Rule Support for Role-Based Access Control

Axel Kern – Claudia Walhorn
Beta Systems Software AG
Josef-Lammerting-Allee 14
50933 Köln, Germany
{axel.kern | claudia.walhorn}@betasystems.com

ABSTRACT

The administration of users and access rights in large enterprises is a complex and challenging task. Role-based access control (RBAC) is a powerful concept for simplifying access control. In particular, Enterprise Roles spanning across different IT systems are increasingly used as a basis for company-wide security management. However, the administration of roles in large organisations can become quite cumbersome and needs to be automated.

During the past years, rules have been used to support automation of user and access rights administration. We discuss different rule-based approaches and propose a new method called rule-based provisioning of roles which combines the advantages of rules and roles.

Experiences made during implementation of this approach are presented in two case studies. The results are evaluated and show that role-based access control in combination with rule-based provisioning can be successfully used in practice. A high level of automation can be achieved.

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), rules, access rights, SAM Jupiter, Provisioning Engine, Rule Engine, directories, case studies

SACMAT'05, June 1–3, 2005, Stockholm, Sweden.

1. INTRODUCTION

Role-based access control (RBAC) has established itself as a solid basis for today's security administration needs, as documented in the RBAC96 model [16] and the subsequent NIST standard [7]. In particular, Enterprise Roles spanning across different IT systems are increasingly used in medium and large organisations as a basis for company-wide security management. An Enterprise RBAC model (ERBAC) has been described in previous papers [11, 13] and was implemented as a core functionality of the commercial identity management product SAM Jupiter [15].

Roles as the sole basis for security administration have some significant drawbacks, however.

- If roles only contain explicit authorisations, large organisations need quite a large number of different roles. What is needed is a more generic definition of roles (see e.g. [11]).
- Although the usage of roles reduces the administration effort considerably, quite a lot of work is still necessary to assign roles to users (so-called security provisioning). Efforts are made to automate the provisioning process (see e.g. [13]).

Rules are a well-known technique used for automation purposes. In this paper we show that in combination with roles, the use of rules enhances the current ERBAC model significantly. The resulting system fits well into current IT environments and reduces the efforts for identity management and access control administration considerably. Practical experience was gained when introducing this concept during the implementation of the identity management software SAM Jupiter in several large companies.

Having given an initial introduction and motivation for our work, the rest of this paper is structured as follows. We discuss several administration approaches using roles and/or rules including our concept of rule-based provisioning of RBAC in section 2. This is followed by a brief summary of related work in section 3. We present two case studies in section 4 showing the experiences made with introducing RBAC and rule-based provisioning in large organisations. An evaluation of these experiences concludes our paper.

2. ROLES AND RULES

2.1 Role-Based Administration

In the past years, role-based access control (RBAC) has established itself as a solid basis for today's security administration needs. Based on the RBAC96 model [16, 6], a

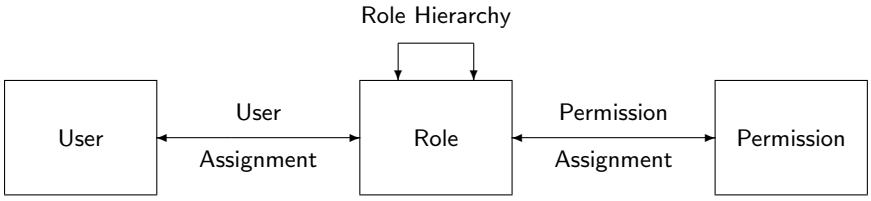

Figure 1: Role-Based Access Control (RBAC)

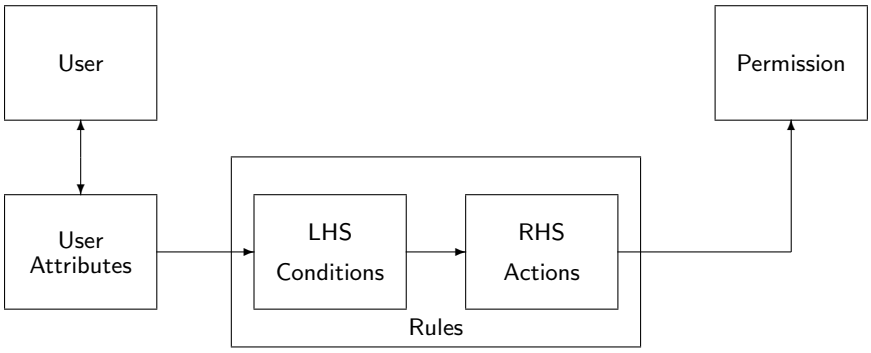

Figure 2: Rule-Based Access Control

NIST standard proposal [7] was accepted by the American National Standards Institute, International Committee for Information Technology Standards as ANSI INCITS 359-2004.

In the RBAC model, permissions are not directly assigned to users but are instead collected in roles (see figure 1). Every role represents a special scope of permissions. A user is assigned to one or more roles, thereby acquiring permissions defined for the roles.

To facilitate the administration of permissions, roles can be organised in role hierarchies in which permissions are inherited. Permissions can easily be taken away or added to a role if necessary. This is then reflected in the permissions granted to all users that are assigned directly or indirectly to the role.

The standard RBAC model also contains sessions. During a session, a user can activate one or more of the assigned roles. Each session is associated with one user, whereas a user can have several sessions at the same time. As we are concentrating on the administration of RBAC in this paper, and sessions are not relevant in this context, we do not deal with them further here.

The RBAC approach has the following strong advantages:

- RBAC is a simple and proven administration concept.
- Roles allow business-oriented, non-technical administration.
- Roles provide a solid basis for security audits.

However, the RBAC approach also has some disadvantages:

- A potentially high administration effort, as roles are principally static.
- Organisational and functional changes that often occur in large enterprises always require the reassignment of a considerable amount of roles. The resulting administration work is normally quite expensive.

2.2 Rule-Based Approaches

We will discuss some alternative strategies for dealing with these disadvantages. As already pointed out in [1] and [11], the decision that determines which roles a user receives is based on the user's attributes. As these attributes are normally already stored in a human resources (HR) database, a directory or a security administration system, the attributes can be used to automate the process of assigning roles. This can be done by utilising rules. In the following, we discuss different approaches to applying rules.

2.2.1 Rule-Based Access Control

The grouping of access rights can be achieved by using rules instead of roles. Rules consist of a condition in the so-called left-hand side (LHS) and one or more actions on the right-hand side (RHS) [9]: When the expression on the LHS is true, the action on the RHS is executed. In our context, the LHS expressions are based on user attributes (see figure 2). A rule such as

```
IF   User-Function = "Developer" AND
     User-Company  = "Beta Systems"
THEN Execute Java Compiler
```

gives all users of company "Beta Systems" with the function "Developer" *Execute* access to the Java compiler. Thus, a rule makes it possible to give one or more access rights to a whole group of users making rules quite a powerful and dynamic administration tool. Changes in user attributes automatically change the user's access rights. As the attributes are often provided automatically – e.g. from an

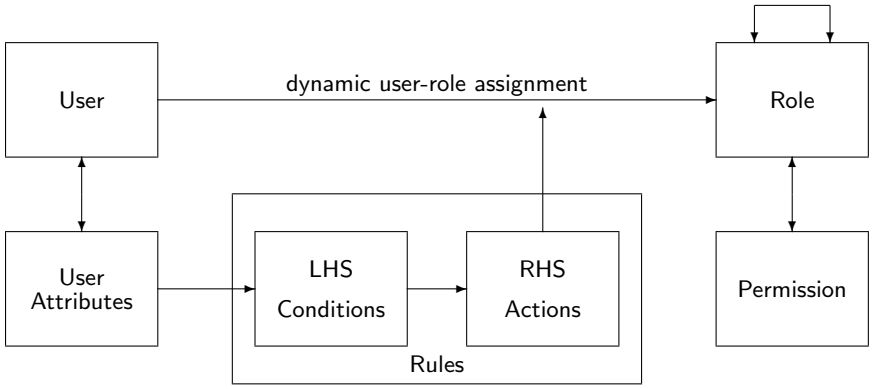

Figure 3: Rule-Based RBAC (RB-RBAC)

HR database – administration work is considerably reduced. Furthermore, the correctness of the access rights is high, as long as user attributes and rules are correct.

Compared to roles, however, rules have some drawbacks:

- When using rules, the assignment of access permissions to users is highly dynamic. Thus, it is difficult to obtain an overview concerning who is allowed to do what. In particular, this makes it difficult to provide a sound audit function (see also [10]).
- As as result, it is not easy to maintain rules in the long run because it is difficult to foresee the impact of rule changes.
- Rules completely lack the capacity of roles to build "business roles" which contain all permissions for a specific business function and can be assigned by non-technical administrators.

For these reasons, the use of rules alone for administering access rights does not seem feasible. Nevertheless, rules are used as a method for grouping permissions in access control systems. An example is the mainframe security system CA-ACF2, which features authorisations through rules based on a very restricted set of user attributes.

Another example is introduced for application security systems in [12]: Rules are used to build sets of processes and objects, which in turn are used to define authorisations. These authorisations can then be grouped to form roles and used in an RBAC system. The main reason for using roles in this context is the reduction of complexity.

2.2.2 Rule-Based RBAC

As the use of rules alone does not seem to be feasible for enterprise-wide user administration, we now look at approaches which combine rules and roles. In [1] the so-called Rule-Based RBAC (RB-RBAC) is introduced: In contrast to the standard RBAC model, roles are not assigned directly to users. Instead, rules using user attributes compute the role assignments for users dynamically (see figure 3). Some user attributes which might be used for this purpose in our context include organisation, job function or location.

The concept of combining rules and roles has the following advantages:

- This approach offers most of the advantages of RBAC, including the possibility to use role hierarchies.
- Rules feature dynamic role assignments based on user attributes and thus reduce administration effort.

If changes in the role assignments are necessary, e.g. due to organisational or operational changes, no manual assignments and deassignments of roles are necessary. Often, only the user attributes change thus adapting the user's role assignments automatically. For structural changes, the appropriate rules are written or changed.

However, this approach has some disadvantages:

- Large organisations probably have quite a lot of rules, which makes it difficult to maintain an overview of who has which permissions.
- The missing overview makes it especially difficult to fulfil auditing requirements.
- Furthermore, it is often difficult to foresee the impact of a new rule or the modification of an existing rule.

2.2.3 Rule-Based Provisioning of RBAC

As described in the previous section, the most important advantage of applying rules is the possibility to automate role assignments. The main drawback – at least when used for large user populations in complex environments such as banks and other large organisations – is the lack of overview and audit capability. This is mainly due to the highly dynamic quality of the approach, which implicitly connects users to roles.

To overcome these disadvantages, we propose an approach which is more "static": We separate rule-based provisioning and the role-based administration system. With "provisioning" we describe the process of bringing users, their attributes and further information needed for entering access rights into the RBAC system (and thus "provisioning" the user with the needed access rights). This process is normally automated by taking information from HR databases, corporate directories or other information bases in the enterprise. Such databases contain information about employees entering or leaving the company and data such as employee number, organisational unit, location or job description.

Thus, the provisioning process encompasses two steps (see figure 4):

1. Provisioning enters new users, deletes old ones and changes attributes for existing users in the RBAC system.

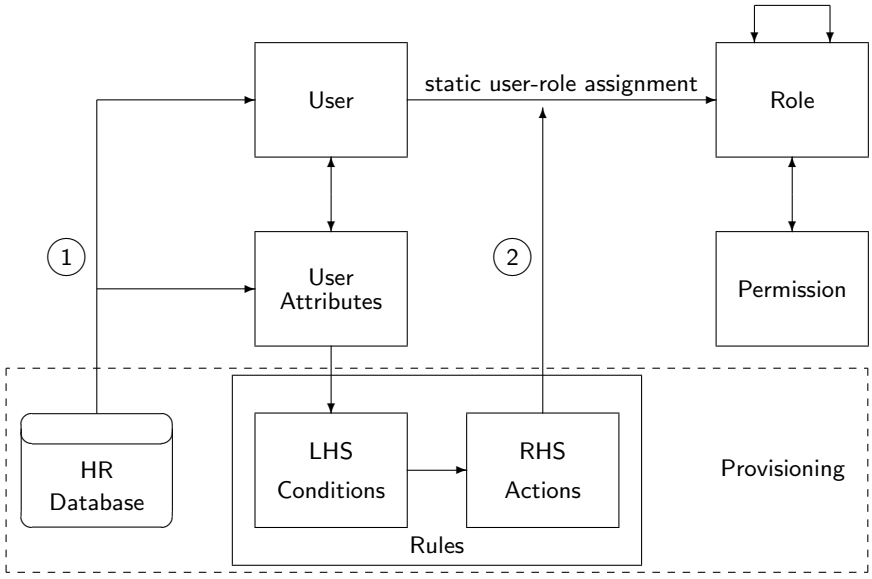

Figure 4: Rule-Based Provisioning of RBAC

2. Based on the attributes of the users, the process computes which role assignments the users should receive and adds new role assignments and deletes obsolete assignments accordingly.

By separating provisioning and the RBAC administration system, we are thus able to combine the advantages of roles and rules:

- As the RBAC system contains explicit role assignments, the administrator has a good overview of the actual authorisations that a user has in the system.
- For the same reason, the system provides good auditing capabilities, which is very important in many industries such as banks and insurances.
- Rules are used to compute the role assignments by exploiting information from existing databases.
- Using these rules, a high grade of automation can be achieved, thereby considerably reducing the manual administration effort.
- Automation using rules also reduces the probability of errors and leads to a higher security level as it prevents errors made in manual administration.

For RB-RBAC, we mentioned that it is often difficult to foresee the impact of a new rule or the modification of an existing rule. This drawback also exists for rule-based provisioning. As a remedy, an impact analysis feature simulates rules and allows the rule administrator to check the impact of changes before they are activated.

A disadvantage of rule-based provisioning compared to RB-RBAC is the more static use of rules: Changes in rules or attributes do not take effect immediately on permissions of users. However, we think that in practice – at least for large user populations – this disadvantage is outweighed by the improved overview and auditing capabilities. Normally, a daily update of user information and their role assignments is sufficient. As organisational changes, job changes etc. are normally known in advance, such changes can be scheduled for the provisioning run at a set date. In urgent cases, manual administration directly in the RBAC system is possible. Manual actions that contradict the defined rules are detected and corrected during the next provisioning run.

3. RELATED WORK

Al-Kahtani and Sandhu introduce a model and language for attribute-based user-role assignment, called Rule-Based RBAC (RB-RBAC) in [1]. We have already discussed this approach in section 2.2.2. In [3] this model is extended with negative authorisations. As we do not use these in ERBAC, we do not further discuss this issue.

Al-Kahtani and Sandhu also introduce seniority levels of rules: A rule A is senior to a rule B if the users that meet the conditions of rule B are a subset of those meeting the conditions of rule A. Seniority requires that an order be defined for the attributes used in rules. Seniority levels might conflict with role hierarchies: Let us assume that rule A assigns role *R1* and rule B assigns role *R2*, and *R2* is senior to *R1* in the role hierarchy. Users matching rule B would then acquire role *R1* through role inheritance although it is assigned by a rule with a higher seniority level. The impact of contradicting rule and role hierarchies is explored in [2]. In our context, we do not see seniority levels of rules as an important aspect. This is mainly because for most of the attributes, no clear order can be defined. This point was also confirmed by the practical experiences that we have made in customer projects using rules.

In [5], Chandramouli presents a business process driven framework for defining an access control service consisting of five steps:

1. Identify the business processes.
2. Determine access control requirements.

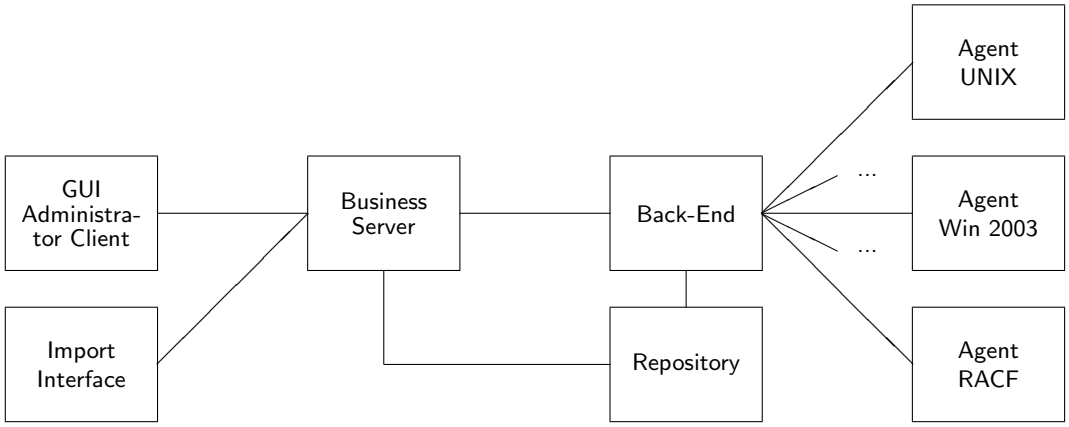

Figure 5: Architecture of SAM Jupiter

3. Define the access control model for the application.
4. Define the access decision rules.
5. Define the access enforcement mechanism.

The framework uses roles and rules: Roles are used as the basic mechanism of the access control model. Rules are defined to provide access to functions in the applications and can authorise either roles or users directly. Rules are thus used as described in section 2.2 and [12].

4. PRACTICAL EXPERIENCE

We have used the approach described in the previous chapter in a number of identity management projects in large organisations. Presenting some case studies, we show in this section how RBAC and rules can be successfully combined in real life. The case studies are based on the experiences that we have made when introducing our identity management solution – SAM Jupiter – in several companies.

4.1 SAM Jupiter

SAM Jupiter is a commercial identity management solution developed by Beta Systems [15]. It provides a central point of administration, giving administrators full control of all IT access control management for employees and resources without compromising on the lowest common denominator of security protection. Interfaces to the specific security systems and applications allow for a consolidation of information in a common security repository using a system-independent conceptual model. When these systems are connected to SAM Jupiter and their data loaded into its repository, administrators work only in the SAM Jupiter environment and no longer need specific knowledge about the systems they administer. This not only consolidates the administration work, but also reduces the need for in-depth knowledge about all underlying systems. All administration work is done in SAM Jupiter and automatically propagated to the underlying systems in the format required.

Figure 5 shows the architecture of SAM Jupiter. It is based on a state-of-the-art 3-tier architecture. The presentation layer is represented by a modern, Web-based graphical user interface which provides access for both central and decentralised administrators. The GUI was developed using a user-centric development process according to ISO 13407 [4]. By ensuring good usability of the administration interface, it is possible to minimise errors and thereby increase overall security. An import interface is provided for automation purposes. The underlying security systems are connected via agents and called target systems (TS) of SAM Jupiter.

The Business Server implements the business logic of SAM Jupiter. This is also where the security and administration policies enforced by SAM Jupiter are defined. The back-end component acts as transaction engine for the repository and provides connections to the supported target systems via agents. The agents run on the target platform, propagate the administrative work completed in SAM Jupiter to the relevant security systems and are also able to load the data to the repository. Standard agents are provided for all major software systems. Customer applications can be easily connected using the SAM Jupiter connector technology. Specific connectors are provided for connecting application security systems, LDAP-based systems, and other company-specific applications.

Using SAM Jupiter, all users and their access rights across all systems in the IT environment of an organisation are administered using RBAC. For this purpose, we have extended the RBAC model to support Enterprise Roles which span over more than one security system and consist of permissions in multiple systems. The resulting model is called Enterprise Role-Based Access Control (ERBAC), which is introduced in [13] and [11] (see figure 6[1]).

As shown in figure 6, ERBAC distinguishes between enterprise and target systems level. On enterprise level, enterprise-wide entities such as users and roles are defined. The target system level describes the underlying security systems and contains the target-system specific entities.

Enterprise Roles include all permissions needed to perform a specific role. Users are then assigned to these roles. The permissions that a user receives through the assignment of a role are propagated to the administered target systems (TS). The Enterprise User definition leads to the creation of user accounts (user IDs) in the TS. 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

[1]For a more comprehensive description of ERBAC and its comparison to the NIST RBAC standard, see [11].

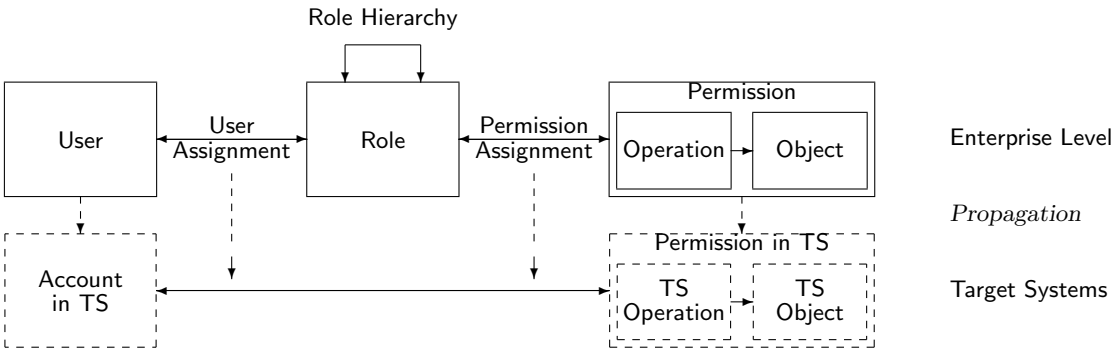

Figure 6: Enterprise RBAC (ERBAC)

does not necessarily cause any update in the target system. The user's accounts receive the permissions defined for the role in the respective TS 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. 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.

A difference between ERBAC and the RBAC96 model lies in the notion of sessions. In our enterprise-wide administration concept, all systems in the enterprise are administered. The actual user sessions, however, are controlled by the respective target systems. Therefore, sessions are not part of the ERBAC model.

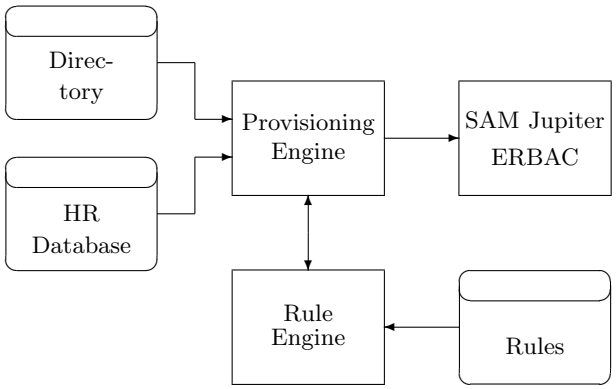

Figure 7: Provisioning and ERBAC

To further reduce administration costs, most enterprises wish to automate administration. The most accurate information about employees can usually be found in the human resources database. Extracted information such as employee number, organisational unit, location or job description can be used to add and delete users automatically as well as update their role assignments. If a new employee starts with the company or changes position, this information is transferred directly from the human resources database to SAM Jupiter. SAM Jupiter automatically transforms the information to role assignments and makes the corresponding updates in the connected target systems. New accounts are created when necessary. In addition, when an employee leaves the company, all of the employee's accounts and access rights are automatically deleted, thus greatly reducing security risks. A prerequisite for automation is the usage of roles that correspond to organisational structures, job descriptions etc.

The functionality described above is called "provisioning". In our system, it is accomplished by the so-called Provisioning Engine, which acts as a front-end for the ERBAC system (see figure 7). It is complemented by a Rule Engine which contains rules as described in section 2.2.3. We use Jess, a Java-based general-purpose rule engine developed at Sandia National Laboratories in the USA [9]. This Rule Engine uses the Rete algorithm to obtain high performance for rule evaluation [8].

The provisioning process principally works in two steps (see also figure 4):

1. User data including all relevant attributes are taken from the HR system and/or other databases such as corporate directories. This data is then compared to the data in the ERBAC repository. Transactions for creating or deleting users and changing user attributes are generated and automatically run against SAM Jupiter. Alternatively, it is possible to work with a differential input which contains only the changes since the last provisioning run. However, comparing the entire user population is much more robust against errors.

2. The users and predefined user attributes are fed into the Rule Engine. All rules are evaluated and a list of roles that the users are supposed to have is generated. This list is compared with the current user-role assignments in the ERBAC repository, and the roles in SAM Jupiter are changed accordingly.

Normally, the Provisioning Engine runs every night to process the changes of the day.

Companies often administer various groups of users differently. For instance, the HR system may not contain all users. Therefore, it is important to flag imported users in the ERBAC repository. A provisioning run then only updates users that have this flag set. Other users are left for manual administration.

4.2 Case Studies

4.2.1 Case Study 1: Bank

A European bank is currently using SAM Jupiter as described in the previous section to administer some 46 000 users. The administered target systems include three RACF systems on the mainframe, about ten Windows domains (NT, 2000 and 2003), SAP, LDAP and application security systems.

Three types of users are administered:

- Internal users (employees)
- External users (e.g., consultants working for the company)
- Technical user IDs needed for automated processes, etc.

Internal and external users are imported from the HR database. For this purpose, all external users are also fed into the HR system. Technical user IDs are administered directly in SAM Jupiter.

More than 80% of the roles that the users need for their work are computed using the Rule Engine and automatically connected or disconnected. This is accomplished by approximately 1000 rules. The rules are based on about 15 user attributes such as 'cost centre', 'branch' or 'organisational unit'. The administration is divided between headquartes and the branches:

- Users in headquarters receive their roles primarily as a result of the cost centre to which they belong. The cost centre mainly describes the function of the user.
- Users in the branches receive their roles mainly according to the branch (location) in which they work. Users often work for more than one branch. In this case, they receive the roles for all of these branches.

These different approaches can easily be reflected in different rule structures for both domains.

Most of the rules are quite simple. They are based on *branch*, *cost centre*, *company* and some other user attributes. A typical rule looks like:

```
IF   User-CostCentre = "AB2500" AND
     User-Company    = "Bank1"
THEN Assign Role "Bank1-Cashier"
```

Rule and role definitions are not static. Organisational, functional and technical changes lead to changing requirements for authorisations. Some typical changes are:

- Changes in the operational structure which encompass new or changed applications, changing job functions etc. which lead to new or changed rules. On the average, two rules are changed per day in the bank.
- Changes in the organisational structure of the bank also occur at longer intervals. For example, branches are merged, resulting in the following changes:
  - The organisational attributes of the users in the merged branches change.
  - The rules referring to the old branches must be adapted.

  In such cases, the rules are migrated in a concentrated action.
- New user attributes are introduced for use in the provisioning process. This does not happen very often – normally about once a year. An example was the introduction of SAP-HR, which provided more information about the users.

To handle these changes of rules in productive environments, a rule life-cycle management has been implemented:

1. New rules are written in a deactivated state.
2. As the change of a rule can have a great impact on user authorisations, it is important to validate new rules before letting them take effect. An incorrect rule may lead to the revocation of important authorisations, which in turn could prevent brokers from trading or ATMs from dispensing money, for example. Validation is performed by simulating a modified rule and listing the total number of affected users and their identifiers. Based on this information, the rule administrator determines whether designated users are affected and whether the rule has the correct effect. The validation can be performed by the following means:
   - Checking whether sample users are or are not affected by the rule,
   - Checking whether the number of affected users matches expectations or
   - Checking whether the number of affected role assignments matches expectations.
3. After testing, a rule is activated and takes effect during the next provisioning run.
4. Due to organisational or business related changes, a rule might later become obsolete and is then deactivated or deleted.

4.2.2 Case Study 2: IT Service Provider

Our second case study describes the identity management solution implemented at an IT service provider. This organisation administers some 150 000 users in RACF, several SAP systems, an LDAP directory and customer-specific application security systems. Because of extensive automation, they manage with three administrators.

They also administer different types of users:

- Internal users are administered in the HR system and imported into SAM Jupiter.
- External users are administered directly in SAM Jupiter; roles and access rights are assigned manually.

The service provider serves several clients with different administration concepts due to differing business requirements. Provisioning of users and their attributes is implemented for all companies. Automated provisioning of roles is performed for one company with about 15 000 users. The roles that a user receives are again computed using the Rule Engine. About 2000 rules based on six user attributes are used. Regular changes of rules occur very infrequently. A large number of rules must be adapted only after reorganisations.

The provisioning process runs about once a week.

Users	64 000	150 000	19 000	10 000	11 500	30 000
Users with roles	36 000	7 500	18 000	6 000	10 500	14 500
Roles per user	1	1	1	1	5	3
Roles	3 000	50	1 800	400	2 800	2 500
Permissions per role	34	3	10	14	2	2

Table 1: Examplary Figures for Users and Roles

4.3 Conclusion

We have made some general observations in our identity management projects with role-based access control. Different types of users may be administered in different ways. Reasons for this can be the following:

- Different business needs.
- Master user data is available in different databases with different attributes. For some types of users, no data might be available at all. In some organisations, the RBAC repository contains the only reliable source for external users.
- More and more companies are outsourcing their IT to service providers. These service providers have customers with entirely different administration philosophies.

Organisations implementing RBAC do not necessarily do so for all of their users. It is common knowledge that the process of defining roles is quite expensive (see e.g. [14]). Organisations might decide to do one of the following:

- Introduce RBAC step by step, e.g. branch by branch.
- Implement RBAC only in areas where they see the greatest benefit. Many service providers, for instance, introduce RBAC only to those customers who request it and are willing to pay for it.

Therefore, it is important to use an access control model which provides roles, but does not force organisations to use them. Our ERBAC system SAM Jupiter allows administration via roles as well as the direct assignment of permissions to user accounts. This conclusion is also backed by table 1 which shows the quantity structure for six companies that have implemented role-based administration with SAM Jupiter. The table lists:

- The total number of users the company administers.
- The number of users which are administered using roles.
- The average number of roles per user.
- The total number of roles.
- The average number of permissions per role. These permissions can be groups or authorisations in the respective target system (see also section 4.1).[2]

The table gives some interesting insights: The total number of roles, the percentage of users administered via RBAC and the average number of permissions per role vary quite widely. On the contrary, it is interesting to note that the average number of roles per user is quite low for all companies and does not exceed five.

When considering the use of RBAC in practice, we have often seen that pure RBAC provides a good methodical basis, but is too expensive to administer. Generic roles (see [11]) and rules are extensions which make the use of RBAC feasible.

Rules have proven to be a very useful supplement to roles. In addition to use as access decision rules (see [5, 12]), rules are primarily used to automate the provisioning process. The following characteristics of a rule-based provisioning function have proven to be essential:

- Rules can easily be adapted to reflect organisational and operational changes.
- Attributes used in the Rule Engine are generic so that new attributes can easily be added.
- Normally, it is sufficient to run provisioning once a day (or even less often). Urgent administration can be performed directly in the RBAC system.

Generally, the rules are quite straightforward and based on a manageable number of attributes (five to fifteen). Although changes of rules do not occur too often, an effective rule lifecycle management as described in section 4.2.1 is essential. The most important rule-related figures of our case studies are summarised in table 2.

	Bank	IT SP
Users	46 000	150 000
Target systems	25	10
Attributes in rules	15	6
Rules	1000	2000
Rule changes per week	10	1

Table 2: Quantity Structure for Case Studies

Our conclusion is that ERBAC with rule-based provisioning has proven to be an efficient concept in practice for the administration of users and their access rights. It allows a high level of automation, thus reducing administration costs while ensuring a high security level.

[2]In most cases, groups in target systems like RACF, ACF2, Windows 2000, LDAP directories or Netware are used.

5. 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] M. A. Al-Kahtani and R. Sandhu. Induced Role Hierarchies with Attribute-Based RBAC. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), Como, Italy*, pages 142–148, June 2003.

[3] M. A. Al-Kahtani and R. Sandhu. Rule-Based RBAC with Negative Authorizations. In *Proceedings of the 20th Annual Computer Security Applications Conference, Tucson, Arizona, USA*, December 2004.

[4] A. Beu, A. Kern, and J. Schwagereit. "Das User Interface ist wunderschön...". Der benutzerzentrierte Gestaltungsprozess nach ISO 13407 in der Praxis. *Java Magazin*, pages 28–35, May 2002.

[5] R. Chandramouli. Business Process Driven Framework for defining an Access Control Service based on Roles and Rules. In *23rd National Information Systems Security Conference, Baltimore, Maryland, USA*, October 2003.

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

[7] 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.

[8] C. L. Forgy. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. *Artificial Intelligence*, 19:17–37, 1982.

[9] E. Friedman-Hill. *JESS in Action. Rule-Based Systems in Java*. Manning Publications, Greenwich (USA), 2003.

[10] G. Gebe. *Managed Authorization Services: Implementing Roles, Rules, and Policies*. Burton Group, Midvale, Utah (USA), December 2004.

[11] 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.

[12] A. Kern, M. Kuhlmann, R. Kuropka, and A. Ruthert. A Meta Model for Authorisations in Application Security Systems and their Integration into RBAC Administration. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT 2004), Yorktown Heights, New York, USA*, pages 87–96, June 2004.

[13] 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.

[14] M. Kuhlmann, D. Shohat, and G. Schimpf. Role Mining – Revealing Business Roles for Security Administration using Data Mining Technology. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT 2003), Como, Italy*, pages 179–186, June 2003.

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

[16] 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.