Enabling Technologies for Role Based Online Decision Engines

Thomas Gebhardt, Thomas Hildmann
Technical University of Berlin

29. February 2000

Abstract

The implementation of an RBAC system at the TUB led us towards a specific meaning of role-based access control, where decisions regarding user permissions are based on online evaluation of a distributed RBAC model, we call this approach role-based online decision (RoBOD). Requirements of our project partners and the integration of our solution into application environments showed that a number of enabling technologies, i.e. authentication and secure communication, are necessary to employ RBAC in networked application architectures.

Contents

1 Introduction **1**

2 Development of RoBOD **2**
2.1 End-to-End Security over the Internet 2
2.2 SPD Party in Germany 2
2.3 MultiPLECX 3
2.4 Role Based Online Decision . . . 3

3 Features of Role Based Online Decision **4**

4 Enabling Technologies **5**
4.1 Authentication 5
4.2 Secure Communication 6
4.3 Role Based Auditing 7
4.4 Features of the Modelling Language 8
4.5 Persistence 8

5 Conclusion **9**

1 Introduction

The interdepartmental research center PRZ of the Technical University of Berlin (TUB) developed its own implementation of an RBAC interpreter. This interpreter and the concepts of it's implementation is a result of the work on several projects and the requirements found during these projects. The TUB's RBAC system is not a result of theoretical motivations worked out to an implementation. It is the result of several requirement analyses and implementations using an incremental, iterative development process. Because of the high practical relevance of this work we decided to give a report of our work until now and how we plan to go on. This can be useful for all users involved in projects with common issues as discussed in this paper and we hope to give some new impulses for further theoretical work on the discussed areas.

Section 2 will describe the history of the RoBOD development until now and will show the motivation of this development.
Section 3 will summarize the features RoBOD is based on.
Section 4 founds the need of a set of technologies needed for RoBOD, summarizes the requirements to the implementation and discusses the TUB's implementation of this technology.
The paper ends with an outlook on our current and future research and implementation issues in section 5.

2 Development of RoBOD

This section will give an overview of the most important development steps and design decisions. It will answer the questions: "What is Role Based Online Decision good for?" and "Why is it designed this way?"

2.1 End-to-End Security over the Internet

Starting point of our work on RBAC systems was the European project E2S (End-to-End Security over the Internet). The TUB had two tasks in this project: 1. Development of security tools [1][2] and 2. as a trial partner using the results of E2S for the TUB's administration [3].
First step was the development of a secure e-mail tool with automatic encryption/decryption and signing. One of the most difficult requirements was the management of fast changing usergroups [4]. For example:

- A user has a personal and a post e-mail adress.
- A user can select a representative for his/her post in case of holidays etc.
- A group leader can invite new users or can cancel the membership of others.
- A group can be a member of an other group.
- If a user changes his/her post (s)he will get the rights of his/her predecessor.

This raised the idea of roles in this groups and permissions bound on the roles.
We started our work on the first RBAC database. The work on other services we had to implement near the e-mail service got us to a centralized RBAC database because the distribution of access and group lists did not match the requirement of a kind of realtime behaviour [5].
One goal of the project was a web-based user-interface enabling the maintenance of the model not only for the RBAC administrator but also for group-leaders down to each user. This brought us to the concept of the self-protection of the RBAC model. This was the first time we realized the issue of access-roles and business-roles. To make the model userfriendly enough to find acceptance we tried to model both classes of roles. From the user's point of view they managed business-roles. The model mapped the business-roles to access-roles [6].

2.2 SPD Party in Germany

Our presentations raised the interest of the administration of the German party SPD. Their requirements differed from the ones of the TUB administration in various points.
E2S introduced intranet services in the TUB's administration. The starting point in this project was:

- No intranet services.

- No security concepts.
- No existing databases.

The SPD project was very different. The administration of the SPD have already had a number of services available in their intranet. The situation looked like this:

- Existing intranet services.
- Basic authentication on WWW servers for each service.
- A number of existing databases.

We needed a mapping between access-roles and basic authentication accounts. Our approach was to build an application level firewall configurations based on our RBAC system.
The result of this project was a web-based application level firewall module which used a ticket mechanism based on cookies for a single-sign-on and which used the role-interpreter for access decisions.
We redesigned the user-interface for the needs of the new project. The work on this user-interface initiated the current work on a model-independent user-interfaced which can easily be adapted. Design and function is separated in this new interface.

2.3 MultiPLECX

Our part at the current project ending in July 2000 is to integrate RBAC in a business-to-business environment (B2B). The new idea is to represent several companies and their business relations in a single RBAC model. The most important issue was the visibility of the modelling details. A member of a company A should only be able to see the representative roles of the company B and vice versa. We needed to introduce this new aspect of visibility into the RBAC concepts [7].

To protect the model for even the provider company of a service we developed a distributed RBAC. Only if the model runs on the company's own machine it can be ensured that they are the only ones with access to the modelling data. The collaboration is realized with interface definitions written in the modelling language.

It is a complex task to build a consistent model with consistent constraints and policies for a single company and even more so for multiple companies. The RBAC system for MultiPLECX is designed to handle different models in a single interpreter environment. Each model is protected against other models.

Result of MultiPLECX is an interpreter for an RBAC modelling language which is complete and object-oriented. Complete in this context means: Has loops, variables and can compute. We have choosen the object-oriented approach because these concepts are approved, flexible and match to the business-views.

2.4 Role Based Online Decision

If we are talking about RBAC we are thinking of the result of our projects. This is always a source of misunderstandings because the approach differs in some major concepts from other RBAC solutions. So we try to specify what our meaning of RBAC is. At the current state we do not really have a definition of RoBOD so we describe this approach with its features.

3 Features of Role Based Online Decision

The requirements and use cases formulated by our project partners where aimed at the usage of RBAC in networked application architectures. Flexibility and the ability to adapt to fast changing environments are key requirements to support the integration of RBAC in business applications like electronic marketplaces. The use of access control mechanisms on the operating system level is not appropriate for this class of applications, since they are often comprised of many distributed components (Applets, Servlets, CGIs, DLLs, etc) implemented on different platforms in heterogenous networks. Therefore these applications have to layer their own security models on top of network and operating system environments. To incorporate RBAC functionality in these client/server architectures it has to be provided as a service accessible all the time by all other application components. We call this special class of RBAC usage "Role Based Online Decision" (RoBOD).

A RoBOD system contains an RBAC model of the application and is queried by application components about the permissions of users, depending on their membership in appropriate roles. The complexity of the RBAC model is application dependent and can range from $RBAC_0$to $RBAC_3$, as described by the RBAC model family[8, Sec. 4]. We worked out a set of characteristic features of RoBOD systems and classified them as necessary (basic) and optional (extended) features. These features where deducted from our requirement analysis or emerged during the incremental development of our implementation.

The basic features of RoBOD are:

- Implements a distributed access decision facility/access enforcement facility (ADF/AEF)[9]. This is a demand of the targeted application architecture.
- Access control of the model is provided by the model itself. In [8, Sec. 5] Sandhu describes how this can be achieved through the use of administrative roles.
- All models of the RBAC model family are supported to provide sufficient modelling power for complex applications.

Our implementation also includes the following extended features:

- Support of the modelling of business and access roles in one model.
- Object oriented modelling, our RBAC interpreter implements a fully object oriented language providing features like encapsulation, inheritance and polymorphism.
- Distribution of the model over several instances.
- Support of role delegation in distributed models.
- The decomposition and distribution of the model allows horizontal and vertical scalability.

There are a number of important advantages to this approach. RoBOD was successfully used in at least two different application environments in the EU project MultiPLECX. Our work on introducing RoBOD systems in the administration of our University is going on. This shows the usability and relevance of this kind of RBAC. The

modular architecture allows to supplement existing systems with RoBOD services. The application and the RBAC model can be developed incremental, starting with simple RBAC concepts and adding complexity as the application evolves. Powerful concepts for workflow management and collaborative work can be integrated with the RoBOD system. A possible disadvantage is, that a failure of the RoBOD service or a communication breakdown can disable the complete application. This can be prevented by the employment of industrial-strength solutions to secure the availability of the RoBOD service (high availability, server clustering, load balancing). The encapsulation of the access control in one system component on one side minimizes the number of possible points of attack, on the other hand a corruption of this component could be fatal. The possibility to distribute the model allows the implementation of a security model which best fits the demands of the specific application. It is of course mandatory, that the RoBOD service is protected by state of the art security technology.

4 Enabling Technologies

The implementation of a RoBOD system has to employ a number of technologies which are not part of RBAC research but whose integration with RoBOD concepts is necessary for the development of useable solutions. We have identified five of those enabling technologies, but since our implementation is still expanded with new concepts and features, there is a good chance that we will find more. The enabling technologies discussed in this section are necessary for implementing the basic features of RoBOD systems and are also sufficient to implement the extended features. For each technology we justify the necessity of use in RoBOD systems, discuss probable requirements and present our implementation as exemplary solution.

4.1 Authentication

To make a decision about the rights of a user it is important to identify the user in a suitable way. The authentication method has to depend on the value of the resources which are protected by the access control system.
Therefore the TUB developed a modular authentication concept with scalable security levels. The following levels of authentication are imaginable:

Terminal-based authentication
: (identification by admittance): A user who has access to a terminal is identified as the owner of the terminal. No user-authentication is needed. The terminal is identified over an identifier given by the underlying network technology (e.g. IP-Number, ARPA-Number, Leased-Line-Identifier, ...).

Username/Password authentication
: (identification by knowledge): A user needs to identify himself by the knowledge of a username with the according password. The terminal can be changed or a username/password-pair can be additionally bound on a terminal.

Public-Key-based authentication
: (identification by ownership): A user is identified by the an encryption using a secret-key which can be validated using the according public-key by the authentication module.

Biometric authentication (identification by physical characteristics): The user is identified by an unchangeable physical characteristic.

All identification methods can be combined to increase the security level.
An authentication can be proceed on each access attempt or once resulting in a credential which can than be automatically checked on any following authorization process (single-sign-on).
An authentication can be used for role based access control decision whenever a mapping between credential or authorization method to the modelled user-identity is possible.
The implementation of the TUB's authentication module divides the authentication process into two phases:

1. The identification of the user and the hand out of a ticket.
2. The demonstration of the ticket to the RBAC interpreter.

We followed the approach to not write the role information into the ticket because of the following reasons:
In a multi-company such as a B2B environment the role membership is confidential. On the other hand the role membership can also be evaluated in the RBAC engine at the decision time. This makes it possible to make a user-role-mapping but conceal the roles the user is member in. The outside world is only able to observe the effect of this memberships.
Other reasons to make the user-role-mapping online are: Constraints on role-memberships which depend on external or fast changing parameters and changes in the model which must take effect directly.

We implemented the tickets as a defined datastructure containing a user-identifier (can anonymize the real-life identity of the user), the life-time of the ticket, a session timeout stamp and the creation time all digitally signed with an authentication-server-key.
The advantage of using a single key to sign all tickets is that only the phase 1 service must be integrated in a public-key infrastructure (PKI). All applications which have to check the validity of the ticket do only need to know the public-key of the authentication-server. In our case we make requests to the RoBOD engine sending the ticket as a parameter and leave the checking of the ticket to the decision service.

4.2 Secure Communication

Secure communication is not only necessary to prevent spying of the transmitted data. It is also a way to avert from replay attacks. A signed ticket (as described in 4.1) sent over an uncrypted communication channel can be picked up and resent to fake request of real ticket owner. If the whole communication between application and RoBOD engine and the communication between authentication service and client machine just as the communication between client machine and application is encrypted there is no risk of ticket hijacking.
The need of a full encryption on each communication channel between applications, clients and the online decision engine is one of the biggest issues of the RoBOD approach. There are a many approaches to encrypt a communication channel. Three typical approaches are

- Application-level encryption of the data (including sequence numbers, timestamps etc. to prevent replay attacks).

- Virtual private networks tunneling the communication between the communication partners (e.g. also possible on lower network layers, e.g. IPv6).
- A point-to-point tunneling of the communication channel (with protocols like SSL).
- Direct and non-wiretappable point-to-point wiring.

The encryption method for the according communication channels depend on the security level of the application, the used platforms, availability of existing secure channels on lower network layers, etc. This makes it impossible to give a recommendation in this paper. We just want to determine that secure communication channels are needed between each communication point in the RoBOD environment.
Nevertheless we can discuss the solutions used in our projects in the past.
In the E2S project we used an SSL encrypted connection between client and web-server hosting the application frontend. The authentication server was also web-based and SSL secured. The communication between application and RBAC interpreter was tunneled over secure shell (SSH) connections.
The SPD used a very common approach using an SSH connection between the application level firewall systems and the RBAC interpreter.
MultiPLECX additionally uses SSL tunneled CORBA connections between the various applications. Such a CORBA over SSL is supported by the ORBacus ORB implementation [10]. SSH connections are not longer used. The incorporation of security services into CORBA is still in progress. Therefore the SSL feature of ORBacus is not yet a part of the official CORBA standard.

4.3 Role Based Auditing

Especially in a multi-company or inter-departmental distributed environment decisions of the RBAC system can have consequences of high magnitudes. But also in a centralized single-company system logging is an important task to ensure the compliance of the security policy. This raises the need of a well organized and non-repudiable logging of every transaction and the according decision from the RBAC system.
As described in 2.3 visibility is a big issue in business-to-business environments. This applies also to logging of employee data in some countries because of local laws (e.g. the German privacy law). The same regulations can force a provider of special services to record every operation on sensitive data.
A role based auditing system must be able to at least protect the logged data and to provide access to the data depending on the role of the querying user. To make a logging information non-repudiable we demand a digital signature of every transaction from all communication partners.
An example: A user U buys a good at the electronic market-place of market-place provider M. The good is distributed by the distributor D and delivered by the supplier S. The first transaction is between U and M. Therefor the logging information needs a signature of U and M on the order of U. The second transaction is between M and D and the third is between D and S. The result has to be a set of logging information with a chain of transactions from U to S with orders which can not be repudiated by any of the business partners. On the other hand it should not be possible for a noninvolved third party or even for a distant link in the chain to spy on the busi-

ness of an other partner.
Transactions between two links in a chain are always physically logged on the storage of one of the two links or a trusted third party.
It must be possible to make statistical analyses on the auditing data without the knowledge of the concrete content.
Data logged in the auditing system must be encryptable in a way that can only be decrypted again with the explicit consent of the two partners.
The system should be able to encrypt on a role based level (e.g. all users of the role X have the ability to decrypt the information).
Our implementation is based on a relational database schema. This schema contains signature data for each entry and can hold one or more public-key encrypted session keys of encrypted data entries. Other tables are holding statistical information, timestamps and system states.
The auditing module handles two requests: Adding data to the logging database and reading data from the logging database. All request can contain a set of encrypted, signed or encrypted and signed data. Data entries can be named by the application. Such symbolic names of data can be used for read requests (e.g. all auditing data dealing with transactions of partner X or user Y).

4.4 Features of the Modelling Language

An RBAC model implemented by a RoBOD system is formulated in some kind of modelling language. If the model is directly encoded into a database a query language has to be used to formulated the queries and the database must be able to express all envisaged states and features of the model. Since RoBOD systems should support all models of the RBAC model family, the modelling language must be able to express concepts like role hierachies and constraints.
To support the role engineering and modelling process our RoBOD system provides an object oriented modelling language. The language design was heavily influenced by Java, C++ and Smalltalk. The access decision facility therefore is an interpreter which at startup is supplied with the model and an initial state, both formulated in the modelling language. Queries to the interpreter are also formulated as statements, usually method calls on model objects, in this language. Changes to the model are too applied by appropriate method calls.
As a consequence of this concept the language has to ensure, that calls to the interpreter can not corrupt the model. Access control mechanisms must be embedded into the language and the model must be provided with all functionality the model needs to protect itself. Our language provides the following features.

- no public access to object attributes
- methods can be protected against overloading
- objects can contain their own access control policy
- the formulation of conditions for the execution of code is enforced
- data structures for the secure storage of informations are provided (ie sets and lists).

4.5 Persistence

In a RoBOD system the state of the RBAC model which contains the access information and

potentially many more informations must be stored to a persistent storage medium. This allows backups of model states and recovery from system crashes.
The storage and retrieval of model state information must be managed in an efficient way because usually the system will stay online during the process and response times should not be affected. Models for large companies and organisations can become very complex and voluminous, therefore the requirement that the storage/retrieval process preserves the model integrity is obvious but not trivial.
In its simplest form an RoBOD system can be implemented as a customized database, in this case the persistence and integrity of the information is provided directly by the database technology. We designed our solution as an interpreter of an object oriented modelling language, where the model state is encapsualted in the objects at runtime. To ensure the persistence of these objects they are stored in a database, in our implementation we used the Object-Relational DBMS postgresql, which is available on multiple plattforms and supports most SQL statements and access via ODBC [11]. The internal representation of the objects in the interpreter is designed to support the efficient conversion into a format suited for storage in the database. To ensure the model integrity, transactional behaviour is implemented on different levels.

- The storage and retrieval of an object has to be a transcation, in case that parts of the operation fail a rollback recreates the last valid state of the object.

- A query to the interpreter can change the state of several objects, therefore it has to be treated as a transaction, if an error oucurs the model must return to the last valid state.

- Transactions can also reside on a higher level above the interpreter, such transactions contain multiple queries to the interpreter, the failure of one of these queries must result in the revocation of all queries since the start of the transaction. This third level of transactional behaviour is one of our current research topics.

5 Conclusion

We have described the development of our RBAC implementation so far and outlined the requirements and enabling technologies upon which our solution is based. The decision to implement an interpreted, object oriented modelling language capable of distributing RBAC models yielded a great number of interesting questions. Since our system is still a work in progress some issues remain open and some topics could not be treated in sufficient detail. Therefore we plan to report more research results on key topics like design of the modelling language, transactions, integration with workflow concepts and design patterns suited for the implementation of RBAC models in our modelling language. Our future research topics will include the external representation of models and the extension of the reflective features of the interpreter.

References

[1] Bartholdt J., *Deliverable E2.4: Secure Telecooperation Software*, End-to-End Security over the internet, 1997

[2] Bartholdt J., et al., *Deliverable F1: Secure Telecooperation Software*, End-to-End Security over the internet, 1997

[3] Nagel K., *User Requirements for Administration Sectors*, E2S/WP.C/TUB/001, End-to-End Security over the internet, 1996

[4] Hildmann T., *Entwicklung eines sicheren, erweiterten E-Mail-Systems*, Diploma Thesis, TU Berlin, 1998

[5] Bartholdt J., *Konzeption und Implementierung einer Zugriffskontrolle innerhalb eines virtuellen privaten Netzwerkes fuer private und oeffentliche Verwaltungen*, Diploma Thesis, TU Berlin, 1998

[6] Bartholdt J., Nagel K., Hildmann T., *Abgesicherte Internet-Umgebung mit Hilfe rollenbasierter Zugriffsmechanismen fuer WWW- und E-Mail-Dienste*, Proceedings of 5. Workshop Sicherheit in vernetzten Systemen, DFN CERT & PCA, p.1-36. March 1998

[7] Hildmann T., Bartholdt J., *Managing Trust between collaborating Companies using outsourced Role Based Access Control*, Fourth ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA, 1999

[8] Sandhu R.S., et al., *Role-Based Access Control Models*, IEEE Computer, 1996, Vol. 29, No. 2

[9] ITU Recommondation X.821: *Data Networks and Open System Communications Security: Open Systems Interconnection - Security Frameworks for Open Systems: Access Control Framework*, November 1995

[10] Object Oriented Concepts, Inc.: www.ooc.com, www.orbacus.com

[11] www.postgresql.org