Design of an Access Control Module for an Instrumentation Gateway

Paul G. Greeff, Gerhard P. Hancke, *Senior Member, IEEE*, and Derrick L. van der Merwe

***Abstract*—This paper describes some of the preliminary work performed on a resource access control module for an instrumentation gateway. The access control module employs a variation of the popular role-based access control (RBAC) scheme described by various authors. The gateway has very little security; hence, any user able to log onto the system is able to control any resource. This module aims to implement a pluggable module whereby users are granted access to various parts of the system depending on their user rights, while giving the administrator a very powerful method of restricting access without sacrificing ease of administration.**

***Index Terms*—Access control, field buses, measurement, networks, security.**

I. Introduction

THE COMPUTER Network and Security Group within the Department of Electrical, Electronic and Computer Engineering (EE&C), in collaboration with Institute of Computer Technology (ICT) Technical University of Vienna (TUV), is investigating the concept of a smart card automation network. The idea is to create a decentralized control system with networks of intelligent nodes (refer to Fig. 1). The idea forms the basis of the Internet gateway for unified automation network access (IGUANA URL: http://www.ict.tuwien.ac.at/iguana/) project. The gateway provides access to a field area network. Typically, a user would connect to the extended services daemon (ESD) through the Internet. The ESD software module queries the connected field area network daemon (FAND) to obtain available resources and/or services.

The problem addressed is the access control capabilities of the IGUANA gateway. Currently, the IGUANA gateway possesses no capability of allocating resources and/or services. Any client may have access to any other client's resources or services.

The access control aims to successfully integrate with surrounding IGUANA software modules while allocating the available resources and/or services of the ESD through the application of an access control scheme.

II. Background

Various access control schemes exist, but they vary in applicability depending on the application. Below follows a brief summary of some of the popular schemes.

Manuscript received January 24, 2005; revised October 18, 2005.

The authors are with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa (e-mail: paul.greeff@eng.up.ac.za).

Digital Object Identifier 10.1109/TIM.2005.862012

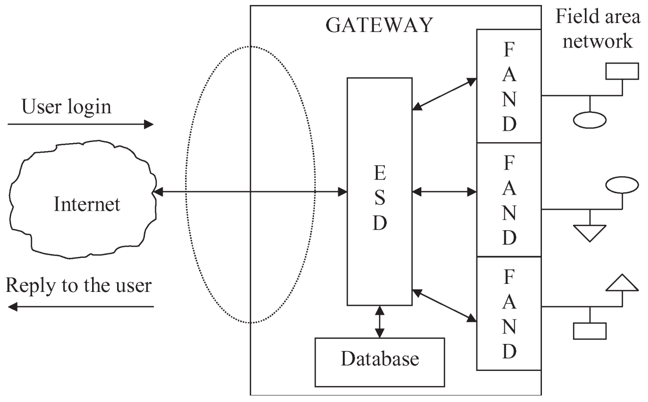

Fig. 1. Current structure of the IGUANA gateway. Note that the dashed circle indicates the placement of the proposed access control module.

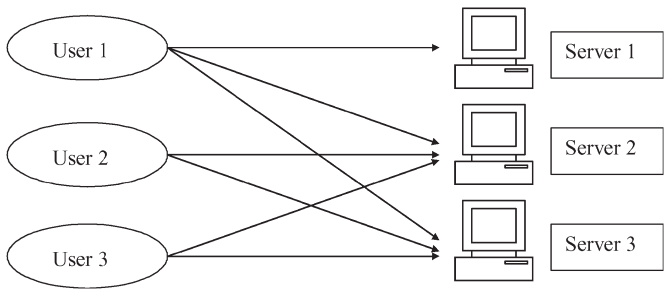

Fig. 2. Model of ACL [4]. User 1 has access to all three servers and the other users have access to server 2 and server 3.

A. Access Control List (ACL)

ACLs [5] are one of the most commonly used access control schemes. The basic principle behind ACLs is that every piece of data, application, or service has a list of users associated with it. The association specifies that a user may, or may not, have access to the specific object. The system will prevent anyone not on the ACL from accessing the record. Fig. 2 illustrates the associations between users and respective objects. Each server may function as a data storage facility, application server, or authentication server.

User 1 can be seen as a system administrator. The administrator needs to maintain the access rights of all the users. In this scheme, it is easy for an administrator to see which users have access to which data and/or applications. Each data and application server has its own ACL. The administrator simply needs to add or remove a user from the respective ACL.

The problem, however, multiplies whenever a person's role in the organization changes, and the amount of resources and services expands.

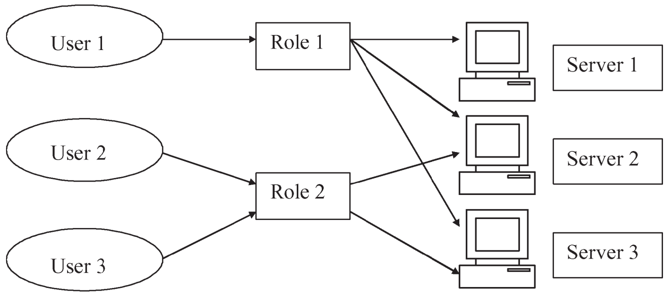

Fig. 3. Model of RBAC associations [4]. User 1 is indirectly associated with permissions regarding all three servers and the other users are indirectly associated with permissions regarding server 1 and server 2.

There exists a more efficient implementation, namely ACL groups [1]. ACL_G is an instance of ACLs where only groups are permitted as entries into an access list. Users are associated with a group, and the group may be associated with specific permissions. The administrator's job is thus simplified.

B. Role-Based Access Control (RBAC)

RBAC is an access control scheme that grants users access to information based on their responsibilities within an organization [6]. Access permissions are associated with roles, and the users are assigned to these roles, as shown in Fig. 3. This scheme creates an opportunity for the administrator to express the access control policy as a logical implementation of how the organization is viewed.

There does not seem to be a great difference between Figs. 2 and 3; however, imagine a network consisting of thousands of users and multiple servers. The task of managing an ACL scheme would be daunting.

The most important benefit of RBAC is that it simplifies the management of the access control policy. The administrator can control access to the system at a level of abstraction that is similar to the structure of the organization. This feature creates a visualization of where, how, and why certain complex access control policies are implemented. If the responsibilities of a user change, so do their roles in the organization. If user 3 were hired as an administrator, it would be a simple task of moving his account form role 2 to role 1. No metadata is kept for the decision-making process.

To provide further administrative efficiency, RBAC allows roles to inherit permissions from other roles. This inheritance creates a role hierarchy [2]. For example, the role "physician" is the hierarchical superior of the role "nurse." The physician contains all the permissions granted to the nurse, including any new permission granted to the role "physician."

RBAC provides the capability to implement any access control scheme through the use of heterogeneous applications on a variety of platforms.

As can be seen in Fig. 4, the core RBAC consists of the following model elements.

- User: a set of clients, both trusted and not trusted, who use the system.
- Roles: a set of named duties or job functions within an organization.

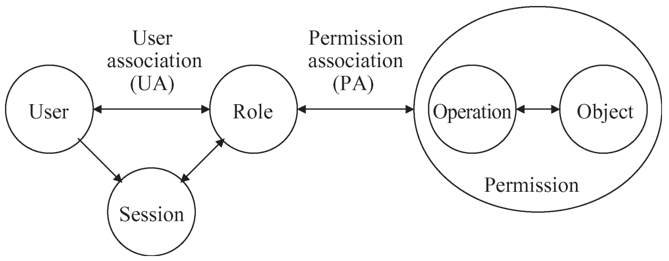

Fig. 4. Relationship between the core RBAC model elements [7]. The double-sided arrows illustrate a many-to-many relationship. The single-sided arrows illustrate a one-to-many relationship.

- Sessions: a mapping between a user and an activated subset of authorized roles.
- Operations: a set of access modes or actions permitted on an object.
- Objects: a passive entity within the system, which must be protected against an attacker.
- Permissions: a logical combination of operations and objects.
- User association: an association between an element in the user entity and the role entity.
- Permission association: an association between an element in the role entity and the permission entity.

A user may be assigned to many roles, but the session element ensures separation of duty. The session element specifies which of the authorized roles a user can occupy at once. The aim is to countermeasure fraud and other conflicts of interest.

Permissions consist of a set of operations that can be performed on a set of objects. Thus, a role must be authorized to read/write from an object, and the user must be assigned to that role. A role may be associated with multiple permissions.

The principle of least privilege also applies to RBAC. The principle states that the user only be given the necessary permissions to fulfill a specified duty; no additional permissions should be given.

The RBAC scheme has become the standard in access control schemes.

C. Other Access Control Schemes

Many other schemes exist; however, the focus of this document is on RBAC and ACLs.

III. Contribution

RBAC is an access control scheme with multiple advantages and very few limitations. RBAC does, however, possess two major weaknesses, namely in the fields of auditing capabilities and dynamic permission validation. Currently, as mentioned, the user is associated with many sessions, and a session may be associated with only one user. The session entity, however, does not log any role activations and operations completed. The session entity [7] implements only the following session functions.

- CreateSession: creates the session and provides the user with a default set of active roles.
- AddActiveRole: activates a role in the current session.

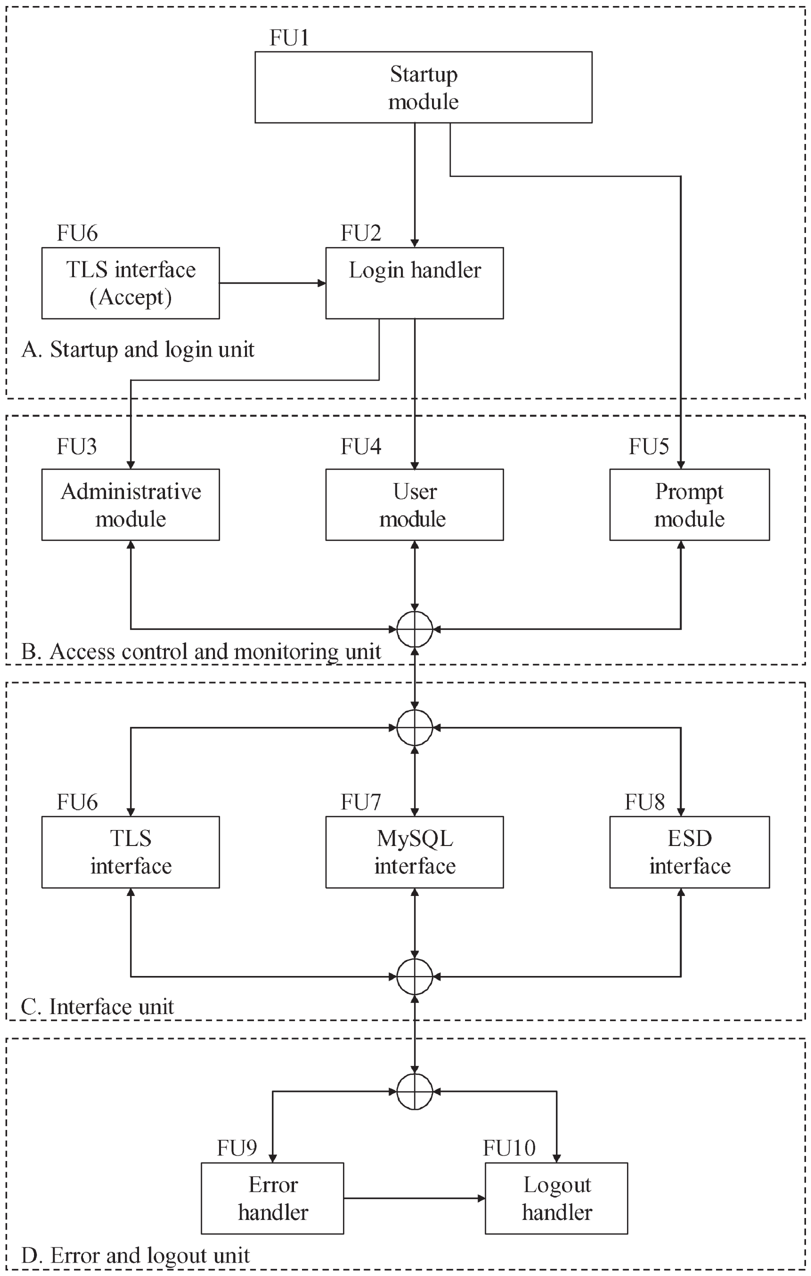

Fig. 5. Functional block diagram of the access control module.

- DropActiveRole: deletes a role from the current session.
- CheckAccess: checks whether the user possesses the requested permissions through role activation.

With the access control module, the session entity consists of subentities responsible for logging role activations and operations performed. A single query can conveniently retrieve information concerning any session associated with a user and/or role. Therefore, you can detect an illegal entity accessing secure context sensitive data.

The second important issue addressed by the access control module is that of dynamic permission validation. Ferraiolo *et al.* [7] do not define any function associated with operation and object creation. Similarly, permissions are only discussed in permission associations with roles.

The access control module implements a completely new way of validating permissions together with the standard create and delete permission scenarios. A query can be issued to the ESD that replies with a list of nodes and data points currently active on all the existing FAN daemons. The administrator may specify all the available operations offered by the ESD. After the operation and object entities have been updated, the validity of all permissions can be checked and activated or deactivated accordingly.

IV. Implementation

A short summary of all the functional units in Fig. 5 is given below.

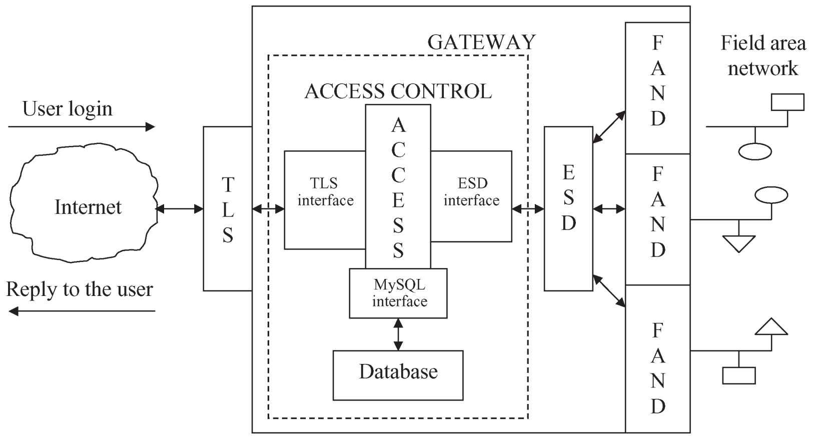

Fig. 6. Proposed structure of the IGUANA gateway. Note that the dashed rectangle indicates the placement of the access control module.

FU1 is a collection of software functions that initialize elements of the access control module. These elements include interface and thread variables. A collection of software functions in FU2 receives a user identification number from FU6 and creates an administrator or client thread. A software module FU3 is used by a remote gateway administrator to configure, maintain, and review the hierarchical RBAC policy. FU4 shows a software module used by a remote gateway user to allocate, on request, required resources. The gateway user chooses which roles to activate and operations to perform. FU5 is then used by the local gateway administrator to review system and MySQL database information. Data are then exchanged over a transmission control protocol/Internet protocol (TCP/IP) connection with the transport layer security (TLS) [8] module using FU6 as an interface. FU7 issues queries to an MySQL server, receives the result, and processes the result. FU8 exchanges data over a TCP/IP connection with the ESD module. FU9 inserts error information (interface, calling module, time, and user) into the MySQL database. The interface unit mostly generates the errors. If an error is fatal, FU9 calls FU10 to log out the user. FU10 is a collection of software functions that stop a specified user's thread.

The access control module software consists of a full core RBAC command set with role hierarchies and dynamic separation of duty (DSD), an ANSI-C administrative access control submodule to ease the tasks of the IGUANA gateway administrator in configuring the RBAC scheme, an ANSI-C user access control submodule to be used by the IGUANA gateway client in activating authorized roles and performing authorized operations, three ANSI-C interfaces to interact with the existing surrounding IGUANA software modules and the newly created TLS software module, and a test MySQL database (implementing RBAC) that can be used for simulation, training, and development purposes.

A. Concept Design

The concept design (Fig. 6) resulted in the implementation of the RBAC scheme through the use of MySQL as a design tool. RBAC does require more processing than ACL, but through the intelligent use of MySQL table referencing and optimization techniques, this problem could easily be overcome. The simple database management structure of MySQL also enhanced the logical organizational representation offered by RBAC.

MySQL makes it easy to manipulate different types of data. The security offered by MySQL is also a huge advantage, easing the task of protecting the database.

The advantages that RBAC possesses over ACL are that an organization's hierarchical structure can be easily represented; it places high emphasis on the principle of least privilege and of separation of duty; and the permissions are associated with the roles and users associated with these roles. Thus, if a user's role in the organization changes, only the user's association needs to be changed.

B. Operation

The administrator/user logs in through the TLS interface; if the administrator/user is authenticated, the user identification number is forwarded to the access control module through TCP/IP connection. The access control scheme (RBAC) is then applied to the user identification number, and the necessary resources and/or services are allocated to the user. The MySQL database maintains all the entities, fields, and elements of the RBAC scheme. An administrator only performs operations on the RBAC scheme (updating entities and reviewing activity and error logs). A user interacts with ESD according the user associations in the RBAC scheme. Because the permission associations are performed on data point (data points are input and output variables in the network) level, it is presumed that the user does have access to the entire data point.

C. Structure

The goal of the access control module is to provide added functionality to the IGUANA gateway through the successful application of RBAC on the resources and/or services offered by the ESD protocol. The access control module is designed to interface with the existing ESD module, a TLS module, and a MySQL server placed on the IGUANA gateway. The access control module is situated on the gateway between the ESD module and the TLS module, as shown in the figure below.

The MySQL server maintains the relational database and tables needed by the access control module. The MySQL server receives, parses, optimizes, and executes the queries issued by the access control module.

The ESD module is responsible for gathering information from any connected FANDs. The access control module has the responsibility of allocating system resources and services offered by the ESD module.

A TLS module was designed in conjunction with the access control module. The TLS [8] module is responsible for authenticating the user (through the use of smart card technology) and implements the secure socket layer (SSL) protocol to encrypt and decrypt data transferred over the Internet. The TLS is beyond the scope of this document.

D. Elements of the Access Control Module

This section contains a brief description of the submodules, interfaces, and relational database that form the access control module.

1) Administrative Submodule: The administrative submodule's function is to provide the administrator of the access control module with the ability to create, update, maintain, and review the entities of the RBAC model.

The RBAC entities are written in uppercase because they represent the names of the MySQL tables that store the elements of the specific RBAC entity.

The access control module implements the following RBAC model entities (Fig. 7).

- Users (USERS): the set of IGUANA gateway users, authenticated by the TLS module, who use the access control module.
- Roles (ROLES): the set of named duties or job functions within the IGUANA project.
- Sessions (SESSION_OPS and SESSION_ROLES): a mapping between an IGUANA gateway user and an activated subset of authorized roles.
- Operations (OPS): the ESD command set, including the event commands regarding the creation, maintenance, and deletion of events processed by the ESD event handler.
- Objects (OBS): the available nodes, data points, and events stored on the ESD module.
- Permissions (PERMS): a logical association of operations and objects.
- User association (UA): an association between an IGUANA gateway user and a role defined by the IGUANA gateway administrator.
- Permission association (PA): an association between a role defined by the IGUANA gateway administrator and an active permission.
- Role hierarchies (ROLESTREE): the roles are arranged in the form of a hierarchical structure, thus depicting a role's position within the IGUANA project (Fig. 8).

The software flow diagram (Fig. 9) shows the flow of data during use of the administrative submodule.

2) User Submodule: The user submodule's function is to provide the user of the access control module the ability to activate or deactivate authorized roles, to perform requested operations, and to review authorized permissions (refer to Fig. 10). The operations performed are commands issued to the ESD module, which in turn replies with an error, basic reply, or an extended response.

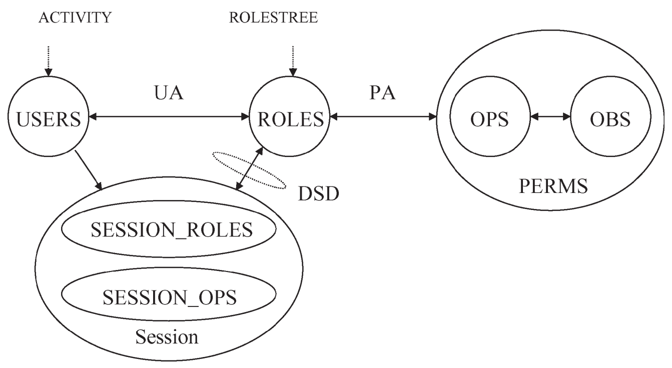

Fig. 7. Relationship between the implemented RBAC model elements of the access control module. The double-sided arrows illustrate a many-to-many relationship. The single-sided arrows illustrate a one-to-many relationship.

E. Interfaces

1) TLS Interface: The TLS interface is implemented as a TCP/IP server. The TLS module authenticates each user that logs onto the IGUANA gateway.

2) ESD Interface: The ESD interface is implemented as a TCP/IP client. The ESD interface connects to the ESD module. The ESD module receives commands from the access control module, performs the operation, and responds with the data requested or an acknowledgement.

3) MySQL Interface: The MySQL interface is implemented as a TCP/IP client. The MySQL interface connects to the MySQL server. All the submodules use the MySQL interface to issue queries to the MySQL server and to return or print the result.

V. Results

The aim of all the experiments was to verify the functionality of all the RBAC sets and relations and not necessarily the fault tolerance of the system. A test database was used and developed for testing purposes. It is worth noting that the construction of the access control module does not allow access to any part of the system unless it is through the user or administrative interfaces. Hence, a failure of, for example, the administrative module will mean that no access is given to any administrative function.

The most important results were obtained by performing the following three experiments.

As explained earlier, one of the important features of the RBAC scheme implemented here is the DSD. The following experiment explored the DSD aspect of the system.

A. Experiment 1: Testing Sets and Relations of the DSD

The experiment was designed to prove that the DSD functions of both the administrative and user submodule work. The administrative submodule was used in the setup of the DSD

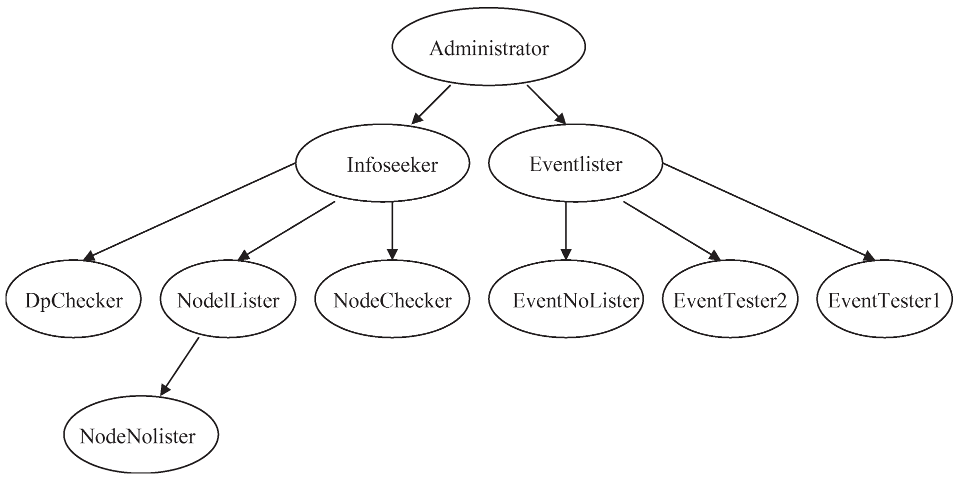

Fig. 8. Example of a role hierarchical structure. Note that this hierarchical structure is used to test the hierarchical RBAC functions of the access control module.

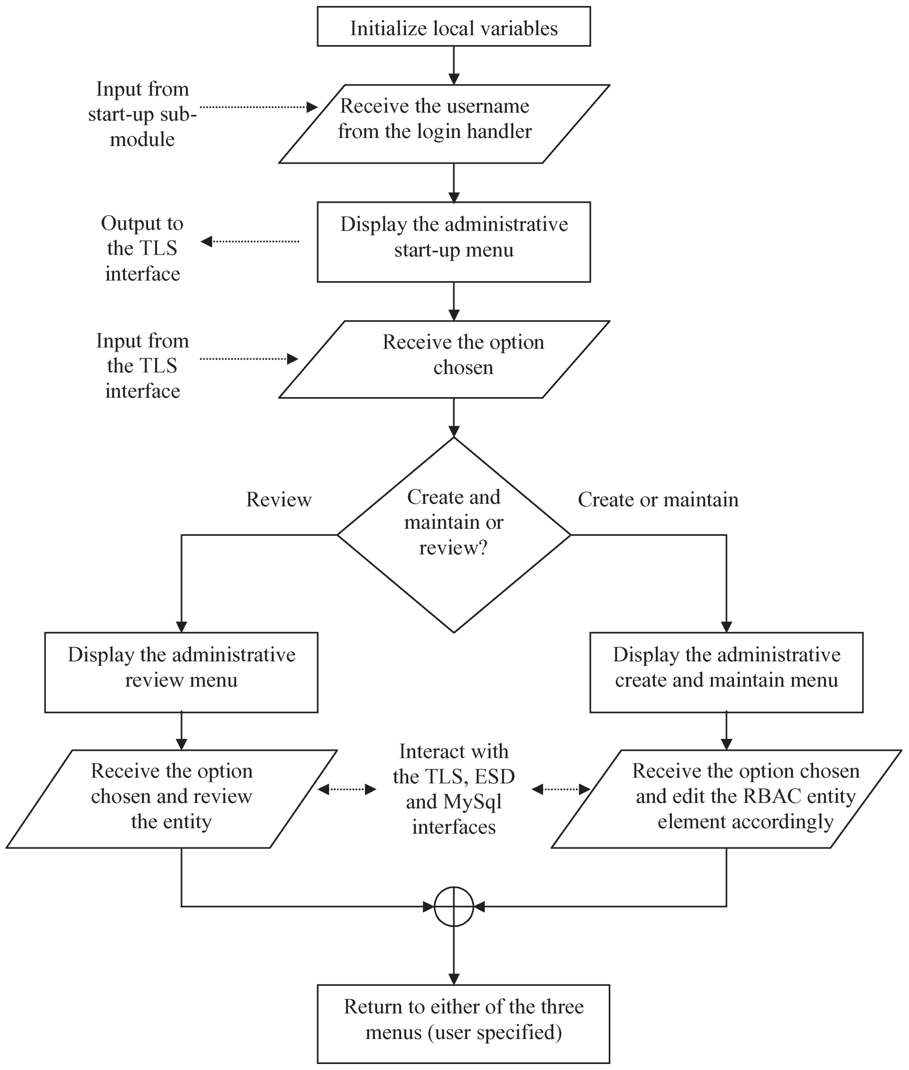
Initialize local variables
Input from start-up sub-module
Receive the username from the login handler
Output to the TLS interface
Display the administrative start-up menu
Input from the TLS interface
Receive the option chosen
Review
Create and maintain or review?
Create or maintain
Display the administrative review menu
Display the administrative create and maintain menu
Receive the option chosen and review the entity
Interact with the TLS, ESD and MySql interfaces
Receive the option chosen and edit the RBAC entity element accordingly
Return to either of the three menus (user specified)

Fig. 9. Software flow diagram showing the primary administrative processes involved.

sets and relations, and the user submodule was used to test the application of the DSD functions and rules upon role activation.

The table in Fig. 11 shows a number of rows generated by the system, indicating two users, derrickvdm and chrisf, performing role activation. Activating ascending roles would be permitted since the role tree incorporates the functions of descending roles. However, conflicting roles are not allowed to be activated simultaneously. This prevents, for example, the administrator from wreaking havoc on user functions. However, DSD does allow the role of the administrator, and any user

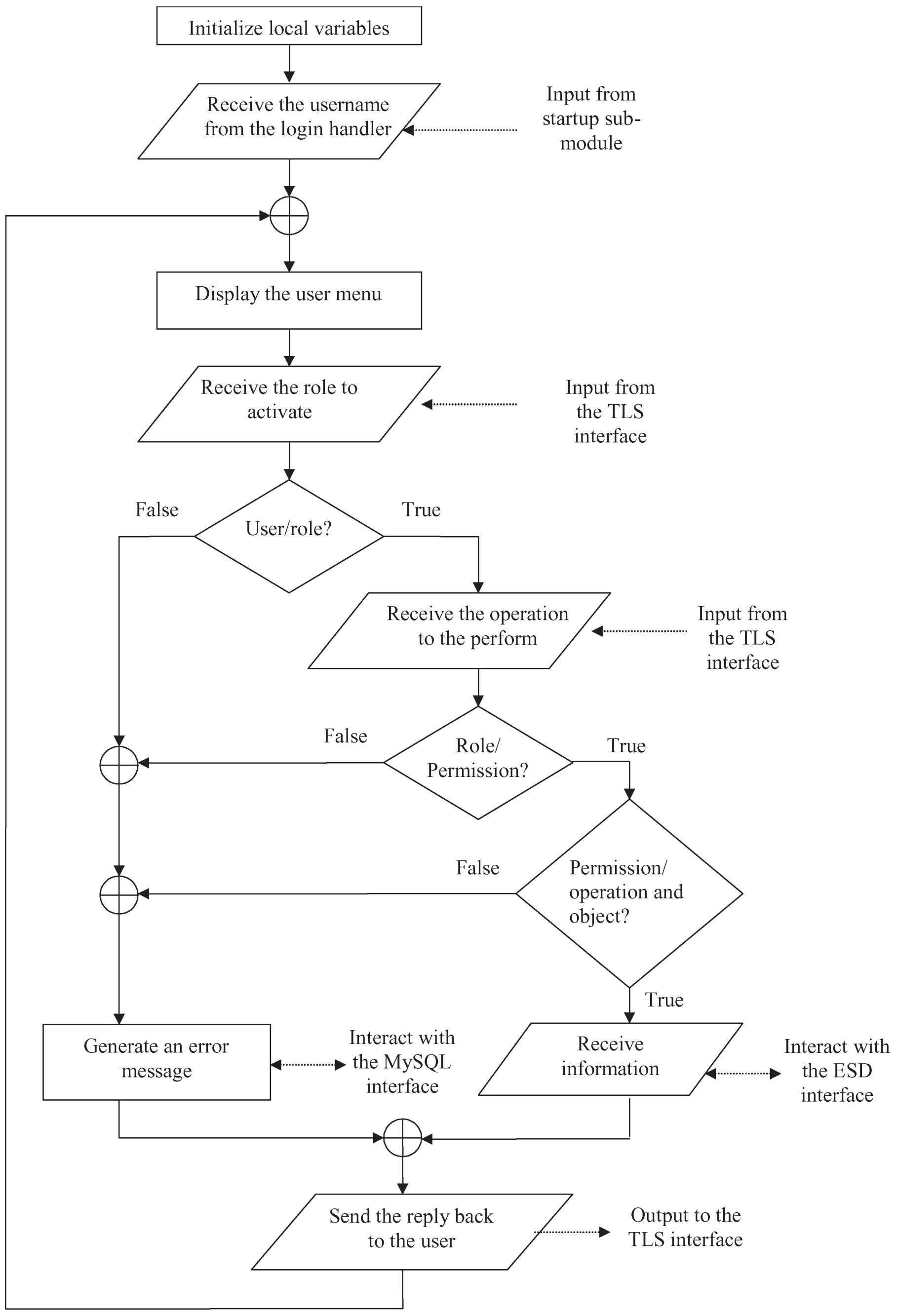
Initialize local variables
Receive the username from the login handler
Input from startup sub-module
Display the user menu
Receive the role to activate
Input from the TLS interface
User/role?
False
True
Receive the operation to the perform
Input from the TLS interface
Role/ Permission?
False
True
Permission/ operation and object?
False
True
Generate an error message
Interact with the MySQL interface
Receive information
Interact with the ESD interface
Send the reply back to the user
Output to the TLS interface

Fig. 10. Software flow diagram showing the primary user processes involved. Note that all decisions are made based on the information contained by the MySQL relational database.

for that matter, to be changed according to requirements by activating or deactivating roles. Fig. 11 shows how roles are granted or denied depending on which roles are activated. Pay special attention to session numbers 4 and 5: eventlister. At first denied, once a conflicting role is deactivated, the eventlister role is activated.

The user submodule successfully enforced the DSD sets.

B. Experiment 2: Testing Sets and Relations of the Role Hierarchy

The experiment was designed to prove that the hierarchical structure (shown in Fig. 8) can be successfully configured or reviewed by the administrative submodule and can be successfully applied by the user submodule. The administrative module was successfully used to configure the role hierarchical structure, and the user was, through role activations, able to receive permissions to ESD nodes and data points.

Fig. 12 shows an extract of the auditing tables. These tables indicate the roles and associated permissions activated through the role hierarchy. Once again note that certain actions are not allowed due to conflicts. Once a conflicting role and associated permissions are deactivated, a new role may be activated. As an example, note session numbers 7 and 8. The SESSION_ROLES table indicates the activated role and the failed attempts. The SESSION_OPS table indicates the operations performed in the active role.

```
mysql> select * from ACTIVITY;
| LogNum | UserId     | LogInTime           | LogOutTime          | Socket | ClientNum |
|   1    | Derrickvdm | 2002-10-30 19:24:11 | 2002-10-30 19:27:10 |   7    |     0     |
|   2    | Chrisf     | 2002-10-30 19:27:24 | 2002-10-30 19:28:30 |   7    |     0     |
2 rows in set (0.00 sec)

mysql> select * from SESSION_ROLES;
| SessionNum | UserID      | RoleName       | RequestTime         | Granted | Status   |
| .          | .           | .              | .                   | .       | .        |
| 4          | derrickmvdm | eventlister    | 2002-10-30 19:24:52 | no      | inactive |
| 5          | derrickmvdm | eventlister    | 2002-10-30 19:25:09 | yes     | inactive |
| .          | .           | .              | .                   | .       | .        |
| 12         | chrisf      | hostnamewriter | 2002-10-30 19:27:40 | no      | inactive |
| 13         | chrisf      | hostnamewriter | 2002-10-30 19:18:02 | yes     | inactive |
13 rows in set (0.00 sec)
```

Fig. 11. Auditing tables of the access control module after the DSD experiment. Note that the status of all the table entries is inactive because both users had already logged out of the access control module, according to the ACTIVITY table.

```
mysql> select * from ACTIVITY;
| LogNum | UserId     | LogInTime           | LogOutTime          | Socket | ClientNum |
|   1    | Derrickvdm | 2002-10-30 18:44:17 | 0000-00-00 00:00:00 |   7    |     0     |
1 rows in set (0.00 sec)

mysql> select * from SESSION_ROLES;
| SessionNum | UserID     | RoleName     | RequestTime         | Granted | Status   |
| .          | .          | .            | .                   | .       | .        |
| 7          | derrickvdm | nodechecker  | 2002-10-30 18:55:57 | no      | inactive |
| 8          | derrickvdm | nodechecker  | 2002-10-30 18:56:32 | yes     | active   |
| .          | .          | .            | .                   | .       | .        |
11 rows in set (0.00 sec)

mysql> select * from SESSION_OPS;
| UserId     | SessionNum | OpName   | FanType | FanId   | FanNodeAddress | DpId |
| .          | .          | .        | .       | .       | .              | .    |
| derrickvdm | 8          | nodeinfo | sys     | gateway | gateway        | none |
| .          | .          | .        | .       | .       | .              | .    |
15 rows in set (0.00 sec)
```

Fig. 12. Auditing tables of the access control module after the role hierarchy experiment.

C. Experiment 3: Testing the Application of RBAC on Generated Events

The experiment was designed to prove that the RBAC control policy can successfully be configured by the administrative submodule and be correctly applied by the user submodule.

The event command set is the largest and most complex command set in the ESD command sets.

The events are broken down into four segments:

1) event criteria: defining a criterion under which an event is triggered;
2) event action: defines the operations to perform when the event is triggered;
3) event description: human readable description of the event;
4) event ICC parameters: unknown event relevance.

The access control module further segments the four segments into the following:

1) operations identified within each of the ESD segments;
2) objects identified within each segment;
3) user variables (such as the maximum number of log entries within the ESD);
4) user textual messages.

The creation and deletion of an event cause the access control module to update the role's permission associations.

The user must be able to maintain and review the newly created event. Note that permission regarding the event is declared inactive if the event is deleted by an administrator directly connected to the ESD. The administrative submodule successfully configured roles, with permissions, to create, delete, and maintain events. The user submodule was successfully used to create and delete events.

The experiments showed (including Figs. 13 and 14) that network latency and the encryption done by the TLS before the information is received by the user play a huge role in packet transfer. Nearly 60% of the response time can be attributed to those two factors. A detailed statistical analysis was not performed. Figs. 13 and 14 are very rough representations of the traffic at the TLS and the MySQL database. They represent the traffic at the C interfaces of the relevant functions.

VI. Conclusion

The goal was to determine whether the access control module could be successfully implemented and used on the IGUANA gateway. The access control module successfully applies the RBAC policy to all the operations offered by the ESD.

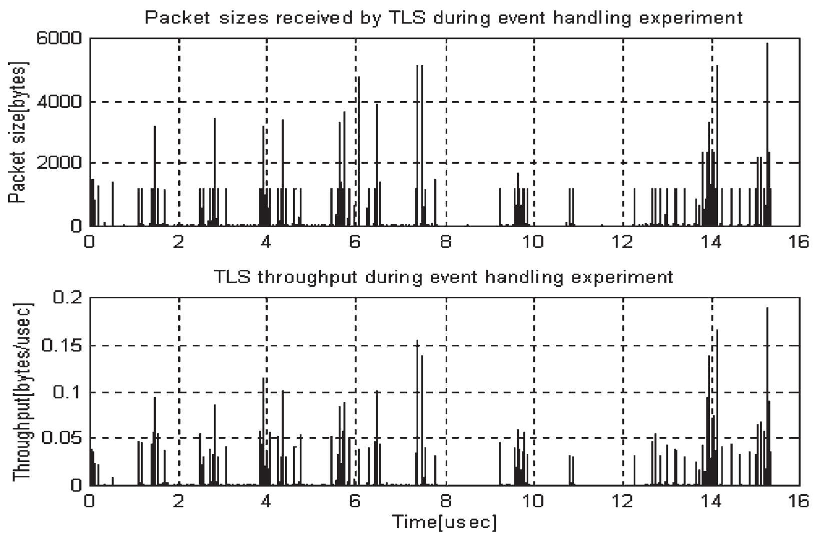

Fig. 13. Throughput achieved to, and packet sizes received at, the TLS module during the event handling experiment.

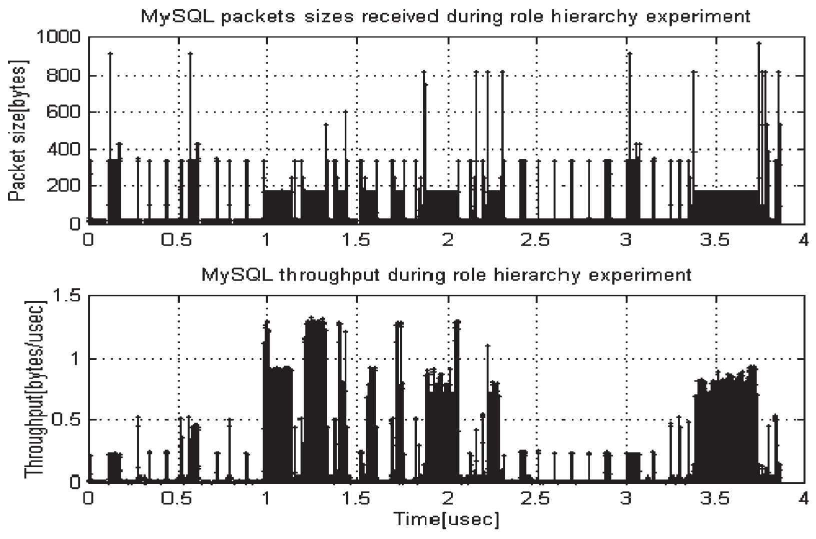

Fig. 14. Throughput achieved to, and packet sizes received at, the MySQL interface during the event handling experiment.

Through these experiments (indicated in Figs. 11 and 12), it is shown that with this type of application of the RBAC scheme with DSD it is possible to effect a system capable of allocating system resources to a number of users, even an administrator, in such a way as to limit the possibility of conflicting roles in an organization to overlap. The roles defined in the RBAC scheme reflect the roles of the users in an organization; hence, user and system maintenance is minimized.

A further advantage of this implementation suggested by looking at Figs. 13 and 14 is that the system overhead is minimal and that resources associated with encryption and possibly authentication (encryption and authentication are part of a project by Shulze [8] and Burger [3]) are the main causes of delay in the system. The simple nature of the RBAC suggests that even with large organizations employing many roles acting on a large number of end points, a very efficient system should be possible. This should be true as long as scalability can be maintained.

The access control module is the beginning of a long line of experimental applications of different access control policies. Further work includes converting the above implementation into a machine-automated system.

References

[1] J. F. Barkley, "Comparing simple role based access control models and access control lists," Nat. Inst. Stand. Technol., Gaithersburg, MD, Rep. No. 973-3346, 1997.

[2] J. F. Barkley, B. Beznosov, and J. Uppal, *Supporting Relationships in Access Control Using Role Based Access Control*. Los Alamitos, CA: IEEE Computer Society Press, 1999.

[3] C. R. Burger, "A security policy for a distributed utility metering system," M.S. dissertation, Faculty EBIT, Univ. Pretoria, Pretoria, South Africa, 2003.

[4] M. P. Gallaher, A. C. O'Conner, and B. Kropp, "The economic impact of role based access control," Nat. Inst. Stand. Technol., Gaithersburg, MD, Rep. No. 07 007 012, 2002.

[5] D. F. Ferraiolo, J. F. Barkley, and D. R. Khun, "A role based access control model and reference implementation within a corporate intranet," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 34–64, Feb. 1999.

[6] D. F. Ferraiolo, J. A. Cugini, and D. R. Khun, "Role based access control (RBAC): Feature and motivations," in *Proc. Annu. Computer Society Application Conf.*, New Orleans, CA, 1995, pp. 241–248.

[7] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001.

[8] S. Shulze, "Secure Internet communication for the IGUANA gateway," Dept. Elect., Electron. Comput. Eng., Univ. Pretoria, Pretoria, South Africa, Project EPR400, 2002. Final year project report.

[9] D. L. Van der Merwe, "Design of an access control module," Dept. Elect., Electron. Comput. Eng., Univ. Pretoria, Pretoria, South Africa, Project EPR400, 2002. Final year project report.

Paul G. Greeff received the B.Eng. and the B.Eng. (honors) degree in electronic engineering from the University of Pretoria, Pretoria, South Africa, in 2000 and 2001, respectively, and is currently working towards the M.Eng. degree in electronic engineering focusing on instrumentation systems and computer networks.

He is currently a Lecturer at the Department of Electrical, Electronic, and Computer Engineering, the Computers Group, University of Pretoria.

Gerhard P. Hancke (M'88–SM'00) received the B.Sc., B.Eng., and M.Eng. degrees in electrical engineering from the University of Stellenbosch, Stellenbosch, South Africa, and the D.Eng. degree in electrical engineering from the University of Pretoria, Pretoria, South Africa, in 1983.

He is currently the Chair of the Computer Engineering Program in the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria. He is responsible for curriculum development and research activities within this program. He partakes extensively in collaborative research programs with research institutions internationally. He is a Professional Engineer and held offices in various national and international scientific and professional bodies.

Derrick L. van der Merwe received the B.Eng. degree in electronic engineering from the University of Pretoria, Pretoria, South Africa, in 2002.