International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014
ISSN 2091-2730

# Review of Security Policy Enforcement for Mobile Devices

Dimpal V. Mahajan[1], Prof. Trupti Dange[2]
[1]Department of Computer Engineering, RMD Sinhgad School of Engineering, University of Pune, India

*dimpal.mahajan27@gmail.com*

[2]Professor, Department of Computer Engineering, RMD Sinhgad School of Engineering, University of Pune, India
trupti.dange.rmdstic@sinhgad.edu

**Abstract**— Mobile devices like smartphones and tablets are commonly used for personal and business purpose. Enterprises are more concerns about security of mobile devices compared to personal mobile subscribers. The challenges an enterprise may face include unlimited access to corporate resources, lack of encryption on corporate data, unwillingness to backup data, etc. Many of these issues have been resolved by auditing and enforcing security policies in enterprise networks. However, it is difficult to audit and enforce security policies on mobile devices. A substantial discrepancy exists between enterprise security policy administration and security policy enforcement. So in this paper we will study different security policies.

**Keywords**— Mobile device, security policy. enforcement, constraints, access control ,confidentiality , android.

## 1. INTRODUCTION

Mobile computing and communications technology has evolved tremendously[4] over the last decade. New mobile computing devices with better design and increased capabilities are released frequently on the market. Consequently, rich mobile services such as e-mail, scheduler, contact synchronization and even scaled-down versions of word processors, spreadsheets and presentation software have become more and more common among mobile users, especially in the enter**p**rises.

Android is the most popular open source[1] and fully customizable software stacks for mobile devices. Introduced by Google, it includes an operating system, system utilities, middleware in the form of a virtual machine, and a set of core applications including a web browser, dialer, calculator and a few others. Now a day's peoples are using Mobile phones and Tablets for accessing Internet. Mobile technology has changed our daily lives in many different ways, such as connecting with people, collecting information, and sharing information. Security concerns are increasing as popularity of Smartphone's and tablets is increasing. Android is open platform[2] that provide few direct protections for the content placed on the phone[2]. Access controls restrict access to application interfaces (e.g., by placing permissions on application components in Android), rather than placing explicit access controls on data they handle. Therefore, what limited content protections exist are largely a by-product of the way interfaces are designed and permissions (often capriciously) assigned. Thus, a malicious application with the appropriate permissions can exfiltrate even the most sensitive of data from the phone. Malware has recently begun to exploit such limitations. Moreover even in the absence of malicious applications commercial interests such as media providers wish to provide content without exposing themselves to content piracy.

Aggressive malware and securing Android-powered devices[3] has focused on three major directions. The first one consists of statically and dynamically analyzing application code to detect malicious activities before the application is loaded onto the user's device. The second consists of modifying the Android OS to insert monitoring modules at key interfaces to allow the interception of malicious activity as it occurs on the device. The third approach consists of using virtualization to implement rigorous separation of domains ranging from lightweight isolation of applications on the device to running multiple instances of Android on the same device through the use of a hypervisor.

The competitive advantages for both mobile businesses and individual users is rovided by the corporate usage of handheld devices[5], such as mobile devices and it has been fastly growing in recent years due to wide dispersion of mobile devices. As the storage capability of mobile devices spans between internal memories, removable memories (e.g., Secure Digital, Multimedia) and

SIM cards, the overall amount of sensitive personal and corporate information that can be stored in those devices can be relevant. Furthermore, some storage volumes (e.g., removable memory cards) are essentially less secure than other volumes (e.g., SIM cards). Hence many companies, when planning to provide their personnel with mobile devices, have to face the typical new threats related to such devices. Unluckily, when compared to personal computers, mobile devices are lacking a number of security features (e.g., effective anti-malware solutions, intrusion detection systems), which can be fundamental in order to protect corporate and personal data as well.

## 2. Review on security policy on mobile devices

- The author in [1], said that the first mass-produced consumer-market open source mobile platform is Android that allows developers to easily develop applications and users to eagerly install them. The ability to install third party applications by users poses serious security concerns. Whereas the existing security mechanism in Android allows a mobile phone user to see which resources an application requires, but she has no choice to allow access to all the requested permissions if she wishes to use the applications. There is no other way of allowing some permissions and denying others. Also there is no way of restricting the usage of resources based on runtime constraints such as the location of the device or the number of times a resource has been previously used.

Third party developers creating applications for Android can submit their applications to Android Market from where users can download and install them. While this provides a high level of availability of unique, specialized or general purpose applications, it also gives rise to serious security concerns. When a user installs an application, she has to trust that the application will not misuse her phone's resources. At install-time, Android presents the list of permissions requested by the application, which have to be granted if the user wishes to continue with the installation. This is an all-or-nothing decision in which the user can either allow all permissions or give up the ability to install the application. As once the user grants the permissions, there is no way of revoking these permissions from an installed application, or imposing constraints on how, when and under what conditions these permissions can be used.

To deal with these problems, we have developed Android Permission Extension (Apex) framework[1], a comprehensive policy enforcement mechanism for the Android platform. Apex gives a user several options for restricting the usage of phone resources by different applications. The user may grant some permissions and deny others. This allows the user to use part of the functionality provided by the application while still restricting access to critical and/or costly resources. Apex also allows the user to impose runtime constraints on the usage of resources. Finally, the user may wish to restrict the usage of the resources depending on an application's use e.g., limiting the number of sms messages sent each day. We define the semantics of Apex as well as the policy model used to describe these constraints. We also de scribe an extended package installer which allows end-users to specify their constraints without having to learn a policy language. Apex and the extended installer are both implemented with a minimal and backward compatible change in the existing architecture and code base of Android for better acceptability in the community.

- In [2], the access of cellular networks worldwide and emergence of smart phones has led to a revolution in mobile content. Users consume various content when, for example, exchanging photos, playing games, browsing websites, and viewing multimedia. Current phone platforms provide protections for user privacy, the cellular radio, and the integrity of the OS itself. However, few offer protections to protect the content once it enters the phone. For example, MP3-based MMS or photo content placed on Android smart phones can be extracted and shared with impunity.
The author[2], introduce the Policy Oriented Secure Content Handling for Android (Porscha) system. . The Porscha system developed in this work places content proxies and reference monitors within the Android middleware to enforce DRM policies embedded in received content. An analysis of the Android market shows that DRM services should ensure: a) protected content is accessible only by authorized phones b) content is only accessible by provider-endorsed applications, and c) access is regulated by contextual constraints, e.g., used for a limited time, a maximum number of viewings, etc.

Porscha policies are imposed in two phases: the protection of content as it is delivered to the phone and the regulation of content use on the phone . For the former, Porscha binds policy and ensures content confidentiality \on the wire" using constructions and infrastructure built on Identity-Based Encryption [6]. For the latter, Porscha enforces policies by proxying content channels (e.g., POP3, IMAP, Active Sync) and placing reference monitor hooks within Android's Binder IPC framework.

- In[3], The increasing popularity of Google's mobile platform Android makes it the prime target of the latest flow in mobile malware. Most of the research on enhancing the platform's security and privacy controls requires extensive modification to the operating system, which has significant usability issues and hinders efforts for widespread adoption. We develop a novel solution called Aurasium that bypasses the need to modify the Android OS while providing much of the security and privacy that users desire. We automatically repackage arbitrary applications to attach user-level sandboxing and policy enforcement code, which closely watches the application's behavior for security and privacy violations such as attempts to retrieve a user's sensitive information, send SMS covertly to premium numbers, or access malicious IP addresses. Aurasium can also detect and prevent cases of privilege growth attacks. Experiments show that we can apply this solution to a large sample of benign and malicious applications with a near 100 percent success rate, without significant performance and space overhead. Aurasium has been tested on three versions of the Android OS, and is freely available.

We develop a deployable technology called Aurasium and which is also a novel, simple, effective, robust technology. Theoretically, we want Aurasium to be an application-hardening service: a user obtains arbitrary Android applications from potentially untrusted places, but instead of installing the application as is, pushes the application through the Aurasium black box and gets a hardened version. The user then installs this hardened version on the phone, assured by Aurasium that all of the application's interactions are closely monitored for malicious activities, and policies protecting the user's privacy and security are actively enforced.

Aurasium does not need to modify the Android OS at all; instead, it enforces flexible security and privacy polices to arbitrary applications by repackaging to attach sandboxing code to the application itself, which performs

monitoring and policy enforcement. The repackaged application package (APK) can be installed on a user's phone and will enforce at runtime any defined policy without altering the original APK's functionalities. Aurasium exploits Android's unique application architecture of mixed Java and native code execution to achieve robust sandboxing. In particular, Aurasium introduces libc interposition code to the target application, wrapping around the Dalvik virtual machine (VM) under which the application's Java code runs. The target application is also modified such that the interposition hooks get placed each time the application starts.

Aurasium is able to interpose almost all types of interactions between the application and the OS, enabling much more fine-grained policy enforcement than Android's built-in permission system. For instance, whenever an application attempts to access a remote site on the Internet, the IP of the remote server is checked against an IP blacklist. Whenever an application attempts to send an SMS message, Aurasium checks whether the number is a premium number. Whenever an application tries to access private information such as the International Mobile Equipment Identity (IMEI), the International Mobile Subscriber Identity (IMSI), stored SMS messages, contact information, or services such as camera, voice recorder, or location, a policy check is performed to allow or disallow the access. Aurasium also monitors I/O operations such as write and read. We evaluated Aurasium against a large number of real-world Android applications and achieved over 99 percent success rate. Repackaging an arbitrary application using Aurasium is fast, requiring an average of 10 seconds.

- In [4], An extended security architecture and policy model to address the lack of flexibility of the current security model employed at mobile computing platforms is proposed. The proposed model has the potential to open up the mobile device software market to third-party developers and it also empowers the users to tailor security policies to their requirements in a fine-grained, personalized manner.
Our work focuses on the Java 2 Micro Edition (J2ME) - one of the most widely used virtual machine execution environments for mobile computing devices today.

Our main contribution is the design and implementation of an extended version of the current J2ME, which we describe **xJ2ME**, from e**x**tended **J2ME**. xJ2ME enables runtime enforcement of a much more expressive class of security policies compared to the current state-of-the-art, allowing for a fine-grained behavior control of individual applications. Furthermore, initial evaluations show no significant, even noticeable, performance overheads. Having introduced the fundamental aspects of the J2ME architecture, in this section we present the extensions to the architecture and modification of its operational aspects that enable the support for fine-grained, history-aware, userdefinable, per-application policy specification and enforcement. We preserve the fundamental aspects of the J2ME security model, such as its domain based nature.

The aim of the work is to provide the support for fine-grained, history-based, application-specific policy specification and enforcement within the J2ME framework. A simple example of a security policy that is supported by our extended J2ME architecture (xJ2ME) is "browser may not download more than 300 KB of data per day". Figure depicts the new Java security architecture. The Run-time Monitor is in charge of making resource access decisions. In order to grant or deny resource access, the Run-time Monitor relies on the Policy Manager to identify the relevant application-specific policy. Once the policy is identified, the Run-time Monitor evaluates its conditions in conjunction with resource usage history information of the system and MIDlet, as obtained from the History Keeper. If the policy conditions are fulfilled, access is granted, otherwise a Security Exception is thrown.

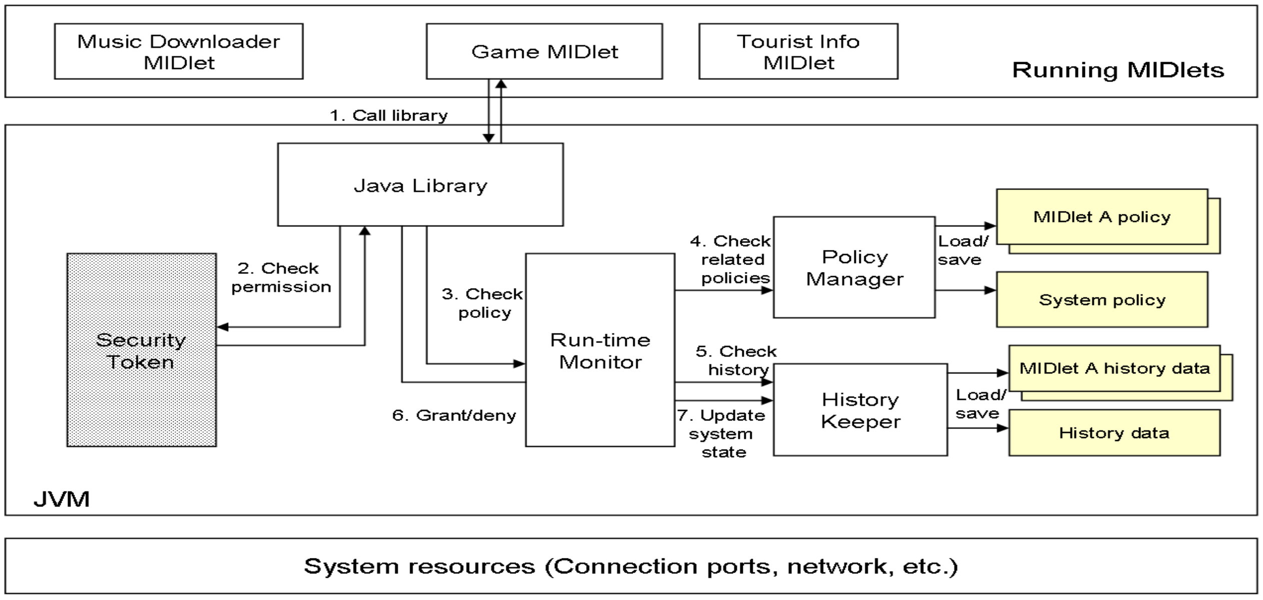

**Fig. Extending the J2ME security architecture with the Run-time Monitor.**

- In[5], The new perspectives related to the security policies and services enforcement for mobile devices is focused . We propose a novel methodological approach to protect the lifecycle of mobile devices aim at the secure management of such devices either directly by the user or by the organization. In order to review the feasibility and impact of our approach in a real scenario, we customized a mobile device with several security features respecting the goal of the proposed lifecycle. A SecureMyDroid is based on our own customization of a Google Android OS version and tested it on actual mobile devices. SecureMyDroid is a prototype developed on a customized version of Google Android OS. It is designed to support and realize practically the secure lifecycle management of the mobile devices as proposed in this work. The basic step of this activity is to obtain, modify and compile the source code of the mobile operating system, so that the new OS image file can be deployed on the mobile device. Starting from this non-trivial result we have realized different features for improving the security management of the entire lifecycle. We sketch next some of these features.

  First of all, we patched the source code of the installation manager in order to inhibit the capability to install new applications on the device. This security feature ensures both that the user cannot install new personal applications (e.g., games, entertainment) on its business device and that the device cannot be compromised by the installation of mobile malwares (e.g., virus, worms, Trojans). SecureMyDroid offers the possibility to disable the use of memory cards (e.g., secure digital cards). This countermeasure can prevent from nimbly copying or moving sensitive business data out of the device.

  We strengthened the relationship between the mobile device and a specific SIM card. Whenever the device is switched on, the association with the SIM is checked; if the device recognizes that the inserted SIM does not match the expected one a dialog prompting for a special PIN is shown. The device remains inactive until the user provides the correct PIN. After a given number of unsuccessful attempts, the device can be permanently blocked.

We placed into the source code of most critical function of the mobile device (e.g., making a call, sending an SMS/MMS/e-mail, connecting to a Wi-Fi network, syncing with a computer) an event logger call. When a stated time slot elapses the collected log messages are transferred to a remote server using HTTPS connections.

SecureMyDroid can inform all the actions to an organization remote server performed by the user on the device through an HTTPS connection. At the same time, the organization can explicitly query the device through simple text messages in order to obtain specifically information such as the current GPS position, the last critical executed operations, the last contacts added to the address book..etc.

Remote wiping is another interesting feature provided by SecureMyDroid. This function enables the organization to send a remote wiping command to the mobile device. We implemented two different wiping modes: a soft mode and a hard mode. A soft wiping is

carried out only on all databases containing personal information (e.g., SMS, MMS, e-mail, address book, calendar). A hard wiping implies the execution of a remove command on the mobile device shell; this completely deletes the OS and a new OS has to be installed on the mobile device to make it again operational. Remote wiping can be activated with a simple text message or can be executed when the mobile device starts with a non-authorized SIM.
Periodically, the security manager of the company can query the mobile device to receive the configuration of the customized OS. When an employee requires an upgrade of his/her customized mobile device, the security management using SecureMyDroid can backup all the data stored in the mobile device and according to the employee privileges can produce a new version of SecureMyDroid. After this, the employee can restore all the useful data on the newly customized mobile device.

When either the device is assigned to a new employee or it must to be permanently disposed, the security management may check and export the log collected on the mobile phone. Moreover, before deleting all data, if needed, he/she can activate the backup of personal information.

## 3. CONCLUSION

So we conclude as Apex allows users to specify detailed runtime constraints to restrict the use of sensitive resources by applications. The Porscha is a content protection framework for Android that enables content sources to express security policies to ensure that documents are sent to targeted phones, processed by endorsed applications, and handled in intended ways. Aurasium is a robust and effective technology that protects users of the widely used Android OS from malicious and untrusted applications. An extension to the security architecture of the Java Virtual Machine for mobile systems, to support fine-grained policy specification and run-time enforcement. SecureMyDroid is a customized version of Google Android OS to support and realize practically the secure lifecycle management of the mobile devices.

## REFERENCES:

[1]M. Nauman, S. Khan, and X. Zhang, “Apex : Extending Android Permission Model and Enforcement with User-defined Runtime Constraints,” *Inf. Syst. J.*, no. c, pp. 328–332, 2010.
[2] M. Ongtang, K. Butler, and P. McDaniel, “Porscha: Policy oriented secure content handling in android,” in *Proceedings Annual Computer Security Applications Conference ACSAC*, 2010, pp. 221–230.
[3] R. Xu, M. Park, and R. Anderson, “Aurasium : Practical Policy Enforcement for Android Applications,” in *USENIX Security 12*, 2012.
[4] I. Ion, B. Dragovic, and B. Crispo, *Extending the Java Virtual Machine to Enforce Fine-Grained Security Policies in Mobile Devices*. Ieee, 2007, pp. 233–242.
[5] A. Distefano, A. Grillo, G. F. Italiano, and A. Lentini, “SecureMyDroid : Enforcing Security in the Mobile Devices Lifecycle,” *Management*, pp. 1–4, 2010.

[6]D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In Proceedings of CRYPTO, 2001