Design and Implementation of a Mobile Device Management System Based on SEAndroid

Meng-Chiao Lee[#1], Jin Kwak[*2], Chung-Huang Yang[#3]

[#]*National Kaohsiung Normal University, Kaohsiung, Taiwan*

[1]chiomle@gmail.com

[3]chyang@nknu.edu.tw

[*]*Ajou University, Suwon, Korea*

[2]*jkwak.security@gmail.com*

***Abstract*— Mobile devices store large amounts of personal information and corporate data, therefore, it is very important to protect the private information of smartphones. In this research, we implements a mobile device management (MDM) system for Android devices, it is consisted of an android system app and a server software. The proposed MDM system can manage security of Android devices by means of extended the classes of Android application context, the Security-Enhanced Linux (SELinux) policy analysis and the SELinux policy update on the kernel layer. Our MDM system can remotely grant or revoke app's permissions, also remotely install or remove applications on the devices. The MDM system app was developed using Google's Android Open Source Project (ASOP) and we field trial our design with Google Nexus 7 II tablet with Android 6.**

***Keywords*— SELinux, SEAndroid, Security Policy, Android Security, Mobile Device Management, AOSP**

1. INTRODUCTION

According to the data from eMarketer, by 2016 global smart phone users will reach 2 billions [1]. In 2016 first quarter, mobile device that operated with android system has reach a market share of 84.1% [2]. As those mobile device simplify our lives and become more and more supportive on all sectors, we have become heavily relies on those devices either on private or business. Consequently, it led to a huge increase of malicious attacks to the android systems [3]. Thus the security of the Android system has become a new headlight.

In this research, we proposed Android systems with a more complete security protection for corporate mobile devices [4]. Base on different threats that has raised over the time, this research will first gather and analysis from the past research, We then enhance the security of the android systems and design and implement a Mobile Device Management (MDM) system. Our system includes an Android system application (client-side) and web-based console (server-side). We use Google Nexus 7 II as test device for evaluation.

Mobile Device Management (MDM) is an industry term; it can be used in the enterprise for remotely manage the mobile devices. MDM can be broken down into four types of mobility strategies [4]: HYOD (Here is your own device), CYOD (Choose your own device), BYOD (Bring your own device), and OYOD (On your own device). For HYOD, mobile devices are bought by the enterprise, assign them to employees within the company. HYOD comes with pre-loaded software and setting for mobile devices, highest security level of all. For CYOD, mobile device are bought by business, assign them to employees within the company, but employee can choose use or not. For BYOD, company employee use own device, but managed the device security level by the company. For OYOD, company employee use own device but no security controlled by company at all, and it has the highest risk of all. This research will use HYOD as the strategy and companies have higher control over the security level with pre-install apps and security setting [5].

Our MDM client-side is an Android device. Android is a mobile operating system that is based on Linux kernel, Android is led by Google's Open Handset Alliances (OHA). Google provides the Android Open Source Project (AOSP) [6] where any firm or person can use its open source to customized Android devices. Android's system architecture are show in Fig. 1 [7] where the bottom is Linux kernel, followed by the next layer using C and C++'s

native library, virtual machine, system service administrator, Java API, and finally the user Apps. Since Android 4.3 version, Security Enhanced Linux (SELinux) was introduced. In this research, we revised Android 6.0 source codes, especially the codes related to SELinux, and created a new system application such as the MDM client software.

Server-side is a web-based console in our system. We revised the Android Push Notification (AndroidPN) [8] software. AndroidPN is an open source project which uses JSP to provide push notification support for Android, it can be run on a tomcat server using xmpp based notification server and a client tool kit. We modified the AndroidPN to act as a remote control server, so that system administrators could remotely control the devices.

In the kernel layer of Android devices, researchers had provided in-depth analysis of SELinux policy (*sepolicy*) files [9-10] in the past, but these sepolicy files are fixed when the kernel start and policies could not been changed until power-on restart again. Therefore, our MDM system provides additional security functions, such as sepolicy analysis, SEAndroid deny log collection, and dynamically remote sepolicy update during run time to increase the classes of application context [11-12].

Also, although some researchers had proposed to remotely adjust Android app's permission [13], we further add another MDM function to validate user installed apps by calculating one-way hash value of any user app and compare this value with hash value of known genuine app, so that administrator could ensure the app has not been tampered with or app installed with an older version with security loophole [14]. Administrator also could remove or re-install the user app. In addition, with some circumstances, administrator could remotely adjust the permissions of application to reduce likely security risks.

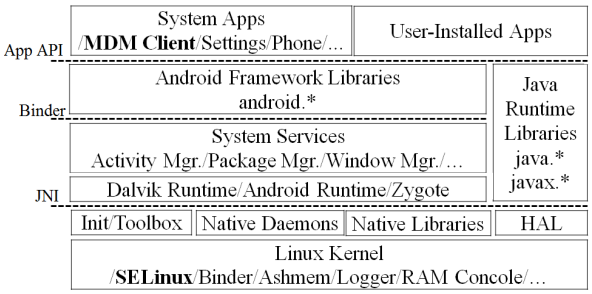

Fig. 1 Android system architecture

2. SELINUX

SELinux was a project of National Security Agency (NSA), and it is a Linux Security Module (LSM). SELinux is a Mandatory Access Control (MAC) mechanism for the linux kernel [15]. SELinux was already installed in several famous linux operating systems (e.g. Ubuntu, Red Hat, and Fedora) (as shown in Fig. 2). Starting with Android 4.3, Google porting SELinux into Android and then Google release Android 6.0 with formally including it in AOSP.

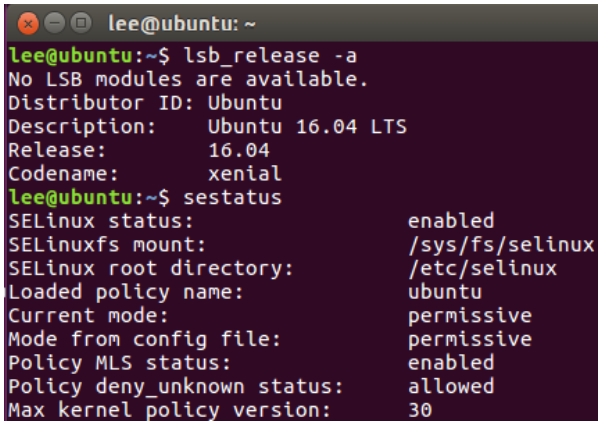

Fig. 2 SELinux on Ubuntu 16.04 LTS

3. SEANDROID

SEAndroid is pored from Linux security module SELinux to Android. The project is managed by the Security Enhancements for Android project; SEAndroid added Mandatory Access Control (MAC) on the default Discretionary Access Control (DAC) mechanism, as illustrated in Fig. 3 [16]. MAC can confine various root exploits and application vulnerabilities [17]. Our research extends original SEAndroid with an online change sepolicy function. Through the default sepolicy stored on the server (sepolicy_X), administrator can control device permission of Camera, GPS, Microphone, or SDcard access, as shown in Fig. 4.

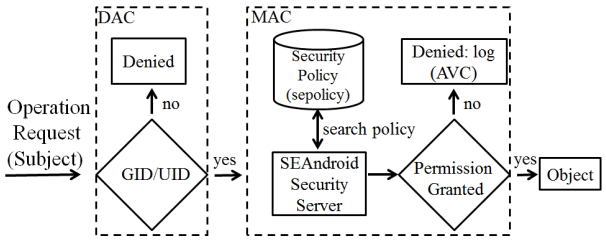

Fig. 3 Simplified view of the SEAndroid

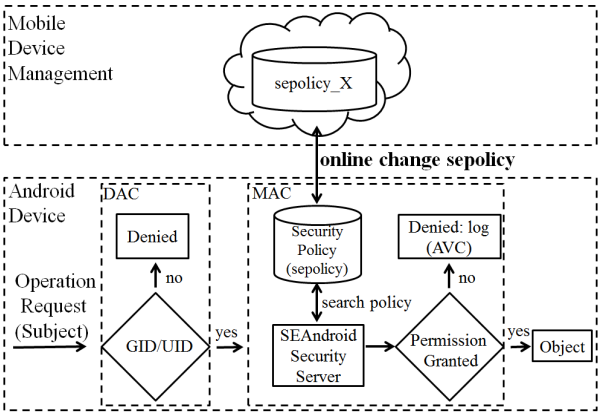

Fig. 4 The proposed modified SEAndroid

The sepolicy file maps user space to system kernel; it used white list alone with a specific programming language. The sepolicy file is a combination of type enforcement (.te) from the source code.

The subject and object are SEAndroid assignation security context (can be also known as security label or label) for every process and file, on android device can viewed by passing the -Z to the "ps" or "ls" command.

SEAndroid kernel policy files are located in [Android source tree]/external/sepolicy. With Android 6.0, this directory contains 98 files, 72 files are "type enforcement" file. In the directory, once it has been made, it will be added into boot.img image. SEAndroid policy files in the Android mobile phone are show in Table 1 [15] where we give locations of all policy files and their functions.

TABLE 1
Android SELinux Policy Files

Policy File	Description
/sepolicy	Binary kernel policy
/file_contexts	Assign file context
/property_contexts	Assign system property context
/seapp_contexts	used to context of app processes and files.
/service_contexts	Assign system server context
/system/etc/security/mac_permission.xml	Maps app signing certificates to seinfo values

4. System Architecture and Development

4.1. System Architecture

This research will produce an Android remote control system. This system will have two parts, MDM Server and MDM Client (Nexus 7 II). Fig. 5 shows architecture of the proposed MDM system. MDM Server will include SEAndroid Management, Permission Management and Application Management; MDM Client will be running customized Android 6.0 and java application. Process between MDM server and MDM client can be breakdown into three items, each of the function are explained as followed:

[SEAndroid Management]: administrator will act according to the SEAndroid denied log (AVC) and analysis sepolicy to understand the policy setting issue. According to different scenario and policy setting, the administrator will send back the new sepolicy and apply to enhance the security protection on the device without rebooting it. Additional, administrator could increase the classes of application context through modify seapp_contexts file.

[Permissions Management]: permission of app could be remotely turned on or off by the administrator so that administrate could protect user's sensitive information which are stored within the device.

[Application Management]: administrate could compare client's app using hash calculation against the pre-calculated hash value stored on the server and administrator could cross check to see if the app's version is correct or not. Furthermore, our system will document the app's information on the remote server to examine for security reason. If administrator have some doubt then he/she can select uninstall or reinstall the app on the mobile device.

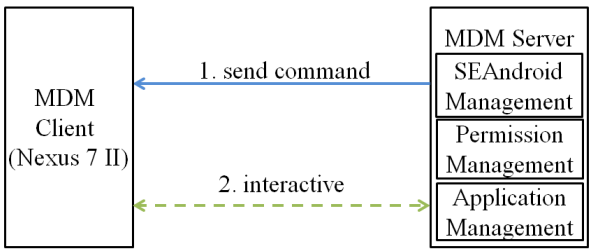

Fig. 5 Architecture of the proposed system

4.2. Development Environment

Our system is developed under the Ubuntu 14.04 LTS operating system. MDM server is based on AndroidPN. The environment of our development and test is shown in Table 2 where MDM client is revised from the AOSP with Android 6.0 source codes. In developing client software, we use Java language, C language and Google Android APIs app development tool. MDM client development and test environment is show in Table 3.

TABLE 2
MDM server development and test environment

Item	Tools
Operation System	Linux (Ubuntu 14.04LTS)
Development Tool	Eclipse
Program Language	Java, HTML, JavaServer Pages, JavaScript
Web Server	Apache tomcat
Database	MySQL

TABLE 3
MDM client development and test environment

Item	Tools
Operation System	Linux (Ubuntu 14.04LTS)
Development Tools	Eclipse, Android Studio, Google Android APIs
Program Language	Java, C
Test Environment	Nexus 7 II (using customized Android 6.0)

On the actual operating, [SEAndroid Management] will update the sepolicy file through editing and execute some commands. According to user install the application, modify seapp_contexts and sent this file to the system default place (android system root directory); normally this will only work once user reboots their device, and however in some circumstances we can forcefully reboot the system to allow the new setting to run in place. In addition, analyzing data has two parts; first, using "dmesg" to collect SEAndriod's AVC from system kernel's log, second, analyse sepolicy with open source code to understand the data. [Permissions Management] using Android's commands to alter user application permission with PackageManager (PM). [Application Management] will be separated into 2 parts, first using hash value calculation. While server-side will use openssl command to calculate administrator approved app with "Secure Hash Algorithm" (SHA-256), at the client-side will conduct MessageDigest to calculate SHA-256 with user's installed app and sent back the calculated file to server for comparison purpose. Second, client-side gathers apps' information and report it back to the server end for later examination purpose.

5. SYSTEM IMPLEMENTATION

5.1. MDM Client

As this research is based on Mobile Device Management (MDM) system with Have You Own Device (HYOD) strategy, administrate has to flash mobile device with images of customized Android 6.0 source code and MDM client application. The MDM client application will provided network information of connection to server (Fig. 6) and log messages (Fig. 7), while server sending the command to MDM client, MDM client will automatically execute the receiving command. The MDM client will show the notification message, shown in Fig. 8 when it received the command.

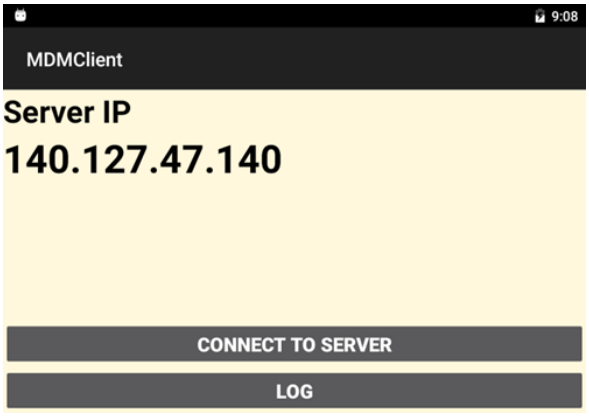

Fig. 6 Interface for the MDM client

Fig. 7 Log message

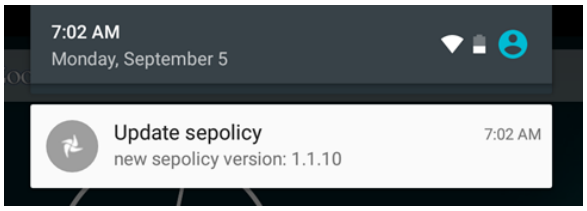

Fig. 8 Notification message

5.2. MDM Server

In this research, "Device Management" screen in the server will show all the devices which are being connected and managed by the server. In the list, it will show which device are being connected, name of the device, connection ID name and device registered date; show in Fig. 9.

Fig. 9 Interface for the MDM server

5.2.1. SEAndroid Management

In "SEAndroid Management" screen on the server, after selecting the device, administer then select sepolicy set "set" button to turn on/off SD card, GPS, Camera or Microphone; shown in Fig. 10 and Fig. 11. Server sends default sepolicy to client. Clicking on the "update" button will trigger sepolicy file to be sent instantly and apply on the device. Selecting "seapp_context update" button will trigger sepolicy file to be send instantly and apply on the device; shown in Fig. 10. Selecting "dmesg log (AVC)" server will get the deny log file which is stored in user's device and server can view it; shown in Fig. 12. Selecting "seinfo" will request device to show the current analysis report of sepolicy file; shown in Fig. 13. Administrator on MDM server can use those two functions (dmesg log and seinfo) to analyze potential problems in sepolicy.

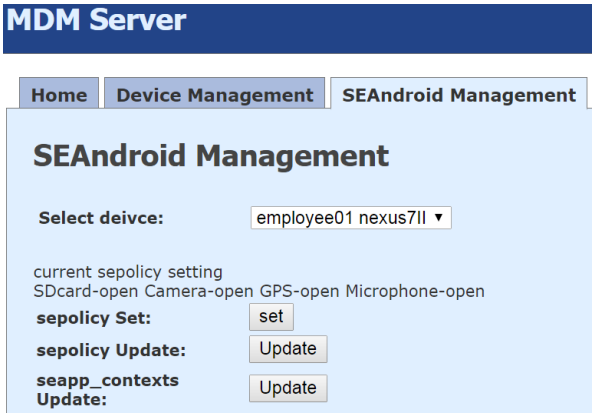

Fig. 10 SEAndroid management screen

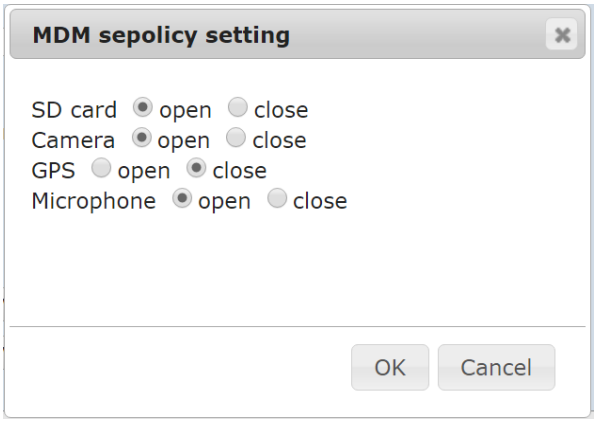

Fig. 11 Select sepolicy screen

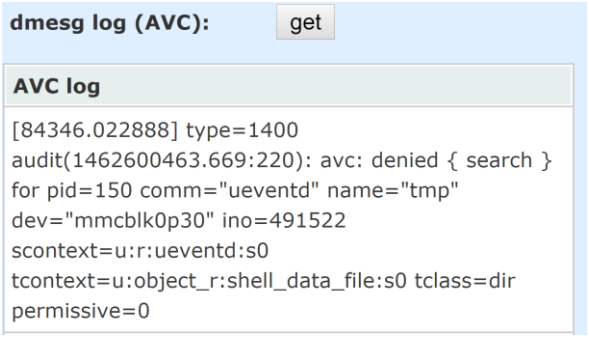

Fig. 12 Interface of the dmesg log

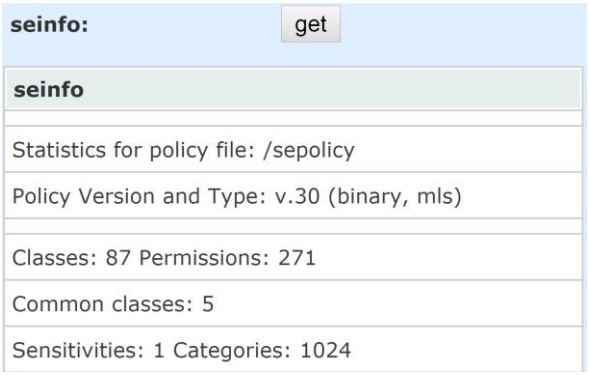

Fig. 13 Interface of the seinfo

5.2.2. Permission Management

"Permissions" screen can manage Android application runtime permission; shown in Fig. 14. Once device is selected, server administrator can use on/off button to turn on/off all (24 items) third party application permission.

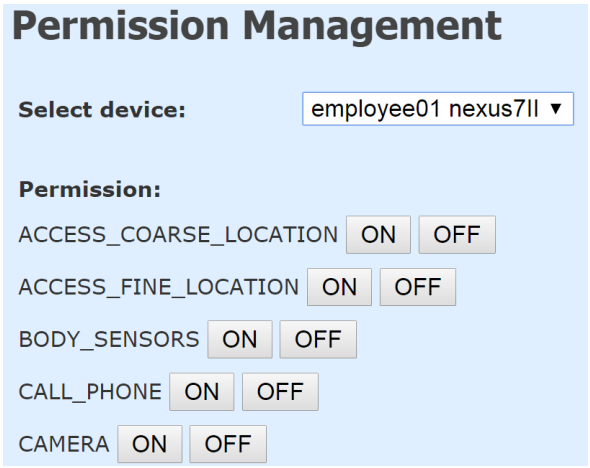

Fig. 14 Permission management screen

5.2.3. Application Management

"Application Management" screen is shown in Fig. 15. Once device is selected, clicking on "calculate" button will perform instant SHA-256 calculation on device's app (APK file) and send the hash value back to server. Once the value has been received by server, the server will compare the value against the MDM server's hash value. If hash value is different, the column will appear in red color. If it's the same, the column will appear in black color. If server cannot find the value to compare with, the column will appear in blue color. Having all the third party applications show on the screen, this allows enterprise administrator to get more information of application (Fig. 16). If the application has potential security threats, administrator can decide whether or not to uninstall or reinstall the user app to eliminate the threats.

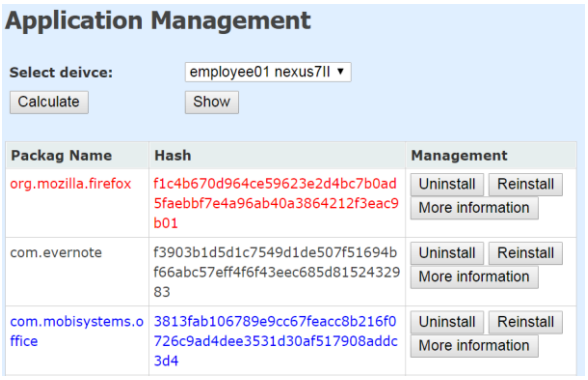

Fig. 15 Application management screen

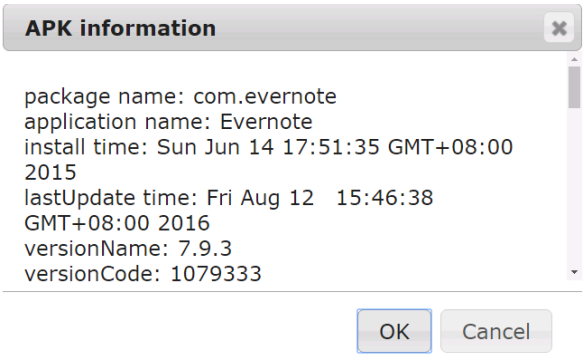

Fig. 16 Interface of the more information of application

5.3. Related Function Comparison

Many researchers had focused on how verification and support various mobile OS in the past [18] and also there are many researchers target how to more accurately verify third party app and it's security feed back [14]. However, few researches are based on the kernel level security issue. It is this research's aim to focus on the kernel security level, therefore, we design and implement an dynamic update SEAndroid file method to improve the device security. We compare existing methods with the proposed approach, as shown in Table 4.

TABLE 4
Comparison of MDM researches

Function \ Related Research	Security Policy based Device Management for Supporting Various Mobile OS [18]	Android Platform Management system [14]	This Research
Application Management	None	Verify and uninstall APK file	Verify, colleting, uninstall and reinstall APK File
Device Management	Camera, Bluetooth, Screenshots, YouTube, Hotspot, Google Play	None	Provide application permission management
Security Reports	None	SCAP Safety Score	None
Kernel level security enhancement	None	None	SEAndroid Function ● sepolicy update (control SDCard, Camera, GPS, and Microphone) ● seapp_contexts update ● collect deny log ● sepolicy analysis
Client installation	Install APK in the Device	Install APK in the Device	Adding third party APK into AOSP then flash the system
Verification	Unique ID, operation system and system version	None	None
Support OS	Android, IOS, Windows Phone	Android	Android

6. CONCLUSION

This research is base on HYOD (Here is your own device) strategy to manage device security. We designed an MDM system to remotely control Android devices and the proposed system can help enterprise administrator to manage Android devices. Our system provides a full support on the SEAndroid (Security Enhanced Android) to lower leakage of sensitive information. We mainly focus on Android's kernel layer to perform real-time analysis on SEAndroid sepolicy, and send reports back to server for further examination of the threats. Based on the feedback information, administrator can improve and send the new sepolicy to the device and have instant effect without device reboot. In addition, based on the application information, the administrator can revise and send the new seapp_contexts file to the device. The administrator can also adjust permission for the app to lower potential threats. The administrator can monitor the safety of the app, if necessary, he/she can uninstall or re-install the app remotely to ensure the minimum amount of bugs exist within the device. In the future, one can further deep research on the SEAndroid so that it is possible to provide more MDM system functions such as refining or automatic produce sepolicy rules.

ACKNOWLEDGEMENT

This work was supported by the Ajou University research fund.

REFERENCES

[1] eMarketer. (2014) 2 Billion Consumers Worldwide to Get Smart(phones) by 2016. [Online]. Available: http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694

[2] Gartner. (2016) Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3215217

[3] P. Faruki, A. Bharmal, V. Laxmi, M. S. Gaur, M. Conti, and M. Rajarajan, "Android Security: A Survey of Issues, Malware Penetration and Defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, pp. 998-1022, Dec. 2015.

[4] P. K. Gajar, A. Ghosh, and S. Rai, “Bring your own device (BYOD): security risks and mitigating strategies,” *Journal of Global Research in Computer Science*., Vol. 4, pp. 62-70, 2013.

[5] C. Yin, L. Liu, and L. Liu, “BYOD implementation: understanding organizational performance through a gift perspective,” in *Proc. PACIS 2014*, 2014.

[6] (2017) The Android Source Code website. [Online]. Available: https://source.android.com/source/index.html

[7] K. Yaghmour, *Embedded Android*, O'Reilly Media, 2013.

[8] SourceForge.net. (2014) Android Push Notification. [Online]. Available: https://sourceforge.net/p/androidpn/

[9] E. Reshetova, F. Bonazzi, T. Nyman, R. Borgaonkar, and N. Asokan, “Characterizing SEAndroid Policies in the Wild,” in *Proc. 2nd International Conference on Information Systems Security and Privacy*, pp. 482-489, 2015.

[10] xda developers.com. (2016) setools-android with sepolicy-inject. [Online]. Available: http://forum.xda-developers.com/android/software/setools-android-sepolicy-inject-t2977563

[11] E. Bacis, S. Mutti, and S. Paraboschi, “App PolicyModules - Mandatory Access Control for Third-Party Apps,” in *Proc. of the 10th ACM Symposium on Information, Computer and Communications Security*, 2015.

[12] U. Kanonov and A. Wool, “Secure Containers in Android - the Samsung KNOX Case Study,” *The Academic Perspective on Cybersecurity Challenges*, 2016.

[13] L. M. Tung and C. H. Yang, “A Remote Control System for Improving the Mobile Device Security Based on SE Android,” in *Proc. International Conference on Advanced Information Technologies*, Taiwan, 2016.

[14] S. C. Chang and C. H. Yang, “Design and implementation of a mobile device management system on android platform,” in *Proc. The E-Learning and Information Technology Symposium*, Taiwan, 2016.

[15] N. Elenkov, *Android Security Internals: An In-Depth Guide to Android's Security Architecture*, No Starch Press, Inc., 2014.

[16] (2013) SELinux Wiki website. [Online]. Available: http://selinuxproject.org/

[17] S. Smalley, and R. Craig, “Security Enhanced (SE) Android: Bringing Flexible MAC to Android,” in *Proc. of the 20th Network and Distributed System Security Symposium (NDSS 2013)*, 2013.

[18] J. E. Lee, S. H. Park, and H. Yoon, “Security Policy based Device Management for Supporting Various Mobile OS,” in *Proc. 2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*, pp. 156-161, 2015.