International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
www.ioirp.com ISSN: 2394-6364, Volume – 1, Issue – 6. May 2015

ENABLING THE EFFICIENT INTEGRITY PROTECTION TO THE OPEN MOBILE PLATFORMS

ELANGOVAN K
PG Student (M.E-Mobile and Pervasive Computing)
Department of Computer Science and Engineering
University College of Engineering, Anna University BIT Campus Trichy, India
elangovan.mee@gmail.com

***Abstract*— The security of mobile devices such as cellular phones and smart phones has gained extensive attention due to their increasing usage in people's daily life. The problem is challenging as the computing environments of these devices have become more open and general-purpose while at the same time they have the constraints of performance and user experience. The propose and implement SEIP, a simple and efficient but yet effective solution for the integrity protection of real-world cellular phone platforms, which is motivated by the disadvantages of applying traditional integrity models on these performance and user experience constrained devices. The major security objective of SEIP is to protect trusted services and resources (e.g., those belonging to cellular service providers and device manufacturers) from third-party code. The propose a set of simple integrity protection rules based upon open mobile operating system environments and application behaviors. This design leverages the unique features of mobile devices, such as service convergence and limited permissions of user installed applications, and easily identifies the borderline between trusted and untrusted domains on mobile platforms. The approach, thus, significantly simplifies policy specifications while still achieves a high assurance of platform integrity. SEIP is deployed within a commercially available Linux-based smart phone and demonstrates that it can effectively prevent certain malware. The security policy of this implementation is less than 20 kB, and a performance study shows that it is lightweight.**

I. INTRODUCTION

This demands then that the solution must be simple but general enough so that most users can rely on default configurations—even after new application installations. On the other side, existing exploits in mobile phones have shown that user downloaded and installed applications are major threats to mobile services. According to F-secure, by the end of 2007, more than 370 different malware have been detected on various cell phones, including viruses, Trojans, and spyware. Most existing infections are due to user downloaded applications, such as Dampig, Fontal, Locknut, and Skulls. Other major infection mechanisms include Bluetooth and multimedia message service (MMS), such as Cabir, CommWarrior, and Mabir. Many exploits compromise the integrity of a mobile platform by maliciously modifying data or code on the device. PandaLab reports the same trends in 2008. Based on the observation that user downloaded applications are the major security threats, one objective of securing mobile terminal should be confining the influence of user installed applications.

This objective requires restricting the permissions of applications to access sensitive resources and functions of mobile operating system (OS), customers, device manufacturers, and remote service providers, thus maintaining high integrity of a mobile device, which usually indicates its expected behavior.

II. RELATED WORK

Considering the increasing attacks through Bluetooth and MMS interfaces, an effective integrity protection should confine the interactions between any code or data received from these communication interfaces and system parts. Toward simple, efficient, and yet effective solution, this propose SEIP, a mandatory access control (MAC)-based integrity protection mechanism of mobile phone terminals. This mechanism is based upon information flow control between trusted (e.g., customer, device manufacturer, and service providers) and untrusted (e.g., user downloaded through browser or received through Bluetooth and MMS) domains. By confining untrusted applications' write operations to trusted domains, the solution effectively maintains runtime integrity status of a device. To achieve the design objectives of simplicity and efficiency, several challenges exist. First, we need to efficiently identify the interfaces between trusted and untrusted domains and thus simplify required policy specification. For example, in many SE Linux systems for desktop and servers, very fine grained policy rules are defined to confine process permissions based on a least privilege principle.

However, there is no clear integrity model behind these policies, and it is difficult to have the assurance or to verify if a system is running in a good integrity state. This analyzes integrity threats on mobile platforms based on infection and compromising mechanisms. The identify salient requirements for security solutions on mobile terminals and propose the strategies toward these requirements. Based on different functional behaviors of subjects in mobile systems, the propose a set of integrity protection rules to control their interactions with others. The solution focuses on the protection of phone and platform management services from untrusted applications such as those downloaded by users or received from Bluetooth/MMS. The present this design and implementation of our solution in a real-world Linux-based mobile phone device, and leverage SE Linux to define a respective policy. The policy size is less than 20 kB in binary form and requires less than 10 domains and types. The demonstrate that this implementation can prevent major types of attacks toward system integrity through mobile malware. The discuss potential integrity assets and threats to mobile platform integrity.

Details of this design and integrity rules are described in, and implementation and evaluation, respectively. The discuss some limitations of this solution in. Related work on integrity model and mobile platform security are presented.

III. DESIGN METHOD

A. EXISTING SYSTEM

In anomaly-based detection, the inverse of this knowledge comes from the learning phase. So theoretically, anomaly-based detection knows what is anomalous behavior based on its knowledge of what is normal. Since anomalous behavior subsumes malicious behavior, some sense of maliciousness is captured by anomalybased detection. If the malware detector employs a signature-based method, its knowledge of what is malicious comes from its repository, which is usually updated/maintained manually by people who were able to identify the malicious behavior and express it in a form amenable for the signature repository, and ultimately for a machine to readThe other input that the malware detector must take as input is the program under inspection. Once the malware detector has the knowledge of what is considered malicious behavior (normal behavior) and the program under inspection, it can employ its detection technique to decide if the program is malicious or benign. Although Intrusion Detection Systems (IDS) and malware detectors are sometimes used synonymously, a malware detector is usually only a component of a complete IDS

B. PROPOSED SYSTEM

The propose a set of simple integrity protection rules based upon open mobile operating system environments and application behaviors. This design leverages the unique features of mobile devices, such as service convergence and limited permissions of user installed applications, and easily identifies the borderline between trusted and untrusted domains on mobile platforms. This approach, thus, significantly simplifies policy specifications while still achieves a high assurance of platform integrity. SEIP is deployed within a commercially available Linux-based smart phone and demonstrates that it can effectively prevent certain malware. The security policy of this implementation is less than 20 kB, and a performance study shows that it is lightweight. Also the using memory read and write option for the storage disk.Also implement the module that is used to view the permissions of the installed applications.The Android platform is a software stack for mobile devices including an operating system, middleware and key applications.

Developers can create applications for the platform using the Android SDK. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use, which runs on top of a Linux kernel.

C. SYSTEM DESIGN

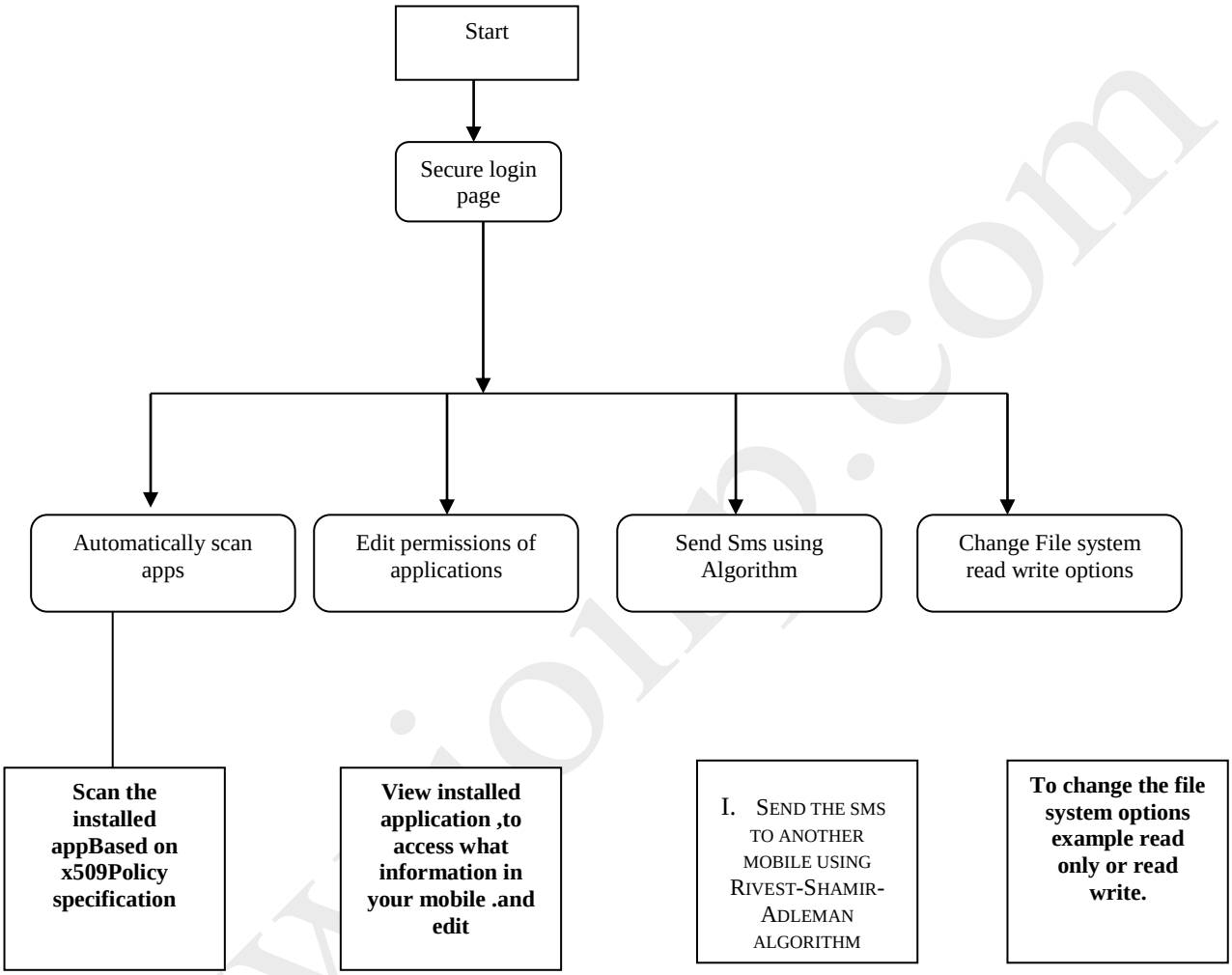

Fig 1. System Architecture

IV. MODULES

A. Malware Detection

This approach, thus, significantly simplifies policy specifications while still achieves a high assurance of platform integrity. SEIP is deployed within a commercially available Linux-based smart phone and demonstrates that it can effectively prevent certain malware. The security policy of the implementation is less than 20 kB, and a performance study shows that it is lightweight. This policy applies to all users of information assets including <Organization-Name> employees, employees of temporary employment agencies, vendors, business partners, and contractor personnel and functional units regardless of geographic location.

This Policy covers all Information Systems environments operated by <Organization-Name> or contracted with a third party by <Organization-Name>. The term "IS environment" defines the total environment and includes, but is not limited to, all documentation, physical and logical controls, personnel, hardware (e.g. mainframe, distributed, desktop, network devices, and wireless devices), software, and information. Although this Policy explicitly covers the responsibilities of users, it does not cover

the matter exclusively. Other <Organization-Name> Information Security policies, standards, and procedures define additional responsibilities. All users are required to read, understand and comply with the other Information Security policies, standards, and procedures.

B. Permission View of Installed Aps

There are many things an app can do to, and for, the phone. There is no reason a game of checkers needs to know friend's phone numbers. In the Permissions option can read a list of some of the most commonly used permissions. The list explains how important they are, what they do, and notes some examples of apps that might legitimately need the permission. This should help to get a basic understanding of what to allow, and when to skip, an app. In this module we check the all installed application permissions. So user can easily see the abstract of the installed application state. Otherwise some applications that access the phone call services.

C. Securing Phone Sms Services

The secure SMS that designed utilizes encryption and decryption, which means that if there is some malware in the middle and tries to intercept or view our short message body it will get nothing but some random bytes. So the message will be secured during transaction.

D. Read/Write Options to File System

In this module that control the file system of the mobile device. So the easily mount file system to read or write options. In read only option any one data from outside world that cannot affect the file system data's. Any time we easily change the file system module through this module.

V. SIMULATION RESULTS

Fig 2.Simulation Secure Login

Fig4. Encryption Method

Fig3.Permission viewer

Fig 5. Security Message

VI. CONCLUSION

Present a simple but yet effective and efficient security solution for integrity protection on mobile phone devices. The design captures the major threats from user downloaded or unintentionally installed applications, including codes. The solution enables very simple security policy development. This implemented that design on an Android platform and demonstrated its effectiveness by preventing a set of attacks. The performance study shows that our solution is efficient by comparing to the counterpart technology on desktop environments. The plan to port this implementation to other Linux-based platforms and develop an intuitive tool for policy development

References

[1] "Google on Android," http://code.google.com/android, 2013.
[2] "Bluebug," http://trifinite.org/trifinite_stuff_bluebug.html, 2013.
[3] "GPE Phone Edition," http://gpephone.linuxtogo.org, 2013.
[4] "JSR-000030 J2ME Connected, Limited Device Configuration," http://jcp.org/aboutjava/communityprocess/final/jsr030, 2013.
[5] "Limo Foundation," https://www.limofoundation.org, 2013.
[6] "Lmbench - Tools for Performance Analysis," http://www. Bitmover.com/lmbench, 2013.
[7] "Maemo," http://www.maemo.org, 2013.
[8] McAfee, "Mobile Security Report 2008," http://www.mcafee. Com/us/research/mobile_security_report_2008.html, 2008.
[9] "MOTOMAGZX Security," http://ecosystem.motorola.com/getinspired/whitepapers/security-whitepaper.pdf, 2013.
[10] National Security Agency, "Security-Enhanced Linux," http:// www.nsa.gov/research/selinux, 2013.
[11] "Openezx," http://www.openezx.org, 2013.
[12] "Pandalabs Quarterly Report," http://pandalabs.pandasecurity.
[13] Com/blogs/images/pandalabs/2008/04/01/Quarterly_Report_ pandalabs_Q1_2008.pdf, 2008.
[14] Trolltech, "Qtopia Phone Edition," http://doc.trolltech.com, 2013.
[15] L. Potter, "Security in Qtopia Phones," LINUX J., http:// www.linuxjournal.com/article/9896, 2013.
[16] Tresys Technology, "SETools - Policy Analysis Tools for SELinux," http://oss.tresys.com/projects/setools, 2013.
[17] T. Krazit, "The Six Secrets to Mobile Computing Success," CNET, http://news.cnet.com/8301-13579_3-9929210-37.html, 2013.
[18] K.J. Biba, "Integrity Consideration for Secure Computer System," Technical Report TR-3153, Mitre Corp., 1977.
[19] A. Bose and K. Shin, "Proactive Security for Mobile Messaging Networks," Proc. ACM Workshop Wireless Security, 2006.
[20] J. Carter, "Using GConf as an Example of How to Create a Userspace Object Manager," Proc. Security Enhanced Linux Symp., 2007.
[21] J. Cheng, S. Wong, H. Yang, and S. Lu, "SmartSiren: Virus Detection and Alert for Smartphones," Proc. ACM Conf. Mobile Systems, Applications, 2007.