Access Control Lab

Authors

Joao Mena - s223186
Tomas Estacio - s223187
Aidana Nursultanova - s212994
Renjue Sun - s181294

November 25, 2022

Contents

1 Introduction **1**

2 Access Control Lists **2**

3 Role Based Access Control **4**

4 Evaluation **7**

5 Discussion **11**

6 Conclusion **12**

Nomenclature **13**

1 Introduction

This report describes the developments, results and conclusions of our work regarding the Access Control laboratory exercise. This exercise is expected to provide the students with practical experience in the specification and implementation of software that enforces an access control policy to a system, as well as comparing and discussing different access control policies.

On the previous laboratory exercise, we developed a printer server application equipped with password-based authentication, where a user would need to be authenticated by the server to use it's services. This is a good start in terms of security, but it's often the case where not every user needs access to the same resources. In order to implement an efficient solution which grants the right permissions to each user, we need a good access control policy which should check every access, enforce least privilege and verify acceptable usage.[3]

With this purpose in mind, we implemented two independent systems with the same core (the software developed for the previous exercise), where we enforce two different policies, one based on an access control list, and another one based on a role-based access control. We also experiment with the flexibility of each policy regarding organisational changes and compared each of their strengths and weaknesses.

At the end of this exercise we have a fully functional software that meets all the proposed objectives, which will be thoroughly explained throughout this report.

2 Access Control Lists

An access control list is a list to control which users are granted or denied access to a particular system resource. Normally, they are used for controlling permissions to a computer system, in our case, to a printer server, enforcing authorization policies in the users in our client/server application. [2]

After analysing the access control scenario proposed on the project's description, we agreed that the best way to implement a list of the usernames that have permissions to do certain functions inside the print server was to use the same database created for the previous assignment, creating two new tables to associate the usernames with the operations that they can perform. The first new table is called *operations_table* and has two columns, one for the operations ID and other for the operation name, while the other new table is called *user_operation_mapping*, used to associate the table with the usernames with the table with the operations ID, having two columns with those values.

The first task required us to implement the necessary code to enforce the access control policy scenario described in the project's description, meaning that all users and the access control list for the printer server must be included in the database, in the already described tables. We started by creating a function that, after the user has authenticated him/herself in the printer server, checks if that user has permission to perform a specific operation in the printer server, controlling the access based on the username and the operation ID in the table *user_operation_mapping*. In order to have a specific order that is defined for every function we create related to the access control list, we designed a "Enum" class with the following order for the operations:

1. print;
2. queue;
3. topQueue;
4. start;
5. stop;
6. restart;
7. status;
8. readConfig;
9. setConfig.

We chose this design for our implementation because, since we already operated with the database for saving the usernames, hashed output and salt, using them to authenticate users to use the print server, and we needed to use shared arguments, like the username, the best solution was to create another table on the database and associate them. This way, the access control policy is not "hardcoded" into the program, because the verification of

the permissions of each user inside the printer server is done by verifying the table in the database which contains the information required. [1]

In order to test the implementation, we created the usernames Alice, Bob, Cecilia, David, Erica, Fred and George with passwords for each of them and insert them in the database, using functions created in the program, so that they are able to authenticate themselves to use the printer server. After that, we inserted in the *user_operations_mapping* table each of operation IDs that the users are able to access:

- Alice is able to access all the operations of the printer server;
- Bob can use the start, stop, restart, status, readConfig and setConfig operations;
- Cecilia can use the print, queue, topQueue and restart operations;
- David, Erica, Fred and George can use the print and queue operations.

Then, in the client side, we authenticate each user and, after entering the printer server, we check the services accessible for each user, to verify that the system works as required, meaning that each user can only perform certain operations inside the printer server.

Finally, to have a better understanding of the implementations made in the program, the following diagram describes the design of our system with the access control list implementation:

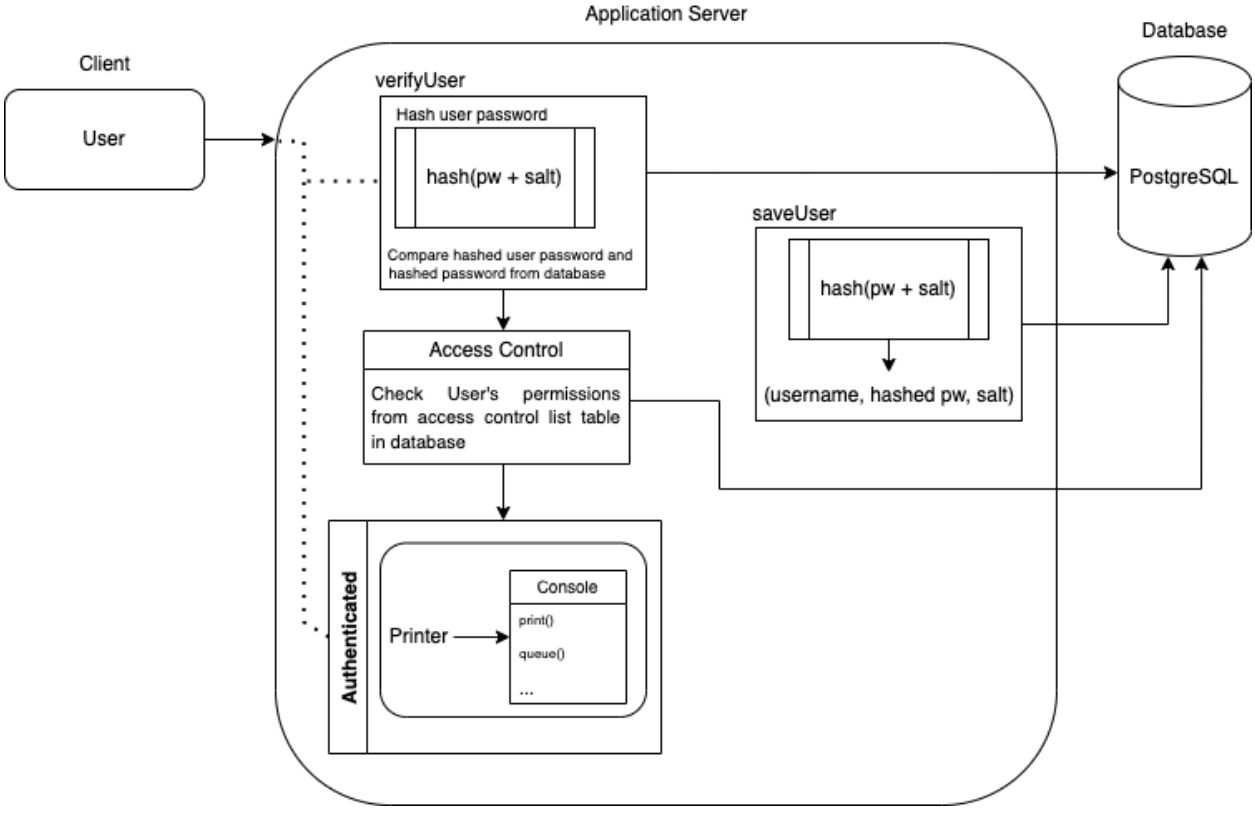

Figure 1: System using Access Control

3 Role Based Access Control

A Role Based Access Control system is a system where the security is managed at a level that corresponds to its hierarchy, meaning that each user is assigned one or more roles (in this case we just assign one role to each user), and each role is assigned one or more permitted to the role in question. These are usually used by companies to implement administrative security that determines the operations that can be executed by people in specific jobs, and assigning them to proper roles. [4] The use of a role based access control system offers two main goals in any security based system: integrity and confidentiality, in a way that it prevents unauthorized people to access specific parts of a system to tamper or destroy data, preserving it to authorized people.

In the scenario that was presented in the project's guideline, we are required to identify roles and define a role hierarchy and permissions for each role, and implement those changes to the code in which we are controlling the access of the system using only an access control list. For the second task, we defined the role hierarchy and access control policy for the company according to the scenario described.

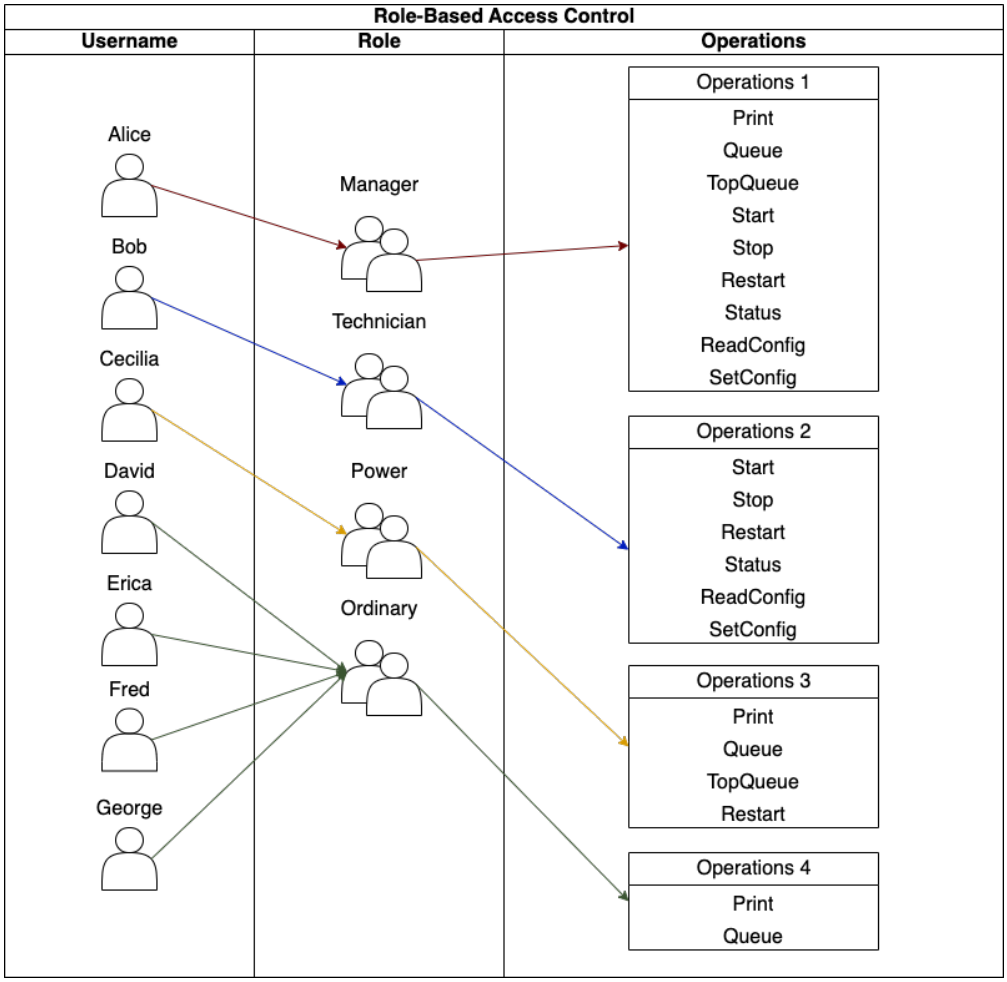

Figure 2: Role Based Acess Control Structure

For the third task, we had to implement the access control policy, previously outlined, in the program, starting with the creation of two new tables in the postgresql's database, named *roles* and *users_roles_mapping*. The first one has one column of the existing roles in our system and one column of the role's IDs, and the second table is the association of the *roles* and the *users* tables, having one column with the usernames and other with the role's IDs. These tables are used for verifying the roles of each user, so that when they intend to access a specific operation inside the printer server, we can verify if the role of the user is one with permissions to perform that operation, by using the created class *RoleAccessChecker*() in the program, which verifies the tables in the database to check which role the user has. After checking the role of the specific user, we get the knowledge of the operations that the user can perform and we are able to deny or grant access to those operations, after the user authenticates to the printer server.

In order to have a specific order that is defined for every function we create related to the role based access control, we designed a "Enum" class with the following order for the operations:

1. Manager;
2. Technician;
3. Power;
4. Ordinary.

The idea behind the design for our implementation was again using the database already described, and it was chosen for the same reasons previously stated, as it is easier to have the information in the same place to associate information about the users, making the system faster and more user-friendly, without compromising the security of the system, because we still protect the database with username and password and authenticate each user that tries to access the printer server. [1]

In order to test the implementation, in the client's side, we load the users, the roles and the operations to the tables in the database so that we can work with that data. After loading the data, we authenticate each user with a password, saving that in the database as described in the previous report, so that they can access the printer server and associate them with the role they have in the company. After that, we try to access different operations with different usernames, so that we can test if the system is working according to the role based access control developed.

Finally, to have a better understanding of the system we have created, we developed a diagram of our system, in which we include the role based access control policy used:

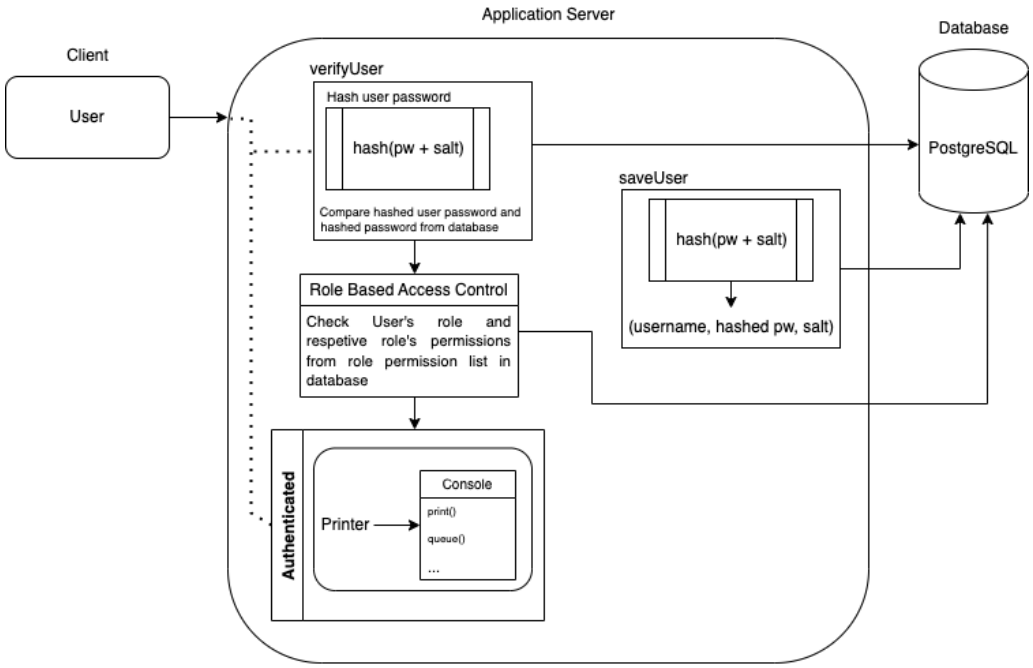

Figure 3: System using Role Based Access Control

For the final task, we were asked to implement some changes in the access control policy scenario that was first presented.

Firstly, we consider that the user Bob leaves the company and George takes over as a technician, so we deleted the username Bob in the table of users, we replaced Bob by George in the table where the user is associated with the roles and deleted the previous George entry in that table. This way, we do not need to assign again all the new operations that George is now able to perform, because that information is already assigned to each role in the database, meaning that having roles assigned in the database facilitates the process of replacing users in the role-based access control system. However, for the system with the access control list, this change is more complex, as we are required to delete the user Bob from the users table and replace every entry that Bob has in the table that associates the username with each operation that the user can perform with the user George, which takes more time and is more complex to manage, comparing to the role based access control system, since the role assignment simplifies the number of changes required to perform.

Secondly, two new users were added to our system, Henry, who is granted the ordinary role and Ida, who is granted the power role. In the role based access control system, the addition of new users is easier than in the access control list system due to the same reasons as before, the number of changes needed in the tables of the database is smaller, since we need to create the user and assign him to just one role in one case, and, in the other case, we are required to create the user and associate him/her to all the operations that he/she is allowed to perform inside the printer server.

Finally, we tested the new changes in both our systems and verified that the new users have access to the correct functions and the user that left the company is not registered in our database.

4 Evaluation

For the evaluation section, we designed a system with a access control list to determine which users are able to perform certain operations inside the printer server, after being authenticated. As described previously, the system developed guarantees that whenever a user tries to use the printer server's operations, we first verify if that specific username has the permission to perform the operation, by accessing the table inside the database that associates each user with the operations that he/she can execute. The following diagram describes the structure of our database tables for the access control list implementation:

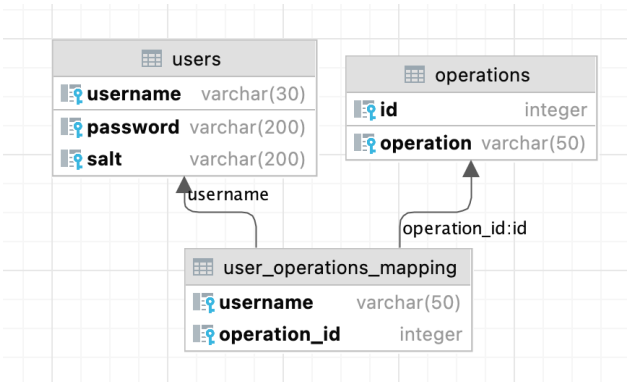

Figure 4: Database tables diagram for Access Control List

After loading all the users and their permissions inside the printer server into our program, we made each user try to access each of the operations available in the printer server, after being authenticated inside the server, and this was the output obtained:

```
Checking access for ALICE:

Start:
Alice successfully started the printer

Status:
Alice reads the status of printer: Printer server is started and can be used

SetConfig:
Alice sets the printer config for parameter 'param_-1598858963'

ReadConfig:
Alice reads the printer config. The value of the parameter 'param_-1598858963' : value1

Print:
File 'file_-1614762653' successfully added to the print queue with job number 1
File 'file_-1265019848' successfully added to the print queue with job number 2

Queue:
job number: 1, file name: file_-1614762653
job number: 2, file name: file_-1265019848

TopQueue:
Alice moves job: 2 to the top of the queue

Stop:
Alice stopped the server

Restart:
Alice restarted the server
```

Figure 5: Alice's access attempts

```
Checking access for ERICA:

Start:
User with name 'Erica' not permitted to invoke operation 'start'

Status:
User with name 'Erica' not permitted to invoke operation 'status'

SetConfig:
User with name 'Erica' not permitted to invoke operation 'setConfig'

ReadConfig:
User with name 'Erica' not permitted to invoke operation 'readConfig'

Print:
File 'file_1553275722' successfully added to the print queue with job number 3
File 'file_1092451247' successfully added to the print queue with job number 4

Queue:
job number: 1, file name: file_1850300301
job number: 2, file name: file_-1277772545
job number: 3, file name: file_1553275722
job number: 4, file name: file_1092451247

TopQueue:
User with name 'Erica' not permitted to invoke operation 'topQueue'

Stop:
User with name 'Erica' not permitted to invoke operation 'stop'

Restart:
User with name 'Erica' not permitted to invoke operation 'restart'
```

Figure 6: Ericas' access attempts

The program ran as expected, meaning the requirements presented for Task 1 were accomplished, as the access control list system developed enforces the policy explained in Section 2.

Next up, we designed a system with a role-based access control, meaning that we added different roles to the users already stored in the database, regarding the operations that each user is allowed to perform. In our system, we defined 4 different types of roles: manager, technician, power and ordinary, that have different sets of available operations associated to them and are stored in the tables in the database used. For that reason, to check if the users are allowed to use a certain operation inside the printer server, after being authenticated, we have to access the table that associates the user with the role, and verify which permissions that role has. The following diagram illustrates the structure of our database tables for the Access Control List implementation:

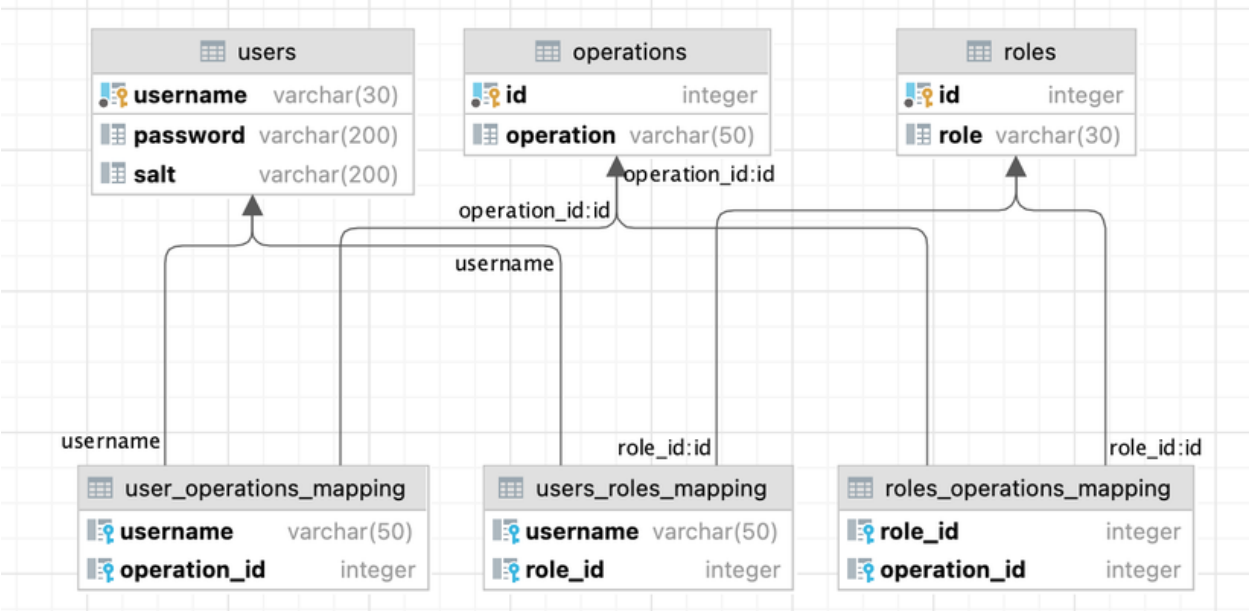

Figure 7: Database tables diagram for Role Base Access Control

After loading all the users, the roles and the operations available for each role, we made each user try to access each operation, after being authenticated in the printer server, and this was the output obtained:

```
Checking access for CECILIA:

Start:
User with name 'Cecilia' and roles [power] not permitted to invoke operation 'start'

Status:
User with name 'Cecilia' and roles [power] not permitted to invoke operation 'status'

SetConfig:
User with name 'Cecilia' and roles [power] not permitted to invoke operation 'setConfig'

ReadConfig:
User with name 'Cecilia' and roles [power] not permitted to invoke operation 'readConfig'

Print:
File 'file_-327341915' successfully added to the print queue with job number 3
File 'file_-1197315530' successfully added to the print queue with job number 4

Queue:
job number: 1, file name: file_-377776722
job number: 2, file name: file_-522496975
job number: 3, file name: file_-327341915
job number: 4, file name: file_-1197315530

TopQueue:
Cecilia moves job: 2 to the top of the queue

Stop:
User with name 'Cecilia' and roles [power] not permitted to invoke operation 'stop'

Restart:
Cecilia restarted the server
```

Figure 8: Cecilia's access attempt with Role usage

```
Checking access for FRED:

Start:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'start'

Status:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'status'

SetConfig:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'setConfig'

ReadConfig:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'readConfig'

Print:
File 'file_-377776722' successfully added to the print queue with job number 1
File 'file_-522496975' successfully added to the print queue with job number 2

Queue:
job number: 1, file name: file_-377776722
job number: 2, file name: file_-522496975

TopQueue:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'topQueue'

Stop:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'stop'

Restart:
User with name 'Fred' and roles [ordinary] not permitted to invoke operation 'restart'
```

Figure 9: Fred's access attempt with Role usage

The requirements presented for Task 2 and 3 were achieved in the role based access control system we designed, enforcing the access control policy discussed in the Section 3.

Finally, the final task was to change some access control specifications in both the systems developed to reflect the organisational changes made in the company: Bob left, George takes over Bob's role and two new people were hired, Henry as an ordinary user and Ida as a power user. In both the designs, the changes were well implemented, as we were able to change the information necessary in the database. After changing these values, we tested them in both implementations, getting the following outputs:

```
Checking access for IDA:

Start:
User with name 'Ida' and roles [power] not permitted to invoke operation 'start'

Status:
User with name 'Ida' and roles [power] not permitted to invoke operation 'status'

SetConfig:
User with name 'Ida' and roles [power] not permitted to invoke operation 'setConfig'

ReadConfig:
User with name 'Ida' and roles [power] not permitted to invoke operation 'readConfig'

Print:
File 'file_-2061406894' successfully added to the print queue with job number 1
File 'file_1904018140' successfully added to the print queue with job number 2

Queue:
job number: 1, file name: file_-2061406894
job number: 2, file name: file_1904018140

TopQueue:
Ida moves job: 2 to the top of the queue

Stop:
User with name 'Ida' and roles [power] not permitted to invoke operation 'stop'

Restart:
Ida restarted the server
```

Figure 10: New user Ida's access attempts with Role usage

```
Checking access for HENRY:

Start:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'start'

Status:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'status'

SetConfig:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'setConfig'

ReadConfig:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'readConfig'

Print:
File 'file_1040978263' successfully added to the print queue with job number 1
File 'file_1364917545' successfully added to the print queue with job number 2

Queue:
job number: 1, file name: file_1040978263
job number: 2, file name: file_1364917545

TopQueue:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'topQueue'

Stop:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'stop'

Restart:
User with name 'Henry' and roles [ordinary] not permitted to invoke operation 'restart'
```

Figure 11: New user Henry's access attempt with Role usage

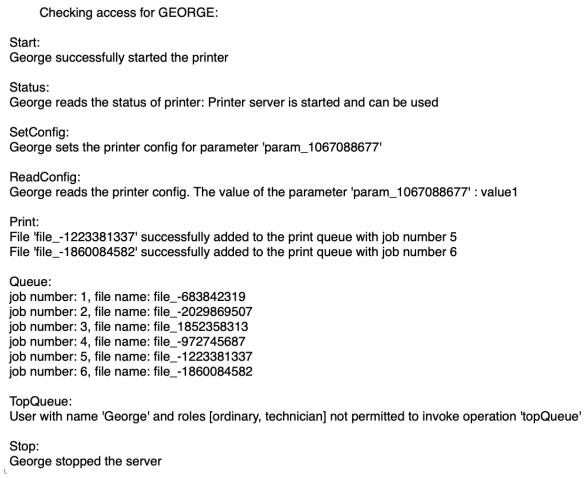

Figure 12: New user George's access attempt with Role usage

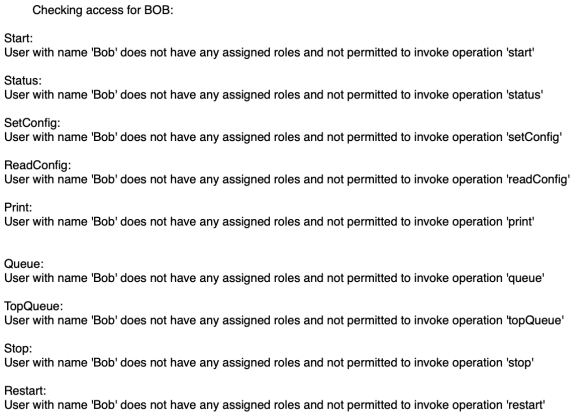

Figure 13: Deleted used Bob's access attempt with Role usage

The results shown were only for the Role Based Access Control implementation due to space constraints reasons, however the result matched our expectations for both implementations, meaning our program includes all the requirements specified in the project's guideline for both policies: it checks every access to the printer server, enforcing least privilege, allowing access to the fewest resources necessary for the user to complete each task, and verifying acceptable usage of the operations in the printer server. Every requirement was checked and tested in the client's side of the program by sending commands to the server.

5 Discussion

After implementing, testing and experimenting with these two different access control policies, we can clearly affirm that they have different applications where they can thrive.

For systems with few users, or where every user needs a specific set of access permissions, the use of ACL makes more sense, while in a large system such as a business' software platform, with various groups of users with the same needs and restrictions, RBAC is much more practical. ACL offers customization at the individual level, but lacks scalability and would not be feasible to effectively enforce in a system with thousands or millions of users, where each of them would need to be analyzed to determine which accesses they would be given and added to the list. On the other side, in the same situation, RBAC would allow for a systems administrator to quickly assign the correct permissions to new users, or change them when a user is promoted or changes departments, but is rather limited if someone has a particular set of resources they need access to that do not correspond to any existing roles.

Both policies have their limitations and they can both be flexible, just at different levels. In our final task, for example, using ACL all new situations had to be addressed individually, new users needed an entry on the policy files (the database tables) with their individual sets of permissions, old users needed all their permissions reviewed and potentially changed to fit the new requirements and deleted users needed their permissions removed. For the RBAC implementation, the new users just needed to be assigned with new roles, the old users with new requirements had their roles updated and the deleted user had his role removed. However, if it was the case that the access requirements in one of those cases did not match any role, the user would need to be assigned a role with more privileges, revoking our "least privilege" requirement, or a new role would need to be created, which would not be an efficient solution if it were to be done every time this situation came up.

All in all, both policies have strengths and weaknesses, and when building a system, a careful analysis must be performed to put together an ideal policy for the respective use case.

6 Conclusion

In conclusion, every requirement in this activity was successfully met with a working solution. An access control policy was enforced on top of our previously built authenticated printer server application, based on an access control list specified externally in a DBMS. We then implemented another access control policy, role based access control, once again on top of our previously developed system, by defining a role hierarchy with specific permissions associated with each role, and assigning a role to every user based on their access limitations. The role based access control policy files were also specified in the DBMS used. Finally, we implemented changes caused by hypothetical changes of permissions and inclusion of new users to both systems, and reflected upon the pros and cons of using each of the policies experimented with.

Future work could include adding an access logger to our code, which keeps a record of which user used which command of the server and at what time, as well as considering the use of an Attribute Based Access Control policy in our system and discuss how it compares with the other policies we used.

Nomenclature

ACL Access Control Lists

DBMS Database Management System

RBAC Role Based Access Control

References

[1] Department of Applied Mathematics DTU Compute and Computer Science. "Protection Mechanisms". In: (), pp. 6–17.

[2] Ben Lutkevich. *What is Access Control List (ACL)? - searchsoftwarequality*. Feb. 2022. URL: https://www.techtarget.com/searchnetworking/definition/access-control-list-ACL.

[3] Charles P Pfleeger. *Security in computing*. Prentice-Hall, Inc., 1988.

[4] *Role based access control: CSRC*. URL: https://csrc.nist.gov/projects/role-based-access-control.