POSTER: API-Level Multi-policy Access Control Enforcement for Android Middleware

Dongdong Tian[1,2], Xiaohong Li[1,2(✉)], Jing Hu[1,2], Guangquan Xu[1,2], and Zhiyong Feng[1,2]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
{tianddong,xiaohongli,mavis_huhu,losin,zyfeng}@tju.edu.cn
[2] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin University, Tianjin, China

Abstract. This paper proposes *MpDroid*, an API-level multi-policy access control enforcement based on the 'Rule Set Based Access Control' (RSBAC) framework. In the *MpDroid*, we monitor and manage resources, services and Android inter-component communication (ICC) based on multiple policies mechanism, so as to restrict the applications access to the sensitive APIs and prevent privilege escalation attacks. When installing an application, we build the mapping relationships between sensitive APIs and the application capability. Each rule in the user-defined and context policies is regarded as a limitation of the application capability. Moreover, system policy is used for matching the illegal ICC communications. Experimental results showed that we can realize the API-level access control for Android middleware, and prevent the illegal ICC communication on the Android 4.1.4.

Keywords: Android middleware · Multi-policy · Permission re-delegation · Inter-component communication · Privilege escalation attacks

1 Introduction

Apex [1] allows to selectively grant permissions at install time, and defines constrains at runtime. MockDroid [2] allows to provide fake or 'mock' data to applications by the user-defined policies. CRePE[3] designs a fine-grained framework by introducing the context policy. Saint [4] proposals a novelty framework that developers design policies based on application requirement. Xmandroid [5] deals with privilege escalation attacks based on the system policy that calling and callee permissions are matched. However, None of them can design a flexible and security framework that comprehensively solves the problem of Android framework.

In this paper, we expand the android framework layer based on the RSBAC. We monitor and manage resources, services and Android inter-component communication (ICC) based on multiple policies mechanism, so as to restrict the applications access to the sensitive APIs and prevent privilege escalation attacks. Our *MpDroid* is integrated into the Android system, which can be used to realize the permission management. It is applied to manage android market applications in this paper.

B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 559–562, 2015.
DOI: 10.1007/978-3-319-28865-9_30

2 MpDroid Architecture

The *MpDroid* Architecture was inspired by RSBAC [6]. In RABAC framework, users should define some policies for constraining capabilities of subject and object. In our framework, AEC is responsible for obtaining the request of the subject and analysis the object type. As the subject always is the application and the object type belongs to resource, service, component or application. Then AEC transfers the request and the object property to the ADM, in which the states of subject and object are loaded. In this paper, we name these states as Application State (AS). AS are three-tuple $S(type, S_{cap}, S_{compo})$, where type is the application type, S_{cap} is the set of application capabilities and S_{compo} is the set of application components.

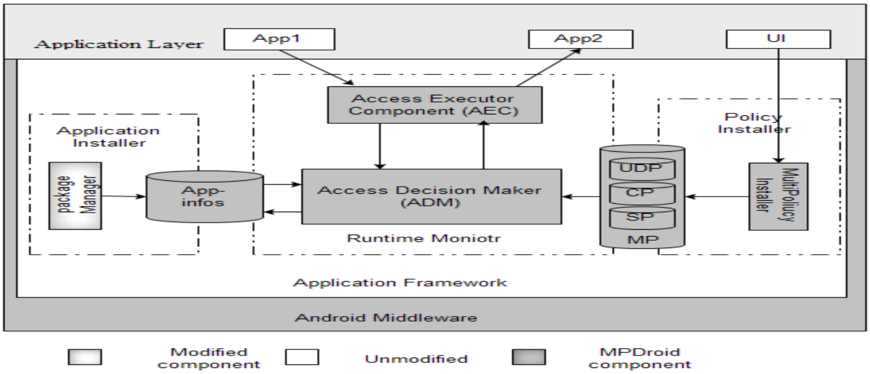

Fig. 1. MpDroid Architecture

3 Experiment

The malicious samples are from VirusTotal Malware Intelligence Service [7], Android Malware Genome Project [8] (Totally 118 applications). The benign samples are from the Google Play application market of top 50 popular software. The experiments can be classified as following:

Table 1. The experiment of access API

Experiment samples	The number of access sensitive API	Access entities of system component	The number of rules for access control
Known Attacks[32,33]	623	Location/Bluetooth Manager, Telephony/SMS Manager, Calendar/ Contact Content Provider, Internet	623 rules in UDP, 977 rules in CP
Walk and Text	5	Contact Content Provider, SMS/Telephony Manager	5 rules in UDP, 10 rules in CP
What's App	9	Contact Content Provider, SMS/Telephony Manager, Internet	9 rules in UDP, 15 rules in CP
Twitter	7	Contact Content Provider, SMS/Telephony Manager	7 rules in UDP, 12 rules in CP

Table 2. The experiment of ICC communication

Experimental Samples	Escalation type	($S_{callingUid}$, $S_{calleeUid}$,Policies)
Malicious contacts manager (READ_CONTACTS) and malicious wallpaper (INTERNET)	Colluding applications	$S_{calleeUid}$.type=untrust_app, $S_{callingUid}$.S_{cap} =read_contacts $S_{calleeUid}$.type = untrust_app $S_{callingUid}$.S_{cap} = Internet Policy = SP
Malicious location manager (ACCESS_FINE_LOCATION) and malicious wall-paper (INTERNET).	Colluding applications	$S_{callingUid}$.type $S_{calleeUid}$.type =untrust_app, $S_{callingUid}$.S_{cap} =access_fine_location = untrust_app $S_{callingUid}$.S_{cap} =Internet Policy = SP
Malicious app (No CALL_PHONE) and vulnerable dialer [17]	Confused deputy attacks	$S_{calleeUid}$.type =untrust_app, $S_{callingUid}$.S_{cap} =NULL $S_{calleeUid}$.type = system_app $S_{callingUid}$.S_{cap} = send_sms Policy = SP
Malicious contact manager (READ_CONTACT) and vulnerable SMS sender (SEND_SMS).	Confused deputy attacks	$S_{calleeUid}$.type =untrust_app, $S_{callingUid}$.S_{cap} =read_contacts $S_{calleeUid}$.type = system_app $S_{callingUid}$.S_{cap} = send_sms Policy = SP

The subject which sends request to the Service/Providers are tagged by AEC. Our policy serves as a kind of firewall, making it much more difficult for applications to use the default permission to access the sensitive data. We test experiment samples by applying UDP and CP to the system to prevent the application access to the sensitive API. For example, the application, Walk and Text, gains the telephone number and device id, and uploads that to the remote server. The application accesses 5 sensitive API. In the experiment, we success in managing every behavior using policies. The ICC communication can be defined as the tuple $(S_{callingUid}, S_{calleeUid}, Policies)$ .The *MPDroid* runtime control is achieved by mapping the AS and policy to the parameters. We use the tuple $(S_{callingUid}, S_{calleeUid}, Policies)$ to realize access control (Table 2). Attacks targeting confused deputies in system component are tackled by the system policy. By assigned application types, we can address the ICC between colluding applications.

Table 3. The time consuming comparison of access API

	Sample Numbers	Number of access the sensitive API	Average time consumes(ms)
The original API access	50	269	0.149
MpDroid API access	50	281	0.399

Table 4. The time consuming comparison of ICC

	ICC call times	Average time	Std. Dev(ms)
The Original Reference Monitor	80721	0.168	18.932
MpDroid ICC	87453	6.334	45.128

As Table 3 shows, When running the applications in the original system, the results is nearly 0.153ms. When running the applications in the *MpDroid*-based Android system, the average time is 0.399ms. Table 4 lists our performance results. In total 80721 ICC calls occurred during the testing. The average runtime for original Reference Monitor time is 0.168ms, and the MpDroid ICC time is 6.334ms.

4 Conclusions

In this paper, we propose a multi-policy access control enforcement *MpDroid* based on RSBAC framework. Multiple policies makes our framework more efficient to resist the diverse attacks. The experiment results shows that we can realize the API-level access control for Android middleware, and prevent the illegal ICC communication on the Android 4.1.4. The system policy time consuming is 6.334ms. However, from the experiment, we learn some collusion attacks that can not be fully tackled by system policy. Besides, we hope we can make more efficiency policy to deal with more attacks.

Acknowledgments. This work has partially been sponsored by the National Science Foundation of China (No. 91118003, 61272106, 61003080) and 985 funds of Tianjin University, Tianjin Research Program of Application Foundation and Advanced Technology under grant No. 15JCYBJC15700.

References

1. Nauman, M., Khan, S., Zhang, X.: 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, ASIACCS 2010 (2010)
2. Mueller, K., Butler, K.: Flex-P: flexible Android permissions. In: IEEE Symposium on Security and Privacy, Poster Session (2011)
3. Conti, M., Nguyen, V.T.N., Crispo, B.: CRePE: context-related policy enforcement for Android. In: Tsudik, G., Magliveras, S., Ilić, I., Burmester, M. (eds.) ISC 2010. LNCS, vol. 6531, pp. 331–345. Springer, Heidelberg (2011)
4. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically rich application-centric security in Android. In: IEEE Computer Society, ACSAC 2009 (2009)
5. Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.: XManDroid: a new Android evolution to mitigate privilege escalation attacks. Technische Universität Darmstadt; 2011a [Technical Report; Technical Report TR-2011-04]
6. Ott, A., Fischer-Hübner, S.: The 'rule set based access control'(RSBAC) framework for linux. In: Proceedings of the 8th International Linux Kongress (2001)
7. VirusTotal Malware Intelligence Services. https://secure.vtmis.com/vtmis/
8. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: S&P. IEEE Computer Society (2012)