Special Issue Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, www.ijtrd.com

Enhancing Secluded Security for the SDK Mobile Platforms using Integrity Protection

S. Vinod Kumar[1], K. Shouryadhar[2].
[1,2]Ravidra college of Engineering for Women, Department of computer science and Engineering, Kurnool, Andhra Pradesh, India.

***Abstract*----**Mobile Applications are a rapidly developing segment of the global Mobile Market. They consist of software that runs on a mobile device and performs certain tasks before the user of the Mobile Phone. They can be downloaded physically through USB / WIFI from a desktop or can be downloaded by a web server over internet. The security of mobile devices such as cellular phones and smartphones 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. We propose and implement an 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.

Index Terms *- Integrity protection, open mobile platforms, Smartphone security.*

1. INTRODUCTION

G*enerally with the increasing computing scalability and network connectivity of mobile devices such as cellular phones and smartphones, more applications and services are deployed on these platforms. Thus, their computing environments become more open than ever before. The security issue in these environments has gained considerable attention nowadays. According to McAfee's 2008 Mobile Security Report, nearly 14 percent of global mobile users have been directly infected or have known someone who was infected by a mobile virus. More than 86 percent of consumers worry about receiving inappropriate or unsolicited content, fraudulent bill increases, or information loss and theft, and more than 70 percent of users expect mobile operators or device manufacturers to preload mobile security functionality the number of infected mobile devices increases remarkably according to McAfee's 2009 report.*

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 installed. 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. 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.

A. Mobile Application Downloads

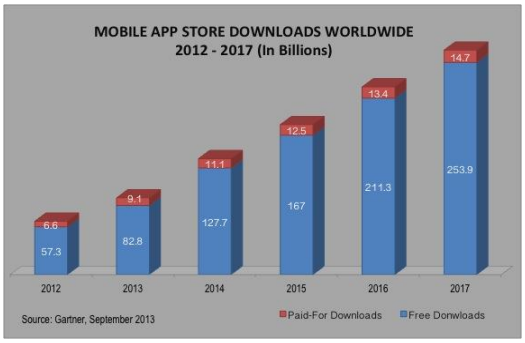

Figure 1: Free Mobile Application Downloads Worldwide

Impact on Business:

Boom in Business-As more and more software development focuses on smartphones, a new industry is building up to help developers create and rapidly deploy mobile applications.

Statistics such as these have motivated an eruption of developers who are hoping to cash in on this newfound mobile boom.

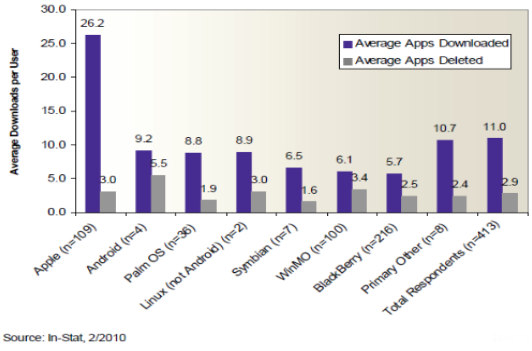

Figure 2: Average Downloading Mobile Applications

Excel Growth:

Some fight wars with words, others with numbers. Hardly a day passes without new data on mobile apps, the small applications that can be downloaded to smart-phones to perform all kinds of feats, such as accessing social networks, playing games and identifying unknown music.

Mobile applications are literally paving the way for companies to generate a ton of sales and revenue. The main reason developers all over the world are calling mobile apps a "gold rush" is because companies of all sizes can greatly flourish by creating mobile applications.

II. MOBILE THREAT TRENDS

We focus our study on the integrity protection of mobile platforms. Particularly for this purpose, we study the adversary model of mobile malware from two aspects: integrity assets and attack mechanisms. Note that, in this paper, we consider attacks from application level.

A. Malware.

Malware, short for malicious software, is software used to disrupt computer operation, gather sensitive information, or gain access to private computer systems. It can appear in the form of code, scripts, active content, and other software. 'Malware' is a general term used to refer to a variety of forms of hostile or intrusive software.

Malware includes computer viruses, ransomware, worms, trojan horses, rootkits, keyloggers, dialers, spyware, adware, malicious BHOs, rogue security software, and other malicious programs; the majority of active malware threats are usually worms or trojans rather than viruses. In law, malware is sometimes known as a **computer contaminant**, as in the legal codes of several U.S. states. Malware is different from defective software, which is a legitimate software but contains harmful bugs that were not corrected before release.

B. Assets for Platform Integrity

Resources of network service provider. A mobile device usually consists of sensitive data from network service provider, such as those stored in SIM card for network and service profiles. Unauthorized access to these data can compromise the running behaviour of the device and communications between the device and wireless network.

Device data and status settings. Modern mobile devices are employed with many sensors, such as timer, GPS, touch screen, and webcam. Manipulating these sensors without authorization from the user can cause unexpected behaviour of a device. Also, a device provides many status setting functions such as those for 3G, WiFi, Bluetooth, screen brightness level, and battery.

Resources of mobile user. A user stores many personal data on device, such as messages, address book, and online credentials. Many online service providers store user data on device side, such as online bank and entertainment services, which are targets for malware. By sending SMS/ MMS messages and making hidden phone calls to premium phone numbers, a malware can generate monetary cost to a mobile user.

We give the idea of some example malware and their infection mechanisms and target integrity assets.

The Mobile Malwares and their Behaviours

- DAmpig, Fontal, Locknut these are infected by Bluetooth, MMS, internet and modify system files and configurations, disable application manager and phone services.
- Cabir,CommWarrior,Mabir malware's propagation through Bluetooth, MMS which scan new devices with Bluetooth,sends user data and malicious code to new targets without authorization.
- Doomboot is affected with Bluetooth, MMS and blocks access to memory card, delete system files and installed application files and data.
- Skulls is by Bluetooth and blocks access to memory card, delete system files and installed application files and data.
- Redbrowser and Mquito malwares is by downloading java applications which sends SMS messages to premium rate number at a rate of 30 and 40 rupees per message.

C. Mobile Phone Applications Types

In this section, we first classify the major players in a mobile phone system from a security perspective, and identify their interactions and the potential security problems that stem from these interactions. We then define the security requirements for a solution to address these problems. In general, we expect a solution that ensures protection of the security-critical (trusted) applications in their use of operating system and user-space service resources in a system that also runs untrusted (e.g., downloaded) applications.Finally, we expect that phone systems that achieve such requirements be capable of proving that to remote parties (e.g., the mobile banking client prove its phone system integrity to the bank). We first classify the entities on the mobile phone system into four categories.

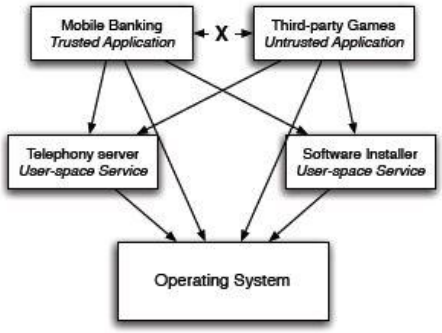

Fig. Mobile phone systems consist of trusted applications, untrusted applications, user-space services that provide function to both trusted and untrusted applications, and the operating system.We prohibit untrusted applications from communicating with trusted applications.

Trusted Applications: Trusted applications are the applications that are entrusted with the processing of security-critical data. These applications must not receive any untrusted inputs as described below. If one of these applications is compromised, then the phone system is compromised. Such applications can include both pre-installed and thirdparty applications, such as a mobile banking application. We assume that trusted third-party applications possess a certificate of trust from an acceptable authority.

Untrusted Applications: Untrusted applications are those that are not entrusted with any security-critical data. Such applications may be compromised without compromising the phone system. Such applications include third-party downloaded applications, but may also include pre-installed applications that do not perform security-critical operations.

User-Space Services: Phone systems typically consist of several user-space programs that provide services to other applications. Examples of these include the software installer,

the telephony server, windowing server, GPS server, etc. These services cater to both trusted and untrusted applications. Such services are also trusted in that if one is compromised, then the phone system is compromised.

Operating System: The phone's operating system

(e.g., Linux, Symbian, Windows Mobile, etc.) is also trusted. These entities can interact leading to security problems as described below.

Untrusted and Trusted Interactions We prohibit untrusted applications from communicating with trusted applications directly or indirectly. This prohibition protects the secrecy and integrity of trusted applications. From an integrity perspective, such communication is undesirable because if an untrusted game or other malicious application is allowed to modify a file read by trusted application or send an IPC to a trusted application, then it may impact the integrity of that trusted application. From a secrecy perspective, we also prohibit flows from trusted applications to untrusted applications to prevent leakage security-critical data. In general, there is no need for a security-critical application to provide data to an untrusted application.

Untrusted and Service Interaction User-space services perform operations for both the trusted and untrusted applications. For example, an untrusted game may call the telephony server to check battery status or send GPRS data requests to its server, so some interaction with the telephony server must be permitted. In processing such requests, we expect that every user-space service will prohibit operations that would result in an information flow between a trusted and an untrusted application. This means that each user-space service must be able to mediate operations that may access security-critical data and that the service must enforce the expected access policy. In addition, each user-space service must protect itself from requests from untrusted applications, as userspace services are trusted by our trusted applications.

III. DESIGN OF SEIP

This section presents design details and integrity rules of SEIP for mobile platform based on discussed security threats and our strategies. Although we describe within the context of Linux-based mobile systems (specifically, LiMo platform), our approach can be applied to other phone systems such as Symbian, as they have similar internal software architecture. One assumption is that we do not consider attacks in kernel and hardware, such as installing kernel rootkits or reflashing unauthentic kernel and filesystem images to devices. That is, our goal is to prevent software-based attacks from application level.

A. Trusted and Untrusted Domains

To preserve the integrity of a mobile device, we need to identify the integrity level of applications and resources. In mobile platforms, typically trusted applications such as those from device manufacture and wireless network provider are more carefully designed and tested as they provide system and network services to other applications. Thus, in our model, we regard them as high-integrity applications or subjects. Note that completely verifying the trustworthiness of a high-integrity subject, for example, via static code analysis, is out of the scope of SEIP. As aforementioned, our major objective is to prevent platform integrity compromising from user installed applications. Therefore, by default all user installed applications later on the platform are regarded as low integrity. In some cases, a user installed application should be regarded as high integrity, example, if it is provided by the network carrier or trusted service provider and requires sensitive operations such as accessing SIM or user data, for example, for mobile bank and payment applications. For an applica-tion belonging to third-party service provider, its integrity level may be based on the trust agreement between the service provider and user or manufacturer/network provi-der. For example, an antivirus agent on a smartphone from a trusted service provider needs to access many files and data of the user and network provider and should be protected from modification of low-integrity software; therefore, it is regarded as high integrity. Other high-integrity applications can be trusted platform management agents such as device lock, certificate management, and embedded firewall.

B. Subjects and Objects:

The design distinguishes subjects and objects in OS. Basically, subjects are active entities that can access objects, which are passive entities in a system such as files and sockets. Subjects are mainly active processes and daemons, and objects include all possible entities that can be accessed by processes, such as files, directories, filesystems, network objects, program and data files. Note that a subject can also be an object as it can be accessed by another process, for example, being launched or killed. In an OS environment, there are many different types of access operations. For example, SELinux predefines a set of object classes and their operations. For integrity purposes, we focus on three access operations: create, read, and write. From information flow perspective, all access operations between two existing entities can be mapped to read-like and write-like operations.

The network manager framework in LiMo creates and maintains all network connections and profiles for different applications, such as packet data protocol (PDP) sessions for GPRS and access point associations for WiFi connections. Another example, Gconf daemon (gconfd) stores configuration data for individual phone applications, which can only access their data via GConf APIs, and gconfd is the only subject that can physically (in OS point of view) read and write the objects. Not only for those objects in regular OS such as files and sockets, our design protects objects internally maintained by these service daemons which affect the integrity of a platform.

Rules for Information Flow Control:

Rule 1: create(s; o) Ã L(o) = L(s), where L(x) is the integrity level of subject or object x: when an object is created by a process, it inherits the integrity level of the process.

Rule 2: create(s1; s2; o) Ã L(o) = MIN((L(s1);L(s2)): when an object is created by a trusted process s1 with input/request from another process s2, the object inherits the integrity level of the lower bound of s1 and s2.

These two rules are exclusively applied upon a single object creation. Typically,

Rule 1 applies to objects that are privately created by a process. For example, an application's logs, intermediate and output files are private data of this process. This rule is particularly applied to Type I trusted subjects and all untrusted subjects. Rule 2

applies to objects that are created by a process upon the request of another process. In one case, s1 is a server running as a daemon process, the s2 can be any process that leverages the function of the daemon process to create objects, e.g., to create a GPRS session, or access SIM data. In another case, s1 is a common tool or facility program that can be used by s2 to create object. In these cases, the integrity level of the created object is corresponding to the lower of s1 and s2. This rule is applied to Type III trusted subjects as aforementioned.

Rule 3: can read(s; o) Ã L(s) · L(o): a low integrity process can read from both low and high integrity entities, but a high integrity process can only read from entity of the same level.
Rule 4: can write(s; o) Ã L(s) ¸ L(o): a high integrity process can write to both low and high integrity entities, but a low integrity process can only write to entity of the same level.

C. Trusted Subjects:

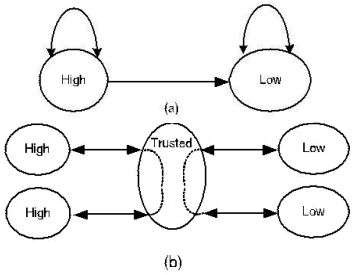

Figure 3: Information flows are allowed between high- and low-integrity entities, directly or indirectly via trusted subjects

In this project it is distinguished three types of trusted subjects on mobile platforms, according to their functionalities and behaviors. Different integrity rules are applied to them for integrity protection purpose.

Type I trusted subjects. This type includes high-integrity system processes and services such as init and busybox, which are basically the trusted computing base (TCB) of the system. Only high-integrity subjects can have information flow to those subjects, while un-trusted subjects can only read from them. Type I trusted subjects also include preinstalled applications from device manufacture or service provider, such as dialer, calendar, clock, calculator, contact manager, and so on. As they usually only interact with other high-integrity subjects and objects, their integrity level is constant during runtime.

Type II trusted subjects. These are applications provided by trusted resources, but usually read low-integrity data only, such as browser, MMS agent, and media player. They are usually predeployed in many smartphones by default, However, they mostly read untrusted Internet content or play downloaded media files in flash memory card. These subjects usually do not communicate with other high-integrity subjects in most current smartphone systems, and they do not write to objects which should be read by other trusted subjects. Therefore, in our design, we downgrade their integrity level during runtime without affecting their functions and system performance.

Type III trusted subjects. These are mainly service daemons such as telephony, message, network manager, interprocess communication (IPC), device status manager (reading and setting hardware status), and application and platform configuration services. Usually these subjects need to interact with both low- and high-integrity subjects.

Dealing with IPC:

A low-integrity process creates an IPC object and writes to it, a high-integrity process cannot read from it, according to our integrity rules. In many mobile Linux platforms such as LiMo, OpenMoko, GPE, Maemo, and Qtopia, D-Bus is the major IPC, which is a message-based communication mechanism between processes. A process builds a connection with a system- or user-wide D-Bus daemon (dbusd). When the process wants to communicate to anther process, it sends messages to dbusd via its connection. The dbusd maintains connections of many processes, and routes messages between them. A D-Bus message is an object in our design, which inherits integrity level from its creating process.

D. Program Installation and Launching:

An application to be installed is packaged according to particular format, i.e., .SIS file for Symbian and .ipk for many Linux-based phone systems, and application installer reads the program package and metadata and copies the program files into different locations in local filesystem. As the application installer is a Type III trusted subject specified by policy, it can read both high- and low-integrity application packages. Also, according to our integrity Rule 2 and 5, it writes (when installing) to trusted part of the filesystem when reads high-integrity software package, and writes to untrusted part of the filesystem when reads low-integrity package.

On one aspect, this enhances the security as a malicious application cannot be launched to a privileged process, which is a major vulnerability in traditional OS; on the other aspect, this simplifies policy specification in a real system, which can be seen in next section.

V. IMPLEMENTATION

We have implemented SEIP on a real LiMo platform. Our implementation is built on SELinux, which provides comprehensive security checks via Linux security module (LSM) in kernel. Also SELinux provides domain-type and role-based policy specifications, which can be used to define policy rules to implement high-level security models. However, existing deployments of SELinux on desktop and servers have very complex security policies and usually involve heavy administrative task. Furthermore, current SELinux does not have an integrity model built-in. On one side, our implementation simplifies SELinux policy for mobile phone devices based on SEIP. On the other side, our implementation augments SELinux policy with built-in integrity consideration.

A. Trusted and Untrusted Domains

All Linux system binaries (e.g., init, busybox), shared libraries (/lib, /usr /lib), scripts (e.g., inetd, network, portmap), and nonmutable configuration files (fstab.conf, inetd. conf, inittab.conf, mdev.conf) are located in a read-only cramfs filesystem. Also, all phone related application binaries, configurations, and framework libraries are located in another cramfs filesystem. All mutable phone related files are located in an ext3 filesystem, including logs, tmp files, database files, application configuration files, and user-customizable configuration files.

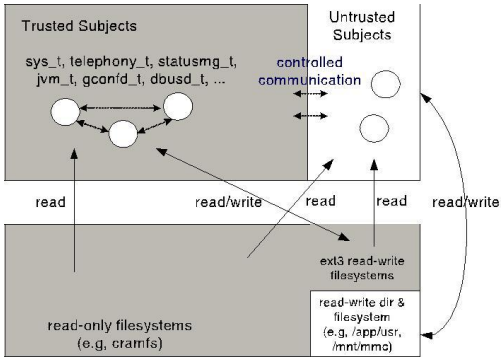

Fig. 2. Trusted and untrusted domains and allowed information flow between them on evaluation platform

all read-only filesystems and part of ext3 filesystem where phone related files are located are regarded as trusted, and user writable filesystems are regarded as untrusted. By default, processes launched from trusted filesystems are trusted subjects, and processes launched from untrusted filesystems are untrusted subjects. Note that our approach does not prevent trusted user application from being installed on the device. For example, a trusted mobile banking application can be installed in the trusted read-write filesystem, and the process launched from it is labeled as trusted.

B. Securing Phone Services

The telephony server provides services to typical phone-related functions such as voice call, data network (GSM or UMTS), SIM access, messages (SMS and MMS), and GPS. An application calls telephony APIs (TAPI) to access services provided by the telephony server, which in turn connects to the wireless modem of the device to build communication channels. In our LiMo platform, message framework and data network framework are dedicated for short message and data network access services. An application first talks to these framework severs which in turn talk to the telephony server. Security controls for those services can be implemented in their daemons.

Different levels of protection can be implemented for secure voice call. For example, one policy allows that only trusted applications can make phone calls, while untrusted application cannot make any phone call, which is the case in many feature phones. For another example policy, un-trusted applications can make usual phone calls but not those of premium services such as payment-per-minute 900 numbers. Different labels can be defined for telephone numbers or their patterns. In our implementation, we allow untrusted applications to call 800 toll-free numbers only. Similar design is used in message framework.

Fig. 3 shows the workflow for a typical voice call. A client application calls tapi_call_setup() to initialize a phone call with TelCallSetupParams t, which includes the target phone number and type (voice call, data call, or emergency call), and a callback function to handle possible results. The telephony server provides intermediate notifications including modem and connection status to the client. Once the call is established with the modem, the telephony server sends the connected indication to the TAPI library which in turn notifies the application via the registered callback function about the status of call (connected or disconnected), and then the application handles the processing.

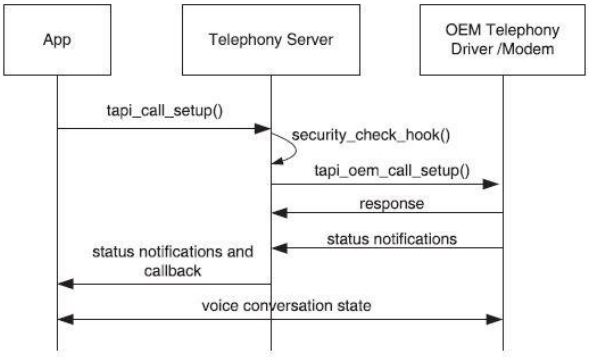

Fig 3. Secure telephony server.

LIMITATIONS

Although we have implemented our design in some major services of our evaluation platform including IPC (D-Bus), telephony, device status manager, and system configuration service, obviously this is not a complete list for a whole platform. due to lack of source code, we do not have implementation on data network service and message service. In general, the framework services of a mobile phone device can be provided by many different vendors, such that a complete implementation so far is not feasible in our prototype. One of our design goals is to ease the integration of security between functional frameworks. Typically, a framework provider just needs to identify sensitive functions or APIs that need to be controlled for integrity purpose, declare a set of corresponding permission names, and insert a common security hook function into the API implementations of the service functions, which is implemented in a trusted library based on our integrity rules. We believe this significantly releases the burden of security considerations for system framework developers.

CONCLUSION

In this paper, we present a simple but yet effective and efficient security solution for integrity protection on mobile phone devices. Our design captures the major threats from user downloaded or unintentionally installed applications, including codes and data received from Bluetooth, MMS, and browser. We propose a set of integrity rules to control information flows according to different types of subjects in typical mobile systems. Based on easy ways to distinguish trusted and untrusted data and codes, our solution enables very simple security policy development. We have im-plemented our design on a LiMo platform and demon-strated 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. We plan to port our implementation to other Linux-based platforms and develop an intuitive tool for policy development.

References

[1] National Security Agency, “Security-Enhanced Linux,” http:// www.nsa.gov/research/selinux, 2013.

[2] L. Potter, “Security in Qtopia Phones,” LINUX J., http:// www.linuxjournal.com/article/9896, 2013.

[3] T. Krazit, "The Six Secrets to Mobile Computing Success," CNET, http://news.cnet.com/8301-13579_3-9929210-37.html, 2013.
[4] K.J. Biba, "Integrity Consideration for Secure Computer System," Technical Report TR-3153, Mitre Corp., 1977.
[5] A. Bose and K. Shin, "Proactive Security for Mobile Messaging Networks," Proc. ACM Workshop Wireless Security, 2006.
[6] J. Carter, "Using GConf as an Example of How to Create a Userspace Object Manager," Proc. Security Enhanced Linux Symp.,
a. 2007.
[7] J. Cheng, S. Wong, H. Yang, and S. Lu, "SmartSiren: Virus Detection and Alert for Smartphones," Proc. ACM Conf. Mobile Systems, Applications, 2007.
[8] D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," Proc. IEEE Symp. Security and Privacy, 1987.

Author's Details

S. Vinod Kumar is working as Assistant Professor in the Department of Computer Science and Engineering at RCEW. He has obtained his masters degree in Computer Science from Sree Vidyanikethan Engineering College (SVEC) Tirupati. He has obtained his bachelors degree in Information Technology from St John's College of Engineering and Technology (SJCET).

Mr. K. Shouryadhar is working as Assistant Professor in the Department of Computer Science and Engineering at RCEW. He has obtained his masters degree in Computer Science and Engineering from G. Pullareddy Engineering College (GPREC) Kurnool. He has obtained his bachelor's degree in Information Technology from the Stanley stephen Engineering college, Kurnool He has worked in the following industries: RYK - Rajiv Yuva Kiranalu, Kurnool - a subsidiary Project of DRDA - IKP, A.P Govt. NAPLTech - Navabharat Agro Products Limited's Tech division - Hyderabad.