A Meta Model for Authorisations in Application Security Systems and their Integration into RBAC Administration

Axel Kern – Martin Kuhlmann – Rainer Kuropka – Andreas Ruthert
Beta Systems Software AG
Hermann-Heinrich-Gossen-Str. 3
50858 Köln, Germany
{axel.kern | martin.kuhlmann | rainer.kuropka | andreas ruthert}@betasystems.com

ABSTRACT

This paper presents a new concept for efficient access rights administration and access control. It focuses on the special requirements of application security and reflects experiences from the implementation of security for large industry application systems.
Application security shows a considerable inherent complexity due to the large number of combinations of objects and processes for which access rights must be defined. Based on practical experiences, this paper introduces a new approach for the implementation of access control for application systems which reduces this complexity. After describing the challenges for such an approach, we introduce process spaces and object spaces as a basis for authorisations. We show how they make application security maintainable, controllable and offer sufficient flexibility for reaction to changing business needs. In addition, we discuss how a separation of administration and access layers allows for convenient administration as well as optimised access decision performance in business-critical applications. To facilitate the integration of this rule-based concept into enterprise-wide security administration, we show how application security can be integrated into role-based access control (RBAC) systems. In particular, this goal is achieved by enhancing Enterprise RBAC (ERBAC) with variable roles. These roles can contain variable process and object spaces referencing user and role attributes. Finally, we give a short overview over related work.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Management, Security

Keywords

Automated identity management, security provisioning, security administration, role-based access control (RBAC), enterprise role-based access control (ERBAC), enterprise roles, application security, SAM Jupiter

SACMAT'04, June 2–4, 2004, Yorktown Heights, New York, USA.

1. MOTIVATION

Application security is a security world of its own. While topics such as firewalls, intrusion detection, virtual private networks and security features are increasingly the focus of discussions and publications, little is being said about application security. This is clearly a mistake, as the protection of application data - in terms of both integrity and authenticity - is an essential element for an integrated, comprehensive, enterprise-wide security infrastructure. As an initial approach to this topic, it makes sense to take a closer look at the challenges that arise in application security.
Although a database can be protected effectively using the means provided by system security, these measures cannot achieve differentiated protection of the objects contained in the database. Any attempt to implement an authorisation concept for millions of bank accounts or insurance policies based upon system security concepts seems nearly impossible. Arranging users in groups containing identical users and the grouping together of authorisation objects in units specific to particular target groups merely creates huge amounts of authorisations and opens the door to problems concerning administration and the quality and auditing of the mapped authorisations.
Protection of the objects alone does not meet all requirements. We must also consider the ability to describe protection of objects depending on the processes accessing the objects. Should it be possible to view data via Web interface and to maintain this same data via a host application? This differentiation introduces a new dimension of complexity for the authorisations to be mapped.
If an authorisation concept is to be structured according to how an organisation operates, then it must use the organisation's business processes and the individuals who perform these processes as reference points. Both the attributes of the acting individuals and the attributes of the processes and protected objects should be considered integral parts of

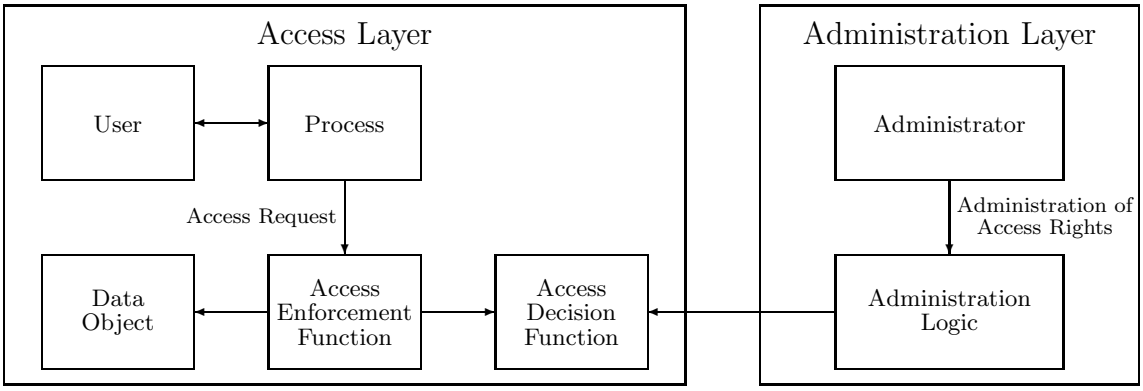

Figure 1: Application Security Overview

the authorisation concept. An authorisation resulting from such a concept could be formulated as follows:
An employee assigned to the Berlin branch of a bank can debit sums less than 100 EUR for all accounts in the branch office in which he is currently working.
An additional challenge lies in the establishment of roles for the purpose of bundling such authorisations. Roles are usually formulated in general terms and do not normally incorporate user attributes and attributes of the involved objects and processes. With eBusiness, the use of application servers and portals, and the increasing interconnectedness of IT infrastructures, the consistent enforcement of enterprise-wide security policies has become a primary concern.
In summary we can say that a solution of the challenges described here resolves the complexity of application security and makes a significant contribution towards increasing ease of administration, transparency, and suitability for practical use. The following factors are therefore decisive for the success of a meta model for authorisations in application security systems and their integration into RBAC:

- Authorisations can be qualitatively described in the same way for different security concepts.
- Authorisations are not described according to technical security concepts; rather, they reflect the security policy for each respective business process.
- Bundled authorisations based on business processes represent roles which are used within the context of role-based access control (RBAC).
- Administration is simple and efficient.
- The authorisations can be used in standardised form for authorisation verification in different applications on different platforms.
- Authorisation checks meet the performance requirements for business-critical applications, while at the same time remaining easy to administer.

The following text describes a meta model which takes into account all of the factors for success enumerated above.

2. CHALLENGES OF APPLICATION SECURITY

From our experience in the design and implementation of security systems and their administration in a number of large organisations (e.g., in the financial industry), we know that there is a fundamental difference between security for application systems and system security. We first face the situation that the number of system security concepts is relatively small due to the rather small number of standard operating systems and standard middleware systems (e.g., databases). In contrast, the components that provide security for applications are often proprietary. There are a great many of them, all following various concepts. The rising popularity of web applications has fuelled the development of access control systems available as commercial products (e.g., Netegrity Site Minder or Tivoli Access Manager).
Before describing a general application security management concept, we shall provide a brief definition of application security and review an ISO access control model which meets the needs of application security. We present some characterisations of application security and list the main challenges regarding its design and implementation.

2.1 What is Application Security?

2.1.1 Definition

An application security system provides controlled access to data stored in a computer system and to processes used by subjects to manipulate this data (e.g. view, change, delete, transfer).
As a basis for further discussion, we can use ISO 10181-3 [7], which defines a standardised model for access decisions and the enforcement of access decisions. The model describes the following (see Figure 1):

1. The access rights for application security are defined and stored in a repository via administration.
2. When a user starts a process or application, the access enforcement function (AEF) is called.
3. This function enforces the access decision by calling the access decision function (ADF) and allowing the required access to the data objects if the response from the access decision function is positive.

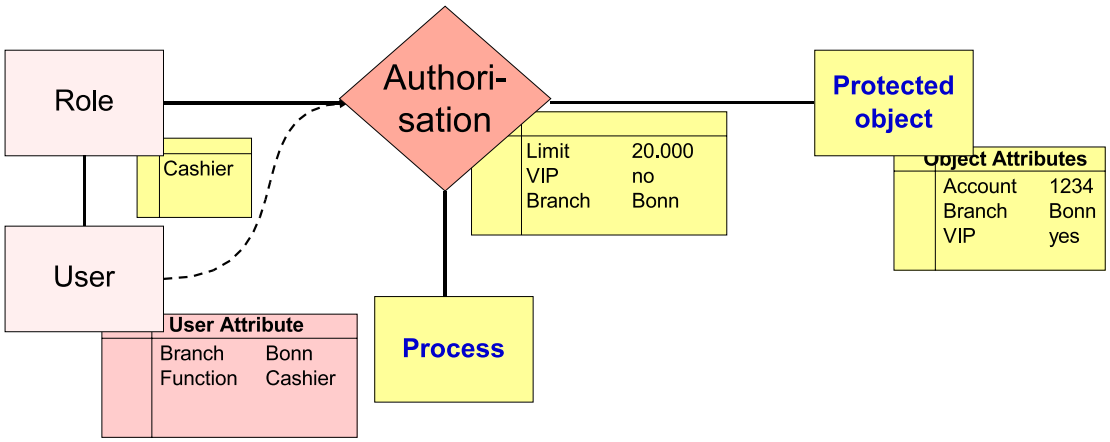

Figure 2: Problem of Integrating Application Security into an RBAC Concept

4. The access decision function compares the given process data with the defined access rights and returns *Access* if an access right could be found, and *No access* otherwise.

The introduction of an access enforcement layer has a high impact on application development, as the entire application architecture must be adapted accordingly. Therefore, many organisations do not implement a separate access enforcement layer, but depend on the programmers to call the access decision function when needed.

2.1.2 *Characteristics of Application Security Systems*

Examples for processes include transactions in a mainframe environment or distributed (web) services; the data to be accessed may be a bank account, a contract or a patient record, for example.
To control the access of users via a process to data objects, an "access decision function" performs an access permission check. The access decision function uses the following criteria for determining access permission or denial:

- User attributes (e.g., the user's organisational unit)
- Attributes related to the process (such as amount of withdrawal)
- Attributes of the data object (i.e., part of the data, such as account balance)

2.1.3 *Example*

A bank cashier is allowed to access an account via the service *Withdraw from account* if it is a VIP account belonging to the branch Berlin-Mitte or Berlin-Wedding and the maximum amount is less than or equal to 10 000 EUR.
The following constraints define the authorisation space for this access right definition:

1. Branch of account = (Berlin-Mitte or Berlin-Wedding)
2. Account type = VIP
3. Maximum amount <= 10 000 EUR

The attributes used for the access decision are not usually technical attributes (such as the "read/write/execute" attributes used in system security), but terms reflecting the semantics of the action to be performed and the data to be used ("business-oriented language"). The access rights should therefore be administered with business-oriented expressions.
On the other hand, the execution of the access decision function is highly time-critical because decision time has an impact on the execution time of the business function. Many companies require a response time between 10 and 100 milliseconds for the access decision. This means that the decision process must be optimised on an IT-technical level.

2.2 Where are the Main Challenges?

Several challenges for the design and implementation of application security become apparent from the characterisation provided above.

2.2.1 *Complexity*

The complexity of the described ternary relation (user - process - object) is quite high in the application security domain in comparison with system security:

- In the system security domain, we normally find a large number of users and a large number of objects, but only few processes (such as read, write, execute in file systems). This means that the complexity of one element of the relation (processes) is quite low. Therefore, authorisations in system security are normally defined as a binary relation with the process as attribute.
- In the application security domain, however, we find a large number of users, a large number of objects and a relatively high number of processes (e.g., all IMS transactions of an organisation). Therefore, we have a high complexity of all elements in the ternary relation.

The following example comparing authorisations for Windows 2000 system security and the application area for a company with 10 000 users illustrates this higher level of

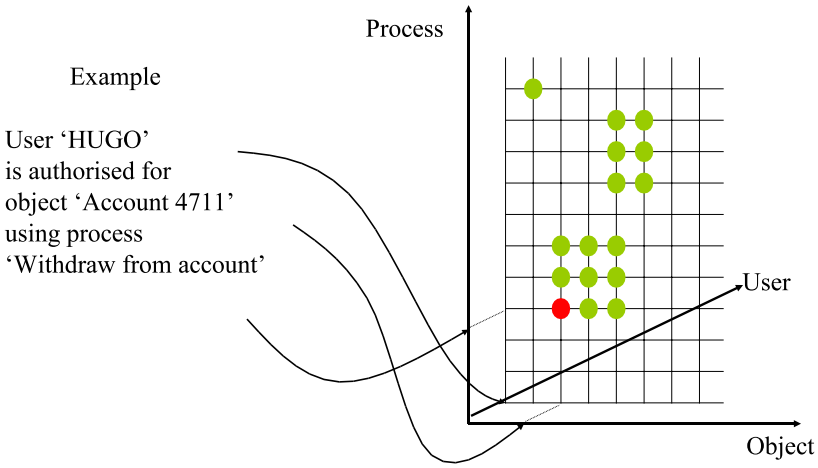

Figure 3: Authorisations - Points in a Three-Dimensional Space

complexity. Considering 6 access methods and 1000 system objects for which authorisations can be established in the Windows 2000 domain, we find 60 million potential authorisations. Considering that every user is actually authorised for 100 Windows 2000 objects using one access method each, we reach a sum of 1 million actual authorisations. In the application area, the company must authorise their users on 1000 contracts using 100 different processes (such as *conclude contract*, *change contract amount*, *change address*). This results in 1 billion possible authorisations. As every user has on average authorisations for 100 contracts via 20 specified processes, there are 20 million actual authorisations.

The number of actual authorisations is a measure for the performance requirements of the system, the number of potential authorisations is a measure for the complexity of the administration.

A further aspect of application security not mentioned in the above example is the additional complexity resulting from process attributes, such as posted amount.

2.2.2 Flexibility

Organisations are subject to a continuous process of change. The reasons for such change vary. Some examples include mergers, acquisitions or business process re-engineering activities. It is therefore unlikely that the structure of the application security system will remain unchanged for a long period of time.

If, for example, a bank decides to separate their accounts into VIP and non-VIP accounts, the access rights to the account must be able to reflect the *VIP* attribute of the accounts. As a result, it must be possible to supplement the application security system with new authorisation attributes easily. Therefore, we need a flexible, semantic-free representation of all information required for the administration of the system. These attributes can be part of the users, the processes or the objects that are to be protected.

2.2.3 Generic Application Security

Many modern companies have different objects that require protection, such as accounts, contracts, etc., each with different processes and different attributes. This requires generic application security that is able to handle access rights on all these different objects, processes and attributes.

2.2.4 Integration of Application Security into an RBAC Administration Concept

Modern enterprises use roles to administer user access rights in their IT environments. Of course, the administration of application security must be integrated into the role concepts. Roles define bundles of access rights which are then assigned to users [14]. Roles exist independently of users and do not take any characteristics of the users into account.

Authorisations in application security, however, are built using sets described by rules using authorisation attributes of the user. This means that authorisations contain references to values which are not known when a role is created. Figure 2 shows an example of this situation. A user is to receive the role *Cashier* for all accounts in his branch. Such an authorisation cannot be specified using the classic RBAC concept. A cashier role would have to be created for every branch. This would lead to an unmanageably large number of roles.

2.2.5 Access Decision Performance

The performance of access decisions is of vital interest to companies because an access decision is required for each individual access request. Slow access decisions are business-critical; e.g., for a bank the availability of its core banking applications is directly related to turnover and therefore to profit. In spite of the high number of attributes involved from different sources, the business requirement for high performance access decisions must be met.

3. SOLUTION

3.1 The Meta Model for Application Security

3.1.1 Definitions

Before starting, we would like to define the relevant terms in the application security domain. As already mentioned,

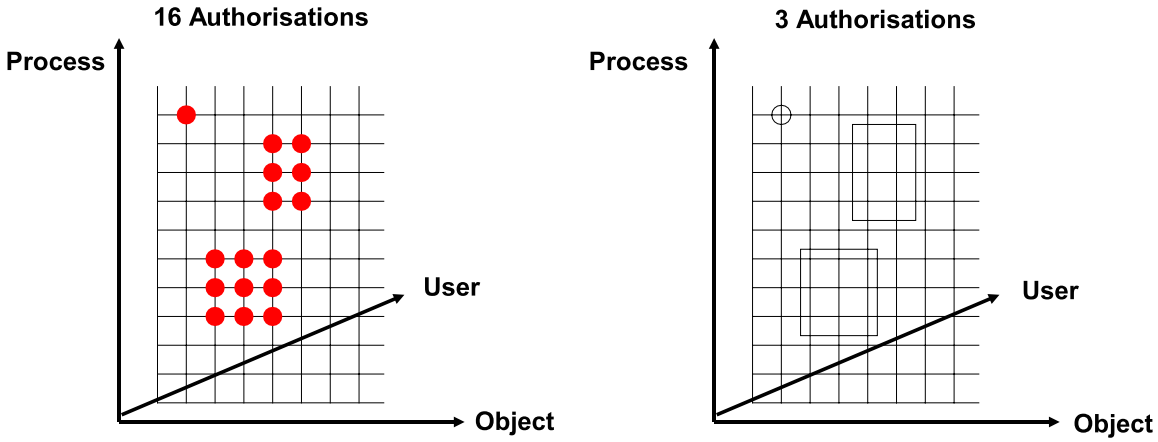

Figure 4: Authorisations - Enhancement Using the Descriptive Approach

the main issue is to define which user is allowed to use which object via which process.
The **user** is the representative of a real-life user in the IT system. In this context, it is also the subject of an access right.
A **process** is an action through which a user accesses an object. A process is part of an application. For example, the process *Withdraw from account* is part of the application *Account administration*. Considering access control, the process is the entity via which a user is authorised to access an object.
An **object** is the instance of a system resource which is to be protected. Examples for objects include a directory in Windows 2000 or a bank account in the application *Account administration*.
An **authorisation** can be defined as a triple (cf. [3])

(user, process with execution attributes, object)

or - more abstractly -

(specified set of users, specified set of processes with execution attributes, specified set of objects).

Therefore, every single authorisation can be defined as a point in a three-dimensional space which is built using all users, all processes (with execution attributes) and all objects (see Figure 3).

3.1.2 *Characteristics of the Meta Model*

A user is characterised by a number of attributes and their values. For example, the user *Smith* is assigned an attribute *branch* with the value *Bonn*. On the other hand, by specifying one or more attributes it is possible to define a set of users. For example, the set *all cashiers in branch Bonn* is defined as all users for which the attribute *branch* has the value *Bonn*. Attributes which are used to specify a set of users are called authorisation attributes of the user.
In the following, a set of users is mapped to the entity *role* and can be used for the definition of access rights (see section 3.5.1). As an alternative, we could have defined sets of users implicitly via rules. However, explicit definition of roles normally fits better to the business processes as it allows the specification of business roles and provides a sound base for auditing purposes. The attributes of a user can nevertheless be used to automate the assignment of roles.
A **process** is connected with two types of attributes:

- Like users and objects, the process definition can have attributes. These attributes can be used to specify processes or sets of processes. Attributes which are used to specify a set of processes are called authorisation attributes of the process.
- The process is executed with specific values for attributes of the process execution. For example, the process *Withdraw from account* will be executed with a specific amount. Attributes used during the execution of processes are called execution attributes of the process.

An **object** is characterised by a number of attributes and their values. For example, the account *5050505* is assigned the attributes *branch* with the value *Bonn* and *account type* = *VIP*. This again allows the definition of a set of objects by specifying one or more attributes. For example, the set *all VIP accounts of branch Bonn* is defined as all account objects where the attribute *branch* has the value *Bonn*, and the attribute *account type* is equal to *VIP*. Attributes used to specify a set of objects are called authorisation attributes of the object.

3.2 Reduction of Complexity

The following section leads to an administration model for application security that is able to meet the requirements of higher complexity.

3.2.1 *Using the Meta Model of Application Security*

Using our meta model and the definitions of users, processes and their attributes, we can define any authorisation in a security system as follows:
Authorisation = (specified set of users/role, specified set of processes with specified execution attributes, specified set of objects), where

- the set of users is specified by rules using the authorisation attributes of the user,

- the set of processes is specified by rules using the authorisation attributes of the process,
- the set of objects is specified by rules using the authorisation attributes of the object.

Using the three-dimensional presentation already introduced in Figure 3, the result is shown in Figure 4. It is obvious that the number of authorisations can be reduced considerably by using rules.

3.2.2 *Using Rule-Based Description for Sets of Objects and Processes in Authorisations*

Referring to the definitions of application security in section 2.1, we propose the following model:
Users are the subjects of authorisations in a system. They are normally human users, but there are also technical user IDs used by automated processes. The user record contains a set of standard and company-specific attributes. These include attributes describing the user, such as name, title, and telephone number, as well as authorisation attributes such as the users branch, organisational unit, job function(s) and so on. These latter attributes provide the basis for the application security authorisations as described below.

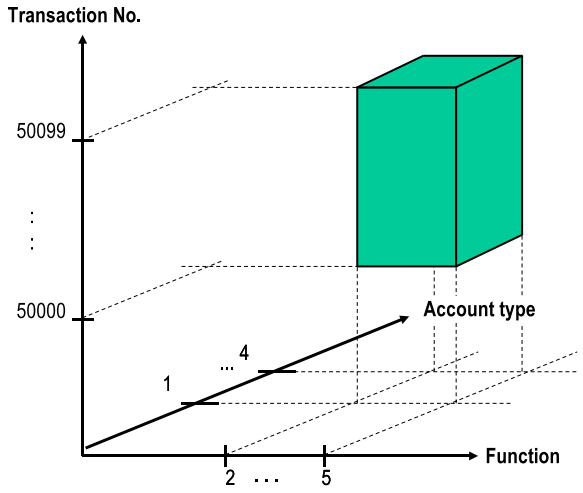

Figure 5: Process Space "T500*,F2-5,T1-4"

By defining rules using the process attributes, it is possible to group processes, e.g., a set can be "all transactions from number range *50000 - 50099* with functions *2 - 5* and type = *private*". With these attributes we can construct a multi-dimensional space containing all elements of the process set. We define a **process space** as a set of processes described by such a set of rules (see Figure 5).
As there are usually a number of different process types, process spaces are classified by a class attribute. Classes differ between different installations. Examples for such classes might be *IMS transactions* or *Web services*.
To facilitate administration, process spaces that are organisationally linked are bundled in process space folders. These folders are used for pre-selections in the user interface (see Figure 7).

It is possible to group objects by defining rules using the object attributes, e.g., a set can be "all private savings accounts (type 1 to 4) from number range *4711**". With these attributes we can construct a multi-dimensional space containing all elements of the object set. We define an **object space** as a set of objects described by such a set of rules (see Figure 6).

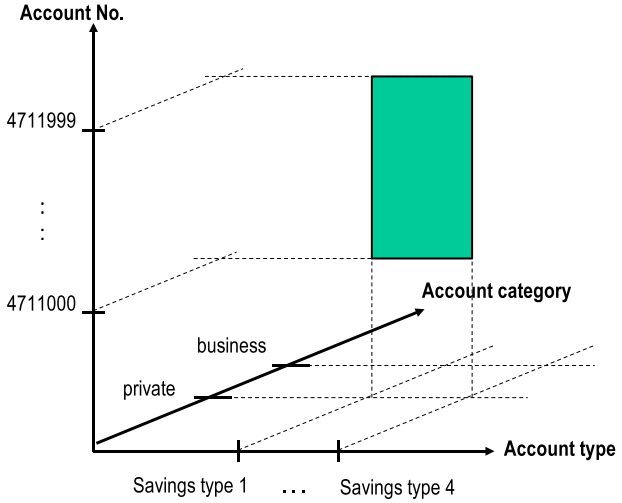

Figure 6: Object Space "Accounts S4711*,T 1-4,p"

As there are usually a number of different types of objects, object spaces are classified by a class attribute. Classes differ between different installations. Examples for such classes might be *bank accounts* or *contracts*.
To facilitate administration, object spaces that are organisationally linked are bundled in object space folders. These folders are used for pre-selections in the user interface (see Figure 7).
An **authorisation** in the application security system authorises a user or - as previously mentioned - a set of users represented via a role to execute specified processes on specified objects. It is defined as a triple of the already defined entities user/role, process space and object space. A typical authorisation is defined by the following steps:

1. A process space folder is assigned to a user/role. Such a folder contains process spaces which are linked together in terms of a particular business topic and are described by a business term that is also comprehensible to a non-technical administrator.
2. An object space folder is now assigned to this tuple. Such a folder contains object spaces which are linked together in terms of a particular business topic and are described by a business term that is also comprehensible to a non-technical administrator.
3. We now attain a matrix of process spaces and object spaces for this user/role (see Figure 7). In this matrix we can define which permissions the user/role receives. Every 'x' in Figure 7 indicates that the user/role is authorised for the appropriate combination of process space and object space.

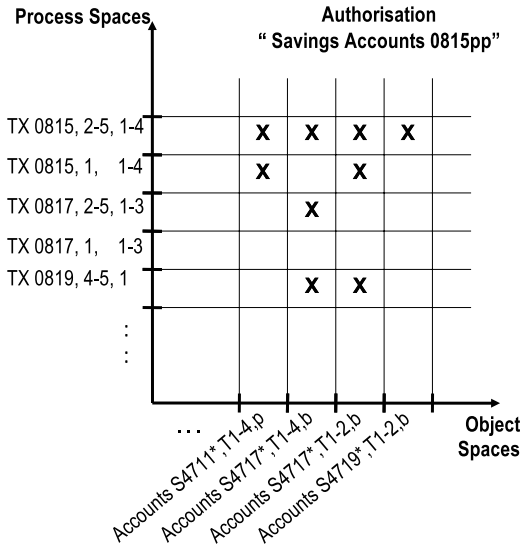

Figure 7: Authorisation Matrix

By using the process described above, we have reduced the high complexity of application security authorisations in two steps:

1. By grouping processes and objects in process spaces and object spaces using rules, we have considerably reduced the number of instances relevant for our authorisations.

2. By grouping process and object spaces in folders and combining them to form authorisations using an authorisation matrix, we have rendered the administration more comprehensible.

These measures allow even large organisations to manage their usually large authorisation space in the application security domain.

3.3 The Flexible and Generic Approach

To ensure that the application security system can be easily supplemented with new authorisation attributes, we need a generic, semantic-free representation of all information needed for the administration of the system:

- All entities are represented by attribute/value combinations.

- All rules use an attribute/operator/value representation.

Generic approaches have the disadvantage that checks for valid values and syntax are not possible. We therefore use an attribute repository containing type, syntax and valid values for all attributes used in the systems.

3.4 Enhanced Access Decision Performance

The meta model described in the previous section is convenient for administration, and the access control information can theoretically be used directly by an access decision mechanism (e.g., as described by the ISO standard in section 2.1). However, in practical implementations, some additional constraints for the access decision mechanism of an application system exist:

- For business critical applications such as core banking applications, the access decision mechanism must demonstrate optimal performance and availability.

- For regulatory compliance and to satisfy security policies of the organisation, it must have a sound audit functionality.

The first point in particular is not in accordance with administrative convenience and a possibly delegated administration model. The data model underlying our security meta model is optimised for administration; this will have a negative impact on the performance of the operative access control decisions: The data needed for the access decision must be collected from different sources and combined at run-time.
On the other hand, a representation of authorisation data optimised for performance would be difficult to handle by human administrators.
To solve this conflict, we use a two-layered architecture for our system, consisting of an administration and an access layer (see Figure 1). The administration layer is optimised to simplify the use for human administrators. The integration of our model with RBAC will consequently be for this layer. Additionally, we introduce an access layer consisting of the access control data optimised with respect to the performance of the access decision. This version of the access control data can be physically located "near" the access decision mechanism, which also improves availability. Between these two layers, access control data is converted by appropriate routines. To provide a general solution, both layers use the generic approach as described above.

3.5 Integration of Application Security into an RBAC Administration Concept

As already stated, it is state-of-the-art for medium and large organisations to administer users and their access rights using roles. Therefore, it is very important to integrate the administration of application security - which normally contains the most important and critical authorisations of an enterprise - into an RBAC concept. In the following we show how this goal can be achieved. We are using the Enterprise Role-Based Access Control Model as our basis. However, the presented concept is more general and can also be incorporated into other RBAC models.

3.5.1 The Enterprise Role-Based Access Control Model (ERBAC)

Roles are a powerful concept for simplifying access control. In Role-Based Access Control (RBAC), permissions are not directly associated with users, but are instead collected in roles. Users are then assigned to these roles, thereby acquiring the roles' permissions. A role normally contains all rights needed in an organisational unit or for a specific job function [14, 5]. In 2001, an RBAC standard was proposed which defines the core and extended capabilities of roles [6]. In [9] and [8] the Enterprise-Role Based Access Control

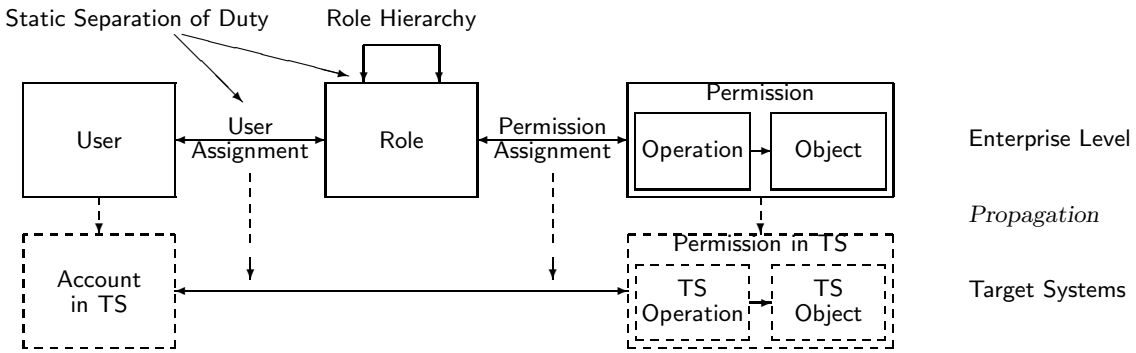

Figure 8: The Enterprise Role-Based Access Control Model (ERBAC)

Model (ERBAC)[1] was introduced. Enterprise Roles allow the administration of users and their access rights across all systems in the IT environment of an organisation. Enterprise Roles span over more than one target system and consist of permissions in multiple systems. These permissions are specific to a target system and can be of various natures.

Figure 8 shows the resulting ERBAC model. Enterprise Roles include all permissions needed to perform a specific business role. Users are then assigned to these roles. The permissions a user receives through the assignment of a role are propagated to the administered target systems. The Enterprise User definition leads to the creation of user accounts (user IDs) in the target system. A permission can be any operation for an object in one of the underlying target systems. The assignment of a permission to an Enterprise Role does not necessarily cause any update in the target system. The permissions defined for the role are propagated, and the users accounts receive the associated permissions in the respective target system only when a role is assigned to the user. The process is the same, of course, when permissions are added to or removed from roles.

In addition to the core RBAC features, a general role hierarchy is supported. Enterprise Roles can be assigned to other roles in a directed acyclic graph (DAG). Child roles inherit all permissions from their parent roles (including all permissions that these roles inherit). A user assigned to a child role thus receives all permissions assigned to this role, plus all permissions which the role inherits from its ancestors. Separation of Duty is implemented in ERBAC by rules defining constraints between roles. These rules are evaluated when assigning users to roles and connecting roles to other roles, thus preventing a user from receiving illegal combinations of roles, even in the presence of a role hierarchy.

3.5.2 Integration of Application Security into ERBAC

Our application security system is a specific target system and as such is integrated into our ERBAC model. However, as we have already shown in section 2.2.4, it is not easy to use typical application security authorisations in roles, as they contain variables referencing e.g. user attributes. Some examples include:

- A loan manager may grant loans only up to a specific amount.
- A bank cashier may only work with a specific set of customer accounts.

It would be possible to build separate loan manager roles for every different maximum amount or separate bank cashier roles for every range of customer accounts. Obviously, this is not a good solution as it would lead to a large number of similar roles.

The entities of our application security meta model can be mapped to ERBAC in Figure 8 as follows: The administration layer maps to the ERBAC enterprise level, whereas the access layer (see section 3.4) is our target system. Application security permissions contain process spaces and object spaces which can be compared with operations and objects in standard RBAC. All entities in ERBAC are supplied with attributes which can be used to define variable role definitions. For application security permissions, we use three types of attributes in particular:

- Process space attributes, such as *account type* (see e.g. Figure 5),
- Object space attributes, such as *branch*,
- Execution attributes, e.g. the *maximum amount* allowed for a withdrawal from an account. These attributes are compared with the actual values of the transaction during run-time. In the example above, the transaction is rejected if the *actual amount* to be withdrawn is greater than the defined *maximum amount*.

To provide flexible roles for application security, we leave one or more of these attributes in a role permission assignment variable, meaning that we define a reference to attributes of other ERBAC entities instead of defining fixed values. It is possible to reference attributes from the user, the user role assignment or the role. Using some real-life examples, we go into more detail about these three cases in the following:

[1]For a more comprehensive description of ERBAC and its comparison to the proposed NIST RBAC standard see [8]. ERBAC has been implemented in the commercial security provisioning and identity management tool SAM Jupiter [13].

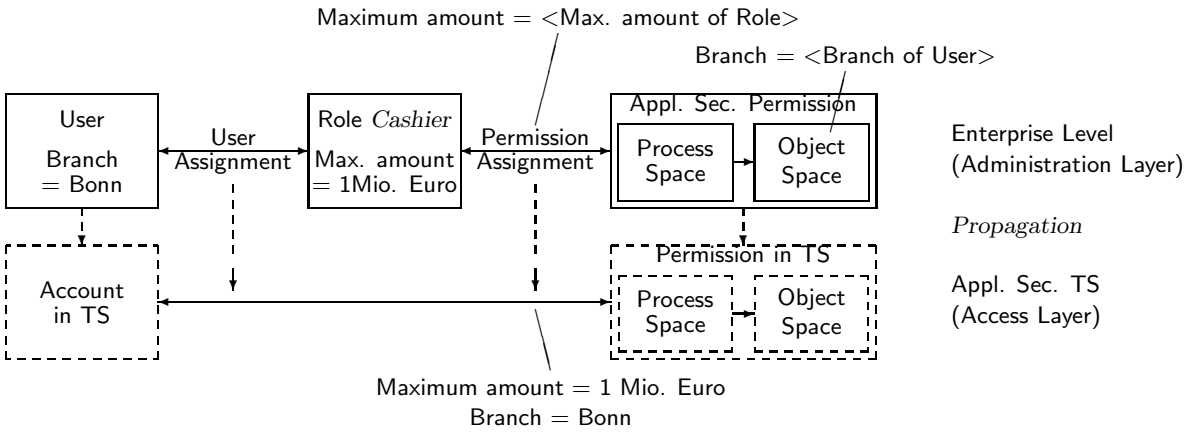

Figure 9: Role with Generic Application Security Permission

- **Attributes from the user record:** As described in section 2.1, a user record contains a number of attributes. These attributes can be used to define a *data space* which is referenced by *variable* permissions of the role. These references are only evaluated when the role is assigned to a user. Thus, we are able to define *generic* roles for such items as job functions which can be restricted to a specific *data space* for every assigned user. This results in a much smaller number of roles and makes administration easier.

  Figure 9 shows a real-life example from a bank: A role *Cashier* is defined which will allow access to all accounts in the branch of the user. Therefore, we assign a permission containing the process space *post for account* and an object space *accounts* with the attribute *branch* = <*branch of assigned user*> to the role. When assigned to a user, the variable in the object space is filled with the value of the attribute *branch* of the user record resulting in the desired permission for this user. If the user's branch changes, the permission is adapted accordingly. Thus, role and role assignment become resistant to organisational changes.

- **Attributes from the user assignment:** We can also specify attributes from the user assignment to roles and use them as references in variable permissions. The functionality is quite similar to that described for user attributes above, but has some important differences. We can define different attributes for every role a user is assigned instead of always using the one defined in the user record. If, for example, an individual has different jobs in different branches of a company, we assign him the appropriate roles and specify the attribute *branch* with the appropriate value for each assignment.

- **Attributes from the role definition:** Similarly, we can specify attributes from the role definition and use them as references in variable permissions. This feature does not provide additional flexibility when using the role, as in the two cases described above. However, referencing role attributes in the permissions can make sense as a means of facilitating the administration of changes. We then only need to change the role attributes instead of all referencing permission assignments.

  An example is shown in Figure 9: The permission assignment receives a constraint "Maximum amount = <*Max. amount* of Role>", which refers to the *Max. amount* attribute in the role definition.

The use of attributes in variable roles is also possible in the presence of a role hierarchy:

- User attributes are valid for all roles to which a user is connected, be it directly or via inherited roles.

- Attributes specified for a user assignment also apply for roles which are inherited via this assignment. Possible conflicts which may occur when the same role is assigneed to a user via different inheritance paths are normally solved by cumulating the assigned attribute values. For example, a user is directly connected to the role *Cashier* with attribute *branch* = *Bonn* and also inherits role *Cashier* via another role with assignment attribute *branch* = *Berlin*. As a result, our user is authorised for the role *Cashier* for both branches.

- Role attributes are not affected by role hierarchies, of course. Attributes apply only to the role to which they are directly assigned.

4. RELATED WORK

[12] describes how the access control system of a legacy bank application can be mapped to the proposed NIST RBAC standard. In the conclusion, it is suggested that it should be possible to attach rules to roles. An example would be a rule which says that the target object may only be accessed when it has the same *branch* attribute as the user holding the role. Our generic role definitions as described in the previous section fulfil these requirements.
Like other authors, we have emphasised elsewhere that automation of user administration is very important (see e.g.

[9, 10]). User data from databases such as HR systems is used to automatically create, maintain and delete users in the RBAC system. [1] presents a rule language which is used to assign roles to users by evaluating user attributes. The use of rules based on user attributes is similar to our approach. However, the two approaches differ considerably: [1] uses rules to define which user receives which role. In contrast, our rules define which permissions a user receives when he is assigned a role.

The Administrative Enterprise RBAC concept (A-ERBAC) in [11] describes the administration of ERBAC itself. It resembles the concept presented in this paper in some aspects, as it also describes the administration of an application. For example, the implementation of A-ERBAC also separates administration and access layer to provide fast operation of the access check. A permission in A-ERBAC is defined by a triple (operation, object, scope). Whereas the tuple (object, scope) is similar to the definition of an object space in this paper, the number of allowed operations is quite small, so that process spaces are not needed.

We know of several case studies for application security systems for health care organisations (see [2, 4]). These systems fulfil the special requirements in this field. In contrast, our approach is more general and meets the needs of most commercial enterprises.

5. CONCLUSION AND FUTURE WORK

This paper presents a new approach for the practical implementation of access control for application systems. Application security shows a considerable inherent complexity due to the large number of combinations of objects and processes for which access rights must be defined. We propose a solution which reduces this complexity. The introduction of process spaces and object spaces makes application security maintainable, controllable and offers sufficient flexibility for reaction to changing business needs. Together with a separation of administration and access layers, our approach fulfils the requirement of optimised access decision performance in business-critical applications.

With the integration of this rule-based concept into RBAC, we achieve a further reduction of complexity on the administration layer. Additionally, this concept provides the possibility of integrating application security with an enterprise-wide role concept (ERBAC).

Our approach was mainly derived from experiences made working with conventional distributed and mainframe application systems. It would be interesting to see if and how this concept is influenced by web services architectures in which it might be difficult to decide where applications reside and where security checkpoints should be posted. Emerging standards such as SAML or XACML (see www.oasis-open.org) will influence application security, and as a future task we will analyse the impact of these technologies on our model.

6. REFERENCES

[1] M. A. Al-Kahtani and R. Sandhu. A Model for Attribute-Based User-Role Assignment. In *Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA*, pages 353–362, December 2002.

[2] B. Blobel, R. Nordberg, and P. Pharow. Privilege Management und Zugriffskontrolle in verteilten Gesundheitsinformationssystemen. In P. Horster, editor, *D·A·CH Security – Bestandsaufnahme, Konzepte, Anwendungen, Perspektiven*, pages 374–382. syssec, March 2003.

[3] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *IEEE Symposium on Security and Privacy*, pages 184–194, 1987.

[4] A. A. El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *Proceedings of the Fourth IEEE International Workshop on Policies for Distributed Systems and Networks, Como, Italy*, pages 120–131, June 2003.

[5] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, editors. *Role-Based Access Control*. Artech House, Norwood (MA), USA, 2003.

[6] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, August 2001.

[7] *ISO/IEC 10181-3, Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework*, 1996.

[8] A. Kern. Advanced Features for Enterprise-Wide Role-Based Access Control. In *Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA*, pages 333–342, December 2002.

[9] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the Role Life-Cycle in the Context of Enterprise Security Management. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002), Monterey, California, USA*, pages 43–51, June 2002.

[10] A. Kern, M. Kuhlmann, and R. Wick. Ein Vorgehensmodell für Enterprise Security Management. In P. Horster, editor, *Sichere Geschäftsprozesse*, pages 81–90. IT Verlag für Informationstechnik, Höhenkirchen, September 2002.

[11] A. Kern, A. Schaad, and J. Moffett. An Administration Concept for the Enterprise Role-Based Access Control Model. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), Como, Italy*, pages 3–11, June 2003.

[12] A. D. Marshall. A Financial Institution's Legacy Mainframe Access Control System in Light of the Proposed NIST RBAC Standard. In *Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA*, pages 382–390, December 2002.

[13] *For more information about SAM Jupiter see http://www.sam-security.com.*

[14] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, February 1996.