the use of corporate network. Such a solution is not full proof because the corporate data is at a security risk, when the device is not connected to the corporate network.

In this study, we formulate a generic BYOD policy from a corporate data security perspective, study the possible security breaches on that policy from Android devices' perspective, and propose ways to defend against them. We propose a solution architecture for Android-based mobile devices. Our approach, unlike the existing BYOD solutions, provides data security, preserves privacy, and consumes less energy. Our approach successfully detects (a) root status of the device and (b) malicious apps, which steal information or subvert information. Our detection is 70 % accurate when tested on real malicious applications. Finally, we present limitations of our approach.

Acknowledgements

First and foremost I would like to express my sincere gratitude to my advisor *Dr. Vinayak Naik* for giving me an opportunity to pursue my thesis work under his expert guidance. This thesis work would not have materialized without your support and guidance. I respect and admire your research driven style of working which inculcated the research orientation in me. Your patience, faith and confidence in me gave me the liberty to learn and explore all dimensions of this particular research area without the fear of failure. I would like to thank my esteemed committee members *Dr. Gaurav Gupta* and *Dr. Sandip Bapat* for agreeing to evaluate my thesis work.

Heartfelt thanks to my seniors, batch mates, juniors and friends for their everlasting support throughout the thesis work. Special thanks to *Vidushi Chaudhary* for her support and motivation throughout the thesis work.

I would further like to thank Indraprastha Institute of Information Technology, Delhi for providing an excellent research oriented environment, world class faculty, administration and state of the art facilities. I feel lucky and proud to be a part of one of the best research led institute in the world.

Special thanks to my parents and my sister for their continuous support and motivation.

Contents

List of Figures

List of Tables

Chapter 1

Research Motivation and Aim

In 1990's IT computing infrastructure in corporate organisations used client server computing. The computing infrastructure was mostly limited to desktop or PC only connected to the servers via Local Area Network (LAN). The employees were expected to come to their workplace and perform their duties from their respective desktops or PC's. In the later part of the decade the technological advancements led to the popularity of portable computers known as laptops. The IT computing infrastructure in corporate organisations now used client server computing comprised of PC's and Laptops connected to the servers via LAN or Wireless Local Area Network (WLAN) or Virtual Private Network (VPN). The IT department ensured the security and privacy of corporate data via network based operating systems and firewalls which monitored all the data going into and out of the corporate organisation. The task of corporate data security and privacy was relatively easy then because IT department in the corporate organisation had to worry only about standard Windows based desktops or laptops all using the same operating system.

Enterprise computing went mobile after Blackberry introduced smartphones in the year 1999. Blackberry smartphones supported push email, mobile telephone, text messaging, Internet faxing, Web browsing and other wireless information services [34]. A smartphone is a mobile phone with a mobile operating system having more advanced capability than a feature phone [36].

Year 2000 onwards corporate organisations started giving their managerial employees the blackberry devices to facilitate their global business operations. This improved the employee productivity too since the employees were available round the clock. The corporate data security and privacy was ensured by the administrative tool set and integral security features of Blackberry devices like built in encryption, remote wipe, Blackberry server that ties into Microsoft Exchange etc. All this enabled the IT department in the corporate organisations to manage their employee smartphones.

In the last few years, unlike the past, the computing devices being used by corporate organisa-

tions and those used by consumers are converging giving rise to a new phenomenon known as consumerization of IT. It refers to the trend of information technology first emerging in consumer market and then spreading to business and government organisations. The rapid proliferation of smartphones and tablets from companies like Google, Apple, Nokia, Samsung and many more has fuelled this phenomenon of IT consumerization immensely. Table 1.1 shows the popular companies and the mobile operating systems launched by them.

Company	**Mobile OS**	**Type**
Google Inc., Open Handset Alliance	Android	Free and Open Source
Apple Inc.	iOS	Closed source, proprietary, on top of open source Darwin core OS
Microsoft	Windows	Closed source, proprietary
Research In Motion Limited, trading as Blackberry	Blackberry	Closed source, proprietary
Samsung Electronics	Bada OS	Closed source, proprietary
Accenture on behalf of Nokia	Symbian OS	Closed source, proprietary

Table 1.1: Mobile Operating Systems launched by popular companies

The user friendly and appealing features like tocuhscreen, application stores etc of smartphones and tablets attracted the corporate employees to personally own these devices. Slowly it was observed that managerial level corporate employees who had the corporate blackberry device had started keeping personal smartphone too for their personal use. The rapid proliferation of smartphones made the feature phones almost obsolete amongst the corporate employees. Corporate organisations realized this trend and came up with a corporate policy popularly known as Bring Your Own Device (BYOD) towards the end of the year 2009 [35].

1.0.1 Bring Your Own Device

Bring Your Own Device (BYOD) is a policy, adopted by corporate organisations worldwide, which allows the employees to bring their personal devices to their workplace and use them to access the privileged company information and applications while being both inside and outside the workplace [35]. BYOD is also known as Bring Your Own Technology (BYOT), Bring Your Own PC (BYOPC), Bring Your Own Phone (BYOP).

The important advantages of the BYOD policy in corporate organisations are as follows:

1. **Increased employee productivity:** BYOD policy enables the employees to be available round the clock thus increasing employee productivity. The familiarity and proficiency of the employees with their own devices and software contribute to increased employee productivity.

2. **Improved employee satisfaction:** BYOD policy enables the employees to work on the devices of their choice and use the softwares they like thus improving employee satisfaction. Earlier it was mandatory for the employees to use the devices provided by the corporate organisation to do their work irrespective of their likes and dislikes for that device. This was one of the primary reasons because of which employees starting keeping different devices for work and personal use.

3. **Reduced costs:** BYOD policy reduces the corporate expenditure since the corporate organisations now don't need to provide their employees with workstations and reimburse the cost of the mobile device. According to analysts by 2017 more than 30 % companies expect to stop providing devices to their employees [28]. The corporate organisations not only save the cost of the device itself but also save the ongoing costs associated with hardware upgrades, software upgrades and installation of new software.

4. **Attractiveness to job seekers:** An organization BYOD policy attracts the employees thus leading to lower attrition rates and attracting better talent to the organization from outside.

5. **Increased flexibility:** A BYOD policy enables it's employees to work remotely without requiring to carry multiple devices to satisfy their personal and work needs because they will have everything they need in one device itself.

6. **Newer Technology:** Employees tend to keep themselves updated with the latest technology related to the mobile device or software whereas the corporate organisations hardware and software updates are subject to the up gradation cycle in the IT policy. Thus corporate organisations adopting BYOD policy can reap the benefits of the latest technology.

The main challenges in implementing the BYOD policy in corporate organisations are as follows:

1. **Corporate data security:** Corporate data security is one of the biggest concerns of the corporate organisations in implementing a BYOD policy. Permitting employees to access the privileged corporate information and applications on their device can lead to pertinent corporate data being compromised [22]. The problem of corporate data security becomes even more important in cases when the employees lose their personal device or leave the company.

2. **Employee privacy:** The employees are apprehensive that the corporate organisations may spy or track their activities and may have access to their personal passwords, websites and information on their personal device if used for both work and personal use [8].

3. **Manual provisioning of devices:** In large corporate organisations it is almost impossible for the IT department to scale to thousands of employees with dozens of different device types, operating system platforms and Wi-Fi drivers [22] [21].

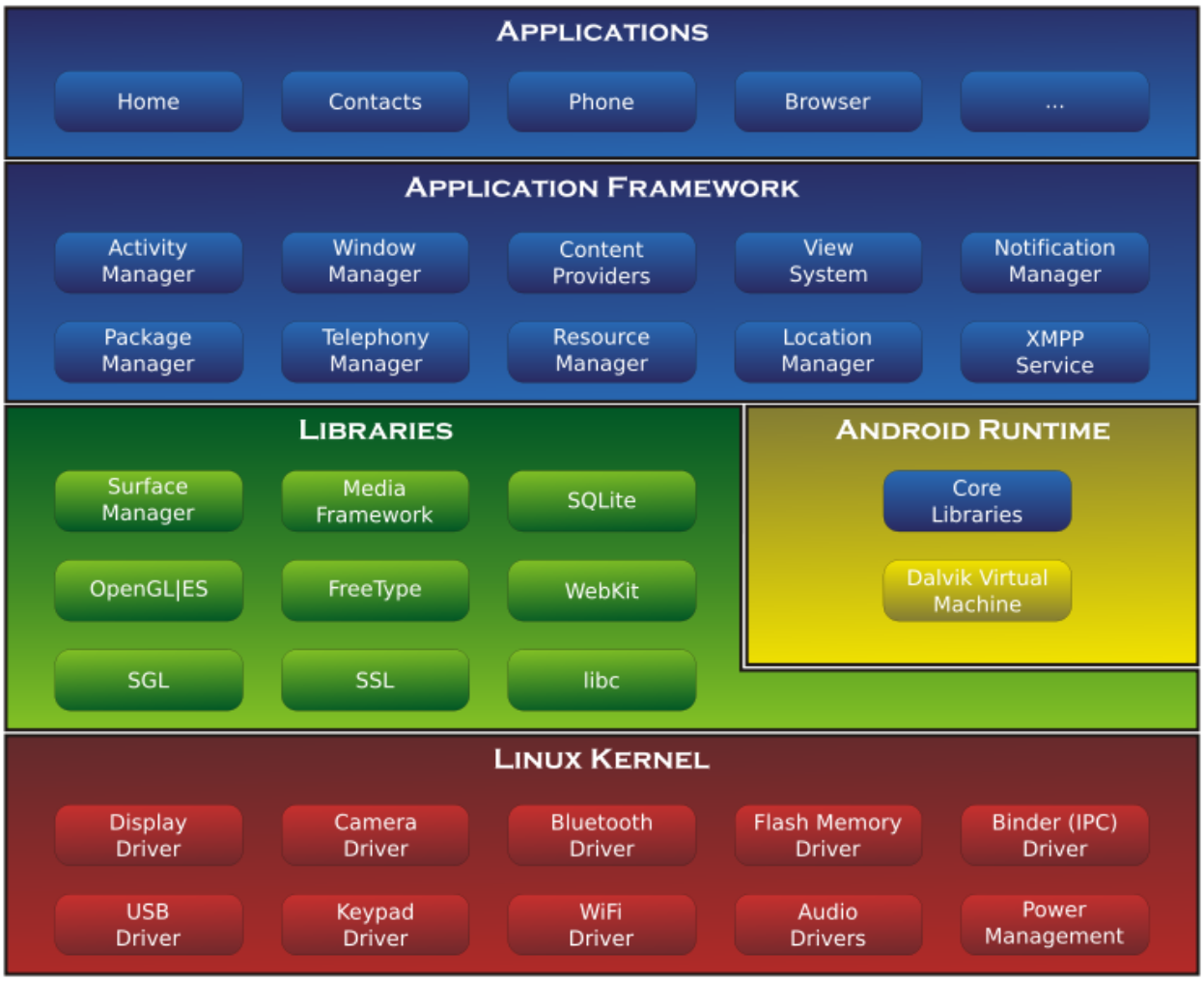

Figure 1.1: Android OS architecture: Layers in the Android stack
[32]

4. **Troubleshooting:** The IT department in corporate organisations will face a tough task to quickly analyse problems associated with diverse devices on the network [22] [21].

1.0.2 Android OS

Android is a mobile operating system developed by Open Handset Alliance (OHA). OHA is a group of 84 hardware, software and telecommunication companies who aim to develop and advance open standards in for mobile devices [1]. Android OS was first released in 2007. Android is open source and google releases the code under Apache License.

Android is a Linux based OS and consists of a kernel based on Linux kernel version 2.6. Android 4.0 and further releases are based on Linux kernel 3.x [32]. Android OS architecture, as shown in Figure 1.1 is composed of 4 layers:

1. **Linux Kernel:** It is the bottom most layer of the Android stack and provides the following functions:
    - Memory management
    - Power management
    - Hardware drivers
    - Network stack
    - Support for shared libraries
    - Security settings
2. **Libraries and Android Runtime:** Above the Linux kernel layer is the set of libraries written in C/C++ which provides the core functionality needed by the developers and device owners. The core libraries are then bundled with a customized Java virtual machine known as Dalvik Virtual Machine to provide the Android run time environment, which is where applications run [14]. Each Android application runs in its own Dalvik Virtual Machine (VM). Dalvik VM runs in its own virtual machine. Android applications are compiled into .dex files which are run by the Dalvik VM.
3. **Application Framework:** It exposes Java API for application developers thus allowing the developers to build applications that take full advantage of the device hardware. Developers are given full access to the framework API used by the core applications, which is designed to simplify the reuse of components.
4. **Applications:** It is the topmost layer of the Android stack and this is where our applications fit. This layer includes several standard applications that come preinstalled with any Android device such Dialer, Web browser, Contact manager etc.

1.1 Research Motivation

BYOD is the indispensable future of IT in corporate organisation because of the immense benefits (as stated earlier in section 1.0.1) it brings along with itself. Testimony to this fact is the Gartners [11] recent prediction that by 2017, half of the employers will require the employees to use their own devices for work purpose.

BYOD typically spans smartphones and tablets and Android is the leader in mobile OS. Year 2013 Q1 figures for handset sales released by Gartner [12] reveals that almost 75 % of all smartphones sold in Q1 were Android based (as shown in Table 1.2) bearing testimony to the fact that Android is the leader in mobile OS.

The security problem in BYOD is centered around the end point device since it is used to access pertinent corporate information. Android, unlike iOS, Windows, Blackberry etc, being a free and open source mobile OS platform is more susceptible to security breaches. According to a

Operating System	**1Q13 Units (in Thousands)**	**1Q13 Market Share (in %)**	**1Q12 Units (in Thousands)**	**1Q12 Market Share (in %)**
Android	156,186.0	74.4	83,684.4	56.9
iOS	38,331.8	18.2	33,120.5	22.5
Blackberry	6,218.6	3.0	9,939.3	6.8
Windows	5,989.2	2.9	2,722.5	1.9
Bada	1,370.8	0.7	3,843.7	2.6
Symbian	1,349.4	0.6	12,466.9	8.5
Others	600.3	0.3	1,242.9	0.8
Total	**210,046.1**	**100**	**147,020.2**	**100**

Table 1.2: Worldwide Smartphone Sales to End Users by Operating System in 1Q13 [12]

report from Kaspersky labs [17] about 99 % of all mobile threats are targeted towards Android devices. Malicious applications is one of the major sources of mobile threats in Android. Android application capabilities are restricted by application permissions which the application declares at the time of installation and cannot be changed at later point of time. A user can either choose to grant all requested permissions or do not install the application at all. Permissions allow applications to access specific data and capabilities on a device, including location, contacts, SMS messaging, identity information, and the ability to access the Internet. However it is difficult for end-users to evaluate the ill-effects of the permissions requested at the time of application installation. Further Android follows an open application distribution model which allows the users to download applications from various places such as Googles Android Market, Amazons Appstore for Android etc. Google's Android market performs some security checks when the applications are submitted but expects that community will participate in identifying malicious applications.

Therefore it becomes imperative to study the security implications of allowing Android based mobile devices as a part of BYOD policy. Gartners [13] survey also shows that BYOD is top concern for enterprise mobile security.

1.2 Research Aim

In this work we formulate a generic BYOD policy, from a corporate data security perspective, which every organisation would like to implement. Keeping in view the framed BYOD policy we study the possible security breaches and come up with techniques to detect the security breaches if such a BYOD policy is adopted for Android based mobile devices. Further we propose a BYOD solution architecture which can be adopted by corporate organisations.

1.2.1 Advantages

The work presented in this report has several advantages:

- Corporate data security accomplished to some extent.
- Employee privacy preserved.
- Minimal battery drainage.

Chapter 2

Related Work and Research Contributions

2.1 Related Work

The work presented in this report is related to the area of security breach detection in Android based mobile devices being used in corporate organisations as a part of the BYOD policy. In this section, we discuss some closely related work and present novel research contributions in context to existing work. We categorize the related work into 3 categories namely security implications of BYOD policy and existing solutions, detecting the root status of the Android based mobile devices and Android malware detection.

2.1.1 Security implications of BYOD and existing solutions

There has not been much of publishable research regarding security implications of BYOD, to the best of our knowledge. Scarfo [26] discusses two security models around BYOD namely Access Control and Device Control. In the access control model IT is a service and IT resources are provided as cloud services by application and desktop virtualization. Virtual desktop and virtual applications are managed and secured centrally, end users just work on the image of the environment in the datacenter. On the other hand Device control model is based on the Mobile Device Management (MDM) technology based on the full control of mobile devices. Scarfo further suggests that the future is possibly the integration of hands off approach [1] and some key features of MDM. He further states that BYOD adoption style should be simple and friendly and the necessary constraints should be enforced only in critical situations by considering the right kind of roles and tasks giving the employees the liberty to choose their devices. Wei et al. [31] identify the undesirable activities of malicious android application in the enterprise and identify some solutions to secure the data and mitigate the security risks.

[1] Hands off approach refers to providing IT resources as cloud services: Desktop virtualization, application virtualization, application stores in private and public cloud style [26].

Existing BYOD solutions include Mobile Device Management (MDM) [35] and Network Access Control (NAC) [29] solutions which are either used independently or in combination. Typical MDM solutions are client server based solutions where employees mobile device is a client and the corporate server represents the server end. This solution operates in a 24*7 mode and is used by companies to fully control the employee mobile device. The primary disadvantages of this type of solution is employee privacy breach and heavy battery drainage due to it's continuous mode of operation. Typical NAC solutions ensure end point security i.e. corporate server holding privileged company information. These solutions typically examine the security status of the mobile device trying to connect to the corporate network and if the device meets the security compliance criteria then it authenticates the users logging into the network and determine what they can see and do. The primary disadvantage of NAC solution is that it fails to provide any security for the corporate data residing on the employee mobile device when the device is not connected to the corporate network.

Recently Samsung has released a comprehensive enterprise mobile solution named *Samsung Knox* [24] for Android based mobile devices to be used as part of the BYOD policy. Three key features of Samsung Knox include platform security, application security and mobile device management. In order to ensure platform security, secure boot procedure is used which prevents unauthorized operating systems and software's from loading during the start up process. Secure Boot requires the device boot loader, kernel, and system software to be cryptographically signed by a key verified by the hardware. Samsungs TrustZone-based Integrity Measurement Architecture (TIMA) makes use of ARM trust zone hardware providing continuous integrity monitoring of the linux kernel. TIMA detects malicious attacks, which tend to attack the kernel and boot strap processes, and notifies the corporate IT department via MDM. Application level security is ensured using the application containers, on device encryption and virtual private network support. Application containers separate Android environment within the mobile device, completed with its own home screen, launcher, applications, and widget. Applications outside the container cannot use Android Inter Process Communication (IPC) with applications inside the container thus preventing data leakage. Finally the MDM solution enables the corporate IT department to monitor, control and administer all deployed mobile devices.

2.1.2 Detecting the root status of the Android based mobile devices

Jang et al. [16] proposed three methods to detect rooting attacks on Android based smartphones namely Inter Process Communication (IPC) monitoring based rooting attack detection, signature based rooting attack detection and activity based rooting attack detection. IPC based method detects the processes suspicious of rooting attack by analysing the number of occurrence of IPC pipe messages related with the attempt to carry out the rooting attack. Signature based method detects applications suspicious of rooting by decompiling the application and finding whether the application contains a cross-compiled file and extracting the file characteristics. An application may be a rooting module if the ELF character strings are present in the executing

code. Activity based method monitors the number of packets being transmitted outside the device, number and characteristics of processes in progress and the events occurring in the system through CPU consumption rate, Wi-Fi and 3G network. All three methods are for detecting whether a particular application or process is trying to root the Android based mobile device. They do not provide information on the root status of the device after the rooting attack has completed.

2.1.3 Android malware detection

There has been an extensive research in the area of Android malware detection. Techniques used to detect Android malwares can be classified into 2 types namely:

- **Static analysis:** Analysis of the Android applications without actually executing the application on the Android based device.
- **Dynamic analysis:** Evaluation of the Android applications by actually executing the applications on the Android based device.

Android malware detection using static analysis

Zhou et al. [40] detect malicious Android applications on popular Android markets using permission based behavioural footprinting scheme and heuristics based filtering scheme. Permission based behavioural footprinting scheme is used to detect new samples of known Android malwares. The manifest file of each known malware application is processed to study the permissions requested by the malware and succinctly summarizes the wrongdoings into a so called permission based behavioural footprint. This footprint is used to detect new samples of known Android malwares. Heuristics based filtering scheme is used to detect malwares that have been not reported before. It recognizes suspicious behaviours from possibly malicious applications and detects certain Android features that may be misused. These features help to identify new unknown malware applications. Kim et al. [19] and Zhou et al. [40] also consider API calls in their respective works while detecting malicious applications. Dicerbo et al. [7] also detect malicious applications by using Android security permissions. Wu et al. [37] developed a system named DroidMat which detects malicious Android applications by analysing the requested permissions, intent message passing etc from each applications manifest file and regards components such as activity, service, receiver as entry points for tracing API calls related to permissions. Different clustering algorithms are used to infer different intentions of Android malware followed by the use of Singular Value Decomposition (SVD) method to decide the number of clusters. The system finally uses kNN algorithm to classify Android applications benign or malicious. Sanz et al. [25] also use machine learning techniques to detect malicious applications by extracted permissions from the application itself. Egners et al. [9] identify attacks to user's device using applications requesting non-suspicious permissions providing insights about the permissions

which can be misused by malicious applications. Li et al. [20] analyze the behaviour of malicious applications in Android using the permissions seeked by the application.

Android malware detection using dynamic analysis

Isohara et al. [15] detect malicious Android applications by logging all the system calls and filtering events made by an application. The logs are further analysed and searched for signatures described by regular expressions to detect a malicious activity. Kefei et al. [18] design and implement a network packet collection tool for Android platform. It does real time capturing of packets moving in and out of the Android network equipment by using Libpcap. Filtering rules can be applied further to detect malicious Android applications sending out sensitive data. Zhao et al. [38] proposed AntiMalDroid to detect malicious Android applications. The proposed framework uses logged behaviour sequence as a feature to detect Android malware effectively in runtime. They also extend the malware characteristics database dynamically. Burguera et al. [4] propose Crowdroid to detect Android malware by tracing system call behaviour and converting then into feature vectors followed by the application of k-means algorithm for detecting malware. Enck et al. [10] propose TaintDroid which tracks the flow of information in Android providing real-time privacy monitoring in smartphones. Information flow is tracked by using variable level tracking (provided by VM interpreter) within untrusted application code, message level tracking between applications, method level tracking for system provided native libraries and file level tracking to ensure that the information retains its taint markings. Blasing et al. [2] execute applications in an isolated environment and log system level interactions to identify malicious applications.

2.2 Research Contributions

In context to closely related work, this report makes the following novel contributions:

1. The work presented in this report is the first step in the direction of studying and detecting security breaches in Android based mobile devices from BYOD perspective. Although there has been some work done in the area of Android malware detection but the detection of security breaches in Android based mobile devices from BYOD perspective is the novel research contribution of this work.

2. We use one class classifier approach using permissions as discriminatory features to identify malicious applications.

3. We use system call logging for Android malware detection. These system call logs generated by an application are checked for signature patterns. The signature patters are made by studying and analysing the system call logs of known Android malware families.

4. We propose a BYOD solution architecture for Android based mobile devices using our

own detection mechanisms. This solution architecture preserves employee privacy, reduces battery drainage and provides corporate data security to some extent.

Chapter 3

Proposed Solution Approach

We formulate a generic and comprehensive sample BYOD policy[1] (Section 3.1) from corporate data security perspective for a corporate organisation. We thereafter identify some of the prominent threats (Section 3.2) to corporate data security w.r.t. formulated policy. In Section 3.3 we find methods and techniques to detect the threats and attacks identified in section 3.2. In the end we propose a BYOD solution architecture (Section 3.4) using the techniques found in the Section 3.3.

3.1 Comprehensive sample BYOD policy

1. The personal mobile device which the employee intends to use for corporate work, as a part of the BYOD policy, needs to be registered with the corporate IT department.

2. It is mandatory for the employees to use the original device operating system and keep the device updated with security patches and updates as released by the manufacturer.

3. Employees will not root or jailbreak their device. Rooted or jail broken devices will be prohibited from accessing the corporate data and network.

4. It is mandatory for the employees to install and run a *BYOD Security Suite* application which monitors the security status of the mobile device. This application can be downloaded from the corporate network. Attempts to uninstall this application without prior approval from corporate IT department may lead to un-registering of the device from the corporate BYOD program with immediate effect.

5. It is mandatory for the employees to use the corporate e-mail client application named *BYOD E-Mail* on their mobile devices. This client application can be downloaded from the corporate network.

[1]Some of the points of this BYOD policy have been adopted from the BYOD tool kit released by the Digital Services Advisory Group and Federal Chief Information Officers Council [30], Govt. of USA

6. It is mandatory for the employees to use the corporate calendar client application named *BYOD Calendar* on their mobile devices. This client application can be downloaded from the corporate network.

7. It is mandatory for the employees to use the corporate Virtual Private Network (VPN) client software named *BYOD VPN* to access corporate network services.This client software can be downloaded from corporate network.

8. Employees must comply with the password policies of the organisation (for BYOD email, calendar and VPN client applications), password expiration (45 days) and password history (15).

9. Employees will not download or transfer any kind of corporate data to their personal devices.

10. Employees are prohibited from backing up the corporate data to any other device or third party cloud services like Dropbox etc.

11. Employees are required to password protect their personal devices.

12. Employees agree not to share the device with other individuals due to the business use of the device.

13. Employees resolve to delete sensitive[2] and confidential corporate data that may have been unintentionally downloaded and stored on the device while viewing e-mail attachments.

14. In case the mobile device is lost or stolen the employee is obligated to inform the corporate IT department immediately after it comes to his/her notice. The corporate organisation reserves the right to delete the corporate client applications on the mobile device, without any prior notice, in all such incidents.

15. The corporate organisation reserves all rights to perform manual and automated scanning of the mobile device, any time, without any prior notice, for anomalies and limit the computing privileges of the mobile device registered for BYOD program and take other administrative or legal action in case of failure to comply with the above mentioned rules of the policy.

3.2 Data security threats w.r.t. formulated BYOD policy

In corporate organisations, the threats to corporate data can be categorized as follows:

- **Inside Threats:** Threats to corporate data from the employees itself.

[2]Sensitive corporate data is the proprietary information which if compromised can cause serious harm to the organisation owning it.

- **Outside Threats:** Threats to corporate data from the outside world including malware applications etc.

The prominent security breaches w.r.t. the formulated comprehensive sample BYOD policy are as follows:

1. The employee, intentionally or unintentionally, roots or jailbreaks his/her device after registering the device with the corporate IT department under BYOD policy.
2. The employee, intentionally or unintentionally, uses applications on the device which are malicious in nature.
3. The employee, intentionally or unintentionally, installs and uninstalls malicious applications on the device.
4. The employee, intentionally or unintentionally, changes the incoming and outgoing server and port settings of the corporate email client application named *BYOD E-Mail*.
5. The employee, intentionally or unintentionally, downloads the corporate e-mail attachments to the mobile device and does not delete them.
6. The employee, intentionally or unintentionally, makes a backup of corporate documents to other external device or third party cloud services like Dropbox etc.

The threats to corporate data security listed above is not an exhaustive list but represents few prominent possible security breaches.

3.3 Detection of threats to corporate data security

We find some methods and techniques in order to detect the threats to corporate data security (as discussed in section 3.2).

3.3.1 Detecting the root status of the Android based mobile device

Rooting of an Android device refers to the process of getting unrestricted access or superuser permissions to the Android OS. Only unrooted mobile devices are allowed in the BYOD policy since the root access circumvents the security restrictions put in place by the Android OS and there is no real effective way to tell what the application intends to do with superuser permissions. Some of the undesirable consequences of using applications with root permissions are as follows:

- Replace the corporate and personal email client applications with a modified one.
- Delete corporate as well as personal files such as applications and application data.

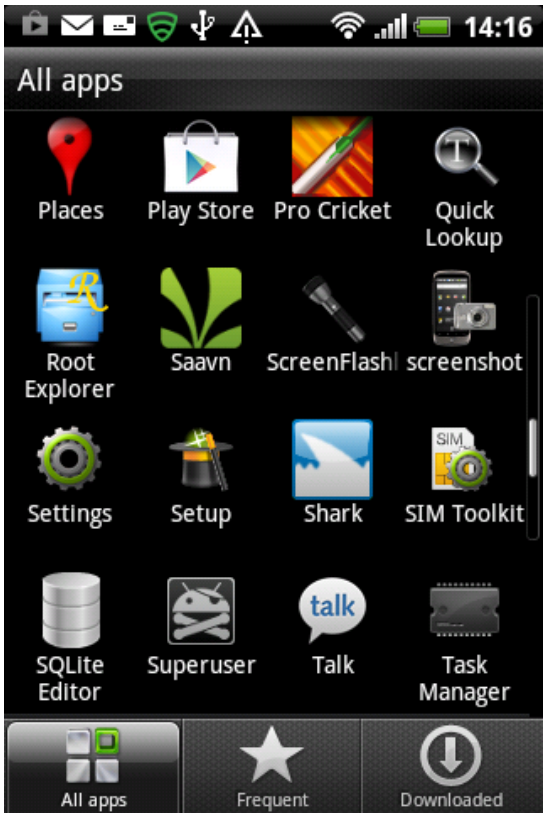

Figure 3.1: Superuser application in a rooted Android device

- Steal corporate as well as personal data.
- Download and install other applications which makes calls and SMS to premium numbers costing the employees heavily.
- Download and attempt to install a different modified ROM.
- Replace the keyboard with the version that logs keystrokes.

Rooting is mainly required for advanced and potentially dangerous operations like modifying or deleting system files, uninstalling manufacturer installed applications and to get low level access to hardware itself [33]. Therefore it becomes very important to detect the rooting status of the Android based device.

The process of rooting varies from device to device. Usually it involves the exploitation of a security bug in Android [16] and thereafter copying the *su binary* to a location in the current process's PATH (e.g. */system/xbin/su*) and granting it executable permissions with the chmod command [33]. Typically during the process of rooting the Android based device a superuser application is also installed, as shown in Figure 3.1, which supervises applications that are granted root or superuser rights.

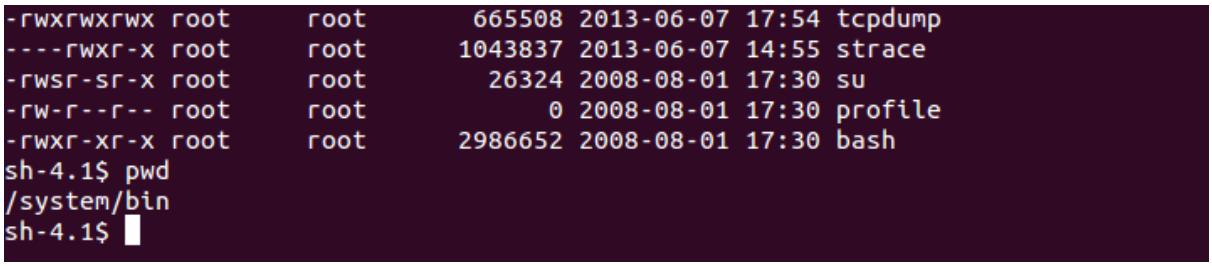

Figure 3.2: Screenshot showing the superuser binary *su* file in the */system/bin/* directory in adb shell of a rooted Android device

We use 5 methods to detect whether an Android device is rooted. Method 1,2 and 3 are automated whereas Methods 4 and 5 are manual in nature and are specific to HTC devices.

Method 1

We observe that in rooted Android devices the superuser binary *su* file is present in one of the following three directories:

- */system/bin/*
- */system/sbin/*
- */system/xbin/*

Figure 3.2 shows that the superuser binary *su* file is present in the */system/bin/* directory. Therefore in order to detect the root status of the Android device we check for the superuser binary file existence. If the superuser binary file exists the device is said to be rooted else it is unrooted.

Method 2

We observe that in rooted Android devices *Superuser.apk* file is present in the */system/app/* directory as shown in Figure 3.3. Therefore in order to detect the root status of the Android device we check whether the *Superuser.apk* file is present on the device. If the superuser application apk file is present the device is said to be rooted else it is unrooted.

Method 3

In rooted Android devices the superuser command successfully executes. Therefore we detect the root status of the Android device by executing superuser command from the code. If the superuser command executes successfully the device is said to be rooted else it is unrooted.

```
-rw-r--r-- root     root       994673 2011-12-13 20:54 IndicInputMethods.apk
-rw-r--r-- root     root       196640 2008-08-01 17:30 Superuser.apk
-rw-r--r-- root     root       933408 2011-12-13 21:44 HTCCamera.odex
-rw-r--r-- root     root      1014332 2011-12-13 21:32 SetupWizard.apk
-rw-r--r-- root     root        19418 2011-12-13 21:44 LiveWallpapersPicker.apk
-rw-r--r-- root     root       202192 2011-12-13 21:44 HtcVideoPlayer.odex
-rw-r--r-- root     root       267399 2011-12-13 21:44 HtcAddProgramWidget.apk
-rw-r--r-- root     root       143872 2011-12-13 21:44 GSD.odex
-rw-r--r-- root     root       200728 2011-12-13 21:44 Smith.odex
sh-4.1$ pwd
/system/app
sh-4.1$
```

Figure 3.3: Screenshot showing the *Superuser.apk* file in the */system/app/* directory in adb shell of a rooted Android device

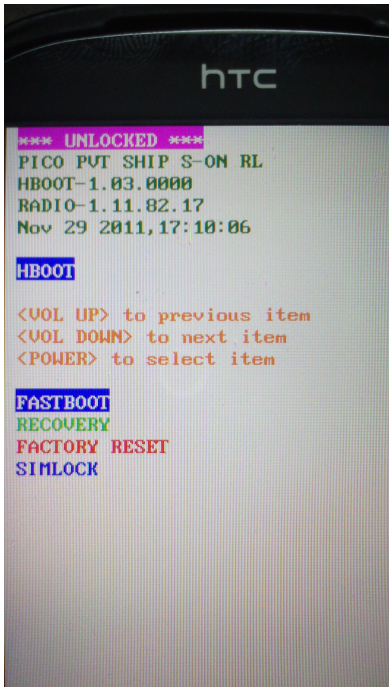

Figure 3.4: Bootloader mode in HTC Explorer A310e

Method 4

Checking for the S-ON and S-OFF status by booting into HBOOT (as shown in Figure 3.4) in the HTC device. S-OFF means that the NAND portion of the device is unlocked and can be written to. The default status of HTC devices is S-ON which means that neither can you access certain areas of the system nor can you guarantee a permanent root. S-OFF status does not necessarily imply that the device is rooted. Similarly S-ON status does not necessarily imply that the device is unrooted. It may or may not be rooted in case the status is S-ON. Figure 3.4 shows the bootloader screen of a rooted Android device with S-ON status. However if the status is S-OFF the probability of the HTC device being rooted is more. This method is manual in nature and is applicable for only HTC based Android devices.

Method 5

Checking whether the bootloader is locked or unlocked. In order to root HTC Android devices the bootloader has to be unlocked. However a unlocked bootloader (as shown in Figure 3.4) does not necessarily imply that the device is rooted, but there is a high probability of a device being rooted in case the bootloader is unlocked. Reason being that most people from non technical background unlock their bootloaders for rooting purpose only. It is important to note here that the bootloader can be relocked too by using the key provided by HTC during the bootloader unlocking process. This method is manual in nature and is applicable for only HTC based Android devices.

If the mobile device is found to be rooted then it is immediately unregistered from the BYOD policy followed by revoking of the access to the corporate server. In case the mobile device is not rooted then we identify the malicious applications installed on the device.

3.3.2 Detecting malicious application in Android based mobile device

In BYOD since the same device is being used for personal and corporate use therefore it becomes important to ensure that no data is being stolen from the device in any form. This ensures not only corporate data security but employee privacy too. Apart from the data theft there should be no malicious application installed on the device which makes calls or SMS to premium numbers without the knowledge of the employee. In Android based mobile devices, applications are the biggest source of stealing data from the device. We use both static and dynamic analysis techniques to identify the potential malware applications on the employees device.

Detecting malicious Android applications using static analysis

We use one class classification approach to detect malicious applications statically. One class classification is also known as unary classification. It tries to distinguish one class of objects from all other possible objects by learning from a positive class training dataset. Traditional

classification problems try to distinguish between two or more classes with the training set containing objects from all other classes. One class classification is based on similarity computation where we find the similarity of the new object (here android application) with the existing / training dataset. The use of one class classification is justified here since our problem is to detect whether an application is malicious or not and we are not interested in the further classification or category in which the malicious application falls. One class classification consists of four steps which are as follows:

1. Positive class training dataset
2. Characterization and training of the classifier
3. One class classification
4. Performance evaluation of the classifier

We use a positive class training dataset of 200 malicious applications from the dataset provided by the Android malware genome project[3] [39].

In order to characterize the classifier we use android permissions as discriminatory features. The use of android permissions as discriminatory features is justified since every Android application has to explicitly declare the permissions it requires in it's manifest file. This requirement of declaring permissions in the central design point of Android security architecture so that by default no application has permission to perform any operation that would adversely impact other applications, the operating system, or the user [6]. Since Android follows the sandboxing approach, sandboxing applications from each other, applications must explicitly share resources and data which they do by declaring permissions needed for additional capabilities not provided by basic sandbox [6]. The permissions are declared statically by the Android applications and the user is prompted for consent at the time of application installation. Android OS does not provide any mechanism for granting permissions dynamically. Therefore in order to detect malicious Android applications we use permissions as discriminatory features in one class classifier.

In order to study and identify the set of discriminatory features (here permissions) we analyse 200 Android malware applications. We identify 25 most commonly used permissions by the Android malware applications. Table 3.1 shows the list of permissions most commonly used by Android malware applications and the description of each permission. Figure 3.5 shows the mapping between the top 25 permissions sought by Android malware applications and the percentage of malware applications requesting that particular permission.

The second step of the classification algorithm is to define a set of discriminatory features which can be used to identify malicious applications. We use only the top 16 permissions, which are

[3]Thanks to Yajin Zhou and Xuxian Jiang for sharing the Android malware application dataset.

Permission	Description
INTERNET	Allows applications to open network sockets
READ_PHONE_STATE	Allows read only access to phone state
ACCESS_NETWORK_STATE	Allows applications to access information about networks
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage
SEND_SMS	Allows an application to send SMS messages
RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting
RECEIVE_SMS	Allows an application to monitor incoming SMS messages, to record or perform processing on them
ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks
READ_SMS	Allows an application to read SMS messages
WRITE_SMS	Allows an application to write SMS messages
WAKE_LOCK	Allows using PowerManager Wake Locks to keep processor from sleeping or screen from dimming
READ_CONTACTS	Allows an application to read the user's contacts data
CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed
VIBRATE	Allows access to the vibrator
WRITE_APN_SETTINGS	Allows applications to write the apn settings
ACCESS_COARSE_LOCATION	Allows an app to access approximate location derived from network location sources such as cell towers and Wi-Fi
ACCESS_FINE_LOCATION	Allows an app to access precise location from location sources such as GPS, cell towers, and Wi-Fi
WRITE_CONTACTS	Allows an application to write (but not read) the user's contacts data
CHANGE_WIFI_STATE	Allows applications to change Wi-Fi connectivity state
MOUNT_UNMOUNT_FILESYSTEMS	Allows mounting and unmounting file systems for removable storage
READ_LOGS	Allows an application to read the low-level system log files
READ_HISTORY_BOOKMARK	Allows an application to read (but not write) the user's browsing history and bookmarks
CHANGE_NETWORK_STATE	Allows applications to change network connectivity state
GET_TASKS	Allows an application to get information about the currently or recently running tasks
PROCESS_OUTGOING_CALLS	Allows an application to monitor, modify, or abort outgoing calls

Table 3.1: List of top 25 most commonly used permissions by Android malware and their description

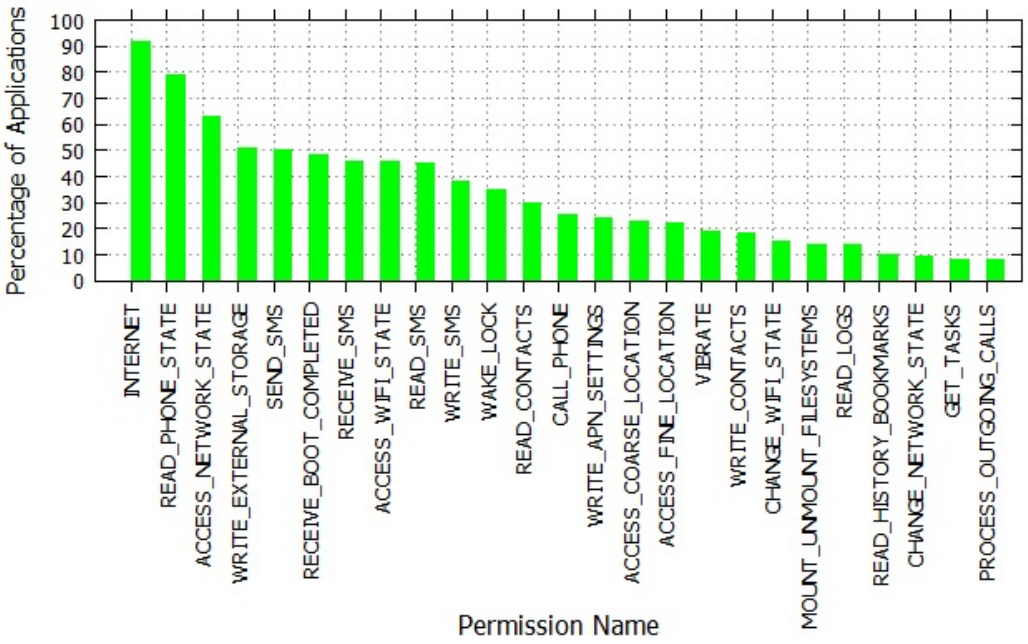

Figure 3.5: Most common permissions requested by Android malware applications

used by more than 20 % malware applications, as discriminatory features to identify malicious applications. In order to train and develop the one class classifier we use the Chaudhary et al. [5] algorithm for weight and score computation [4] of each feature or permission as shown below.

Classifier feature or permission set X is:

X= (x_1, x_2,x_3 . . . ., x_n),where n = Number of features or permissions. In our case n=16 for identifying malicious applications.

Let **W_i = Weight of the discriminatory feature or permission i s.t.**

$$\sum_{i=1}^{n} W_i = 1 \tag{3.1}$$

Algorithm 1 [5] shows the approach of calculating the weight of each feature or permission where the whole process is repeated until the accuracy is optimal. The result of the algorithm shows the contribution of each feature or permission in identifying or detecting the malicious applications. Therefore higher the weight the more important is the permission in identifying the malicious application.

One class classification approach used here is based on similarity computation. Score [5] of the feature or permission is a unique value which represents that feature in comparison to the

[4]This algorithm for weight computation of each feature has been adopted from Chaudhary et al. [5] work on Contextual Feature Based One-Class Classifier Approach for Detecting Video Response Spam on YouTube

Input: A list L of features or permissions.
Result: Weight of each feature or permission.
initialization;
Assign equal weight to each feature (or permission) s.t

$$\sum_{i=1}^{n} W_i = 1 \tag{3.2}$$

Run the classifier and calculate the accuracy of the system, say accuracy1.
for *each feature (or permission) f in L* **do**
    Remove feature (or permission) j from L;
    Adjust weights of rest of the features (or permissions) s.t.

$$\sum_{\forall i \neq j} W_i = 1 \tag{3.3}$$

    Run the classifier and check the accuracy of the system, let accuracy2;
    Let

$$\Delta = \textbf{percentage change in accuracy / 100} \tag{3.4}$$

    if *((Significant change in accuracy))* **then**
        Removed feature (or permission) is an important feature/permission and weight corresponding to this feature/permission should be high;

$$feature_{weight_i} = feature_{weight_i} + \mu \tag{3.5}$$

    else
        Removed feature (or permission) is not an important feature/permission and weight corresponding to this feature/permission should be low;

$$feature_{weight_i} = feature_{weight_i} - \mu \tag{3.6}$$

    end
end

Algorithm 1: Algorithm for Weight computation [5] of each feature or permission.

training dataset. **S_i = Score of the permission or feature i s.t.**

$$0 \leq Si \leq 1 \tag{3.7}$$

Score of the feature or permission in our case here is taken as the percentage of malicious applications using that permission.

Let y = Percentage of malicious applications requiring a permission i.

$$Score_i = (y/100) \tag{3.8}$$

Based on weight and score of each permission, we compute the final value of the permission which is the product of it's score and weight. Finally we compute the C_{value} which represents the similarity or resemblance of the application with the target class and recognizes the malicious nature of the application.

$$c_{value} = \sum_{i=0}^{n} W_i * S_i \tag{3.9}$$

If the computed C_{value} is greater than a threshold value then the application is considered to be malicious else it is considered to be of unknown category i.e. it may or may not be malicious. In this way the one class classifier is developed. Finally we evaluate performance of the classifier on a testing dataset in Chapter 4.

Detecting malicious Android applications using dynamic analysis

It has been observed that there are applications which request for minimal set of permissions and appear to be legitimate but they are actually malicious. On the other hand there are a number of applications which ask for more permissions than required. Such applications however do not misuse those privileges. There are also many legitimate applications which use the same permissions as used by malicious applications. Static analysis method fails in these type of situations. Detecting malicious applications using static analysis fails to capture the actual intent of the application accurately. Therefore in order to overcome the shortcomings of static analysis method it becomes necessary to detect malicious applications using dynamic analysis technique.

We use 50 malicious applications from the Android malware genome project [39] as the training dataset. We log the application activities occurring at the operating system level. We use the strace utility to log all the system calls (List of important system calls and their description is given in Table 3.3) made by an application. Further we analyse the log to find the footprints or signatures of the malicious activity of the application. Thereafter we derive some rules and signatures described by regular expressions, as shown in Table 3.2, to detect the malicious

applications.

Type of threat	Rule/Signature Pattern
Information Leakage	.*recv.*www.*famaz0n-cloud.*
Information Leakage	.*write.*Vodafone.*
Information Leakage	.*read.*Processor.*ARM.*
Information Leakage	.*connect.*family.*addr.*
Information Leakage	.*socket.*
Information Leakage	.*bind.*inet_addr.*
Information Leakage	.*read.*Hardware.*
Information Leakage	.*read.*CPU.*
Information Leakage	.*read.*MIPS.*
Abuse of root	.*system/bin/su.*
Abuse of root	.*system/xbin/su.*
Abuse of root	.*system/sbin/su.*
Abuse of root	.*execve.*/busybox.*
Rooting attempt	.*(execve|read).*/asroot.*
Rooting attempt	A process forking or cloning more than 30 processes

Table 3.2: Rules and Signtaure patterns for detection of malicious Android applications using dynamic analysis

3.3.3 Detecting the installation and uninstallation of applications on the Android based mobile device

An employee can install and uninstall applications of his/her choice on the mobile device registered under the BYOD policy. It may so happen that the employee, intentionally or unintentionally, installs a malicious application on the device and uninstalls the same after some time. Such an application can also steal corporate data as well as personal data from the mobile device. It becomes important to detect and log events of application installation and uninstallation because of the following reasons:

- It reduces the unnecessary drainage of battery which is a critical resource since the module or service to detect whether the application installed is malicious or not is triggered only when a new application is installed after the initial scanning of the device.
- It is important to know the security status of the mobile device and keep the log updated. This is helpful in situations when a malicious application is installed and uninstalled after some time because the mobile device now need not to be rescanned but only the log needs to be updated.

In order to detect and log such events of application installation and uninstallation we use logcat, which is an audit framework on the Dalvik Virtual Machine to monitor the application behaviour. Even though logcat dumps limited events but we successfully find system generated log lines which captures the application installation and uninstallation event. We use 50 different

System Call	Description
read()	Reads from file descriptor
write()	Writes to a file descriptor
fork() and clone()	Creates a child process
exit()	Terminate the current process
open()	Open a file or device
close()	Close a file descriptor
link()	Make a new name for a file
unlink()	Delete a name and possibly the file it refers to
execve()	Execute program
chmod()	Change permissions of a file
stat()	Get file status
lseek()	Reposition read/write file offset
getpid()	Get process identification
getuid()	Get real user id
access()	Check user's permission for a file
mkdir()	Create a directory
rmdir()	Remove a directory
pipe()	Create an interprocess channel
brk()	Change the amount of space allocated for the calling process's data segment
ioctl()	Control input/output devices
fcntl()	File control
dup()	Duplicate a file descriptor
getppid()	Get parent process ID
gettimeofday()	Get the date and time
lstat()	Get file status
munmap()	Unmap pages of memory
statfs()	Get file system statistics
socketcall()	Socket system calls
mprotect()	Set protection of memory mapping
sigprocmask()	POSIX signal handling functions
readv()	Read data into multiple buffers
writev()	Write data into multiple buffers
sysctl()	Read/write system parameters
mlock()	Lock pages in memory
munlock()	Unlock pages in memory
poll()	Wait for some event on a file descriptor
semget()	Returns the semaphore indentifier associated with the given key
semop()	Used in semaphore operations such as signalling and waiting
recv()	Receive a message from a socket
msgget()	Creates or return results from a message queue

Table 3.3: Important system calls description

applications to analyse and derive the application installation and uninstallation signature patterns. Table 3.4 and 3.5 show the application installation and uninstallation signature patterns derived from logcat.

S.No.	Signature patterns
1	^D/SM:PackageReceiver.*package added:*
2	^I/RegisterService.*android.intent.action.PACKAGE_ADDED*
3	^D/PackageManager.*.embedded.*New package installed in*
4	^D/installd.*DexInv:.*success*

Table 3.4: Signtaure patterns for application installation

S.No.	Signature patterns
1	^I/UninstallAppProgress.*Finished uninstalling pkg:*
2	^I/installd.*unlink /data/dalvik-cache/data@app@*
3	^D/SM:PackageReceiver.*package removed:*
4	^I/PackageManager.*Removing non-system package:*

Table 3.5: Signtaure patterns for application uninstallation

3.3.4 Detecting corporate email security breaches

E-Mail communication is an integral part of every enterprise and therefore plays a vital role in the company's BYOD policy. It is one of the mediums via which the corporate data security can be breached easily. All the inbound emails must be routed through the corporate mail server which filters spam emails, phishing mails etc. Similarly all outbound emails must be routed through the corporate SMTP server so that the corporate data cannot be sent out from the corporate email id via a fake SMTP server to the rival companies.
Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks. Electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for relaying. For receiving messages, client applications usually use either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP).

Whenever an e-mail is sent, the mobile e-mail client application interacts with the SMTP server or mail server (as shown in Figure 3.6) to handle the sending [3]. The host SMTP server may have conversations with other SMTP servers to deliver the e-mail. In corporate organisations the SMTP severs are programmed specifically to monitor all outgoing emails so that no privilege company information is leaked by any employee. We observe that mobile email client applications permit the user to configure the incoming and outgoing server settings (as shown in Figure 3.7 and Figure 3.8). We analyse the working of the com.htc.android.mail application in the HTC

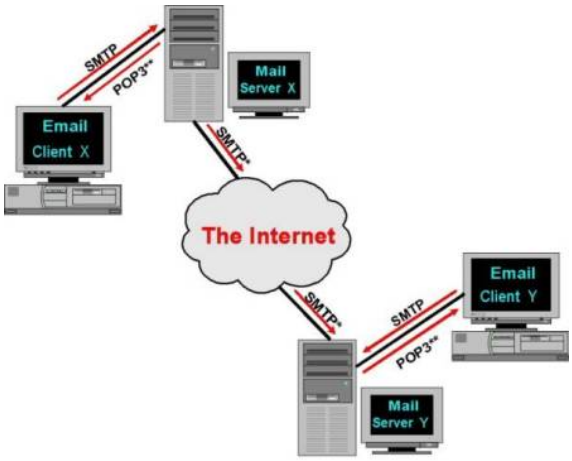

Figure 3.6: Working of email and protocols

Explorer A310e handset.

Therefore it is possible that the employees may configure fake SMTP server settings on their mobile device for the corporate e-mail and send out privilege corporate information via emails bypassing the corporate SMTP server or mail server. This is serious kind of threat which needs to be overcome.

We further observe that all the incoming and outgoing mail server settings for an email account are stored in mail.db database and the accounts table in it, as shown in Figure 3.9, created by the com.htc.android.mail application.

In order to detect and overcome threats arising from the changes in incoming and outgoing mail server settings, we propose the use of a customized corporate e-mail client application where all such incoming and outgoing server settings are hard coded or any changes to such settings are monitored. As a proof of concept we show the use of trigger on a sqlite database which generates a message every time the changes are made to the accounts table. This method can be used by corporate email client application to detect the instances of tampering with the incoming and outgoing mail server settings in the corporate e-mail client application.

We do not recommend the use of the same email client application for personal and corporate use since in that case the employee privacy is breached if the mail.db database and the accounts table is monitored by the corporate organisation.

3.3.5 Detecting the downloaded corporate e-mail attachments and corporate documents on the device

The problem of detecting the downloaded corporate e-mail attachments and corporate documents is a hard problem since once the document is downloaded to the device it is difficult to distinguish between the employees personal document and the corporate document. Monitoring

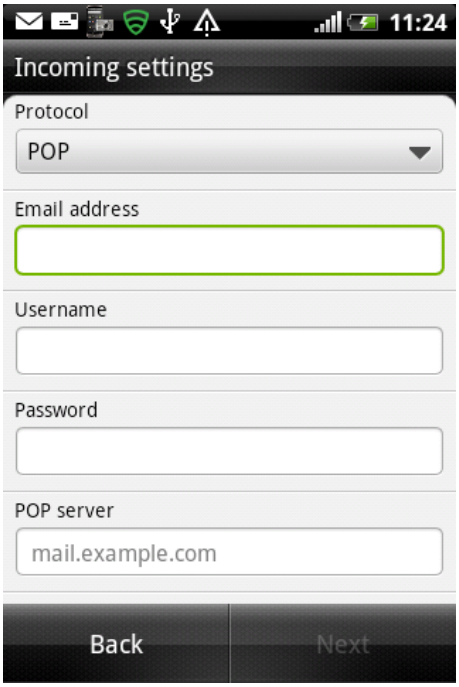

Figure 3.7: Screenshot of screen permitting the configuration of incoming server settings for e-mail

Figure 3.8: Screenshot of screen permitting the configuration of outgoing server settings for e-mail

Figure 3.9: Screenshot of the entries in the accounts table of the mail.db database of htc mail application

all the documents on the device is not a feasible solution since it leads severe employee privacy breach. Differentiating personal and corporate documents by using distinct naming conventions for corporate document is not a good solution since the employee themselves can rename the corporate documents once downloaded to the device. The partitioning of the mobile device memory into two parts (i.e. one for corporate use and one for personal use) so that the downloaded corporate documents are directed to the corporate partition may not be acceptable to the employees. This remains an open problem. We propose the use of customized corporate e-mail client application for mobile devices which either disables the download link in the corporate e-mail or opens up the documents in the web based office suite similar to google docs eliminating the need and choice of downloading the document at all.

3.3.6 Detecting the event of backup of corporate documents to external device or cloud

The problem of detecting the event of backup of corporate documents to external device or cloud is again a hard problem due to same reasons as mentioned in section 3.3.5. The inability to differentiate between corporate and personal documents makes it impossible to detect the event of backup of corporate documents to external device or cloud. Monitoring all the documents on the device is not a feasible solution since it leads to severe employee privacy breach. Similar to section 3.3.5 we propose that the corporate documents should not be allowed to be downloaded or transferred to mobile device in any way. This remains an open problem.

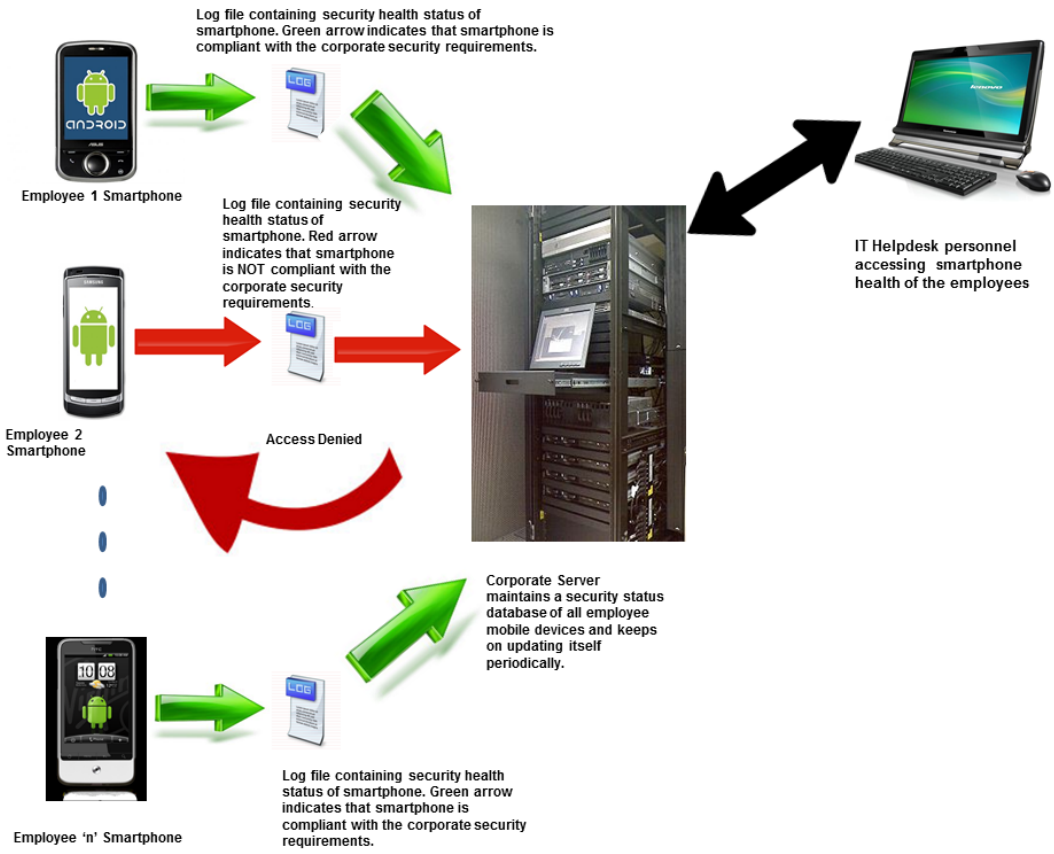

Figure 3.10: BYOD solution architecture

3.4 Architecture of proposed solution

In this section we propose a BYOD solution architecture (as shown in Figure 3.10) which utilizes the methods or techniques to detect security breaches (as mentioned in Section 3.3).

The BYOD solution will work as follows:

1. Every mobile device, registered under the BYOD policy, will run the following applications provided by the corporate IT department:
    - *BYOD E-Mail* which is a corporate email client application for Android based mobile devices.
    - *BYOD Calendar* which is corporate calendar client application for Android based mobile devices.
    - *BYOD VPN* which is a corporate VPN client application for Android based mobile devices.
    - *BYOD Security Suite* which is a corporate application to monitor the security status of the Android based mobile device.

2. *BYOD Security Suite* applications performs the following functions:

   (a) It monitors and logs the root status of the mobile device periodically using the technique mentioned in section 3.3.1.

   (b) It detects malware Android applications on the mobile device using static and dynamic analysis techniques mentioned in Section 3.3.2. Further it alerts the employee about the same and logs details of such applications.

   (c) It monitors, detects and logs the installation and uninstallation of applications including corporate applications namely *BYOD E-Mail*, *BYOD Calendar*, *BYOD VPN* using the technique mentioned in Section 3.3.3.

   (d) It monitors,detects and logs instances of tampering with the *B*YOD E-Mail database and tables containing the incoming and outgoing server settings.

3. The log file generated by the *BYOD Security Suite* application on each mobile device is sent to the corporate server.

4. The log file generated by each mobile device is analysed for anomalies at the corporate server. Such a mechanism saves device battery since all the processing is performed at the server end.

5. In case some anomalies are found in the log file, the corresponding mobile device is denied access to the corporate server and his/her mail access is blocked (as shown in Figure 3.7 for Employee 2 Smartphone).

6. In case no anomalies are found in the log file the corresponding mobile device continues to have access to the corporate server and the the relevant information.

Chapter 4

Results and Discussion

4.1 Root status of the Android based mobile device

We use 5 methods to detect the root status of the Android based mobile device. The methods are as follows:

1. **Method 1** Checking for the superuser binary file existence. If the superuser binary file exists the device is said to be rooted else it is unrooted.
2. **Method 2** Checking whether the superuser application apk file is present on the device. If the superuser application apk file is present the device is said to be rooted else it is unrooted.
3. **Method 3** Checking by executing superuser command from the code. If the superuser command executes successfully the device is said to be rooted else it is unrooted.
4. **Method 4** Checking for the S-ON and S-OFF status by booting into HBOOT in the HTC mobile device. This method is HTC specific and manual in nature.
5. **Method 5** Checking whether the bootloader is locked or unlocked.This method is HTC specific and manual in nature.

We test the above 5 methods on different devices, selected randomly, and get the results which are summarized in Table 4.1.

We observe that the Methods 1,2 and 3 accurately detect the root status of the device. Method 3 can fail in situations when the user is prompted to grant or deny the super user privileges on execution of super user command from the code and the user denies the super user privileges. We observe in Method 4 that even if the HTC device status is S-ON, the device can be rooted (as is the case with the device with IMEI No: 359918041010851. S-OFF status also does not necessarily imply that the device is rooted, it may or may not be. In Method 5 we clearly know that if the bootloader is locked the device is unrooted since in HTC devices the rooting process

IMEI	Method 1	Method 2	Method 3	Method 4	Method 5
359918041010851	Rooted	Rooted	Rooted	S-ON	Unlocked
352264051538128	Unrooted	Unrooted	Unrooted	Non HTC device	Non HTC device
353346054349980	Unrooted	Unrooted	Unrooted	Non HTC device	Non HTC device
359462045267837	Rooted	Rooted	Rooted	Non HTC device	Non HTC device
359462049208464	Unrooted	Unrooted	Unrooted	Non HTC device	Non HTC device

Table 4.1: Results of different methods for detecting root status of Android based mobile devices

involves the unlocking of the bootloader as the first step. However if the bootloader is unlocked then it cannot be said surely, on the basis of the results of the method 5 only, that the device is rooted or unrooted since after unlocking the device it may or may not have been rooted.

4.2 Analysis of malicious Android applications

4.2.1 Static analysis to detect malicious Android applications

We use a testing dataset of 200 applications to evaluate the effectiveness of one class classifier, using permissions as discriminatory features, to identify malicious Android applications. We get an accuracy of 73.5 % in detecting malicious Android applications. The reason for not getting cent percent accuracy can be attributed to the fact that there is a lot of permission overlapping between malicious and non malicious Android applications. We further find out that permissions like INTERNET, READ_PHONE_STATE, ACCESS_NETWORK_STATE cannot be used as discriminatory features with heavy weight in the one class classifier since these permissions are also used by non malicious applications heavily. Therefore the permission based filtering method is not able to specifically detect those malicious applications which access and steal the device information like IMEI, IMSI number etc.

Table 4.2 shows the set of permissions, used as discriminatory features to identify malicious applications in the decreasing order of their weights. The table also provides the possible reason due to which the malicious applications use that particular permission thereby making it a discriminatory feature. Only the top 16 permissions out of 25 permissions identified in Table 3.1 are used in the discriminatory feature set since the remaining permissions do not contribute significantly to the accuracy. We find that the threshold C_{value} comes out to be 0.065 and maximum C_{value} is 0.400816. All applications having a C_{value} or similarity score lower than the threshold value are considered non malicious and the applications having C_{value} or similarity score greater than the threshold value are considered malicious.

Permission	Weight	Remarks
SEND_SMS	0.2500	Granting this permission to applications costs money to the users, therefore heavily used by malicious applications which automatically send send SMS to premium numbers.
CALL_PHONE	0.1800	Granting this permission to malicious applications costs money to the users. However this permission is not as heavily used as the SEND_SMS permission.
RECEIVE_BOOT_COMPLETED	0.1200	Granting this permission to applications enables them to restart automatically after the mobile device boots up and malicious applications need this permission to carry on with their malicious activities.
READ_SMS	0.0850	This permission enables the malicious applications to read the users SMS and gain information about them leading to privacy breach.
WRITE_APN_SETTINGS	0.0800	Allows applications to write the apn settings
RECEIVE_SMS	0.0650	This permission enables the malicious applications to monitor incoming SMS messages, to record or perform processing on them.
WRITE_SMS	0.0500	This permission enables the malicious applications to write SMS messages automatically costing money to the users.
READ_CONTACTS	0.0450	This permission enables the malicious application to read the user's contacts data thus leading to privacy breach.
ACCESS_WIFI_STATE	0.0320	This permission enables the malicious applications to determine if the mobile device is connected to Wi-Fi network.
WRITE_EXTERNAL_STORAGE	0.0290	Allows an application to write to external storage.
WAKE_LOCK	0.0280	Allows using PowerManager Wake Locks to keep processor from sleeping or screen from dimming.
ACCESS_COARSE_LOCATION	0.0092	This permission is used by malicious applications to access approximate location derived from network location sources such as cell towers and Wi-Fi leading to privacy breach.
ACCESS_FINE_LOCATION	0.0068	This permission is used by malicious applications to access precise location from location sources such as GPS, cell towers, and Wi-Fi leading to privacy breach.
ACCESS_NETWORK_STATE	0.0067	This permission is needed by malicious applications to determine whether the internet connectivity is available or not.
READ_PHONE_STATE	0.0018	This permission enables the malicious applications to get the device details like IMEI no., IMSI etc.
INTERNET	0.0015	This permission is used by malicious applications to access the internet and send and receive data.

Table 4.2: Discriminatory features or permissions in the order of decreasing weight to identify malicious applications

4.2.2 Dynamic analysis to detect malicious Android applications

In order to evaluate the effectiveness of our system call logging and analysis method to detect malicious applications we use a testing dataset of 50 applications. We collect the system call logs of the 50 applications and search for signature patterns in the log. To our surprise we get an accuracy of just 46 % in detecting malicious Android applications. After the introspection of the results we find that the reason for low accuracy is the incomplete capturing of the Android Inter Process Communication (IPC) in the system call logs.
Android binder is used for IPC communication in Android. Android binder is a lightweight Remote Procedure Communication (RPC) Mechanism. Each Android application is composed of 4 different components namely activity, service, content provider and broadcast receiver. Data exchange between different components is realized through IPC, if the specific components belong to different processes (applications). This communication works with so called Intents. A content provider can also be queried by an activity via IPC and returns the result.The Binder framework communication is a client server model. A client will initiate a communication and wait for response from a server. The Binder framework uses a client-side proxy for communication. On the server side, a thread pool exists for working on requests. The Binder kernel driver is the heart of the Binder framework. The Binder kernel driver supports the file operations *open, mmap, release, poll* and the system call *ioctl* [27]. So even though the system call logs were capturing the *ioctl* system calls but the call logs do not reveal the binder transactions and binder transaction data hidden in the ioctl system call. Binder transactions cause transition of data from one transits data from one binder interface to another binder interface.

4.3 Installation and uninstallation of Android applications

In order to detect the events of application installation and uninstallation we use logcat which is an audit framework on the Dalvik Virtual Machine to monitor the application behaviour. We analyse the logcat while applications are installed and uninstalled on the device and identify the signature patterns (as described in Section 3.3.3) capturing such events. We use these signature patterns to detect the events of application installation and uninstallation.

In order to test the accuracy of the signature patterns we take a set of 50 Android applications and install them on the Android device and capture the logcat log in a file. Thereafter we search the log file for signature patterns. Every time we were able to accurately identify the application installed for all the 50 applications.

Similarly we randomly choose any application to uninstall on the device and capture the logcat log in a file. On searching the log file for signature patterns we were able to accurately identify the application uninstalled for all the 50 applications.

The above tests establish the correctness and accuracy of our signature patterns for application installation and uninstallation.

4.4 Incoming and outgoing mail server settings of email client

We propose a customized corporate email client application for corporate use where either the incoming and outgoing corporate mail server settings are hard coded or the sqlite database and the table in it containing the server setting information is monitored using triggers. As a proof of concept we implement a trigger on a table named accounts in the mail.db database of a custom application. We observe that the trigger successfully records any kind of changes or tampering done with the accounts table in the mail.db database. The proposed BYOD solution architecture can make use of this technique to detect any kind of tampering done with the corporate email client application database and the table containing the incoming and outgoing mail server settings.

4.5 Downloaded corporate e-mail attachments

The problem of detecting the downloaded corporate e-mail attachments is hard and remains open. Once the corporate e-mail attachment is downloaded to the mobile device it is hard to distinguish between corporate document and the personal document on the mobile device. Monitoring all the documents on the mobile device is not a solution since it will lead to severe employee privacy breach. Distinct naming conventions for corporate e-mail attachments too fail to differentiate between corporate and personal documents since the employee can rename the corporate documents once downloaded to the device.

4.6 Backup of corporate documents

The problem of detecting the process or event of backup of corporate documents is again a hard problem and remains open because of the same reason as mentioned in section 4.5. It is difficult to distinguish between personal and corporate documents on the mobile device therefore it becomes hard to detect which document is being backed up on external device or cloud.

4.7 Deficiencies in proposed solution architecture

The proposed solution architecture addresses some of the important challenges in detecting security breaches in Android based mobile devices being used in byod policy for corporate organisations. However we observe some deficiencies or limitations of the proposed solution architecture which are as follows:

1. It fails to address the problem of distinguishing between corporate documents and personal documents on the mobile device.

2. The solution fails to provide remote installation of corporate applications and uninstallation of malware applications on the mobile device. In order to remotely install and uninstall applications the device should be rooted for it and as part of BYOD policy we do not permit rooted devices due to security concerns.

3. Presently the corporate applications and security suite application on the mobile device is being installed as user applications. However the effectiveness of the proposed solution architecture can increase considerably if the corporate applications are installed as system applications since then the corporate applications would have been able to receive commands remotely and uninstall malware applications installed a user applications.

4. The proposed solution is applicable for only the unrooted Android based mobile devices.

5. The solution architecture also fails to capture the scenarios or events where if the corporate e-mail attachment is accidentally downloaded to the mobile device and thereafter deleted by the employee according to the BYOD policy, but forensically it is possible to extract the document since a simple delete does not actually remove the downloaded document from the device memory, provided it is not overwritten [23]. Therefore even after deleting the downloaded document the employee can possibly retrieve it.

Chapter 5

Conclusion

In this work we formulate a Bring Your Own Device (BYOD) policy for corporate organisations from the corporate data security perspective. We further study the possible security breaches if such a BYOD policy is employed by the corporate organisations for Android based mobile devices. We find and implement methods and techniques to detect some of the possible security breaches which could have a major impact on the corporate data security. We find that our method to detect the root status of the Android based mobile device is successful for all the devices used for testing purpose. The one class classifier using permissions as discriminatory features to detect malicious Android applications has an accuracy of more than 70 %. However the dynamic analysis method to detect malicious Android applications has low accuracy since it fails to capture the interprocess communication in Android. The trigger based method to detect the changes in the incoming and outgoing mail server settings successfully detects all such attempts to tamper with the mail settings of the corporate email client application. We further identify the limitations of our proposed solution approach. Unlike MDM and NAC the methods used to detect security breaches in Android based mobile devices reduce the battery drainage, provide corporate data security and conserve employee privacy by monitoring minimal set of activities on the employee mobile device.

We also propose a BYOD solution architecture which utilizes the methods and techniques to detect the security breaches in Android based mobile devices registered under BYOD policy of the corporate organisation.

Chapter 6

Future Work

The work presented in this thesis report addresses some of the challenges related to detecting security breaches in Android based mobile devices being used in Bring Your Own Device (BYOD) policy for corporate organisations. The future work includes addressing other security challenges like differentiating between corporate and personal mail attachments and documents on the mobile device if the BYOD policy permits the downloading of email attachments on the mobile device. The other extension of this work is the development of the kernel based module which captures the Android interprocess communication along with other system calls. Such a kernel module will be able to detect malware Android applications accurately and precisely along with the information the malware applications are accessing and sending out.

Bibliography

[1] Alliance, O. H. Industry leaders announce open platform for mobile devices. http://www.openhandsetalliance.com/press_110507.html.

[2] Blasing, T., Batyuk, L., Schmidt, A.-D., Camtepe, S. A., and Albayrak, S. An android application sandbox system for suspicious software detection. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference* (2010), IEEE, pp. 55–62.

[3] Brain, M., and Crosby, T. How e-mail works. http://computer.howstuffworks.com/e-mail-messaging/email3.htm.

[4] Burguera, I., Zurutuza, U., and Nadjm-Tehrani, S. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (2011), ACM, pp. 15–26.

[5] Chaudhary, V., and Sureka, A. Contextual feature based one-class classifier approach for detecting video response spam on youtube. In *Privacy, Security and Trust (PST), 2013 11th International Conference* (2013), IEEE.

[6] Developer, A. Application permissions. http://developer.android.com/guide/topics/security/permissions.html.

[7] Di Cerbo, F., Girardello, A., Michahelles, F., and Voronkova, S. Detection of malicious applications on android os. In *Computational Forensics*. Springer, 2011, pp. 138–149.

[8] Digital, C. A. Byod deluge. http://www.cadincweb.com/wp-content/uploads/2012/04/CAD_BRAD_Managing_the_BYOD_Deluge_with_a_BYOD_Blueprint.pdf.

[9] Egners, A., Meyer, U., and Marschollek, B. Messing with android's permission model. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference* (2012), IEEE, pp. 505–514.

[10] Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (2010), pp. 1–6.

[11] Gartner. Gartner predicts by 2017, half of employers will require employees to supply their own device for work purposes. http://www.gartner.com/newsroom/id/2466615.

[12] Gartner. Gartner says asia/pacific led worldwide mobile phone sales to growth in first quarter of 2013. http://www.gartner.com/newsroom/id/2482816.

[13] Gartner. Gartner survey shows byod is top concern for enterprise mobile security. http://www.gartner.com/newsroom/id/2048617.

[14] Hoog, A. *Android software development kit and android debug storage*, 1st ed. Syngress, 2011.

[15] Isohara, T., Takemori, K., and Kubota, A. Kernel-based behavior analysis for android malware detection. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference* (2011), IEEE, pp. 1011–1015.

[16] Jang, W.-J., Cho, S.-W., Lee, H.-W., ill Ju, H., and Kim, J.-N. Rooting attack detection method on the android-based smart phone. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference* (2011), vol. 1, pp. 477–481.

[17] Kaspersky. 99 http://www.kaspersky.com/about/news/virus/2013/99_of_all_mobile_threats_target_Android_devices.

[18] Kefei, C., and Yanglei, C. Design and implementation of network packets collection tools based on the android platform. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference* (2012), IEEE, pp. 2166–2169.

[19] Kim, S., Cho, J. I., Myeong, H. W., and Lee, D. H. A study on static analysis model of mobile application for privacy protection. In *Computer Science and Convergence*. Springer, 2012, pp. 529–540.

[20] Li, J., Gu, D., and Luo, Y. Android malware forensics: Reconstruction of malicious events. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference* (2012), IEEE, pp. 552–558.

[21] Networks, A. Conquering todays bring your own device challenges. http://www.arubanetworks.com/pdf/technology/whitepapers/WP_BYOD.pdf.

[22] Networks, M. Byod best practices - requirements and challenges. http://www.merunetworks.com/collateral/white-papers/2012-wp-byod-implementation-whitepaper-for-wlan-security.pdf.pdf.

[23] Robin Verma, Anuradha Gupta, A. S., and Gupta, G. Forensically important artifacts resulting from usage of cloud client services. http://www.acsac.org/2012/program/case/Gupta.pdf?OPENCONF=5ee217b66d1070adf9361cc65b890df2.

[24] Samsung. Samsung knox. https://www.samsung.com/global/business/business-images/resource/white-paper/2013/06/Samsung_KNOX_whitepaper_June-0.pdf.

[25] Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P., and lvarez, G. Puma: Permission usage to detect malware in android. In *International Joint Conference CISIS12-ICEUTE12-SOCO12 Special Sessions*. Springer Berlin Heidelberg, 2013, pp. 289–298.

[26] Scarfo, A. New security perspectives around byod. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2012 Seventh International Conference* (2012), IEEE, pp. 446–451.

[27] Schreiber, T. Android binder - android interprocess communication. http://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf.

[28] Singh, S. By 2017, companies will expect workers to get their own device to work. http://articles.economictimes.indiatimes.com/2013-05-04/news/39027006_1_byod-own-device-distinguished-analyst.

[29] Synder, J. M. Network access control : A whirlwind tour through the basics. http://www.networkworld.com/podcasts/itr/2008/PM4-NAC-JoelSnyder.pdf.

[30] USA, D. G. Bring your own device. http://www.whitehouse.gov/digitalgov/bring-your-own-device#sample2.

[31] Wei, X., Gomez, L., Neamtiu, I., and Faloutsos, M. Malicious android applications in the enterprise: What do they do and how do we fix it? In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference* (2012), IEEE, pp. 251–254.

[32] Wikipedia. Android (operating system). https://en.wikipedia.org/wiki/Android_%28operating_system%29.

[33] Wikipedia. Android rooting. http://en.wikipedia.org/wiki/Android_rooting.

[34] Wikipedia. Blackberry. https://en.wikipedia.org/wiki/BlackBerry.

[35] Wikipedia. Bring your own device. http://en.wikipedia.org/wiki/Bring_your_own_device.

[36] Wikipedia. Smartphone. http://en.wikipedia.org/wiki/Smartphone.

[37] Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., and Wu, K.-P. Droidmat: Android malware detection through manifest and api calls tracing. In *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference* (2012), pp. 62–69.

[38] Zhao, M., Ge, F., Zhang, T., and Yuan, Z. Antimaldroid: An efficient svm-based malware detection framework for android. In *Information Computing and Applications*. Springer, 2011, pp. 158–166.

[39] Zhou, Y., and Jiang, X. Dissecting android malware: Characterization and evolution. In *Security and Privacy (SP), 2012 IEEE Symposium* (2012), pp. 95–109.

[40] Zhou, Yajin, Z. W. W. Z., and Jiang, X. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *19th Annual Network and Distributed System Security Symposium, 2012* (2012), vol. 1.