

## SECURITY MANAGEMENT IN A DISTRIBUTED OPEN ENVIRONMENT

Monika Calitz<sup>a</sup>, Dr R. von Solms<sup>a</sup>, Prof S. H. von Solms<sup>b</sup>

<sup>a</sup> Faculty of Computer Studies, Port Elizabeth Technikon, Private Bag X6011. Port Elizabeth, 6000, South Africa

<sup>b</sup> Rand Afrikaanse Universiteit, Johannesburg

### 1. INTRODUCTION

A recent corporate trend emphasizes business restructuring and global strategies on a business level. There has been a tendency to decentralize control and put decision making closer to the operational centres. With modern information technology, especially telecommunications systems, decision making and operational control can be delegated to operational units and control information can be available to headquarters on a real time basis.[1]

In the traditional computing environment, centralized processing was associated with centrally managed and secured information. Typically, a person from the operations department responsible for enrolment of users would grant access to resources such as terminals, application programs, etc, while a database administrator would be responsible for managing access control to the database.

However, the computing environment is becoming increasingly decentralized and distributed to possibly remote sites. Users at remote sites will have to apply for certain resources and access rights from the central site. These rights are based on a description of the job to be done and/or resources and access rights required. The security of information is thus managed at the central site. This is undesirable because the requester and his/her needs are not known at the central site and this can pose a serious security threat. The ideal situation would thus be that information security should be managed at the remote site.

This paper will address this issue and describe how information security can be managed effectively in a distributed environment at the remote site in a non-technical way.

## **2. CENTRALIZED VS DISTRIBUTED SECURITY MANAGEMENT**

In establishing information security, most systems provide user identification and verification. For most systems, this means that the users sign on to the system with their name and are then required to supply a password. Another step of information security is the securing of data. In an environment where data is properly secured, each data set in the system has an owner and the owner's identification and the data set classification are part of the data. In such a secure environment, each owner must specify for his data sets who can access them and the manner in which the data set can be accessed. The manners of access are : read, write, alter, delete, and execute. If data security is to work properly, then each valid user is authorized by the owner to access the data set and the user's authorized actions are identified. When a user tries to access a dataset, the system validates the user as well as the type of access that user is attempting to gain. Unauthorized attempts to access data are recorded and can therefore be investigated by management. [1]

In a centralized computing environment, someone requiring access to any information system resources would be required to apply to the data processing department. Such a person would be assigned a user identification and password by the data processing department. In addition access to certain resources, applications, files, objects had to be granted explicitly to the requester as well. This might have been a combination of read/write rights on certain operating system files, assignment of home directories or libraries, access to run certain applications, and rights to certain tables within a corporate database, etc.

A further problem is that if two or more employees are doing the same job, this process has to be repeated and no guarantee can be given that the same data access rights are requested by and granted to the different employees.

These problems have certainly escalated as organisations have distributed their information systems to remote sites. To solve these problems, information security will have to managed to a certain extent at the individual remote sites. This leads to another problem. No system specialists and/or technical personnel may reside at the remote sites. If information security is to be distributed along with the information systems to remote sites, then the security management mechanisms must be of such a nature that a non-technical manager can be empowered to control security at his/her local site.

### 3. MODEL FOR OPEN SYSTEM SECURITY (MOSS II)

The MOSS II model [4] proposes a possible solution to this problem by using role profiles .

To manage information security from a business point of view, roles should be associated with certain job functions in a company. Each of the roles is allowed to perform certain transactions. Each transaction in turn, will access certain objects, such as files, database tables, or records and will be allocated the required rights to it. A profile of what is allowed to be done is created for each possible role.

The role oriented approach has the advantage that security can be decentralized due to the simplification of the administering environment. The MoRP model [3] proposes a 4-level model where level 1 is the individual user profile, level 2 is the role profile, level 3 is the transaction profile and level 4 is the resource profile. The skills required for administering profiles range from high technical skill at level 4 through low technical skill at level 1.

To demonstrate the use of roles, the following example describes a situation from a banking environment, where there may be roles for bank tellers, accountants and managers. Each of these jobs may perform a finite set of transactions, e.g. :

	<b>View Customer Account Balance</b>	<b>View Budget Details</b>	<b>Change Budget Details</b>
<b>Bank Teller</b>	allowed	not allowed	not allowed
<b>Accountant</b>	not allowed	allowed	not allowed
<b>Manager</b>	allowed	allowed	allowed

**Fig 1. Role/Transaction Table**

The definition of roles and the creation of the associated profile. i.e. the identification of transactions that each role may perform as well as the allocation of rights to required resources, should take place centrally. These profiles are created on a strict need-to-know basis. These roles, with their underlying profiles are made available at remote sites where they can be assigned to and withdrawn from employees by a non-technical manager. The roles and associated transactions form an abstraction from the technical details of specific access rights on objects such as programs, files, tables, records and fields. According to MOSS II, a non-technical manager at a remote site can then assign employees at that site to predefined roles as required, without the technical knowledge about such objects. With this, decentralized security management would be effectively implemented and managed.

#### 4. IMPLICATIONS OF ROLE PROFILES

A user may be assigned more than one role. At login, a user would be required to choose from a menu in what role he would like to work for that session. The menu should be tailored to that particular user's role assignments. The role chosen by the user becomes the "active" role. For the duration of that session, the user would then be allowed to perform only the transactions associated with this active role, even though he may have been assigned other roles with a different set of transactions by the security manager.

Time restrictions could be included in the enrolment of users. This would have the effect that a user can only work in a particular role during certain hours of the day.

All employees performing the same job will have the same rights, which should be completely adequate to do the required job effectively. Local branch managers can then also deactivate the link of an employee to all roles when the employee is on holiday, or absent from office. Complete information access control can thus be delegated to branch managers through the use of role profiles.

#### 5. THE ESTABLISHMENT OF ROLE PROFILES

I. Mohammed and D. Dilts [2] propose an 8 step methodology to establish a dynamic user-role-based security model. The steps include the building of an initial user-role definition hierarchy, the identification of events (transactions), the identification of protected entities (objects) and the assigning of privileges.

To define, create and implement role profiles for the MOSS II model in full, the following tasks should be performed. This is described using an example.

##### Task 1 - Identify Roles.

A person with adequate business knowledge, at the central office, determines what typical roles are encountered in a banking environment. Assume only three roles are identified, i.e. bank teller, accountant and manager as in figure 2.

<b>Roles</b>
Bank Teller
Accountant
Manager

**Fig 2: Roles in Banking Environment**

**Task 2 - Identify Transactions.**

Identify all the typical transactions performed at a bank. Again, this is done centrally by a business analyst and the result in our example may be: view customer account balance, view budget details and change budget details, as in figure 3.

Transactions
View Customer Account Balance
View Budget Details
Change Budget Details

**Fig 3: Transactions in Banking Environment****Task 3 - Determine Role/Transaction Table.**

A business analyst determines which of these transactions need to be performed by each of the roles. A table, such as figure 1 should be compiled.

**Task 4 - Determine Rights to Transactions.**

A person, with knowledge of the application systems, determines which rights are required to perform each of the identified transactions. These rights includes access rights to data objects, rights to resources, etc. Figure 4 will be compiled.

Transaction	Object 1: Customer Record	Object 2: Budget Record
View Customer Account Balance	read	none
View Budget Details	none	read
Change Budget Details	none	write

**Fig. 4: Transaction Rights**

**Task 5 - Assign Rights to Transactions.**

The database administrator assigns the necessary rights to each of the transactions.

The following, figure 5, relationships between the identified roles, transactions and rights to objects can be established.

<b>Roles</b>	<b>Transactions</b>	<b>Objects</b>
Bank Teller	View Customer Account Balance	Customer Account Record
Accountant	View Budget Details	Budget Record
Manager	View Customer Account Balance	Customer Account Record
Manager	Change Budget Details	Budget Record
Manager	View Budget Details	Budget Record

**Fig 5: Roles/Transaction/Rights Relationships**

The following, figure 6, Transaction/Object/Rights File will be created following these tasks.

**Transaction/Object/Rights File**

<b>Transaction</b>	<b>Object</b>	<b>Rights</b>
View Customer Account Balance	Customer Record	read
View Budget Details	Budget Record	read
Change Budget Details	Budget Record	write

**Fig 6: Transaction/Object/Rights File**

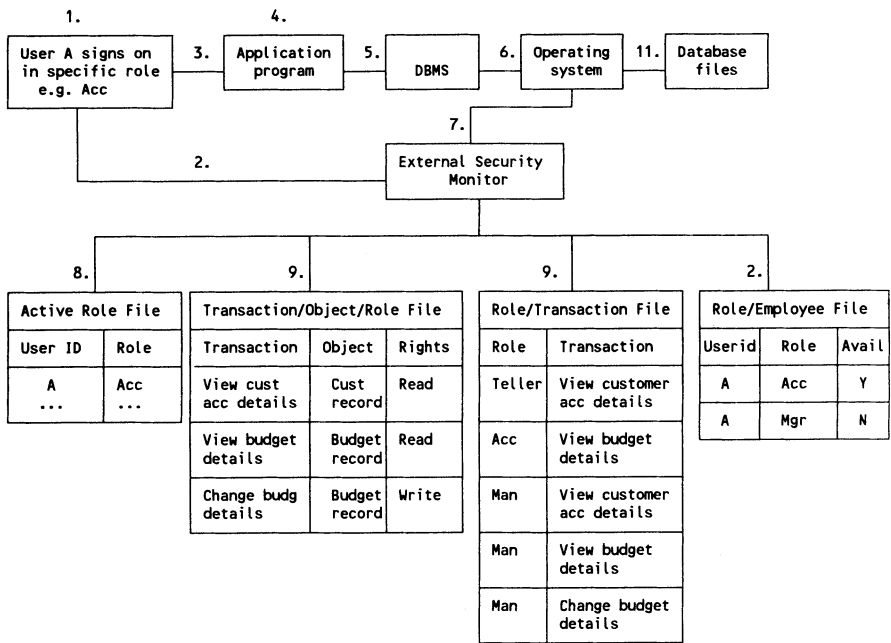
At this stage the roles as well as the accompanying profiles have been defined and can be made available to the remote sites. At a remote site, a non-technical manager has to grant rights to employees working at that branch, to enable them to have access to all the objects required to perform their work. To manage the security at the site, an application program will use the role profiles, previously defined, and will allow the manager to assign each employee to the roles required by his or her work, or deassign them from roles when he or she is on leave or changes jobs within the organization.

## **6. THE MECHANICS OF ROLE PROFILES**

When a user signs on to a specific role, that role becomes active and that user will only be allowed to perform the transactions associated with that role for that session. This role chosen by the user will be activated in an 'Active Role File', see figure 7. When the user wants to perform a certain transaction the following series of tasks is performed.

Tasks performed when a user wants to execute a certain transaction:

1. User A attempts to log on and chooses to work in role R.
2. The external security monitor checks whether user A may log on to role R, and whether role R is available to user A at this particular time.
3. User A executes an application program. The application program performs a transaction T.
4. To execute the transaction T, the application program requires certain resources (objects).
5. The application program passes these requests onto the database management system.
6. The database management system passes it onto the operating system.
7. Before the operating system 'fetches' the object from the physical database files, the External Security Monitor (ESM) is activated.
8. The ESM checks in the Active Role File which role is currently active for user A.
9. The ESM then checks whether the transaction T user A wants to execute is allowed in the role R, which is currently active for user A.
10. If the transaction is allowed, the ESM will give the operating systems clearance to make the resources available to user A according to the Transaction/Object/Rights File.
11. The operating system will fetch the necessary data items and pass them back to the requesting application program.



**Fig 7: Mechanics of Role Profiles**

The issue of the External Security Monitor that liaises with the operating system and finally determines whether the user should or should not be allowed to complete the transaction, will not be discussed in this paper. It forms part of a related research project.



## 7. A PROTOTYPE SYSTEM

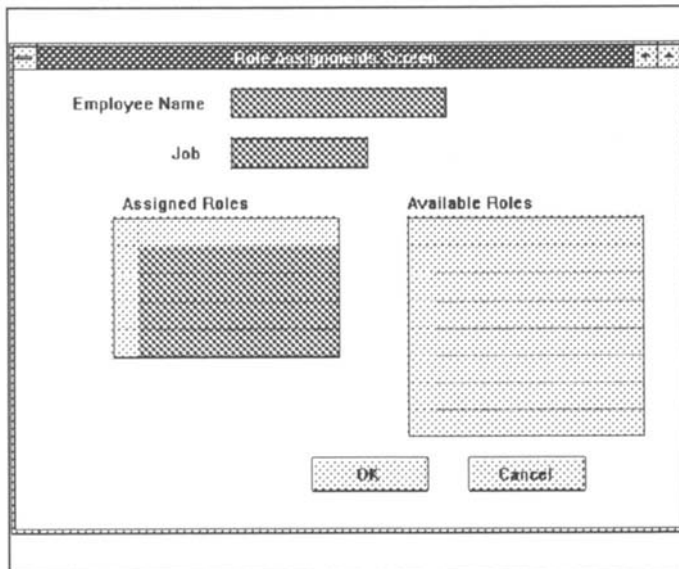
A complete demonstration will be given of a prototype of a system that uses role profiles to manage information security in a non-technical manner at remote sites, as described in this paper.

The demonstration will .....

- \* Show the following files, created for use by the ESM:
  - Transaction/Object/Role File
  - Role/Transaction File
  - Role/Employee File

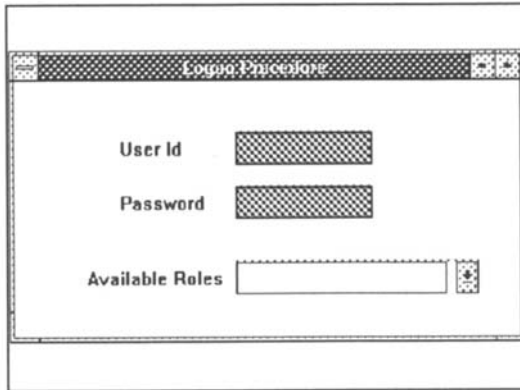
The "transaction/object/role file" and the "role/transaction file" would typically be created at the central site and distributed to the remote sites. The "role/employee file" is created and updated by the application that allows a manager to assign roles.

- \* Demonstrate how a manager can assign users to roles by means of a user-friendly, graphical interface, which hides the technical detail from that manager. See Fig 8.



**Figure 8:** Assignment of Roles to employee by manager

- \* Demonstrate that the "employee/role file" is updated by the assignment of an employee to a role or the revoking of a role from an employee.
- \* Demonstrate the login procedure of a user, showing how that user will be required to choose one of the roles assigned to him or her, for that session. See Fig 9.



The image shows a graphical user interface window titled "Login Procedure". The window has a standard title bar with minimize, maximize, and close buttons. Below the title bar, there are three input fields arranged vertically. The first field is labeled "User Id" and contains a shaded rectangular box. The second field is labeled "Password" and also contains a shaded rectangular box. The third field is labeled "Available Roles" and contains a list box with a scroll bar and a small icon on the right side, indicating a dropdown or list of options.

**Fig 9: Login Procedure**

- \* Simulate the invocation of the ESM to check that the role chosen by the employee is available to him or her at that time.
- \* Demonstrate the creation of the "active role file" which identifies the which role has been chosen by a particular user for that session. This file exists only for the duration of that session and is deleted when the user logs out.
- \* Simulate the invocation of the ESM, when an application program run by a particular user attempts to perform a transaction. The ESM will use the "active role file" to determine the currently active role. It will use the "role/transaction file" to check whether the transaction is permitted for that role and the "transaction/object file" to determine whether the transaction has the required rights to the object it is attempting to change. Users should not be allowed to perform transactions which are not permitted by the currently active role, even though another, non-active role may be permitted to perform that transaction.

## 8. SUMMARY

This paper addresses security management in a distributed open environment where processing and data have been decentralized. Access to certain objects like files and records is restricted to certain transaction, which in turn can only be performed by certain roles. Users are assigned one or more roles, but have to choose one role at login, which remains the active role for that session.

Access to objects is controlled at the remote site by a non-technical manager. This paper proposes a prototype which will provide a graphical user-interface to manage access to objects within a business environment, without requiring the user of that prototype to know technical details, or even the objects involved.

## 9. REFERENCES

- [1] Management of Information Technology, Caroll W. Frenzel, Boyd & Fraser Publishing Co, 1991, pp 475-476, pp 496-499
- [2] Design for dynamic user-role-based security, Imtiaz Mohammed & David M. Dilts, Computers & Security Volume 13 Number 8
- [3] The management of computer security profiles using a role-oriented approach, S.H. von Solms, Isak van der Merwe, Computers & Security Volume 13 Number 8
- [4] MOSS II A Model for Open System Security, PWJ van Zyl, Dr MS Olivier, Prof SH von Solms