

# Assignment 3 Data Security

Mukhtiar Khizar (s182696)

Sarker Tama (s232913)

Turki Yassine (s231735)

Zdrale Dimitrije (s231734)

November 2023

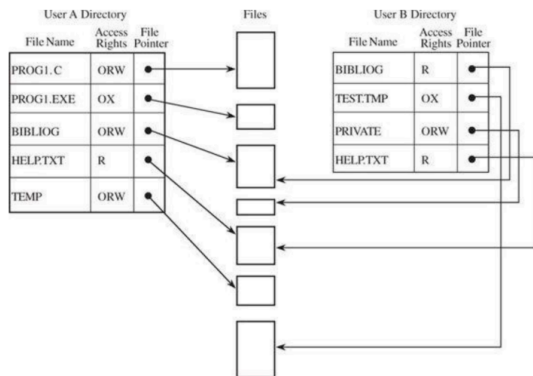


# 1 Introduction

In the previous implementation of the program, an authorized client was securely authenticated prior to gaining access to invoke operations on the remote print server. However, in that scenario, the client has the privilege to gain access to all the methods that the print server provides after the authentication process. Therefore, a problem arises when there are multiple clients and each is only allowed to have access to some specific operations. This is where access control comes into play which in essence sets limits for an entity to access some resource(s) and regulates this process. Thus, implementing such a mechanism in a distributed system that is comprised of client(s) and a server paves the way for a significant challenge, in other words, finding a footing that allows accessing some resources to permitted clients and prevents unauthorized access. To address this problem and get a better understanding, it is crucial to look into the basic paradigm of access control laid forth by Graham and Denning (1972) [2] who define it as “a subject is permitted to access an object in a particular mode, and only such authorized accesses are allowed”. The subject in this interpretation represents the entity that is trying to gain access to some resource, which is the object, and possibly perform some actions on it. Furthermore, the interaction that an entity is allowed and able to perform on any given resource is referred to as access modes. These key terms can also adhere to the components comprised in the client-server distributed system of the print server program. The entire mechanism behind granting some specific set of privileges to authorized users can be based on establishing a security policy from which all access control rules can be derived (P.Pfleeger et al., 2015 [3]). The developed software in this project sheds light on the aforementioned issue and an access control policy is adopted which is stored in a database management system. In the first iteration of development, specific users are granted different privileges based on the requirements specified for each individual. Further development incorporates the use of role-based access control based on a hierarchy which is the first thing that is developed in the second phase. Conclusively, a comparison of the implementation of the two mechanisms is reflected

## 2 Access Control Lists

P.Pfleeger et al. (2015) [3] argues that in most cases the operating system executes the functions pertaining to access control. For instance, some of the basic files can only be accessed and controlled by the operating system. Furthermore, the programs that constitute subjects (i.e. users) are also initiated and terminated by the operating system. Nonetheless, these mechanisms do not take into account facilitating the implementation of differentiated and finely detailed access control. Therefore, an external file can be employed that defines this policy. This analogy resonates well with the current print server prototype and based on the postulation laid forth by Anderson (1972) [2], access control will always be invoked upon the execution of the program and any attempt to gain access to some resource is validated. The process itself should be resistant to tampering and is supposed to work correctly. This construct is referred to as a reference monitor and serves as a conceptual theory behind the enforcement of security policy. However, in this report it is assumed that the policy is held secure and its confidentiality and integrity maintained. Similar assumption is also made for the source code of the program while also considering a secure channel for the communication between entities in this system. One of the ways to implement access rights in the program is to use some sort of a file directory which explicitly states which subject has what control right to some resource in the system. In essence, all the files that a particular user has access to are listed in the file directory pertaining to that user. Allowing the alteration of this file would result in compromising the access rights to some resource to an unauthorised entity. Therefore, safeguarding this file is utmost crucial in protecting the privileges entitled to a legitimate user for some resource. The most common rights that some user can perform on a file or resource are read, write, and or execute. This is demonstrated in Figure 1a below as an example for some hypothetical files and users, each having their own rights:



(a) Access rights to files for hypothetical users

	File A	Printer	System Clock
User W	Read Write Own	Write	Read
Admin		Write Control	Control

(b) Access Control List

Figure 1: Access rights and Access Control List

However, instead of maintaining a directory file for each user's access rights, an alternative approach would be the access control list which can be derived from the access control matrix and corresponds to the columns of the latter which is essentially a table where the left most column represents users and the top most row shows the files. Each entry in between axes represent the rights/privileges that a user has on a given file. The former approach can become quite cumbersome because the increased number of users would proportionally increase the number of file directories. Whereas, employing the use of the latter would not require making an entry in the individual directory of each user. The illustration for this can be seen in Figure 1b.

Similar mechanism can be implemented on the current print server program with multiple users, each having their own access rights. Furthermore, in this project the list is stored in a database management system which specifies the access to some operation for an individual user

and is invoked when the program is executed. The functions access table in the database serves this purpose and can be seen in Figure 2a below:

id	userid	functionid
1	Alice	1
2	Alice	2
3	Alice	3
4	Alice	4
5	Alice	5
6	Alice	6
7	Alice	7
8	Alice	8
9	Alice	9
10	Bob	4
11	Bob	5
12	Bob	6
13	Bob	7
14	Bob	8
15	Bob	9
16	Cecilia	1
17	Cecilia	2
18	Cecilia	3
19	Cecilia	6
20	David	1
21	David	2
22	Erica	1

(a) Access to different operations for each individual user

id	functionid	functiontitle
1	1	print
2	2	queue
3	3	topqueue
4	4	start
5	5	stop
6	6	restart
7	7	status
8	8	readconfig
9	9	setconfig

(b) Functions table in the database

Figure 2: Access Control Information

The users and functions are imported as foreign keys in this table and are derived from the user's table, which is the same as in the original implementation of the program, and functions table that are essentially the operations on the print server. The latter can be seen in Figure 2b

Moreover, as indicated in the policy, Alice is able to gain access to all the operations on the print server after successful log-in and Bob has the privilege to access only a few of the operations. This is demonstrated in Figure 3 below:

```

ApplicationServer x Client x
/Library/Java/JavaVirtualMachines/jdk-21.jdk/
-----From Server: Server is Connected-----
1. Login
0. Exit
Enter Choice :1
Enter Username = Alice
Enter Password = acoolpassword
0. Exit
1. setconfig
2. readconfig
3. status
4. restart
5. stop
6. start
7. topqueue
8. queue
9. print
Enter Choice :

```

```

ApplicationServer x Client x
/Library/Java/JavaVirtualMachines/jdk-21.jdk/
-----From Server: Server is Connected-----
1. Login
0. Exit
Enter Choice :1
Enter Username = Bob
Enter Password = somepassword
0. Exit
1. setconfig
2. readconfig
3. status
4. restart
5. stop
6. start
Enter Choice :

```

Figure 3: Access rights for Alice (left) & Bob (right)

The same holds true for each individual user based on their access rights defined in the policy.

### 3 Role Based Access Control

In the previous section, granting access to the invocation of operations on the print server was based on the access rights that one specific individual had. Although the access to an operation is defined and restricted based on user privileges, this can get difficult to maintain when scaling the system. Therefore, to circumvent this issue it is crucial to define and add different roles in the policy with each role having its own set of access rights. Thus, when adding or removing users they are accordingly assigned or revoked access to a given resource respectively. For instance, an administrator in the company with the remote printer might have the privilege to access almost all operations whereas a desk clerk should only be able print a document and access to the rest of the methods are rendered unnecessary for them [3] This aids in defining a more robust policy for managing access control for users based on their roles. A role can be most commonly postulated as allocating actions within a system under some name [1]. Furthermore, this can be extended to the current implementation of the program based on the roles and their hierarchy that are illustrated in Figure 4 below:

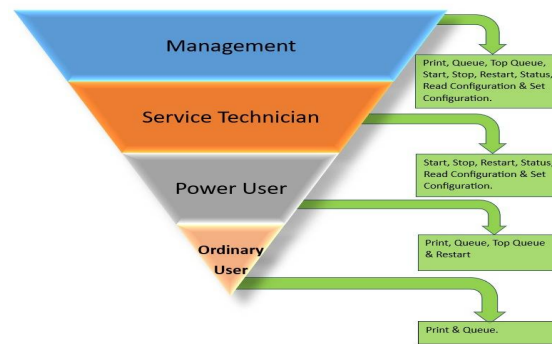


Figure 4: Role Based Hierarchy

The roles themselves are explicitly defined in a new table in the database and the entire connection between all the components of the system in the database management system can be seen in an entity relationship diagram and is depicted in Figure 5:

ER Diagram

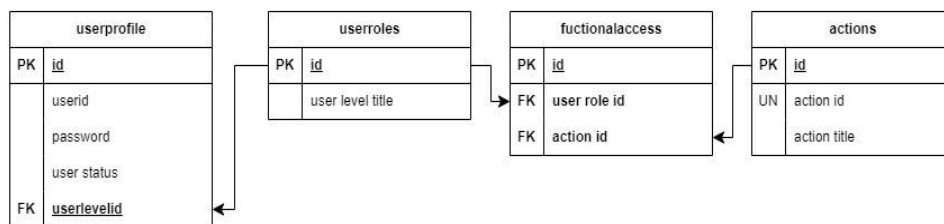


Figure 5: ERD of the Database Management System

The roles are divided into different levels and imported as a foreign key in the user profile where each individual user is assigned a level, contrary to the last section where access rights were granted based on the specific user. This can be seen in the actual table in the database represented in Figure 6 below:

	id	userid	password	ulevel	activeuser
<input type="checkbox"/>	12	alice	a665a45920422f9d417e4867efdc4fb8a04a1311fa07e99...	1	1
<input type="checkbox"/>	13	bob	a665a45920422f9d417e4867efdc4fb8a04a1311fa07e99...	2	1
<input type="checkbox"/>	14	cecilia	a665a45920422f9d417e4867efdc4fb8a04a1311fa07e99...	3	1
<input type="checkbox"/>	15	david	a665a45920422f9d417e4867efdc4fb8a04a1311fa07e99...	4	1
<input type="checkbox"/>	16	erica	a665a45920422f9d417e4867efdc4fb8a04a1311fa07e99...	4	1
<input type="checkbox"/>	17	fred	a665a45920422f9d417e4867efdc4fb8a04a1311fa07e99...	4	1

Figure 6: User's profile table

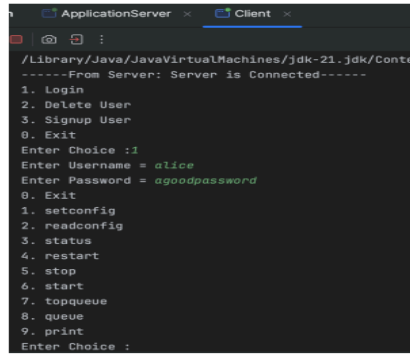
In addition, the program also keeps track of whether the user's profile exists or is deleted. The functions are the operations that can be invoked and are referred to as actions in this diagram and is similar to the previous implementation of access control which can be seen in Figure 2b. Moreover, the allocation of privileges amongst different roles are subsequently defined in the functional access table. The roles are divided into several levels and derived from the user role table. These two tables indicate where significant changes are taking place in refining the security policy and incorporates the role based access control mechanism. Both can be seen in the database is depicted in Figure 7 below:

	id	roletitle
<input type="checkbox"/>	1	management
<input type="checkbox"/>	2	servicetechnician
<input type="checkbox"/>	3	poweruser
<input type="checkbox"/>	4	ordinaryuser

	id	roleid	functionid
<input type="checkbox"/>	1	1	1
<input type="checkbox"/>	2	1	2
<input type="checkbox"/>	3	1	3
<input type="checkbox"/>	4	1	4
<input type="checkbox"/>	5	1	5
<input type="checkbox"/>	6	1	6
<input type="checkbox"/>	7	1	7
<input type="checkbox"/>	8	1	8
<input type="checkbox"/>	9	1	9
<input type="checkbox"/>	10	2	4
<input type="checkbox"/>	11	2	5
<input type="checkbox"/>	12	2	6
<input type="checkbox"/>	13	2	7
<input type="checkbox"/>	14	2	8
<input type="checkbox"/>	15	2	9
<input type="checkbox"/>	16	3	1
<input type="checkbox"/>	17	3	2
<input type="checkbox"/>	18	3	3
<input type="checkbox"/>	19	3	4

Figure 7: Role table dividing each role into different levels (left) & Assigning functions to different roles based on their level (right)

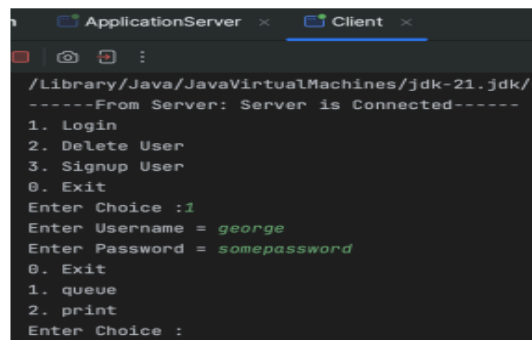
When a user runs the system, they will be prompted with the login screen that is printed on the console requesting credentials. The program then authenticates the user based on the same principles as defined in the original implementation from previous assignment, i.e. password is securely stored as a one way strong cryptographic hash and the server keeps track of each session to prevent a possible replay attack in the future. After successful authentication process, the user can only access operations on the print server based on the privileges they are allocated that is indicated in the policy stored in the database management system. For instance, if someone from the management were to log-in they would be able to access all the operations on the printer. This is demonstrated in Figure 8 below where Alice is a level 1 user defined in the database and has all the access rights:



```
ApplicationServer x Client x
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java
-----From Server: Server is Connected-----
1. Login
2. Delete User
3. Signup User
0. Exit
Enter Choice :1
Enter Username = alice
Enter Password = agoodpassword
0. Exit
1. setconfig
2. readconfig
3. status
4. restart
5. stop
6. start
7. topqueue
8. queue
9. print
Enter Choice :
```

Figure 8: Management user log-in

On the contrary, George is an ordinary user and falls under level 4, thus, having access to only two operations and this is represented in Figure 9:



```
ApplicationServer x Client x
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java
-----From Server: Server is Connected-----
1. Login
2. Delete User
3. Signup User
0. Exit
Enter Choice :1
Enter Username = george
Enter Password = somepassword
0. Exit
1. queue
2. print
Enter Choice :
```

Figure 9: Ordinary user log-in

Similarly every role has its own designated level each with its own set of accesses to operations. Furthermore, to ease the comparison of the two implementations, sign-up and delete methods have been developed which will be reflected upon in the next section.

## 4 Evaluation

Now we are facing certain changes in the example company hierarchy. These changes must be reflected accordingly both on the ACL and RBAC implementation of the access control policy.

Implementing any changes within company hierarchy can often require one or the other policy. RBAC is useful if the changes encompass already existing role, and it is quicky to assign/reassign them. However, sometimes the existing roles are not enough, and that's when ACL's flexibility comes in handy.

First, we will address the changes in the RBAC implementation. Since Bob has left the company, his role and user profile must be deleted from the database altogether. As George takes over the service technician responsibilities, we must create a new role that is specific for George, encompassing both his previous and service technician privileges. These changes can be seen in the figures below.

```

ApplicationServer x Client x
/Library/Java/JavaVirtualMachines/jdk-21.jdk/
-----From Server: Server is Connected-----
1. Login
0. Exit
Enter Choice :1
Enter Username = George
Enter Password = anothergoodpassword
0. Exit
1. setconfig
2. readconfig
3. status
4. restart
5. stop
6. start
7. queue
8. print
Enter Choice :

```

Figure 10: George's available actions

id	roletitle
1	management
2	servicetechnician
3	poweruser
4	ordinaryuser
5	newRoleForGeorge

(a) George's new role

id	roleid	functionid
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	5	4
11	5	5
12	5	6
13	5	7
14	5	8
15	5	9
16	5	1
17	5	2
18	3	3
19	3	6

(b) Functions available for George's new role (roleid = 5)

Figure 11: George's Role and Associated Functions



Next, we look at the two new employees: Henry and Ida. Henry is set to receive the ordinary user privileges, while Ida gets the power user role, just like Cecilia. The changes are visible in the figures 12 below.

```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/
-----From Server: Server is Connected-----
1. Login
2. Delete User
3. Signup User
0. Exit
Enter Choice :1
Enter Username = henry
Enter Password = abombasticpassword
0. Exit
1. queue
2. print
Enter Choice :

```

(a) Henry's available actions

```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/
-----From Server: Server is Connected-----
1. Login
2. Delete User
3. Signup User
0. Exit
Enter Choice :1
Enter Username = ida
Enter Password = insanepassword
0. Exit
1. restart
2. topqueue
3. queue
4. print
Enter Choice :|

```

(b) Ida's available actions

Figure 12: Henry and Ida's available Actions

Implementing these changes required a generation of a new role for George as combination of two existing ones, which might not be the best solution for such a change. However, the process for Henry and Ida was quite simple: registering a user and assigning an already existing role to them.

As for the ACL, the approach is somewhat different. As before, Bob is removed from the database, and in order for George to receive necessary permissions, we just add all the service technician functions to his permitted action list. As for the new employees, Henry and Ida, they are registered, and are given access to functions respective of their role in the company hierarchy. The changes can be seen in the figure below.

					id	userid	fuctionid
<input type="checkbox"/>	Edit	Copy	Delete		7	Alice	7
<input type="checkbox"/>	Edit	Copy	Delete		8	Alice	8
<input type="checkbox"/>	Edit	Copy	Delete		9	Alice	9
<input type="checkbox"/>	Edit	Copy	Delete		10	George	4
<input type="checkbox"/>	Edit	Copy	Delete		11	George	5
<input type="checkbox"/>	Edit	Copy	Delete		12	George	6
<input type="checkbox"/>	Edit	Copy	Delete		13	George	7
<input type="checkbox"/>	Edit	Copy	Delete		14	George	8
<input type="checkbox"/>	Edit	Copy	Delete		15	George	9
<input type="checkbox"/>	Edit	Copy	Delete		16	George	1
<input type="checkbox"/>	Edit	Copy	Delete		17	George	2
<input type="checkbox"/>	Edit	Copy	Delete		18	Cecilia	3
<input type="checkbox"/>	Edit	Copy	Delete		19	Cecilia	6
<input type="checkbox"/>	Edit	Copy	Delete		20	David	1
<input type="checkbox"/>	Edit	Copy	Delete		21	David	2
<input type="checkbox"/>	Edit	Copy	Delete		22	Erica	1
<input type="checkbox"/>	Edit	Copy	Delete		23	Erica	2
<input type="checkbox"/>	Edit	Copy	Delete		24	Fred	1
<input type="checkbox"/>	Edit	Copy	Delete		25	Fred	2
<input type="checkbox"/>	Edit	Copy	Delete		29	Ida	1
<input type="checkbox"/>	Edit	Copy	Delete		30	Ida	2
<input type="checkbox"/>	Edit	Copy	Delete		31	Ida	3
<input type="checkbox"/>	Edit	Copy	Delete		32	Ida	6
<input type="checkbox"/>	Edit	Copy	Delete		40	Henry	1
<input type="checkbox"/>	Edit	Copy	Delete		41	Henry	2

(a) New users and George with their permitted functions

```
SELECT * FROM `userprofile`;
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Restore column order | Number of rows: 25 | Filter rows: Search this table

Extra options

					id	userid	password
<input type="checkbox"/>	Edit	Copy	Delete		1	Alice	c058556caa06689eb0928c062452e4d625bdc468983d43e1d...
<input type="checkbox"/>	Edit	Copy	Delete		3	Erica	6b86b273f34fce19d8b804eff5a3f5747ada4ea22f1d49c0...
<input type="checkbox"/>	Edit	Copy	Delete		4	Cecilia	37bc885f08c043f1cb9852956e01ca0d07ba664a5eee14fc41...
<input type="checkbox"/>	Edit	Copy	Delete		5	George	6158081d8937a06b2fede2ed345fc661bc1024e6295d283510...
<input type="checkbox"/>	Edit	Copy	Delete		6	David	4b227777d4dd1fc61c6f884f48641d02b4d121d3fd328cb08b...
<input type="checkbox"/>	Edit	Copy	Delete		7	Fred	ef2d127de37b942baad06145e54b0c619a1f22327b2ebbcfb...
<input type="checkbox"/>	Edit	Copy	Delete		8	Henry	ca20941cc576dafeb141a58d85db28b5d6f18d852da734258f...
<input type="checkbox"/>	Edit	Copy	Delete		9	Ida	1db5bd4516dfde002e1eb4000d773ad8a1912a28307f404677...

(b) Database with the new user accounts

Figure 13: User Changes and Database Updates

As mentioned, the users are registered in the database with the hash of their chosen password.

## 5 Discussion

We have seen throughout this assignment that the goals given to us could be achieved using either ACL or RBAC. It was apparent, however, that some tasks were easier to handle with ACL, whereas we found that RBAC was very suitable for other tasks.

When comparing these two access control schemes, we have noticed some pros and cons for both. Firstly, RBAC is optimal for large organizations, where roles are predefined and assigned permissions upon creation. Therefore, a role is generally not modified/changed, and it is thus very easy to assign a specific role to a specific person. The maintenance process using RBAC includes the insertion, deletion or modification of a role. If these roles are thoroughly designed, the organization will not have to modify them unless to solve a new specific need. Meanwhile, when using ACL, one has to define access rights to every user, which can be tedious within large organizations, and the process is prone to human errors, which could lead to fatal consequences if a user possesses rights he should not have. Nonetheless, ACL offers a lot of flexibility compared to RBAC, which makes it highly suitable for small or rigid hierarchical organizations. For example, a startup will need to continuously update the access rights to its employees, as their role is rapidly changing due to low staffing. This constant change in responsibilities would be tedious to implement with RBAC, which would entail creating a new role each time a user would be granted new responsibilities. This is because if we modify the policy of a role, the changes will affect the whole group that shares the same role, and not solely to the specific individual. Furthermore, in the case where we have a rigid hierarchical organization, such as the military, it is extremely important to define boundaries with respect to user rights and allowed actions. This makes ACL more suitable, as we are more in control of which individual can do what specifically. With RBAC, creating a new role would only make sense if the new responsibility would serve the purpose of additional users that share the same role.

It is capital to consider security aspects when talking about these access control schemes. ACL and RBAC are both used to ensure that access to specific functions, files, or resources is only granted to users with permission. As mentioned before, ACL is prone to human errors, especially if the company is large or if the number of operations is large (constantly updating user rights will eventually lead to a user having access to confidential files or prohibited functions). It is also hard to keep track of which permission should have been removed for a specific user, and removing permissions is also tedious using ACL since we have to go through each user and manually remove the obsolete permissions. The task becomes even more challenging when considering the fact that some permissions can be deleted when they should not, causing a lack of efficiency and difficult maintainability with the ACL scheme. However, if we were to use RBAC, we would avoid these errors and efficiently ensure that no matter the number of users, they will always have a defined access to functions and data, and the policy will remain constant. Nevertheless, the fact that RBAC entails having multiple users sharing the same role could lead to some security concerns. Indeed, in most situations, we will end up with users having more permissions than they should have, which could be potentially harmful in military organizations for example. The fact that defining a new role each time a user would like to have additional permission will likely entail that the user will be moved up to the next hierarchical position, which would lead to the user of having access to more than he should have.

It is thus logical to use ACL with small organizations, as it is not convenient to create a new role specifically for one person. However, for larger organizations, RBAC would be the ideal policy since it scales well and granting new user permission to a role would allow more efficient updates in access for a whole group.

## 6 Conclusion

This report extensively delves into the implementation of data security measures within a program, particularly focusing on access control mechanisms. Initially, it identifies the limitations of the previous authentication system, where authorized clients gained access to all available methods on the remote print server. This led to the necessity of implementing access control to restrict client-specific operations. Two primary access control methods, Access Control Lists (ACL) and Role-Based Access Control (RBAC), were thoroughly discussed. ACL emphasizes the definition of access rights for individual users in a file directory format, specifying the permissions of each user for various resources. In contrast, RBAC introduces a hierarchical role-based approach, allocating predefined roles with associated privileges to users, ensuring scalability and easier maintenance in larger organizations. The evaluation section highlights the advantages and challenges associated with ACL and RBAC. ACL's flexibility suits smaller or rigid hierarchical organizations, offering detailed control but posing challenges in scalability and maintenance. On the other hand, RBAC's predefined roles excel in larger organizations, simplifying user management but potentially leading to excess permissions for some users. Conclusively, both ACL and RBAC serve as viable solutions for enforcing authorization policies, each tailored to specific organizational needs. Small or rigid hierarchical organizations may benefit from ACL's granularity, whereas larger organizations could find RBAC more efficient due to its scalability. The report emphasizes the importance of choosing the right access control mechanism based on the organization's size, structure, and dynamic user roles. It is worth mentioning that all the requirements defined in the project description were satisfied with both protocols. In future work, exploring hybrid approaches that combine the strengths of ACL and RBAC could be a valuable direction. Additionally, continuous refinement and adaptation of access control policies based on organizational changes remain a crucial aspect of ensuring robust data security. Automating this process instead of manual modification would be a great improvement.

## References

- [1] Luigi Giuri. *Role-Based Access Control: A Natural Approach*. 1996.
- [2] G. S. Graham and P. Denning. Protection—principles and practice. In *Proc AFIPS Spring Joint Computer Conf*, pages 417–429, 1972.
- [3] Charles P. Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies. *Security in Computing*. Prentice Hall, 5th edition, 2015.