A Model for Context-dependent Access Control for Web-based Services with Role-based Approach *

Ruben Wolf, Thomas Keinz, Markus Schneider
FhG Institute for Secure Telecooperation (SIT), 64293 Darmstadt, Germany
email: {ruben.wolf|thomas.keinz|markus.schneider}@sit.fhg.de

Abstract

Controlling access to resources is one of the most important protection goals for web-based services in practice. In general, access control requires identification of subjects that intend to use resources. Today, there are several identification mechanisms for subjects, providing different security levels. However, some of them are only suitable to be used in specific environments. In this paper we consider access control to web-based services that also depends on the strength of identification mechanisms as a context-dependent parameter. Furthermore, we show how to model this context-dependent access control by using role-based concepts.

1. Introduction

In the past years, an enormous diversity of technical applications based on web technology has been developed. These applications can be identified in various fields such as electronic shopping, trading with information, applications in the telematic area, or even accessing grid computing services via the web. Dealing with services provided over open networks, security questions like controlling access to offered services are of central interest.

Access control is strongly related to the question of user identification. There are several technical solutions which are used in practice, e.g., challenge & response protocols based on digital signatures and public key X.509 certificates (we consider X.509 certificates due to their wide deployment), passwords, and even cookies. The decision, which of these identification mechanisms has to be applied, eventually depends on the protection goals in the corresponding application and the desired security level. High-level protection usually requires stronger mechanisms. Furthermore, there are possibly desired functionalities that can only be achieved with specific identification mechanisms. Thus, the usage of specific services may only be allowed to be used after identification with specific mechanisms. On the other hand, there are situations where different identification mechanisms can be reasonable for accessing services from different locations, e.g., identification via private key / certificate from the user’s own computer and identification via passwords from a public computer. In such a context, the user’s permissions should also depend on the strength of the identification mechanism. The idea of role-based access control (RBAC) has been extensively discussed. RBAC is considered to be attractive because of its advantages regarding the support for efficient administration of large numbers of users, and its property of being policy-neutral, i.e., being able to deal with a wide range of security policies [4].

In this paper, we propose the usage of RBAC concepts. In this context, we do not use roles by mapping structures inside an institution to a role hierarchy as this is usually done. Instead, we model user classes into roles combined with the identification mechanism they use. Furthermore, we show how well-known RBAC concepts can be applied to model cases in which identification mechanisms can be used as a parameter to be evaluated in access control.

2. Service Requirements

The WWW is a basis for many business models. Today, there is a variety of services that can be accessed via the web. Examples are web-based email services, notification services, online auctions, virtual communities, collaboration platforms, information portals, online libraries, grid computing, information brokerage, procurement systems. These services can be considered to be bundles of various basic services which may comprise one or more operations on objects at the web server side. If specific operations are not allowed for all —possibly unknown— users, the identity of the requesting user has to be verified through a well-chosen identification mechanism, adequate for the values to be protected. In practice, web-based services are usually offered with exactly one identification mechanism chosen by the web service provider.

---

*This work was supported by the Federal Ministry of Education and Research in the ILAB project.
We distinguish two classes of web-based services. There are free services, which means that users do not have to pay for them, or chargeable services where customers have to pay for use. Charging can be realized according to various payment models, e.g., pay per use or pay per subscription based. In general, before a customer uses a chargeable web service he has to register. Services can also be for free but may still require registration. In the registration phase, a customer usually has to give some personal data and may also select some service packages containing the desired features. Then, a user account can be created for which the user can be mapped after he has been identified. In previous works, the realization of identification and access control have been assumed to be static, i.e., no varying identification mechanisms have been considered. Here, the goal is to consider identification mechanism as a context parameter for the access control decision. This kind of context-dependent access control can be motivated by several examples. Consider a user with the intent to access a web service from his own computer but also from a public computer somewhere in the world. Assume that from his computer at home this user uses his private key and his certificate for identification. Dependent on the concrete service, the high security level provided by private key / certificate identification can be necessary, e.g., because of sensitive data. Assume now, that a user wants to use a web-based service while he is traveling and only having access from untrusted machines. Such a user would not want to expose his private key to a potentially hostile environment. Thus, less critical operations should be allowed when he uses password identification, but more critical operations should be refused.

In another example, it is possible that specific service packages can have much more value for the customer than others. Also in such a case, it might be necessary that the protection level of the identification mechanism is high enough. This means, that according to a specific web service provider's policy, high-valued service packages — e.g., usage allowed only to members of a premium customer class instead of normal customer class — might require adequate identification mechanisms. Furthermore, the choice of the payment model for chargeable services and the identification mechanism should not be independent for security reasons. E.g., when offering services in a pay per subscription model, a service provider could have the interest to prevent two or more users from sharing one customer account to save costs. This functionality can be achieved with identification based on digital signatures and certificates, since users usually avoid to share their private keys that can be used for the creation of binding signatures. However, malicious account sharing cannot be prevented with passwords. In a pay per use model, password identification is sufficient. Figure 1 summarizes the aspects mentioned above in a domain model as used in software engineering.

3. Role-based Access Control

In this section, the main ideas in RBAC are presented. Since the general concepts of RBAC are well understood and extensively described in the literature [4, 7], we will only briefly describe the key aspects.

**Basic idea.** The central terms in RBAC are user, role, and permission. Therefore, we have sets USERS, ROLES, and PERMS. For elements of these sets, we have two assignment relations UA ⊆ USERS × ROLES and PA ⊆ PERMS × ROLES. The user assignment UA defines a relation between users and roles, whereas the permission assignment PA defines the relation between roles and permissions. Both relations are many-to-many. The set of all permissions is obtained by the combination of operations and objects, i.e., PERMS = OPS × OBS, where OPS and OBS are the corresponding sets. Types of operations depend on the type of system which is considered. In access control terminology, an object is an entity which contains or receives information, e.g., an object may be a file or some exhaustible system resources. If a user u ∈ USERS changes to a new user category leaving his old role r_{old} then he is simply assigned to a new role r_{new} by getting the permissions of r_{new} and losing the permissions of r_{old}. RBAC facilitates security management, makes it more efficient, and reduces costs.

**Role hierarchy.** Role hierarchies are thought to be one of the most desirable features in RBAC. They are very useful when overlapping capabilities of different roles result in common permissions because they allow to avoid repeated permission-role assignments. This allows to gain efficiency, e.g., when a large number of users is authorized for some general permissions. Role hierarchies RH ⊆ ROLES × ROLES are constructed via inheritance relations between roles, i.e., by the introduction of senior and junior relations between roles r_1 ⊇ r_2 in such a way that the senior role r_1 inherits permissions of the junior role r_2. To put it more formally, if we have roles r_1 and r_2 with r_1 ⊇ r_2, then \(\text{auth_perms}(r_2) \subseteq \text{auth_perms}(r_1)\) and \(\text{auth_usrs}(r_1) \subseteq \text{auth_usrs}(r_2)\). These mappings are defined as

- \(\text{auth_perms} : \text{ROLES} \rightarrow 2^{\text{PERMS}}\)
- \(\text{auth_perms}(r) = \{p \in \text{PERMS} | p' \preceq r, (p, r') \in PA\}\).
The opposite is not true, i.e., if $(\text{auth_perms}(r_2) \subseteq \text{auth_perms}(r_1)) \land (\text{auth_perms}(r_1) \subseteq \text{auth_perms}(r_2))$ is fulfilled, then it is still up to the role engineer to decide whether to introduce an inheritance relation between $r_1$ and $r_2$ or not. Multiple inheritance means that a role inherits both permissions and role memberships from two or more roles. With the concept of multiple inheritance, these hierarchies provide a powerful means for role engineering, i.e., it is possible to construct roles from many junior roles. Role hierarchies are modeled as partial orders.

**Role activation and sessions.** In a role hierarchy, a senior role inherits the permissions from the junior role. But a user does not necessarily have to act in the most senior role(s) he is authorized for. Actual permissions for a user are not immediately given by evaluating the permission assignment of his most senior role(s); these roles can remain dormant. Instead, actual permissions depend on the roles which are activated. A user may decide which roles to activate. Sessions are defined over phases in which users keep roles activated. A user’s session is associated with one or many roles.

**Principle of least privilege.** The principle of least privilege requires that a user should not obtain more permissions than actually necessary. This means that the user may have different permissions at different times depending on the roles which are actually activated. As a consequence, permissions are revoked at the end of sessions.

**Static separation of duty.** In some security policies, there may arise a conflict of interest when users get some user-role assignments simultaneously. The idea of static separation of duties is to enforce some constraints in order to prevent mutually exclusive roles. In the presence of a role hierarchy, inheritance relations have to be respected in order to avoid potential conflicts. The difference here is that these constraints define which roles are not allowed to be activated simultaneously, i.e., the constraints focus on activation of roles. Roles obeying these constraints can be activated when they are required independently, but simultaneous activation is refused.

**Dynamic separation of duty.** The goal of dynamic separation of duty is —similar to static separation of duty— to limit permissions for users by constraints in order to avoid potential conflicts. The difference here is that these constraints define which roles are not allowed to be activated simultaneously, i.e., the constraints focus on activation of roles. Roles obeying these constraints can be activated when they are required independently, but simultaneous activation is refused.

**Further relevant properties.** RBAC is assumed to be policy-neutral. This means that RBAC provides a flexible means to deal with arbitrary security policies. It is highly desirable to have a common means to express a huge range of security policies.

Since popular web-based services may attract large numbers of users, such services pose demanding requirements to security administration systems. Furthermore, providers are interested in changing their business models either for optimization reasons or just to test new ideas. Then, as a consequence, this may imply some changes in the security policy, e.g., introduction of new permissions, new customer categories. This requires adequate support for administration. RBAC simplifies access control management and supports flexibility. Consequently, there arises the question how RBAC can be used to provide efficient solutions for controlling context-dependent access to web-based services exploiting the advantages of RBAC. Previous work on RBAC focuses on different application areas, where roles and their relations are usually given by organizational hierarchies. Here, the situation differs; we model capabilities of users respecting identification mechanisms. This means that roles are not used anymore in the classical way.

### 4. Roles for Identification Mechanisms

The reason for introducing roles was to efficiently assign users to specific sets of permissions. If a member of a role tries to initiate an operation on an object where this role is not authorized for the corresponding permission, then the requested operation is refused. We propose that access control systems, which use identification results for their access control decisions, should take the security level of identification mechanisms into consideration. Consequently, in context-dependent access control, where the relevant context parameter is defined by the identification mechanism, there should be different levels of permissions which depend on the identification mechanism. In classical RBAC, different levels of permissions are aggregated to roles. If the usage of different identification mechanisms implies different permissions, then these mechanisms should influence the generation of roles and hierarchies.

We illustrate this idea by means of an example. If user Alice is authorized to access a service by identifying with a certificate in a challenge and response protocol, then Alice will be assigned to a role $\text{cuser} \in \text{ROLES}$. The role $\text{cuser}$ is authorized for a specific permission set $\text{PS}_{\text{cuser}} \subseteq 2^\text{PERMS}$. Another user Bob which decided to identify himself with a password should be authorized to a role $\text{puser} \in \text{ROLES}$. Analogously, he is authorized for permission set $\text{PS}_{\text{puser}} \subset 2^\text{PERMS}$. If both users identify themselves successfully using their chosen identification mechanisms, they are automatically authorized for the permissions which are assigned to their corresponding roles. With the introduction of roles, administration tasks become more efficient. A whole set of users can obtain new permissions by just assigning new permissions to their specific role. In case of changes in the access policy for roles like $\text{cuser}$ or $\text{puser}$, this adaptation can be done easily.

In practice, roles for customers are more differentiated than shown in the previous example. In real-world examples, there may be more roles necessary in order to express the relevant access control categories for permission
assignment. In this context, there may be further roles for the categorization of subscribed service packages, e.g., such as a role premiumuser for users that are allowed to access highly-privileged services in contrast to the role normaluser for users that have registered for the normal privilege service package. In real-world examples, there may be many more aspects to be considered for the introduction of roles.

However, the roles given in the example above might be sufficient in order to get the idea on how to deal with the problem of different security levels of identification mechanisms in access control. The potential of roles becomes obvious when roles can be arranged in hierarchies as will be shown in Section 5.

5. Role Hierarchies for Protection Levels

After having shown how customers using specific identification mechanisms can be modeled in a role-based approach, we now go one step further by considering role hierarchies. This means that our goal is to construct role hierarchies for the introduced kind of roles. From the considerations in Section 3 it follows that \((\text{auth_perms}(r_2) \subseteq \text{auth_perms}(r_1)) \land (\text{auth_perms}(r_1) \subseteq \text{auth_perms}(r_2))\) is a necessary condition for an inheritance relation \(r_1 \preceq r_2\) of the roles involved. This means that if the necessary condition is fulfilled, it is on the role engineer to decide if there should be an inheritance relation among the relevant roles.

In practice, for some given roles the construction of a role hierarchy requires some role analysis in order to verify whether the necessary condition for inheritance is fulfilled, and in case of a fulfilled condition, it has to be decided whether the roles should really be arranged in an inheritance relation. We demonstrate this process by reviewing the roles introduced in the example of Section 4.

In the previous example we have introduced the roles cuser and puser constructed by respecting context-dependent parameters with the goal that access control decisions also depend on the security level of mechanisms for user identification. When we take a look at the set of permissions obtained by \(\text{auth_perms}(puser)\) then there is no obvious reason why its elements should not be contained in the set \(\text{auth_perms}(cuser)\). Furthermore, we may require permissions obtained by \(\text{auth_perms}(cuser)\) which are not contained in the result of \(\text{auth_perms}(puser)\). Thus, we have \(\text{auth_perms}(puser) \subseteq \text{auth_perms}(cuser)\) which means that the first part of the necessary condition is fulfilled. For the second part of the necessary condition, we have to look at user sets which are authorized for the roles cuser and puser. Users which are assigned to the role cuser can be authorized for the role puser as far as they do not obtain some undesired permissions via this role. But this we have already considered in the first part of the necessary condition. In the opposite direction, users assigned to the role puser should not immediately be authorized for the role cuser. Therefore, we obtain \(\text{auth_usrs}(cuser) \subseteq \text{auth_usrs}(puser)\). Thus, for the concept of modeling identification mechanisms as roles the second part of the necessary condition is also fulfilled. This means that we can build role hierarchies.

We introduce some additional and more differentiated roles in order to demonstrate how a sample role hierarchy for a small service example may look like.

**Role premiumuser.** This role is for a customer which subscribed the premium service package and identifies with his certificate in a challenge and response procedure.

**Role cnormaluser.** This role is for a customer which subscribed the normal service package and identifies with his certificate in a challenge and response procedure.

**Role ppremiumuser.** This role is for a customer which subscribed the premium service package and identifies by sending his password.

**Role pnormaluser.** This role is for a customer which subscribed the normal service package and identifies by sending his password.

**Roleregisteringuser.** This role is for a customer which has not yet subscribed for the provider’s services. In our example, for online registration it might be reasonable to require certificate identification. Since already registered customers should be prevented to register multiple times they are not authorized for the role registeringuser.

**Role anonymoususer.** This role is for customers which do not identify themselves. Usually, in a web service there are some public areas which can be accessed by everyone without the need for identification, e.g., the start page or information to attract new customers.

The role hierarchy for these roles is depicted in Figure 2. There, we show the inheritance relations among the roles introduced above. In Figure 2, we have assumed that customer Alice has subscribed the service package premium and customer Bob has subscribed the service package normal. We assume that all permissions authorized via the normal service package are also autho-
rized via the *premium* service package. Since both customers may have personal permissions for which they are authorized exclusively, we have additional personal roles in the hierarchy. E.g., such permissions may comprise access rights to some personal data like customer profiles, where each customer has the permission to access exclusively his own data and no data from other customers. Since such permissions may depend on the identification mechanism it is reasonable to differentiate between Alice$_p$, Bob$_p$, and Alice$_c$, Bob$_c$. E.g., certificate identification may authorize customers to read and modify data while password identification may only authorize to read data. The role hierarchy shows that *anonymoususer* is the most junior role in the hierarchy. The role *registeringuser* is for users with the intent to use the online subscription facility. These users are not yet contained in the provider’s customer database. For security reasons, we recommend certificate identification for registration. Registered customers belong to at least to one of the roles *pnoruser*, *cnoruser*, *ppremiumuser*, or *cpremiumuser* dependent on the service package they have subscribed at registration and on the way they identify themselves when sending the request. The role *pnoruser* inherits the permissions from *anonymoususer*, whereas both roles *cnoruser* and *ppremiumuser* inherit the permissions of *pnoruser*. As a senior role, *cpremiumuser* inherits the permissions from *cnoruser* and *ppremiumuser*.

6. Automated Role Activation

After having introduced roles and a role hierarchy, we explain how roles can be activated. In general, if a user is assigned to several roles, it is up to him to decide which role(s) he is authorized to activate. In our work, the user does not select the role to be activated directly. Instead, the activation depends on the way the user identifies himself. Thus, one can say that a user selects the role to be activated indirectly. This means that dependent on identification aspects, the user role to be activated is selected by the access control system. In order to describe how and which roles are activated we introduce some mappings. For formalization reasons, we define IDMECHANISMS as the set containing the identifiers of identification mechanisms.

- $\text{assignedr} : \text{USERS} \rightarrow 2^{\text{ROLES}}, \text{assignedr}(u) = \{r \in \text{ROLES} | (u,r) \in \text{UA}\}$. The result of $\text{assignedr}(u)$ is the set of roles for which $u$ is assigned according to UA.
- $\text{juniorr} : 2^{\text{ROLES}} \rightarrow 2^{\text{ROLES}}, \text{juniorr}(rs) = \{r \in \text{ROLES} | r \subseteq r', r' \in rs\}$. The result of $\text{juniorr}(rs)$ is the set consisting of all roles which are in a junior inheritance relation to the roles contained in the set $rs$.
- $\text{allowedr} : \text{IDMECHANISMS} \times 2^{\text{ROLES}} \rightarrow 2^{\text{ROLES}}, \text{allowedr}(im, rs) = \{\text{subset of } rs \text{ authorized for } im\}$. Given an identification mechanism $im$ and a set of roles $rs$, the relation $\text{allowedr}(im, rs)$ filters all roles which are allowed for activation according to some given preconditions to be presented next.

In the following, we will give some sample preconditions for the roles in the hierarchy shown in Figure 2.

- *anonymoususer*: no identification
- *registeringuser*: identification required (certificate-based might be most reasonable), customer not yet registered in customer database
- *pnoruser*: identification by password, customer contained in customer database, customer has subscribed normal service package
- *cnoruser*: identification by certificate, customer contained in customer database, customer has subscribed normal service package
- *ppremiumuser*: identification by password, customer contained in customer database, customer has subscribed premium service package
- *cpremiumuser*: identification by certificate, customer contained in customer database, customer has subscribed premium service package

When the provider system receives a request, roles will be activated dependent on whether the customer identifies himself, how he identifies, whether he has already registered, and for which services he subscribed at registration. Dependent on the fulfillment of these requirements actual roles will be activated. The procedure according to which roles are selected for activation is depicted in Figure 3. With the activation of one role, all roles in junior inheritance relations are activated. In this way, the requesting customer also obtains the permissions of all junior roles.

![Figure 3. Role activation](image-url)
Roles to be activated are selected automatically, i.e., beside the fulfillment of the previous preconditions by using the corresponding identification mechanism a customer has no other possibility to select roles for activation. As it is shown in Figure 3, a non-identifying user can only obtain the permissions of the role anonymoususer. If an unknown customer, i.e., not yet registered, identifies himself, he obtains the permissions of the roles anonymoususer and registeringuser. Registered users get their roles activated according to (1) the UA stored in the customer database and (2) the identification mechanism they actually use. Here, this is modeled by the usage of assignedr, juniorr, and allowedr. Note that a registered customer only obtains the permissions of the role anonymoususer as long he does not identify.

7. Constrained RBAC for Web Services

Constraints in RBAC deal with static and dynamic separation of duty. This means that the RBAC system either controls that no users are assigned to conflicting roles according to the underlying security policy (static separation of duties), or users cannot be activated for conflicting roles simultaneously (dynamic separation of duty).

If we consider the roles that we have introduced in Figure 2, then it is clear that no registered user should be assigned to the role registeringuser. This means that the role registeringuser is mutually exclusive with all other roles that we have introduced, except for the role anonymoususer. This property holds for both static and dynamic separation of duty aspects.

In more concrete business models, there may be other constraints which follow from potential conflicts resulting from the underlying security policy. E.g., consider a model in which a pay per subscription customer should be prevented to access a service with his password. Such a policy would require additional roles not contained in the example dealt with so far. Only to sketch it, a role hierarchy required for this purpose should have password related roles and certificate related roles both being mutually exclusive. In these and many more cases, the application of RBAC mechanisms for avoiding assignment or activation of conflicting roles can be useful.

8. Related Work

In the past years, there have been various contributions to the area of RBAC which emerged as an alternative of classical discretionary and mandatory access control approaches. Ongoing research activities in RBAC resulted in the first proposed NIST standard for RBAC [4]. In the past, RBAC was mainly used for database management and network operating systems. Up to now, there are only a few contributions considering the usage of RBAC in the web context, even though RBAC is assumed to be a promising alternative for this area [5]. But in practice, protection on the web is still dominated by traditional concepts [2], e.g., access control lists. Proposals for the usage of RBAC for the WWW were given, e.g., in [1, 6, 8]. However, there the role concept focuses more on intra-organizational structures. The application of RBAC for controlling access to commercial services has not been considered. In the PERMIS project a role-based access control infrastructure was developed that uses X.509 attribute certificates containing user roles and the permissions granted to the roles [3]. However, certificates have to be renewed after some period, and they can become invalid before expiry date that requires revocation mechanisms.

The second aspect of our work deals with the context-dependency respecting security level of identification for access control decisions. This idea has not been considered so far in literature. In our opinion, this idea is of high relevance for WWW applications.

9. Conclusion

In this work we have shown how to model context-dependent access control regarding security level of identification mechanisms for web-based services with RBAC. Furthermore, we have shown how permissions for requesting users can be made dependent on the strength of identification mechanisms. Using RBAC technology is a promising approach for the realization of context-dependent access control in web-based services. This is especially interesting in cases of large user numbers, frequently changing and flexible business models usually accompanied by adaptations concerning the enforcement of security policies.

References