POSTER: Role Based Access Control For Android (RBACA)

Felix Rohrer, Yuting Zhang, Lou Chitkushev, Tanya Zlateva
Computer Science Department
Metropolitan College
Boston University, MA USA
felixro,danazh,ltc,zlateva@bu.edu

Abstract

Android as an open platform dominates the booming mobile market. However its permission mechanism is inflexible and often results in over-privileged applications. This in turn creates severe security issues. In this paper, we propose a Role-Based Access Control for Android (RBACA) framework to address this problem. We show how the widely used Role-Based Access Control (RBAC) approach can be applied to Android mobile systems in order to mitigate the security risks caused by over-privileged applications. We explain how single users as well as large corporations can make use of RBACA. An initial prototype of RBACA has been implemented and its functionality can be demonstrated.

1 Introduction

Smartphones are an emerging platform for both personal and business applications. IDC data shows that 144.9 million smartphones were sold worldwide in Q1 2012 with 59% of them being Android phones. An important part of Android's security framework is its permission model which is used to access sensitive resources (GPS, WiFi) and functions (sending an SMS). At install time, each application requests its required permissions. Once installed, the applications will never again ask for access confirmation. Many developers handle these permissions with little care leaving applications vulnerable. Malicious applications can launch a permission re-delegation attack by cheating another application into performing a certain job, like sending an SMS to a premium number. Over-privileged applications increase the risk of such an attack, since more permissions are exposed to malicious applications. Android's permission system in particular has the following shortcomings: first, the requested permissions of an application cannot selectively be granted or denied. Second, the permission assignment can only happen during the installation time and third, the permissions cannot be changed or restricted after installation. Since permissions are often required for a short period of time only, many applications are over-privileged for most of their running time. To address these problems, we propose a Role Based Access Control for Android (RBACA) framework. Role Based Access Control (RBAC) is an approach to restrict resource access to authorized users based on their roles. Aiming to support the Principle of Least Privilege, our RBACA enables the users to work in different roles depending on their current activity. This minimizes the exposure of potentially dangerous permissions. We show how RBACA can be used by both non-expert users as well professionals such as the IT department of a large corporation.

Much research has been performed to solve the above mentioned problems. A.P. Khandavilli proposed a mobile role based access control system [1]. Unlike our solution, his solution does not function on standalone devices. Apex allows to select a subset of the requested permissions at install time [2]. However, a previously rejected permissions will never be granted again, effectively crippling the application. Our solution allows a more flexible way of permitting and rejecting permissions by grouping them into different roles which can be changed depending on the current activity, user and context. Ongtang et al. introduced a security framework called Saint that governs fine-grained access control at run-time [3] by analyzing and restricting the communication channels between applications. Saint's policy does not consider single applications and therefore does not provide the type of access control we propose in this paper. Yee et al. proposed a context-related role-based access control mechanism [4]. But roles are defined only on a device level, while our solution allows a role to be defined on a device level as well as on an application level. In addition, by using an external management system, our solution provides a simple configuration mechanism that can distribute a RBACA policy to many mobile devices at the same time. Furthermore, none of the mentioned papers support the logging of sensitive activities that we aim to provide.

2 Our proposed Solution - RBACA

The essential components of an RBAC system are the definition of roles and how they are mapped with users and permissions. In Android, each application App_i is assigned a unique user id uid_i, and a fixed set of permissions P_i. P_i is the maximum set of permissions of App_i. To provide dynamic and fine-grained access control, our RBACA model supports multiple roles for each application. A subset of the permissions for that application is assigned to each role. We denote a set of roles for App_i as $R_i = \{r_i^j\}$, where a role r_i^j is assigned to a permission subset $P_i^j \subseteq P_i$. To simplify role creation, RBACA allows a created role r^j to be associated with multiple applications, or all applications as a system wide role, with a permission set P^j assigned. Then it automatically generates a specific role r_i^j for each associated App_i with a permission set as $P_i^j = P^j \bigcap P_i$. Each application has at least one default role. That is, $|R_i| \geq 1$, and we can assume r_i^0 is the default role for App_i to run.

Based on this primitive RBACA model, a more comprehensive model is under design to describe how roles can be switched manually or dynamically. For example, manual role switching may be triggered when an application requests a permission that is not part of the current role. The user can be informed and could choose to manually switch to another role. In general, we can create low-privileged default roles that do not contain potentially dangerous permissions such as SEND_SMS. If a malicious application tries to perform a permission re-delegation attack, the attacked application will by default be in a low-privileged role denying the request. Automatic role switching can be triggered through an event (such as user login) or context change (such as time, location). For example, a company may provide pre-configured smartphones running the RBACA framework to its employees. To mitigate the risk of data theft, the IT department can create a system-wide role that disables the CAMERA and WRITE_EXTERNAL_STORAGE permission if the mobile device is located on or near company premises.

Our RBACA prototype currently consists of three components. The PolicyEnforcer (PE) intercepts permission requests after they have been granted by Androids own reference monitor. The PE then consults the current role of the requesting application and permits or rejects the request. The second component is the LocalPolicy (LP) application which provides a password protected interface to the user or administrator in order to create, modify and delete roles and users. The third (optional) component is the ExternalPolicy (EP) which is located on an external server and offers similar configuration options as the LP but can distribute these configurations to any number of connected Android devices, which is essential for large environments such as corporations. We plan to offer an API to developers, allowing them to incorporate roles directly into their code. This can be useful for multi-user applications to map roles to users, as well as to restrict access to certain functionality (i.e. for a trial version). We furthermore plan to incorporate a logging mechanism to help analyze suspicious behavior. The log can either be stored locally or periodically sent to a remote server and will provide timestamped information about application execution, sensitive function access and the involved roles and users. One major concern that needs to be addressed is how ordinary users can create roles without expert knowledge. A possible solution is to automatically filter out potentially dangerous permissions and create separate roles for them. Furthermore, a corporation may disallow manual role changes all together, solely relying on role changes based on the mobile device's context.

A running version of RBACA with basic functionalities is completed and ready to be deployed on any Android platform for demonstration. Based on the current primitive RBACA model, a more comprehensive model is under design to include the rules of dynamic role switching. Currently we are adding context-based role switching features into our prototype implementation. The contribution of our proposed solution will be as following:

- Propose a novel approach of integrating role based access control with Android's permission model to enhance its security
- Mitigate risk of malicious applications executing sensitive functions in the background and reduce the risk of permission re-delegation attacks
- Provide easy configuration and management for both single devices and multiple devices in a cooperative environment
- Provide interfaces to both users and developers to take advantage of RBACA.
- Logging mechanism of sensitive functions and role changes to help analyze suspicious behavior

References

[1] A. Khandavilli. A mobile role based access control system using identity based encryption with non-interactive zero knowledge proof of authentication, 2012.

[2] 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*, ASIACCS '10, pages 328–332, New York, NY, USA, 2010. ACM.

[3] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel. Semantically rich application-centric security in android. *Journal of Security and Communication Network*, 2011.

[4] T. T. W. Yee and N. Thein. Leveraging access control mechanism of android smartphone using context-related role-based access control model. In *Networked Computing and Advanced Information Management (NCM), 2011 7th International Conference on*, pages 54 –61, June 2011.