An administration console for the CSAP system

Fredj Dridi, Björn Muschall and Günther Pernul

Department of Information Systems, University of Regensburg
Universitätsstraße 31, D-93053 Regensburg, Germany

Abstract. Our group is involved in the European funded Webocracy project in which an e-government system called *Webocrat* has been designed and implemented. Our responsibility is to provide mechanisms guaranteeing secure and reliable access to Webocrat. According to the security requirements from the city councils involved in the project we have implemented a generic and extensible security architecture called CSAP. In this paper we will shortly describe the CSAP architecture and a web-based administration console supporting the administration of RBAC-based authorization and access control services.

1 Introduction

Webocracy [1] [2, 4] an example of innovative use of state-of-the-art web technologies in order to support direct participation of citizens in democratic processes. In this project an e-government system called the *Webocrat* is implemented, which is modularly designed and composed of several modules (i.e. knowledge management, opinion polling, discussion management). Usually, such systems involve a huge number of heterogeneous users working with the systems under different rights and obligations. In order to form the basis for trust for the whole Webocrat system we implemented a security architecture called CSAP. The purpose of CSAP is to provide appropriate security services for the whole system [1]. Among other services, CSAP provides an implementation of a role-based access control subsystem that conforms to the *Core RBAC* model as defined in the proposed NIST standard [3]. Because of the huge amount of users and the diversity of their requirements the administration of the CSAP system becomes crucial. For managing roles, users and permissions we designed a flexible web-based administration console, called CAC (CSAP administration console).

2 The security architecture CSAP

Systems, like the Webocrat are used in environments in which huge numbers of heterogeneous users (citizens, politicians, government employees, business men, etc.) usually want to share and access large numbers of documents in a controlled way. Some of the documents may contain sensitive information and consequently must not be disclosed to the general public. The purpose of CSAP is to form the basis for trust for the whole system offering "practical and consistent" security by providing services for (a) *user identification and authentication*, (b) *access control and authorization*, (c) *auditing*, and (d) *session management*, which is needed in a Web-based environment to keep track of global security information and to authenticate users only once during a single session.

Within the Webocracy project CSAP will be used by several user partners with differing requirements. For example, the authentication service can be alternatively implemented by

[1] The Webocracy Project is supported by the European Commission under IST-1999-20364. The content of this publication is the sole responsibility of the authors and may not represent the view of the European Commission. The helpful comments and stimulating discussions of our project partners are gratefully appreciated.

using a password scheme, or by using X.509 certificates, or even by a using different techniques. For the authorization service user partners may also have differing requirements. As a consequence CSAP had to offer different security services those are completely substitutable [1].

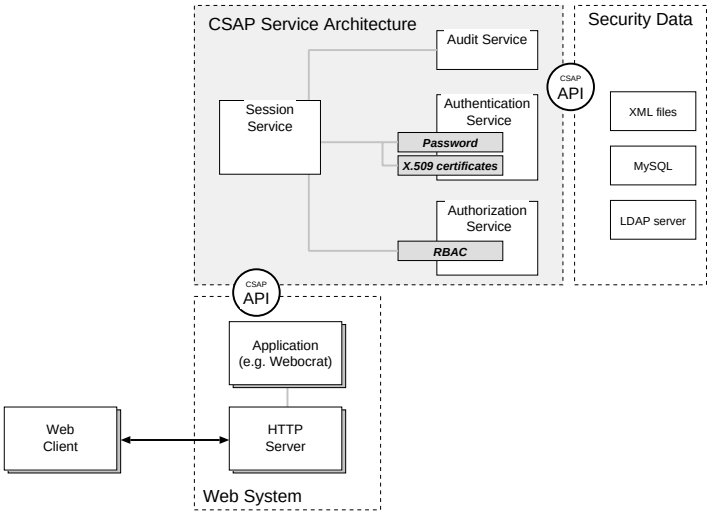

Fig. 1. CSAP Service Architecture

In figure 1 the conceptual architecture of CSAP is displayed. CSAP is based on a "plug and play" architecture which is able to anticipate new security requirements by the integration of new security services or the enhancement of the existing ones. Furthermore, it was required that the way how the security information is stored has to remain configurable and storage mediums and mechanisms have even to be substitutable. For this purpose CSAP had to provide some generic interfaces (APIs), facilities for medium-independent data storage, as well as corresponding configuration tools. One of these will be described next.

3 Administration Console

Among other services, CSAP provides an implementation of a role-based access control subsystem that conforms to the *Core RBAC* model as defined in the proposed NIST standard [3]. Large Web-based e-commerce and e-government applications may have a number of roles up to hundreds and a higher number of users and permissions up to several hundreds of thousands. Consequently, administration of CSAP is difficult and important and managing of roles, users and permissions has to be done carefully and prudently. In other words, a human administrator needs tool support to organize the set of permissions, to define the roles, to assign users to roles and to assign permissions to roles. For this purpose a CSAP administration console (CAC) – a web-based user interface – has been designed and implemented. By using it the CSAP system can be administrated from any common web browser. The architecture of CAC was designed with a special variant of the architectural design pattern MVC (model view controller) called Model 2, that allows an application to be scaled up using EJB in conjunction with servlets and JSP [5].

Figure 2 shows a screenshot of CAC. On the left side a navigation menu is shown. The result of every menu selection is displayed on the right side. In this example permissions

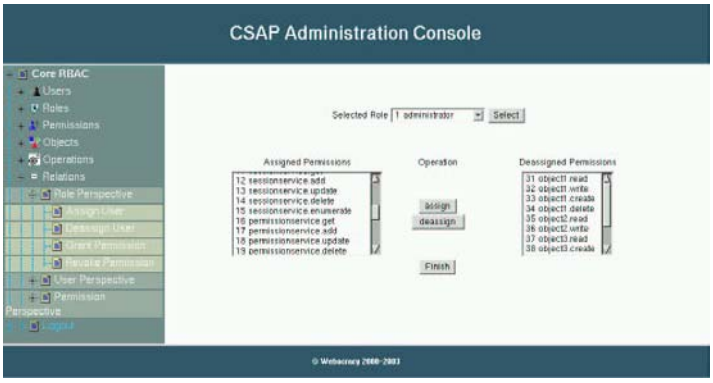

Fig. 2. Screenshot: permission role assignment from role-view

can be assigned or deassigned to/from a role which was selected first. After selecting a role the list of the already assigned permissions and the list of the not assigned permissions are presented. It is up to the administrator to select a certain permission from one list and assign it to the opposite one. The management of the other relations between the roles, users, permissions and sessions (e.g. user-role assignment) is done in a similar way.

CAC allows to start administration with every RBAC element. For example, an administrator may select a permission in order to manage all the related roles, sessions and users. The necessary functions are implemented within CSAP. Those functions are depicted in figure 3 and most are part of the NIST RBAC specification. The functions marked with a dashed line are not defined in the NIST models and additionally implemented by CSAP.

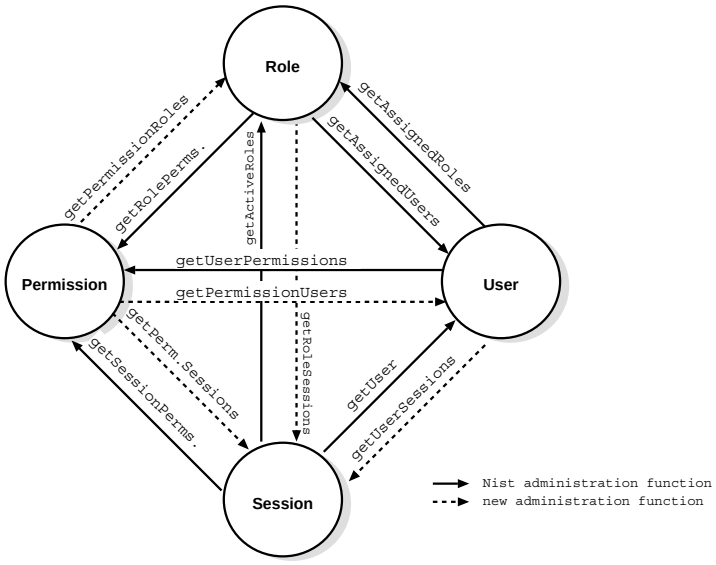

Fig. 3. Administration functions

The function *getRoleSessions* returns the sessions where a given role is activated. This function may be used before updating or deleting a role. From the point of view of a permission the functions *getPermissionRoles*, *getPermissionUsers* and *getPermissionSessions* return the list of the related roles, users and sessions, respectively.

4 Conclusion

We have presented CSAP and its administration. CSAP is currently developed as part of the ongoing Webocracy project. It is an autonomous software module offering programming interfaces to core security services such as authentication, access control and auditing. CSAP provides an implementation of role-based access control. We designed and implemented a web-based administration console for managing roles, users and permissions. It provides several useful administration functions, much more than defined in the NIST proposed standard. Moreover, it provides flexibility by allowing to start the administration with any RBAC element: role, user, session or permission.

References

1. F. Dridi, M. Fischer, and G. Pernul. CSAP – an adaptable security module for the e-government system Webocrat. *Proc. of the 18th IFIP International Information Security Conference (SEC 2003)*, Athens, Greece, 26-28 May 2003.
2. F. Dridi, G. Pernul, and T. Sabol. The Webocracy project: Overview and security aspects. In Schnurr et al., editor, *Professionelles Wissensmanagement: Erfahrungen und Visionen*, pages 401 – 408. Shaker Verlag, Aachen, 2001.
3. D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and Systems Security*, 4(3): 224–274, August 2001.
4. J. Paralic, T. Sabol, and M. Mach. A system to support e-democracy. *Proc. 1st eGovernment Conference. LNCS2455*, pages 288–291, Springer Verlag 2002.
5. Sun Microsystems. Java 2 platform, standard edition (J2SE). 2003. URL http://java.sun.com/j2se/.