A Remote Control System for Improving the Mobile Device Security Based on SE Android

Lun-Ming Tung[1], Chung-Huang Yang[2]
Department of Software Engineering and Management,
National Kaohsiung Normal University
No.116, Heping 1st Rd., Lingya Dist., Kaohsiung City 802, Taiwan (R.O.C.)
[1]k49918135@gmail.com
[2]chyang@nknu.edu.tw

***Abstract*— Mobile devices are becoming more authoritative and developers of the device are continuously striving to combine the device with human life for the enhancement of user intention. On the other hand, mobile device system security also needs to upgrade in order to protect the privacy of users' private messages. The purpose of this paper is to use mobile device permission management systems to protect user privacy information from malicious attacks. We introduce the Android Open Source Project (AOSP), extensible security framework "SE Android". Google and National Security Agency (NSA) developed it. SE Android provides a security application that supports Android systems of permission management. We modify the application combined with a remote server to form remote control software that provides additional support features for Mobile Device Management (MDM).**

***Keywords*— Mobile Device Management, Permission Management, SE Linux, SE Android, Security System**

1. INTRODUCTION

With the features of mobile devices becoming more popular, users usually save private data to mobile devices. The security of mobile devices has raised great attention to enterprises and academia. According to the report from IDC (International Data Corporation), the global smart phone shipments totaled 334.4 million units in 2014 mobile devices (e.g. [1]). It represented 288.3 million units in the first quarter that grew 16% and Android smart phone system occupied 78% in 2015. According to market research firm survey reports pointed out that from 2012 begin global system mobile Android has been leading IOS. As of 2015, the report pointed out that global market share of mobile devices, Android accounted grew from 31% to 54% and IOS accounted grew from 10% to 18% since 2012(e.g. [2]). As the data mentioned above, the mobile devices become more popular and Android devices occupied the largest market share. However, with security concerns, according to international research and consulting firm (e.g. [3]), the investigation report indicates that mobile devices often contain sensitive information and the user should be protected.

In the past research, which was based on the solution of application abuse action device sensitive information, the Gilbert et al. (e.g. [4]) found that the action device of third party application software (third-party apps) may abuse privacy information of user or use their information inadequately. Therefore, to propose automation security validation system: App Inspector, the system will do analyses on device of application and produce reports of potential security and privacy behavior. And Yang et al. (e.g. [5]) found that the action device of malicious software tries to through imitation provides general application software security to take with privacy or sensitive message of behavior (for example: sent newsletter) to escape application software security analysis. It also can control malicious behaviour occurred in certain moments (for example: malicious behaviour only occurred in night). Due to general analysis in malicious software of application cannot operate 24 hours. Hence, the research recommends using the application Context to analyze malicious behavior in mobile device software. As two previous studies provided, most of current data, analyze the behavior of malicious software (applications using mobile device permission to address mobile devices privacy data protection.) However, it needs to be parsed before the users respond in a timely manner. Although there already has a high opportunity to identify the malicious acts by using specific methods, it is not comprehensive. In the same way, this research

also supports to solve the problem of malicious software that made by some permission to obtain user privacy information. This research dismisses the idea of malicious software analysis and tries to directly provide users with the tactic of management. Therefore, the user can control the individual software of permission or all the software in action device. This strategy can ensure the security of user's privacy information. Lastly, this research combines remote control and forms the similar MDM (e.g. [6], [7]) of management mode to provide data of management within mobile devices and reference of protecting important private information for society.

Based on the reasons mentioned above, we develop Android device nexus 7. Used by the United States National Security Council (the National Security Agency; NSA) of SE on the SE Linux with Google offering the AOSP Android open source. This application can operate an application of "seadmin" and isolated the permission of the application. And then AppOps Eclipse can add a key to manage application by specific permissions on all devices. This function combines with a modified AndroRAT program to achieve the effect of remote control. And using SSL Socket mechanism can improve the safety connection problem. We can, through the permission management application, help protect the user's privacy.

2. LITERATURE REVIEW

This study in order to realize the method which needs to pick up the SE Android system for remote access management device, and however to achieve SE Android is Linux kernel Android system replacement for the SELinux kernel. By adding modify SELinux-enabled system components, the applications can obtain API to manage devices. The application system must be used, so TWRP open recovery Mode can install applications on the device and located in the list of system/Priv-app. We can use the device when the system is overloaded and consider the security of the network. The information using SSL Socket mechanism makes delivery channel encryption. As the result indicated, we hope the remote access management can provide MDM for developers to use. Thus, this research describes SELinux, AOSP, SE Android, SSL Socket, MDM and TWRP.

2.1. SE Linux

SE Linux (e.g. [8], [9], [10]) mainly developed by the United States National Security Agency (the National Security Agency; NSA), SE Linux operating structure as in Fig. 1. It currently integrates to the core part of the Linux version, and it provides the function of mandatory access control (Mandatory Access Control; MAC) to improve the General Linux which using the discretionary access control (Discretionary Access Control; DAC). The DAC access control sets by the "rwx" decision. Based on it, the corporate world considers that the most of the problems came from the file setting, reading, writing and executing permissions. If the system administrator has not set the perfect, it will lead the resource leaked. MAC access control needs to derive the file thread source (domain) and file domain and drawn with a policy file than for permission. In Fig. 2, to take the assumption that subjects with an object as an application. Their implementation status of the domain root and source for ABC and policy define objects that can be accessed by domain: ABC domain only xyz. Obtaining object status in DAC mode applications as root, so that all objects can be achieved, however MAC application identity domain defined for root only in the light of policy made for object a XYZ, others are not made.

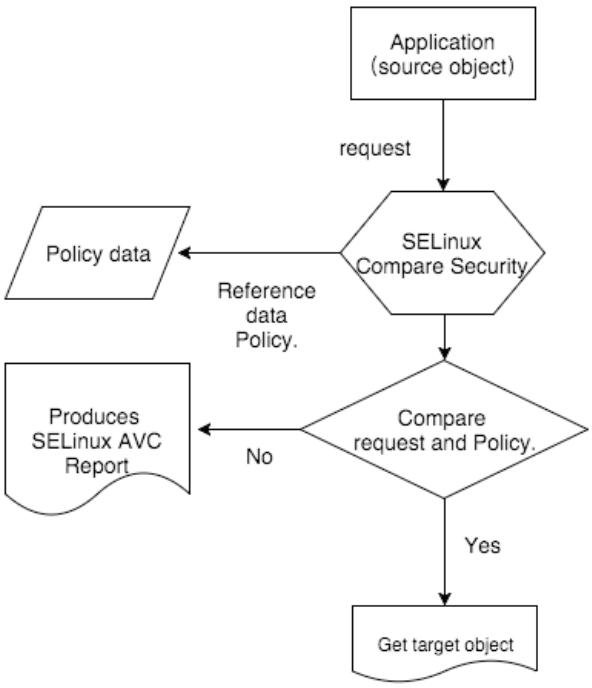

Fig. 1 SE Linux system architecture

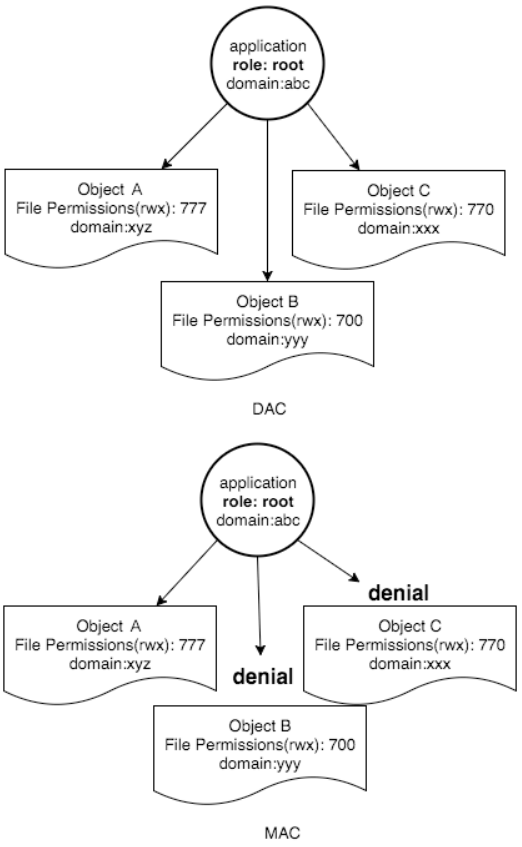

Fig. 2 Compare MAC and DAC

2.2. Android Open Source Project

AOSP (e.g. [11]) is a Google Android development and it has a pipeline to upgrade continually, but it should still judge the effectiveness of the system according to the hardware basis. The AOSP, as representatives of native device with Nexus as a standard point. Most OEMs (Original Equipment Manufacturer; OEM) companies through the action of AOSP developed their own system, such as SONY, HTC, ASUS and Samsung. And other companies in order to enhance the effectiveness or safety of Android system by modifying the AOSP available to others to download and compile a ROM on a network. For example: CyanogenMod.

2.3. SE Android

Starting from Android 4.3 currently supports SE Android (e.g. [12]) system functions. SE Android is the core system from the original Linux kernel changes to SELinux core, which makes the Android system multiple SELinux security mechanisms such as a mandatory access mechanism. The biggest feature of SE Android is that applications can use capacity as user or root access to the system resources. Via the SELinux security policy development, we can set a filter action. If the action cannot go through the filter, it will be denied access. In Fig. 3, the Android system architecture diagram (e.g. [11]) in bold to show what is SE Android modifications, such as SEAdmin were added in system application and added support in the Application Settings set SE Android project. Furthermore, in the API and also added support for SE Android JNI methods and instruction, and will modify the SELinux kernel Linux.

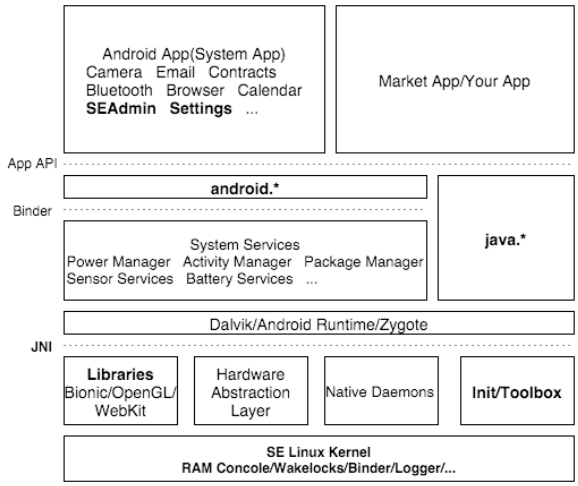

Fig. 3 SE Android system architecture

Table 1 below shows SE Android AOSP source code as an example to modify the content, such as in the AOSP/external/directory, added modify libselinux, libsepol, checkPolicy and sepolicy four new policies, AOSP/packages/apps /directory SEAdmin and two Application Settings to modify, AOSP/bionic/, and AOSP/ List of bootable/recovery and AOSP/build/ of the three modifications to the system running the Foundation, and finally AOSP/frameworks/base/ and AOSP/system/directory of core/and extras/ of the three modified Android system.

TABLE 1
SE ANDROID MODIFICATION

AOSP directory	Modification
external/libselinux	Provides device SE Linux userspace can use resource library. Details can refer to: external/libselinux/README.android
external/libsepol	Provide policy documents, building use policy for userspace.
external/checkpolicy	Provide policy build tool.
external/sepolicy	Provide Policy files for the Android kernel component (*.TE), established by the

	Android.mk and install to device the result.
packages/apps/ SEAdmin & Settings	SEAdmin：An Android application for managing SE Android system (such as loading a new Policy files and manage application permissions). Just support SE Android. Settings：This is application of setting device, it added SELinux configuration on the SE Android device (such as device encryption).
Bionic	Android 'libc' is obtained from the BSD c standard library. On SE Android include SELinux security enhancements.
bootable/recovery	Modify the recovery mode of the content, including recovery init.rc file in the etc directory.
Build	Modify SE Android system's OTA update files.
frameworks/base	1. The JNI added support for SELinux letter shows. Such as: isSELinuxEnabled and setFSCreateCon. 2. SELinux Java class and method definitions. 3. Check the Zygote connection contexts. 4. Wallpaper services and package manager to manage file permissions. 5. Supplement SELinux to support run time additional MMAC[1] and SE Android service.
system/core	1. Toolbox adds SELinux service.(For example: load_policy, runcon) 2. System initialization adds SELinux support. 3. Add SELinux AVC function to access specific object return information document.
system/extras	SELinux support ext4 file system format.

2.4. SE Android Outside of Literature

2.4.1. SSL Socket

SSLSocket is Socket extensions. With the basis on the socket and adds a layer of security protection, we provide a higher level of security, including authentication, data encryption and integrity verification. The authentication is by using digital certificating issuance and using. Encryption prevents the data in the message transfer process being monitored as a result of the loss, even though third parties listening to messages passed. However, without the correct key, it is still unable to get the right message integrity validation to prevent the message from being modified in transit.

2.4.2. TWRP OpenRecovery

TWRP is a custom recovery mod and open source projects can refer to the Web site for details: https://twrp.me/FAQ/. Initially, it developed by four people, and then via a pattern similar to the developer forums, debugs, update. Many people have expressed their views evolved. Custom recovery mode allows developers to install custom applications; you can even install the complete ROM update or modify the operating system Android devices.

2.4.3. Mobile Device Management

Mobile device management (MDM) is in the administrative contents of the company and it can achieve the functions of development, protection, monitoring, integration and management of mobile devices. For instance, smart phones, tablet computers and laptops. MDM's intention was to improve the internal security of mobile devices, while protecting the corporate network security.

3. SYSTEM ARCHITECTURE AND DEVELOPMENT

3.1. System Architecture

The system consists of applications Client-side which is installed on the mobile device and servers which can remotely control your computer, as shown in Figure 4. Client-side notes on setting permissions for mobile device management and connection. The user can realize the part of device on each of the permissions required by the application. Also, the user can manage the requirements of the specific application permissions or other model to manage all the application-specific permission name permissions. The user can set the connecting information in Server to achieve the functions of the remote control. The remote control in Server-side interface provides four operations and it illustrates four pictures respectively. The first one "information control interface" it can display information on mobile devices such as: version Wifi status, network

[1] Middleware MAC to make SE Android system can be simple updates to comply with policies of the MAC mechanism, simplifying the policy syntax (*.TE) complied with policy syntax

status, 3G/4g system and battery status, and also can send warning messages to the device and vibration. The second "obtain address control interface" which can be used by device's GPS chip or network installation location and use simple maps for display. The third "make directory control interface" it can obtain a mobile device's file system directory structure and download files on the device. The final "application access control interface" it is a particular permission which can remotely manage mobile devices, so all applications on the device could not be taken to ensure the privacy of data leakage. In the past, research suggests that mobile devices malicious acts of third party applications may access the user's private data (e.g. [13]), and however, applications need to access the data on the devices. The user must request permission, and therefore this system using administrative rights to privacy information protection on the device. The system can ensure that it is restricting the permissions required by the application and could not be used.

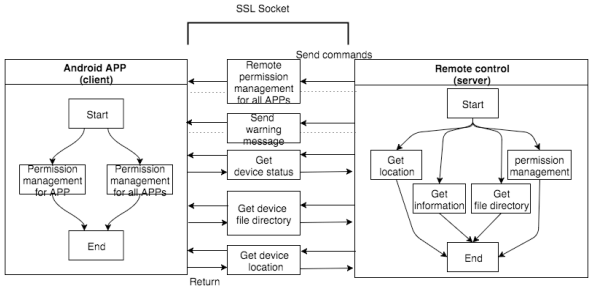

Fig. 4 Architecture of remote permission system

3.2. System Development

As shown in table 2, the system consists of cross-platform programming language Java design and third party open source remote control software for Android AndroRAT changes. Due to it is developed by Java, so as long as the server platform supporting installation of Java computing environment can be run. Client is in Eclipse using Java on Android development tool (ADT) and Google Android APIs developed by. SSLSocket and API with AppOpsManager is most important.

First AppOpsManager contains all of the Android device access[2] control methods, since AppOpsManager only support SE Android system environment and only provides system procedures should be used. The system must run on the Android 4.3 future releases and use of third-party open source backup restore recovery system software TWRP. The client package to support Recovery OTA Update Mode file format (*.zip) to install the application to the system directory. In order to run without modifying the original system TWRP, this study used fastboot command allows TWRP mirror SDK Toolkit document which can be used by the external open storage area on the appliance. However the above process must be unlocked Bootloader, so it is restrictions on this research.

The second important reason in SSLSocket API is that it is for original remote control application open source code in transfer message about security. It will originally only using General transmission agreement Socket modified for additional Shang SSl security agreement of SSlSocket. Also, it will make this research in transfer message must contain a key certification. It can recognize Client and Server end of identity to reach prevention network transmission may occur of intermediaries attack, as Fig5. In addition, the key can then be used by Ubuntu System preload tool: Keytool length defaults to 1024-bit keys, if you want to change the key length can be set by means of directives.

TABLE 2
SYSTEM DEVELOPMENT AND TESTING ENVIRONMENT

Server system environment	Windows and Ubuntu and other operating system that supports Java environment.
Development tool	Java SE Development Kit(JDK)8 Eclipse LUNA Google Android APIs Android Developer Tools
Program languages	Java C languages
Test environment	ASUS Nexus 7 (Android 4.4.4) ASUS Nexus 7 II (Android 5.1.1) LG Nexus5 (Android 5.1.1)
Execution environment	Support Android version after 4.3 system and Bootloader unlocked devices.

[2] Permissions instructions can refer to the following Web site: *http://developer.android.com/reference/android/Manifest.permission.html*

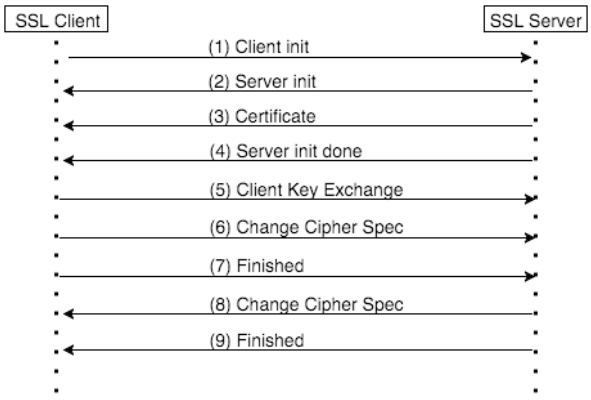

Fig. 5 SSL Socket authentication architecture

(1) SSL client-side supports SSL version, encryption algorithms and other information send SSL server.

(2) SSL server determines the version of SSL and encryption suite and then reply messages to the SSL client-side.

(3) SSL Server using their certificate information to make their public key to sending the SSL client-side.

(4) SSL server sends an initial complete message notification SSL client-side version and encryption Suite consultations have been completed and started a key exchange.

(5) When the SSL client-side verify the SSL server certificate is legitimate, the system will use the server's certificate of public key encryption in SSL client randomly generated Premaster Secret (this is a symmetric encryption key in a 46-byte random number), and the message will be sent to the SSL server.

(6) SSL client-side sends a notification of subsequent transmission will negotiate the SSL server key and encrypt the encryption suite.

(7) SSL client calculates Hash values for the interactive handshake message, using agreed key and encryption algorithm, Hash value, and sends the 'Finished' message to the SSL server. SSL server uses the same method to calculate the interactions of handshake message's Hash value, and compare with the finished decryption of messages. If they are the same, the proof key and encryption suite negotiation will be successful.

(8) SSL server sends a notification of subsequent transmission will negotiate the SSL client key and encrypt the encryption suite.

(9) SSL calculates the Hash value of handshake message and uses agreed key and encryption suite to cope with Hash values. It will send the 'Finished' message to SSL client. SSL client use the same method to calculate Hash values for interactive handshake messages, and compare with the Finished decryption of messages. If they are same, MAC authentication will be successful. The proof key and encryption suite negotiation also will be successful. After SSL client receives the SSL server message, the decryption will be successful, and you will be regarded as the owner of the digital certificate SSL server. The SSL server authentication succeeds. Because this is the only way of SSL server private key to be decrypted from the Client Key Exchange message. Premaster Secret, thus indirectly achieving the SSL client authentication to SSL servers.

4. SYSTEM TESTING AND RESULTS

As the system with a simple GUI and simple method of operation developed by this study, it can provide quick management permissions for device applications in APP Client. It also can be done on the Server side of remote online monitoring and management information and permissions on the device. When users on the Android device finish installing the APP OTA, they can begin using this system. The process refers to fig. 6. In addition, because Android 6.0 has been released on October 5 to Nexus 5, 6, 7, 9, and Player equipment updates and new rights management features in the application of the system. As shown in table 3, the difference between this study and the new Android system settings application and research contributions.

TABLE 3
DIFFERENCE OF RESEARCH AND ANDROID M'S PERMISSION APPLICATION.

Feature	Android M's application	Research
Permission management for APP	Yes	Yes
Permission management for all APPs	No	Yes
Mobile Device Management	No	Yes Part support: such as find device, send warning messages and get devices information etc.
Remote permission management	No	Yes

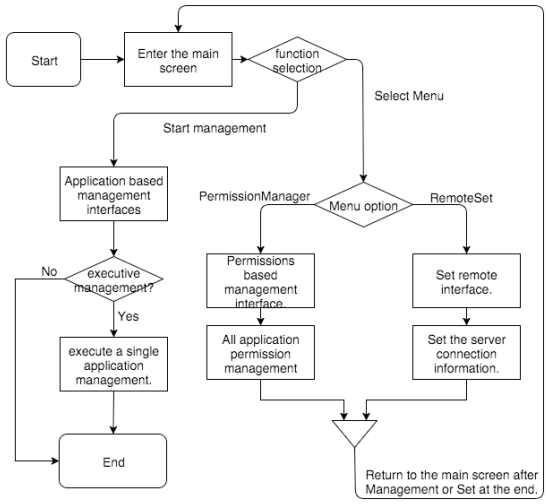

Fig. 6 System flow chart

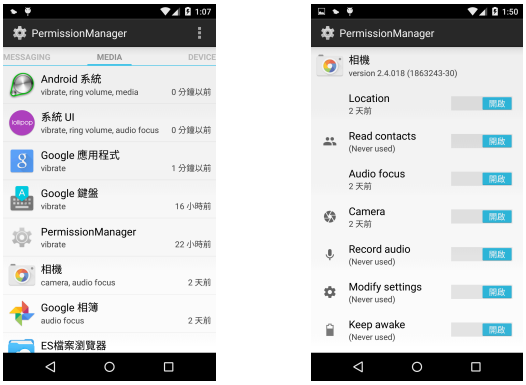

Fig. 7 Interface of single application permission

When APP starts executing, the execution screen will appear, as shown in Fig. 6. At this point the user can slide the permissions page, and select the drawn category, which provide LOCATION, PERSONAL, MESSAGING, MEDIA and DEVICE. The user can select one of the applications to manage the permissions as shown in Fig. 7.

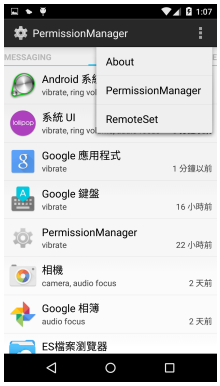

Fig. 8 Single application permission management interface

If users want to switch modes, you can click on the main picture MENU key on the right as shown in Fig. 8, the main option for both PermmissionManager and RemoteSet.

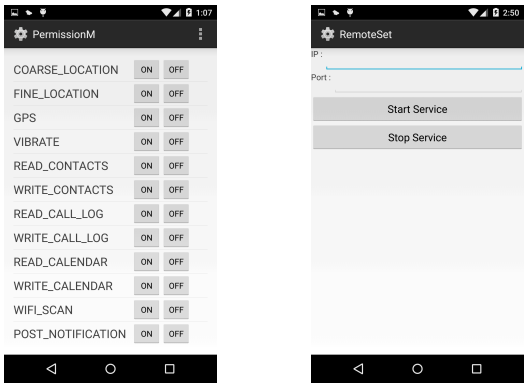

Fig. 9 Permissions management for all APPs and remote connection set.

When the user selects "PermissionManager" option, you can enter a screen to manage all applications. When a user selects "RemoteSet" option, it can be set with the remote server's online IP and Port as shown in Fig. 9. When the online success you can remotely manage mobile devices, start with Server-side image.

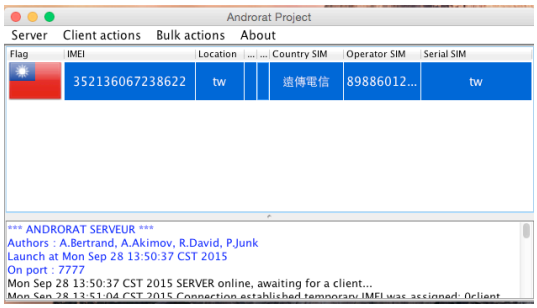

Fig. 10 Remote Server select connection device.

When users finish online setting on the App side, you can see the Server side of online device, as well as the basic information as shown in Fig. 10. This time clicking the device icon into the management of the information screen and managing screen will contain the Home, Permission Manager, Map and the File tree. Then each of the following screens introducing as follows.

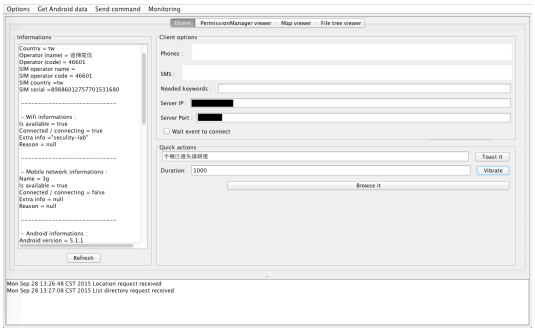

Fig. 11 Device management screen

First, when you enter the device management screen, you will see the Home screen as shown in Fig. 11, where the device appears details such as the Android system information, network information, and equipment information. Here can send warning messages to the device remotely, as well as a shock when the appliance is lost can be found device to make warning message.

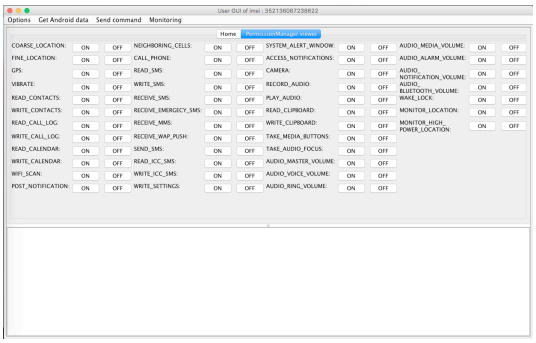

Fig. 12 Application access management screen

Second, the screen Permission Manager shows 43 mobile device application permissions. They are displayed as shown in Fig. 12. And provide managers with online all application-specific permissions on the device control. However, this system only provides permission for the main rights management for all applications on the device. Because the Server-side of the system can be connected to multiple devices and applications on the device will be unknown. They actually are for a single application of rights management difficulties and might cause managers to control permissions. Hence the system does not implement this feature on the Server side.

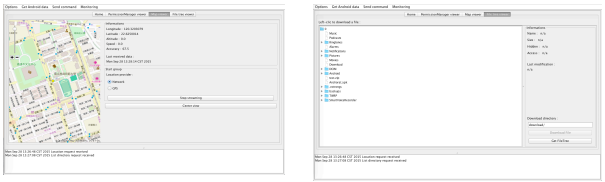

Fig. 13 Application access management screen

The third and fourth screens extend by the original picture in AndroRAT, which as shown in Figure 13. Both functions are described as below, the Map get device address information made by network or GPS, and the picture appears to the left of the map to facilitate managers' monitoring control devices. File tree can get an online installation files directory, and download the file specified to set the path of the address. In other words, it enters on the lower right of the screen text input box the path address and based on the Manager of the system to protect internal data control devices are not being maliciously modified and designed to increase operational safety.

5. CONCLUSIONS

There are two contributions in this study. First and foremost with mobile device management (MDM) based control Google Android API: AppOpsManager. The permissions provided by the project, the development of a remote monitoring and management system, which may limit a third party application. It might have on private data on the device theft or improper use. Second, by modifying the AndroRAT source code, the system can transfer the message protocol and so that it can ensure the safety message to improve the operational security of mobile devices. At present, the system has been applied in the United States National Security Agency (NSA) recommendations and development of a SE Android operating system. The existing system is mainly controlled by remote access to spindle development. Compared to the mobile device management system on the market, this is a new feature, but for complete protection, monitoring and integration, improvement is still needed. This research towards protection of leakage of private data and ensure that it will not be lost on the device in the future.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council of Taiwan (NSC 102-2221-E-017-003-MY3).

REFERENCES

[1] IDC. (Aug 2015) Smartphone OS Market Share, 2015 Q2. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] GlobalWebIndex. (Jun 2015) Android mobile now has huge lead over iOS. [Online]. Available: http://www.globalwebindex.net/blog/android-mobile-now-has-huge-lead-over-ios

[3] Gartner. (February 2015) Mobile Device Security: A Comparison of Platforms. [Online]. Available: https://www.gartner.com/doc/2988420

[4] P. Gilbert, B. G. Chun, L. P. Cox, and J. Jung, "Automated Security Validation of Mobile Apps at App Markets," MobiSys, pp 21-26, 2011.

[5] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, "AppContext: Differentiating Malicious and Benign Mobile App Behaviors Using Context," Technical Research: Security and Privacy(1:3), pp. 303-313, 2015.

[6] K. Rhee, W. Jeon, and D. Won, "Security Requirements of a Mobile Device Management System," International Journal of Security and Its Applications(6:2), pp. 353-358, 2012.

[7] L. Liu, R. Moulic, and D. Shea, "Cloud Service Portal for Mobile Device Management," in e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on, pp. 474-478, 2010.

[8] S. Vermeulen, SELinux System Administration, Packt, Mumbai, 2013.

[9] R. Haines, The SELinux Notebook(4th Edition), 2014.

[10] National Security Agency. (Jan 2009) SELinux Frequently Asked Questions (FAQ). [Online]. Available: https://www.nsa.gov/research/selinux/faqs.shtml

[11] k. Yaghmour, Embedded Android, O'Reilly Media, Sebastopol, 2013.

[12] S. Smalley, and R. Craig, " Security Enhanced (SE) Android: Bringing Flexible MAC to Android," Trusted Systems Research, pp. 20-38, 2013.

[13] M. Backes, S. Bugiel, S. Gerling, and P. von Styp-Rekowsky, "Android Security Framework: extensible multi-layered access control on Android," In Proceedings of ACSAC, pp. 46-55, 2014.

[14] E. Bacis, S. Mutti, and S. Paraboschi, "AppPolicyModules: Mandatory Access Control for Third-Party Apps," In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, pp. 309-320, 2015.

[15] C. Sanders, A. Shah, and S. Zhang, "Comprehensive Analysis of the Android Google Play's Auto-update Policy," Springer International Publishing, pp. 365-377, 2015.

[16] S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna, "Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications," in Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS), pp. 23-26, 2014.

[17] S. Bugiel, S. Heuser, and A. R. Sadegh, "Flexible and fine-grained mandatory access control on android for diverse security and privacy policies," 22nd USENIX Security Symposium, pp. 131-146, 2013.

[18] A. Shabtai, Y. Fledel, and Y. Elovici, "Securing Android-Powered Mobile Devices Using SELinux," IEEE Security & Privacy (8:3), pp. 36-44, 2010.

[19] J. J. Drake, Android Hacker's Handbook, Sons, Indiana, 2014.

[20] MediaWiki. (March 2015) NB SEforAndroid 1. [Online]. Available: http://selinuxproject.org/page/NB_SEforAndroid_1