Policy Based Framework for Securing Enterprise Data on Smartphone

Snehal Desai[1], Sakshi Sachdev[2] , Mehul Jain[3] , Shashikant Hake[4]
Department of Computer Engineering, Sinhgad College of Engineering
University of Pune, India

***Abstract* - There is increase in threat to Enterprise data resident on employee's smartphone, as android has Inter Component Communication and Inter Process Communication as a part of core application functionality. Due to this sensitive data can be stole by applications. In this paper Inter Component Communication is controlled by enforcing policies in android Middleware. Static Enterprise data is also protected by modifying android Linux kernel and granting or denying application to access static data resident on users smartphones according to policies evaluated in android middleware.**

Keywords – security, enterprise, smartphone, android, policy, personal.

I. INTRODUCTION

Android OS is growing its popularity with its expanding market share. In Android anyone who is registered as Android developer can publish his application on android market, due to this malicious applications can be uploaded on market store. And also at install time of any application user has to follow 'ALL OR NOTHING STRATEGY' i.e. if user wants to install that app then user can either allow all permissions or has to give up the ability to install that application. From this it is clear that security infrastructure of smartphone is underdeveloped [11].

In this paper, we consider security requirements of smartphone application and protect Enterprise data from personal applications by controlling ICC between applications by implementing policies in android middleware [2].

Main contribution of this paper can be summarized as:
1. Securing static Enterprise data on smartphones.
2. Framework for securing dynamic information flow.

Section II is related to security mechanism which is used by android today. This section also describes how applications can be identified in android, security labels in android and role of reference monitor in android middleware to control ICC between applications. Section III describes previous frameworks that were built to enhance the security mechanism in android. This section also describes advantages of this framework over previous frameworks. Section IV contains the implementation details of this system. It also represents the Mathematical Model that describes the input, output functionalities along with the success and failure cases. Section V contains expected results of this system. Section VI concludes this paper and describes future work.

II. BACKGROUND

In this we briefly described android's security mechanism.

1. Application Sandboxing:
It is means of separating applications from each other. In this each application is assigned a separate uid and they run as separate process.

2. Application Signing:
Developers have to sign the application code with self-certified key. Applications signed with same certificate shares same uid i.e. they run in same sandbox.

3. Android Permission Framework:
This is provided by android's middleware. Granted permissions in terms of security labels are assigned to application sandbox and inherited by application components. These permissions are also included in manifest file of applications. Reference Monitor in middleware checks permissions at runtime and control ICC between applications.

III. RELATED WORK

There exist many proposals/frameworks for Android's application security such as TaintDroid, Apex, XMandroid, Saint, Quire.

1. XMandroid Framework:
This Framework which extends the Android Reference monitor, in this default reference monitor verifies whether communication link complies to security rules defined in system policy of XMandroid. However XMandroid does consider escalation attacks at kernel level [5].

2. I-ARM-DROID:
This framework enforces security policies on Application. This approach embeds reference monitor in applications by rewriting Dalvik byte code. But this approach needs to modify each application while in our approach we are modifying Android framework, so no need to modify each application [10].

3. Android Permission Extension (Apex) framework:
This policy framework allows a user to selectively grant permissions to applications and impose constraints on the usage of resources. Apex also describes an extended package installer that allows the user to set these constraints through an

easy-to-use interface. Apex is not able to impose constraints between App communications [6].

4. TaintDroid:

It tracks private-sensitive data through third-party applications. It monitors how third-party applications access and modifies user's personal data. It automatically labels data coming from sensitive sources and applies taints as sensitive data propagates through Interprocess communication mediums like files, message passing etc. When tainted data is transferred over the internet, TaintDroid logs the data's labels, the applications responsible for transmitting the data and also destination of the data. So this feedback helps users tracking what mobile applications are doing [7].

IV. TECHNOLOGY

1. Static Enterprise data:

Data which are generated by enterprise applications and Stored in SD card.For example when any attachments are downloaded from enterprise application and after getting stored in internal memory or SD card of smartphone, application does not have control over downloaded data. This data is crucial and should not be leaked. Sometimes data exported from enterprise applications is in encrypted form; however other applications can decrypt it.

This framework is securing static enterprise data by enforcing policies.

2. Concurrent Execution of Applications:

Each Application in android has four components i.e. Activity, Broadcast Receiver, Content provider, Service [8]. And components of each application can communicate with components of other application on the basis of access permission labels, present in manifest file of each application. These permission labels are not sufficient for securing enterprise data as there is still information leakage when any enterprise application calls services of any personal application.

This framework enforces policies in android middleware which will evaluate policies as well as check permissions for ICC. Next question arises how application and data can be classified.

3. a. Application Classification:

There are multiple ways to classify application i.e. Developer signature, market source, etc.

3. b. Data Classification:

Data generated by enterprise application is classified as enterprise data that is based on applications Package name.

4. Policies:

1. Personal Application cannot access the static enterprise data resident on employee's smartphone i.e. if both static data as well as application are enterprise, then application can access the static enterprise data.

2. Components of applications can communicate with each Other if both applications are enterprise, and both of them are personal.

5. Previous Workflow:

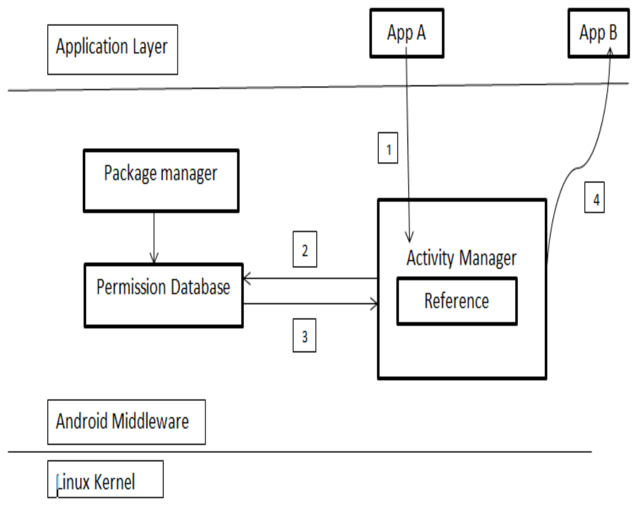

Figure 1. Original Workflow

Reference monitor in android middleware controls ICC by intercepting ICC calls, obtains permissions from permission database and checks permissions. Based on result,allow or deny ICC call. Main drawback of android framework is, it only checks the permissions and does not bother about the type of application.

6. WORKFLOW TO PROTECT STATIC DATA:

As previously described, this framework secures static enterprise data. Static enterprise data can be protected by modifying Linux kernel as well as android middleware.

1. When any application request for accessing data stored on SD card/phone memory, then taint tracker module intercept file open system call and fetch uid of requesting application and file id of requested file.

Taint Tracker: New module added in Linux kernel, which maintains taint set. Taint set contains the uids of each application which had already opened that file.

2. Taint tracker search for uid of requesting application in taint set.

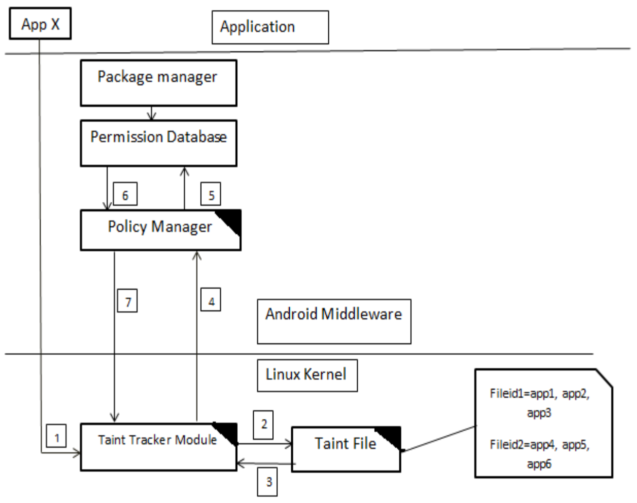

Figure 2. Static work flow

3. If uid of requesting application is found in taint set then access is permitted to requesting application.
4. If not found then it contacts policy manager.
5. Policy manager fetch package names from package manager and evaluate policies.
6. Policy manager gives the result back to taint tracker.
7. Taint tracker allow or deny access to application based on evaluated results of policy manager.

7. Workflow to Control ICC:

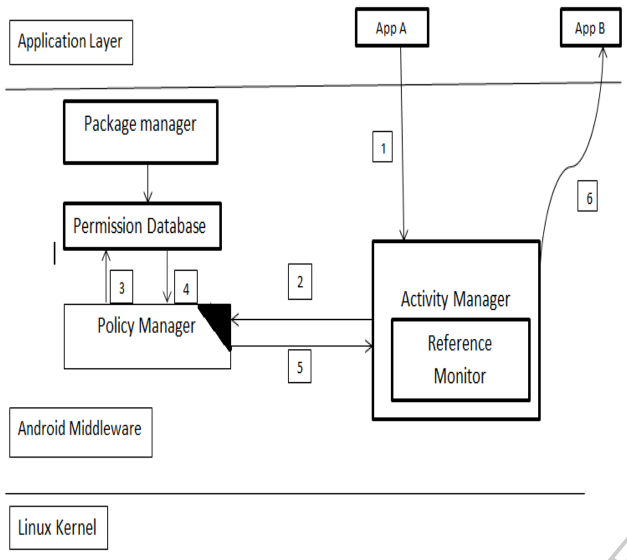

Figure 3. ICC flow control

1. If application A sends request for ICC to application B then ICC call will be intercepted by reference monitor.
2. Reference monitor is redirected to policy manager.
3. Policy manager request permissions from permission database.
4. Permission database gives permissions to policy manager and policy manager accordingly evaluate policies.
5. Policy manager gives result of evaluated policies to reference monitor.
6. Reference monitor checks permission and according to results of evaluated policies and checked permission it will allow or deny ICC.

VI. EXPECTED RESULT

Expected results of this policy based framework are that it will deny any personal application to access enterprise data. It also denies personal application to communicate with enterprise application.

Success Scenario:
Enterprise data is secured by denying access from other applications.

Failure Scenario:
Other applications become successful in stealing enterprise information.

VII. FUTURE WORK AND CONCLUSION

In this paper we have extended the android security model, whose main concern was to secure static enterprise data and to prevent runtime information leakage by enforcing policies in android middleware and modifying linux kernel. Here we are more concentrating on securing enterprise data from other applications, in future we could handover this decision to user, whether to allow or deny any application to access particular data.

REFERENCES

1. Hammad Banuri, Masoom Alam, et. al. "An Android runtime security policy enforcement framework", IEEE paper, 2011.
2. Erika Chin Adrienne, et. al. "Analyzing Inter-Application Communication in Android", IEEE paper, 2011.
3. Prajit Kumar Das, Dibyajyoti Ghosh, Anupam Joshi and Tim Finin "Energy efficient semantic context model for managing privacy on smartphones. "
4. Alastair R. Beresford, Andrew Rice, et. al. "MockDroid: trading privacy for application functionality on smartphones", IEEE paper, 2011.
5. Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, Ahmad-Reza Sadeghi "XManDroid: A New Android Evolution to Mitigate Privilege Escalation Attacks. "
6. M. Nauman, S. Khan, and X. Zhang, "Apex: extending Android permission model and enforcement with user-defined runtime constraints," in Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. ACM, 2010, pp. 328–332.
7. W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel, and A. Sheth, TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in Proceedings of OSDI, 2010.
8. Semantically Rich Application-Centric Security in Android Machigar Ongtang, Stephen McLaughlin, William Enck and Patrick McDaniel Department of Computer Science and Engineering The Pennsylvania State University, University Park, PA 16802.
9. Sven Bugiel, Lucas Davi, Alexandra Dmitrienko,Thomas Fischer,Ahmad-Reza Sadeghi, Bhargava Shastry,"Towards Taming Privilege-Escalation Attacks on Android" falexandra.dmitrienko,ahmad.sadeghi,bhargava.shastryg@sit.fraunhofer.de.
10. Benjamin Davis, Ben Sanders_y, Armen Khodaverdian,and Hao Chen_," I-ARM-Droid: A Rewriting Framework for In-App Reference Monitors for Android Applications" _University of California, Davis fbendavis, bmsanders, aekhodaverdian.
11. Palanivel Kodeswaran, Vikrant Nandakumar, Shalini Kapoor , Pavan Kamaraju, Anupam Joshi, Sougata Mukherjea ,"Securing Enterprise Data on Smartphones using Run Time Information Flow Control" IBM Research India Bangalore, India.