Security Constraint System for Android Devices

[1] P Greeshma Chowdary [2] B Srinivas Goud, [3] Prasad B
[1]II/IV, [2][3]Associate Professor
[1][2][3] Department of CSE, Marri Laxman Reddy Institute of Technology and Management (MLRITM) Hyderabad
[1] greeshmaputtagunta10@gmail.com [2] sree.srinu@mlritm.ac.in [3]bprasad@gmail.com

Abstract: **The applications that we use in our mobile devices often access sensitive data and resources. But when the users data has been misused due to some malicious applications which may lead to leakage of sensitive data and also reflect in privacy. An example is a malicious application records users banking details. The problem starts when the user have installed the application by granting all the privileges on which the user have no control on operating the features. To avoid this problem we propose a context based access control system by which a user can activate and deactivate some of the applications that are already present in the users mobile based on the context that is provided. It can also perform its action in a particular location by using GPS, Wi-Fi etc.. based on the context that is provided by the user. We have preformed many experiments for accessing the data in a particular location based on context.**

***Keywords*: Policy Manager, Policy Controller, Access Controller, End User**.

I. INTRODUCTION

As every Android applications often have access to sensitive data and resources on the user device. Misuse of this data by malicious applications may result in privacy sensitive data leakage and transfer of data to unauthorised persons. When a user install any application by granting all the permission for the application then the user cannot restrict any of the permissions such that any app developer can declare illegal permissions and can record any kind of data from users device with out the permissions of any user. An example would be a malicious application unauthorised recording a confidential business conversation or recording of video with out the knowledge of user. The problem arises from the fact that Android users do not have control over the application capabilities once the applications have been granted the requested privileges upon installation. In many cases, however, whether an application may get a privilege depends on the specific user context and thus we need a access control mechanism by which privileges can be dynamically granted or revoked to applications based on the specific context of the userWith security constraint system policies, employees may be allowed to use smart phones as any users can restricts applications from using the camera and any device resources and privileges that employers restrict while at work while the user device can retain all its original privileges outside the work area. Security constraint system policies are also a necessity for politicians and law enforcement agents who would need to disable camera , microphone , location services, usage of mobile data etc from their devices during confidential meetings while retaining these resources back in non-confidential locations.

The main objective of this system is to remove illegal or unwanted permission for applications from accessing specific data and/or resources. The restrictions specified in a policy are automatically applied as soon as the user device matches the pre-defined locations.

To provide more security for the users from collecting sensitive data and and exposing user to high risks.As Smartphone's are becoming more powerful application developers are advantage of collecting sensitive data and exposing user to high risks by applications useshem inappropriately and without the user's knowledge.

Security for mobile operating systems focuses on restricting applications from accessing sensitive data and resources, but mostly lacks efficient techniques for enforcing those restrictions according to fine-grained contexts that differentiate between closely located subareas. Moreover, most of this work has focused on developing policy systems that do not restrict privileges per application and are only effective system-wide. So User disable all applications from using the camera and any device resources and privileges that employers restrict while at work, while the user device can retain all its original privileges outside the work area. The drawback of the existing system are do not cover all the possible ways in which applications can access user data and device resources, the User leakage of their privacy.

In this system, we propose a security constraint mechanism for Android systems that allows android users to set configuration permissions over their applications' usage of device resources and services at different contexts. Through the system constraint mechanism, users can, for example, set restricted privileges for device applications when using the device at work, and device applications may re-gain their original privileges when the device is used at home. This change in device privileges is automatically applied as soon as the user device matches a pre-defined context of a user-defined policy. The user can also specify a default set of policies to be applied when the user is located in a non-previously defined location. Configured policy restrictions are defined according to the accessible device resources, services, and permissions that are granted to applications at installation time. Such policies define which services are offered by the device andlimit the device and user information accessibility. Policy restrictions are linked to context and are configured by the device user. User can also revoke all the permissions for all the application which are installed in users mobile device.

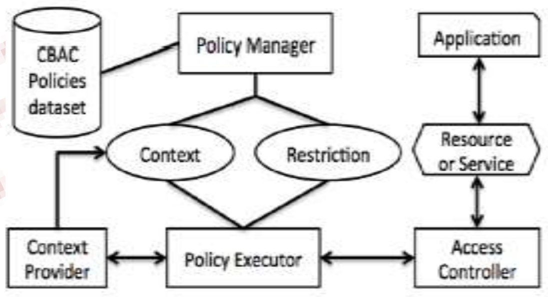

Fig 1:Data flow

II. BACKGROUND

It is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

In this paper, we introduce a security framework for practical and lightweight domain isolation on Android to mitigate unauthorized data access and communication among applications of different trust levels (e.g., private and corporate). We present the design and implementation of our framework, Trust Droid, which in contrast to existing solutions enables isolation at different layers of the Android software stack: (1) at the middleware layer to prevent inter-domain application communication and data access, (2) at the kernel layer to enforce mandatory access control on the file system and on Inter-Process Communication (IPC) channels, and (3) at the network layer to mediate network traffic. For instance, (3) allows network data to be only read by a particular domain, or enables basic security constraint policies such as preventing Internet access by un-trusted applications while an employee is connected to the company's network. Our approach accurately addresses the demands of the business world, namely to isolate data and applications of different trust levels in a practical and lightweight way. Moreover, our solution is the first leveraging mandatory access control with TOMOYO Linux on a real Android device (Nexus One). Our evaluation demonstrates that Trust Droid only adds a negligible overhead, and in contrast to contemporary full virtualization, only minimally affects the battery's life-time.

Smartphone's are now ubiquitous. However, the security requirements of these relatively new systems and the applications they support are still being understood. As a result, the security infrastructure available in current Smartphone operating systems is largely underdeveloped. In this paper, we consider the security requirements of Smartphone applications and augment the existing Android operating system with a framework to meet them. We present Secure Application Interaction (Saint), a modified infrastructure that governs install-time permission assignment and their run-time use as dictated by application provider policy. An in-depth description of the semantics of application policy is presented. The architecture and technical detail of Saint is given, and areas for extension, optimization, and improvement explored. As we show through concrete example, Saint provides necessary utility for applications to assert and control the security decisions on the platform.

A. ANDROID

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as Smartphone's and tablets. Android is designed with a multi-layered security infrastructure, which provides developers a secure architecture to design their applications. Android applications are sandboxed and run inside the Dalvik Virtual Machine, which isolates each application's processes and data. Each application is assigned a user ID (UID) with assigned privileges, and is given a dedicated part of the file system for its own data. While applications have limited access to device resources, developers can explicitly request access for their applications through the Android permission system.

Permission System:

The Android permission system controls which application has the privilege of accessing certain device resources and data. Application developers that need access to protected Android APIs need to specify the permissions they need in the AndroidManifest.xml file which, if inaccurately assigned, can increase the risks of exposing the users' data and increase the impact of a bug or vulnerability. Each application declares the permissions listed in its AndroidManifest.xml file at the time of installation, and users have to either grant all the requested permissions to proceed with the installation, or cancel the installation. The Android permission system does not allow users to grant or deny only some of the requested permissions, which limits the user's control of application's accessibility.

Android Application Components:

Every Android application is composed of four essential components: Activities, Services, Content Providers, and Broadcast Receivers. An Activity defines an application's user interface. The Service component is designed to be used for background processing. The Content Provider component acts as a global instance for a particular application, so that all applications on the device can use it. It stores and manages a shared set of data belonging to the owner application using a relational database interface. Finally the Broadcast Receiver component acts as mailboxes that allows applications to register for system or application events. All registered receivers for an event will be notified by Android once this event happens.

Intents:

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.Intents are objects of the android.content.Intent type. Your code can send them to the Android system defining the components you are targeting. For example, via the startActivity() method you can define that the intent should be used to start an activity. An intent can contain data via a Bundle. This data can be used by the receiving component.

B. LOCATION POSITIONING METHODS

To retrieve the location of the device we rely on WI-FI based positioning technique. In addition to these techniques ,we can also collect location data retrieved from GPS and cellular networks for situations where there is no Wi-Fi coverage in the areas of interest. Moreover, we use public GPS location data in defining policy contexts for areas that have not been previously visited. In this section, we introduce a brief description on methods used in smart phones for locating the devices.

GPS:

The Global Position System (GPS) is a positioning tool available in most smart phones which uses data signals from satellites to compute the position of the device. Data received from satellites contains the time stamp of sending, the orbital information, and the position of satellites. With at least three different satellite signals, the GPS uses the trilateration method to calculate the device's location by measuring the satellite signals time difference or their received signal strengths. The location information provided from the GPS includes the latitude, longitude, altitude, and time. The accuracy of this method is estimated to be in the range of 50 to 100 meters.

Cellular Triangulation:

Cellular triangulation (cell ID) is another positioning approach based on cellular technology. It refers to tracking a mobile phone's current location using radio towers. Through contacting every nearby antenna tower, cellular triangulation can make a measurement of how far away the cellular mobile device is based on the signal it is transmitting. This is done by measuring signal strength and the round-trip signal time. Once this distance is calculated, finer approximations can be done by interpolating transmitting signals between adjacent radio towers. The accuracy of this method varies according to the density of cell towers existing in an area. For instance, the accuracy in large cities can reach up to 10 meters due to the high density of cellular towers that a device could be in contact.

Wi-Fi Positioning Method:

Many Smartphone devices nowadays rely on Wi-Fi-based positioning techniques due to its efficiency in computing the location of a device especially in places when GPS and cellular signals are weak or unavailable . These techniques are based on comparing the device's received Wi-Fi access points (AP) with database fingerprints containing Wi-Fi access points with known geographic locations . In our work, smart phone users will define their own database of Wi-Fi APs which only includes the user's areas of interest. In addition to the Wi-Fi APs, we also store the Received Signal Strength Indication (RSSI) of each AP to enhance the positing accuracy of the device, which in our case, needed to differentiate between nearby sub-areas.

III. DESIGN

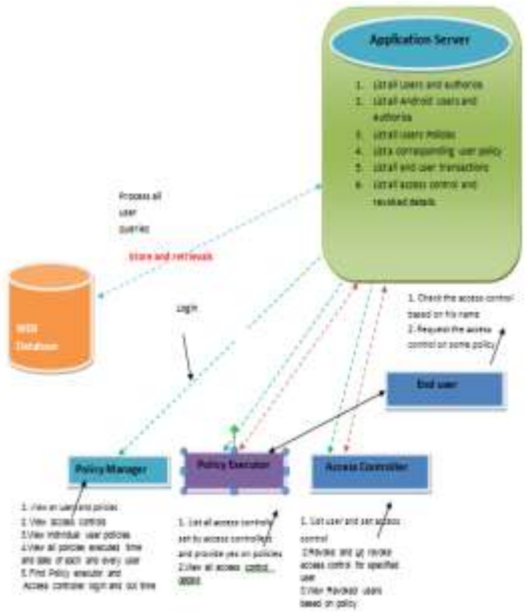
Application Server
1. List all users and authorize
2. List all Android users and authorize
3. List all users Policies
4. List a corresponding user policy
5. List all end user transactions
6. List all access control and revoked details
Process all user queries
Store and retrievals
WEB Database
Login
1. Check the access control based on file name
2. Request the access control on some policy
End user
Policy Manager
Policy Executor
Access Controller
1. View all users and policies
2. View access control
3. View individual user policies
4. View all policies executed time and date of each and every user
5. Find Policy executor and Access controller login and out time
1. List all access control set by access controller and provide yes on policies
2. View all access control details
1. List user and set access control
2. Revoke and up revoke access control for specified user
3. View Revoked users based on policy

Fig 2: Proposed Architecture

A. MODULES

Context Provider :

The Context Provider (CP) collects the physical location parameters (GPS, Cell IDs, Wi-Fi parameters) through the device sensors and stores them in its own database, linking each physical location to a user-defined logical location. It also verifies and updates those parameters whenever the device is re-located.

Access Controller :

The Access Controller (AC) controls the authorizations of applications and prevents unauthorized usage of device resources or services. Even though the Android OS has its own permission control system that checks if an application has privileges to request resources or services, the AC complements this system with more control methods and specific fine-grained control permissions that better reflect the application capabilities and narrow down its accessibility to resources. The AC enhances the security of the device system since the existing Android system has some permissions that, once granted to applications, may give applications more accessibility than they need, which malicious code can take advantage of.

Policy Manager :

The Policy Manager (PM) represents the interface used to create policies, mainly assigning application restrictions to contexts. It mainly gives control to the user to configure which resources and services are accessible by applications at the given context provided by the CP. As an example, the user through the PM can create a policy to enable location services only when the user is at work during weekdays between 8 am and 5 pm.

Policy Executor :

The Policy Executor (PE) enforces device restrictions by comparing the device's context with the configured policies. Once an application requests access to a resource or service, the PE checks the user-configured restrictions set at the PM to either grant to deny access to the application request. The PE acts as a policy enforcement by sending the authorization information to the AC to handle application requests, and is also responsible to resolve policy conflicts and apply the most strict restrictions.

B. INPUT DESIGN

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations. This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design. Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided within an option to select an appropriate input from various alternatives related to the field in certain cases. Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

C. OUTPUT DESIGN

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only. The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

IV. IMPLEMENTATION

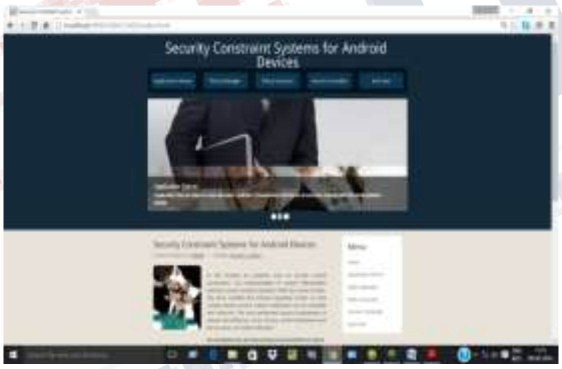

Fig 3: Home Screen

In Fig 3 we see all the modules that are present in the system and we can access all those modules by signing into them.

Fig 4: User Policy

In Fig 4 we can view all the user policies and their access control.we can also see the policy id ,name and accesscontrol.

Fig 5: Authorization

In Fig 5 it is a part of application server module in this we view all the users and we can give authorization to the users.

Fig 6: Access Control

Fig 7: Users Policies Data

Fig 6 is a part of application module in this it will view all users name and controls of the users that he has requested.

In Fig 7 we can view system peripheral state policies, multitasking, intercommunication policies, and user security policies.

In Fig 8 we can view the revoked user data like id,policy control,accesss control,date and status. In this step if the status of the new user is accepted then only his policies will be executed.

Fig 8: Revoked User

Fig 9: End User

In Fig 9 we all the new users can fill the form and request for approval of policies requested by the user.

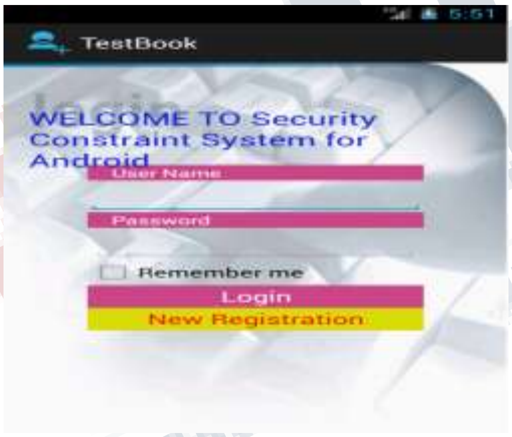

Fig 10: Android Welcome Page

In Fig 10 the user need to get registered with the data then he need to enter the registered username and password to logged in.After login we can access the permissions.

V. CONCLUSION

In this work, we proposed a modified version of the Android OS supporting context-based access control policies. These policies restrict applications from accessing specific data and/or resources based on the user context. The restrictions specified in a policy are automatically applied as soon as the user device matches the pre-defined context associated with the policy. Our experimental results show the effectiveness of these policies on the Android system and applications, and the accuracy in locating the device within a user-defined context.

Our approach requires users to configure their own set of policies; the difficulty of setting up these configurations require the same expertise needed to inspect application permissions listed at installation time. However we plan to extend our approach to give network administrators of organizations the same capabilities once a mobile device connects to their network. In this way, network administrators are able to block malicious application accesses to resources and services that may affect the security of their network. We believe that such an approach is critical for assuring security of corporate networks when organizations allow users to “bring their own devices”.

REFERENCES

[1]. A. Kushwaha and V. Kushwaha, “Location based services using android mobile operating system,” Int. J. Adv. Eng. Technol., vol. 1, no. 1, pp. 14–20, 2011.

[2]. D. Kulkarni, “Context-aware role-based access control in pervasive computing systems,” in Proc. 13th ACM Symp. Access Control Models Technol., 2008, pp. 113–122.

[3]. W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones,” in Proc. 9th USENIX Conf. Oper. Syst. Des. Implementation, 2010, pp. 1–6.

[4]. J. Leyden, (Apr. 2013). Your phone may not be spying on you now—but it soon will be. [Online].Available:http://www.theregister.co.uk/2013/04/24/kaspersky_mobile_malware_infose

[5]. R. Templeman, Z. Rahman, D. J. Crandall, and A. Kapadia, “Placeraider: Virtual theft in physical spaces with smartphones,” in Proc. 20th Annual Netw. Distrib. Syst. Security Symp. (NDSS), Feb. 2013.

[6]. R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, “Soundcomber: A stealthy and context-aware sound Trojan for smartphones,” in Proc. 18th Annu. Netw. Distrib. Syst. Security Symp., Feb. 2011, pp. 17–33.

[7]. Erlingsson, F. Schneider, “IRM enforcement of java stack inspection,” in Proc. IEEE Symp. Secur. Privacy, 2000, pp. 246–255.

[8]. D. Evans and A. Twyman, “Flexible policy-directed code safety,” in Proc. IEEE Symp. Secur. Privacy, 1999, pp. 32–45.