A Policy-based Mechanism for restricting Execution of malicious apps in Android Smartphone

Sanjeev Kumar Dwivedi¹, K.Muthumanickam², Ankur Singh Bist³, Dr. P. C. Senthil Mahesh⁴

¹ Assistant Professor, Department of Information Technology, Krishna Institute of Engineering and Technology, Ghaziabad, India
² Assistant Professor, Department of Computer Science and Engineering, Arunai Engineering College, Tamilnadu, India
³ Assistant Professor, Department of Computer Science and Engineering, Krishna Institute of Engineering and Technology, Ghaziabad, India
⁴ Professor, Annamacharya Institute of Technology and Sciences, Rajampet, Andra Pradesh, India

Email: ¹ sanjeevdwivedi131988@gmail.com, ² kmuthoo}@pec.edu, ³ ankur1990bist@gmail.com, pcsenthimalesh@gmail.com

Abstract—Android is the most widely accepted and used open source Operating System (OS) developed and supported by Google. Compared to other OS, the source code of Android OS can be modified by users to release with their own customized features. The internal features like built-in services and applications offered may differ from one developer to another including the user interface design.

Android phone works based on several applications installed within them. While installing a new application may lead to some malicious software to be installed without the consent of the owner. Some of the application requests the user to access the resource from mobile phone such as SMS, Contacts, Alarm, Calculator, etc. But some of them access the resources which are not requested. This causes the data theft, leakage or loss. This paper proposed a policy based mechanism which incorporates a database about good and bad application which is based on required permission and used permission. The requested permissions of an application is compared with the good and bad application pattern and then its critical APIs are identified. If the predefined pattern matches with good application then it is concluded as secure to install. Otherwise, the user can assign necessary permissions to access the resource in the mobile phone and either the user will install the application or terminate it.

Keywords—Android, Security, Access restriction, Policy Enforcement, Malicious apps.

I. INTRODUCTION

A OS is an very essential for making all hardware components to work according to their intended functions and applications. A mobile OS manages cellular and wireless network connectivity, etc. Typical mobile OSes like Android, Google, Apple IOS, BlackBerry, etc. Android is the most popular and widely used mobile OS relies on Linux with public licence. Android was released under Apache Version 2 open source license. It is designed and developed for various usages of people. Business users have gained a lot by Android because of product maximization and users may keep even their sensitive information or data in Smartphone. Most of the operations are done by several Applications (Apps) which are installed in Smartphone. In Android, the sources and resources are compiled and converted to run on the Dalvik Virtual Machine (DVM) and then packaged up as an Android application package (APK) file.

Google provides the Software Development Kit for developers to build different Android Apps to enhance and attract customer community. Various components of an application are vital part of the Android OS. Typically such components are loosely integrated into the one of the components of Android OS namely Manifest.xml file which takes the responsibility of describing the details of each component and their interaction and it usually includes the required permissions and used permissions. Few important components of an Android Application that are targeted by Malware [8-11] writers are services, activities, content providers, and broadcast receivers. The Activity component handles the user interface and responsible for handling phone-user interaction events. The Services component handles all types of background operations that are relevant to executable applications. The objective of Broadcast receiver component is offering necessary communication translation between installed applications and the OS. Finally, the Content provider component is responsible for resolving database management related issues. By the device manufacturer, the Smartphone are usually sold with some in-built Apps like calendar, message, contacts, alarm, browser, etc., The most important application existing every Smartphone called “Application Stores” which allow the user to download and install some other applications.

While installing an Android app, it may lead to some malicious software to be installed without consent of the owner. This leads to data theft, leakage or loss. The set of permissions are used to compare the applications and then the critical APIs are detected. So the Smartphone needs some security measures when the user installs the applications in their Smartphone. To avoid these malware activities the user must have awareness about the clean and malicious Apps.

As users might use Smartphone to do their daily business and might use them to keep their important data into the phone itself, it becomes an attractive target to malware.
authors. The first malware that targets mobile OS was released in the year 2004 which attacked Smartphone that ran Symbian mobile OS. During the first quarter of 2012, the malware attacks against Android Smartphone were only countable numbers. In 2013, it has increased to 6 million and the count was nearly 7 million during the fourth quarter of 2014 [1]. As more number of users will opt either a tablet or Smartphone to do all their daily activities, it will be increased into huge numbers in near future.

Malwares that target Smartphone can be flooded through many ways like downloading executable application from Playstore and sites, executing applications using Bluetooth option and accessing web resources through internet downloading applications from stores and installing executable using Bluetooth. A malicious app software can be used for performing many illegal activities like stealing a user’s private information, blocking legitimate operations of the device, sending messages by making financial loss, supplying backdoor, etc. Compare to the wired communication medium, propagating a virus through wireless backbone produces more benefits to malware authors because of the following reasons.

- Detecting malicious attacks can be possible only if an operation of an access point deals legitimate Wi-Fi frames
- Existing forensic tools and methods are not sufficient to discover the existence of malware infected device and app.
- Detecting a virus which propagates using Wi-Fi-AP pose additional challenges.

Android Mobile phones are today’s most successful telecommunication medium for increasing the productivity of customers. A mobile phone is typically in-built with either open source or preparatory mobile operating system together with number of attractive applications such as social media, online business, calendar, address book, web browser, etc., by device manufacturer. While customer installing a third party application on their android mobile phone, several security violations may arise.

For example activities such as robbing the customer’s personal information, making calls or sending Short Message Service (SMS) without any consent of the user’s knowledge. This pose serious security policy violation which results into expose of customer’s sensitive data, especially when the standard security mechanisms offered by the mobile phone security software apps are not sufficient to protect such attacks.

In this paper, a policy-based mechanism has been proposed with the intention of classifying good mobile apps from bad mobile apps. This holds the required permission and used permission of an application previously installed. The set of permissions is used to analyze the set of patterns of an application to determine whether either it is a legitimate application or malicious application. When the application matches with good application pattern then it is secure to install. On the other hand if the pattern of an application matches with bad application pattern then the user can either assign necessary permissions to access the resources in the mobile phone or to terminate the application. To ensure the same research team further determine the critical APIs that are used by the application as it directly affect the root of the operating system.

II. RELATED WORKS

Many third party mobile applications are available either for free or purchasable which can be easily downloaded through online. Typically, all applications did not contain authentic information about its developer and manufacturer. Such applications can be used to steal users’ information with their consent. An Application Programming Interface (API) function can be used for performing many services like providing communication between two mobile applications, passing information, or controlling even the entire application. APIs are being called when an app is installed and get executed by the kernel of the OS. API specifications specify the required checks and other APIs to be invoked before and after an API call. Incorrect usage of APIs can lead to robustness and security problems. Some of the critical APIs are listed in Table 1.

TABLE 1. Important critical APIs

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Critical APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>android.content.Context</td>
<td>getFilesDir(),getCacheDir(),getIntDir(),getPackageCodePath(),getCacheDir()</td>
</tr>
<tr>
<td>android.app.DevicePolicyManager</td>
<td>installKeyPair(),uninstallCaCert(),installCaCertificate(),setUserApiPolicy()</td>
</tr>
<tr>
<td>android.app.Icon</td>
<td>setSmallIcon(),setLargeIcon()</td>
</tr>
<tr>
<td>android.app.ActivityNotifications</td>
<td>getActiveNotifications(),getActiveNotifications()</td>
</tr>
<tr>
<td>android.Display.Mode</td>
<td>getPhysicalWidth(),getSize()</td>
</tr>
<tr>
<td>android.ColorStateList.Resources</td>
<td>getColor(),getColorStateList(),getResources(),getColor()</td>
</tr>
<tr>
<td>android.net.VpnService</td>
<td>setAlwaysOnVpnPackage(),getAlwaysOnVpnPackage()</td>
</tr>
<tr>
<td>android.content.Intent</td>
<td>getIntCount(),editIntentAt(),onNavigateUp(),getParentActivityIntent(),finishAffinity()</td>
</tr>
<tr>
<td>android.app.MediaCodec</td>
<td>createEncoderByType(),createDecoderByType(),createEncoderByType(),createDecoderByType()</td>
</tr>
<tr>
<td>android.Buffer.ByteBuffer</td>
<td>getInputBuffers(),getOutputBuffers(),dequequeInputBuffer(),releaseOutputBuffer(),queueSecureInputBuffer()</td>
</tr>
<tr>
<td>android.MediaPlayer.OnTimedTextListener</td>
<td>getTrackInfo(),addTimedTextSource(),selectTrack(),setOnTimedTextListener(),setNextMediaPlayback()</td>
</tr>
<tr>
<td>android.net.Uri</td>
<td>SetBeamPushUri(),setBeamPushUriCallback(),createBeamUri()</td>
</tr>
<tr>
<td>android.net.nsd</td>
<td>setServiceName(),setServiceType(),setPort(),registerService(),discoverServices(),resolveServices()</td>
</tr>
</tbody>
</table>
An Android malware detection app namely, APK auditor [2] proposed towards imposing the trust worthiness of Android market. The APK auditor is a machine-learning based malware detector developed for Android Smartphone and capable of detecting harmful programs to improve both security and privacy of a mobile phone user. The APK framework monitors permission access on different objects and events of an application and analyzed the collected information using machine-learning method to decide whether an application is legitimate or malicious. However APK auditor omitted permission auditing omits the newer application permission.

Few Advertisement (advert) networks for mobile applications required to inspect the layout of each advert to discover the hidden of fraudulent content. Performing such work manually causes errors and failed to support scalability. Liu et al [3] presented a tool namely DECAF. DECAF was designed to detect frauds that are placed in different places in a layout. DECAF applied optimization techniques and automated navigation methods to inspect a huge amount of visual elements to detect the presence of a malware. In addition, DECAF can effectively identify whether advert of an application violates the rules of advert display and placement. But DECAF cannot support Android Smartphone and used for only Windows based applications. The frauds should be identified in software or hardware programs during increased objects namely DECAF.

The MadFraud framework was designed to automatically run many different applications at once to activate and expose advert fraud attacks. The MadFraud automatically identified advert impression click within three steps: First, building the Hyper Text Transfer Protocol (HTTP) request. Next, identification of different advert requested pages and finally, discovering advert clicks through HTTP request tree method. One of the important obstacles of this approach is taking more time to build HTTP request.

Chung et al. [6] proposed a light weight and strict framework based on its permission validation of applications during runtime. Typically, an application can be secured based on implemented permission privileges, thus Android applications cannot be analyzed all pre-installed applications statically to check its approved permissions. Through this approach, a user can able to easily determine a highly privileged application. One of the noted point of this approach is to prohibit unintended app's behavior without using its acknowledgement. However assigning precise permission during runtime is a tedious task.

III. MOTIVATIONS

The following points motivate us to design and develop a policy based mechanism for protecting Android platform against malicious executables.

- Recently, Android Smartphone is most probably used.
- Most of the people use their mobile phones to store sensitive information and do their transactions through online in transit.
- Additionally, many real time applications can be controlled via smart phones.
- As a result, Smartphone is secured with less security policies.
- This attracts malicious attackers to hack and control mobile devices remotely.
- So Smartphone needs some strong security measures before install an application.
- Identifying the untrusted application in mobile environment is a challenging task.

IV. PROPOSED MECHANISM

Research team proposed a policy-based approach which has a database about good and bad application that relies on its required permissions as well as its used permissions of a mobile application to be installed. First, the required permission of an application being installed is compared with the good and bad application pattern. If it belongs to good application patterns then its critical APIs are identified. As an API act as a interface between user and kernel, its functionalities should be verified. When an API is called by an application, the kernel of the Android OS will provide service...
without verifying whether it is legitimate or malicious application. When the application nearly matches with good application then research team claims as it is secure to install. Otherwise the user will grant necessary permission to access the resource in the mobile phone. The user will either decide to install the application or terminate it.

For matching permission of a requested application, the Android Manifest.xml file is mainly used to verify whether the application is assigned with required permission or used permission. Used permissions are the permissions that takes place after an application has been installed. Required permissions are the permissions that are requested during application installation. These permissions are used to distinguish the legitimate application from malicious application.

V. IMPLEMENTATION AND EVALUATION

This section presents the experimental results about the proposed mechanism and also evaluates the overhead on the system and application performance. Research team implemented and tested the proposed policy-based privacy mechanism on the Android Nexus 4 Smartphone running the Android OS version of 4.2.2. To test and evaluate our prototype, research team collected 25 applications from reputed website such as Google Play store. A special software program namely, Android Debug Bridge (ADB) has been used to test each experiment separately by inserting log commands in different places of the OS where amendments were made to examine.

It is very important to assess the feedback as a result of denying a permission which is not actually permitted during installation time. Research team performed logging activity on all 25 mobile applications using the ADB utility to track permissions invoked by checkComponentPermission() function. Figure 2 shows the impact of revoking runtime permissions on each application separately. Application crash is being caused due to poor coding in which the developer failed to include error-handling mechanism during permission validation process.

However, application crashes can be minimized by adding a script through application update process. As far as the performance overhead issue is concerned, we measure the additional time the Android OS takes to execute our updated code which is given in Table 2.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Overhead (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkComponentPermission()</td>
<td>13.204</td>
</tr>
<tr>
<td>Intent-startService()</td>
<td>6.212</td>
</tr>
<tr>
<td>Intent-startActivity()</td>
<td>13.816</td>
</tr>
</tbody>
</table>

The above tabulated results show that the delay caused by enforcing the proposed policies is negligible. Each security profile is associated with set of policies that separate the applications and data in the Smartphone. Users can download a third party application on their Smartphone through
application. Before installing the third party application on their Smartphone, our framework allows the user to set security policies for that application. Our android framework can automatically activate the security profiles based on the current context. Pattern mining technique is used to identify the contrast permission patterns from the malicious applications and clean applications. It required two datasets such as one for keeping clean android applications and other one for dealing malicious android applications. Each android application has one important file called AndroidManifest.xml. The permissions are extracted from AndroidManifest.xml. Each android application was tested by forty three antivirus engines through [7]. If it is passed by all virus tests were considered as clean application otherwise malware one.

VI. CONCLUSION

In the present system, a mobile malware can be identified only after it has been installed. The complexity of malware attack will be increased as the technology develops. Therefore, every system needs an improvement in the future. Day-by-day, thousands of malware are created and it’s not possible to detect all of them instantly by using the current system. Therefore research team proposed a policy based mechanism for preventing the unauthorized execution of a malicious application in Android Smartphone. The proposed policy-based mechanism provides security policy based framework for separating the applications and data on the android platform and then it’s incorporating used permission and Required permissions from embedded permission list and also find out their efficacy in comparing a clean application with a malicious application.

REFERENCES