Data Security, Assignment 2, Group 23

Mukhtiar Khizar (s182696)

Sarker Tama (s232913)

Turki Yassine (s231735)

Zdrale Dimitrije (s231734)

November 2023

Table of Contents

Introduction

The foundational principals of achieving information and data security in the realm of digital systems is based on the CIA (Confidentiality, Integrity, and Availability) triad. However, to fully understand the security requirements of a particular system or an application, a much in depth assessment would be required that incorporates the implementation of relevant security policies and procedures. Thus, when narrowing down to an application that encompasses a distributed system, the exact requirements might vary depending on the required set of functionalities in the application. Nonetheless, the most crucial aspect that should always be taken into account is the process of authentication in such a system. Furthermore, a typical distributed system involves a client (might server as an end-user) that requests services from a remote host, also known as a server. When taking into consideration the vast network facilitated by the Internet, the task of achieving authentication between these two entities can be quite cumbersome. How can a server make sure that some client, requesting from it a service or data, is actually genuine and not someone else and vice versa? Moreover, this is only one of the myriad lenses that the problem of authentication might be viewed from. It is also imperative that the entities do remain authentic throughout the established communication. (Stallings, 2011)

In order to achieve authentication, one of the first steps would require the client to prove themself as authentic. This may be accomplished by client and server agreeing on some credentials that might comprise of a type of identification along with a password, smart card, and or biometrics. Upon establishing a communication and requesting access to services or data that a client is eligible, the latter can use these credentials to gain access. However, the major obstacle to overcome in this endeavour is to make sure that the client submitting these credentials is in fact who they claim to be. This is where multi-factor authentication might come into play, that requires a client to not just input their password but more information. Other additional steps to take can also include the secure storage of a cryptographically strong password. Furthermore, the methods and techniques involving security that ensures the regulation of authenticating genuine users to some resources is covered using access control mechanisms (Lutkevich, 2022)

In addition, the process of authentication involves other major side of the coin, that puts spotlight on the problem of the server being authentic and not a malicious entity that the client is trying to communicate with. Or even worse, both honest entities come under the assumption that they have authenticated each other but instead they are communicating with an intruder. A classic case is analogous to the attack on the Needham Schroeder protocol, in which the authentication principals are violated by an intruder performing a man-in-the-middle-attack. (DS Lecture 2)

A comprehensive study of several security mechanisms will be addressed in this report that touches upon the problem of secure storage and transportation of a password and making the process of authentication more secure. The solution proposed in this paper focuses on countermeasures for aforementioned complications that typically arises in a distributed system comprised of a client and a server. Moreover, the developed software puts in practice, part of the solution to the vast array of problems revolving around authentication. In essence, the client's password is stored as a one-way hash in a database management system that the server then uses to authenticate the latter.

Authentication

The most common method of authentication in client and server applications employ the usage of a password system. The entire *raison d'être* of such a system is to essentially provide a password to verify some identity pertaining to an entity. Thus, this technique is a cost effective and efficient way to preserve a shared secret between two entities (Conklin et al. 2004).

A simple demonstration of this system in visualised in the figure below:

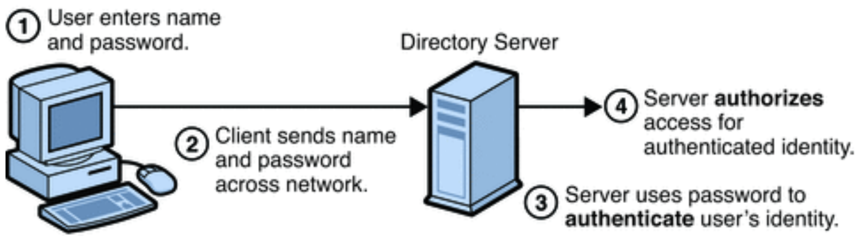

1 User enters name and password.
Directory Server
4 Server **authorizes** access for authenticated identity.
2 Client sends name and password across network.
3 Server uses password to **authenticate** user's identity.

Figure 1: Client authenticating itself to server using ID and password

Source: https://docs.oracle.com/cd/E19424-01/820-4811/gdzeq/index.html

As seen in Figure 1, the client enters their credentials that comprises of some ID and password, which the server might check in its database to see if there is a match and then accordingly grants access to the services and/or data that the latter is entitled to.

However, it is important to consider the penultimate steps that requires establishing trust and initial exchange of credentials between the two entities. Furthermore, there are alternative approaches to the storage of a password and its management paves the way for a variety of complex security concerns.

When storing the password, access control mechanisms are the initial and most crucial steps to take that ensures that unauthorised access is prevented. However, these mechanisms are not covered in this report. Moreover, a potential security risk arises if the password itself is stored as a plain text. Thus, to mitigate the chances of this vulnerability, cryptography can be employed in such an instance.

This is best explained by Stallings (2011) interpretation of the threats revolving around password based authentication systems and gives the analogy of a commonly used scheme on UNIX which does not store a password as a plaintext. A different approach is rather taken into place that uses 7-bit ASCII to convert the user input of plain password to a value of 56-bits which is used as a key to an encryption routine. The algorithm used is altered with a cryptographic "salt" of 12-bits. Moreover, this modified algorithm is then executed on a data block of 64-bits zeros. The output is then used as an input for a second round of encryption. Furthermore, this operation is carried out repeatedly for 25 encryptions and a sequence of 11-characters is obtained by converting the resulting 64-bit output. Conclusively, a user ID with its corresponding hashed password, along with a copy of the salt in its plaintext format, is stored in a password file.

This process is succinctly represented in figure 2(a) below. Furthermore, the verification of the password is demonstrated in 2(b) in which a user logging-in to a UNIX system provides their ID and password. The ID is used as an index to retrieve the corresponding encrypted password and the plaintext salt, and they are both inserted as input in the encryption routine. The resulting output is then compared with the password and the user is allowed access to if there is a match (ibid.,).

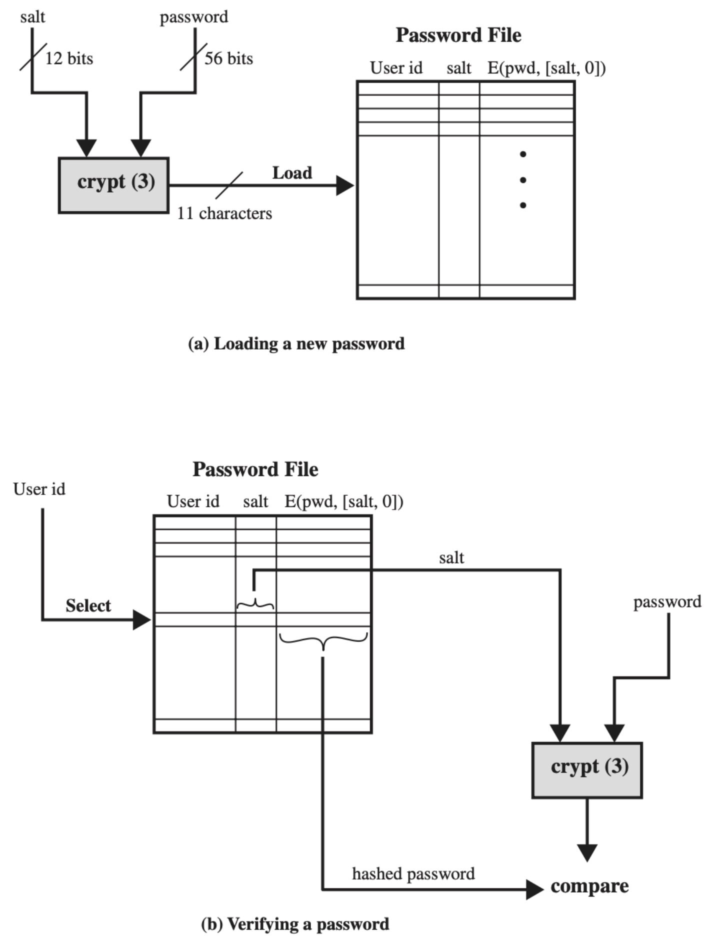


Figure 2 - Password authentication scheme in UNIX
Source - Stallings (2011)

A wide range of attacks have been thwarted by employing this strategy and the design of the encryption routine aims to prevent guessing attacks. The printing server can employ this strategy for storing the encrypted password, ID, and salt in a system file, however, even after incorporating the supposedly secure and safety mechanisms this approach is still susceptible to threats. It is worth mentioning that the algorithm used in the encryption routine, which is crypt (3) in this case, is DES and it has been known to be vulnerable to attacks. For instance, the Morris worm uses a more efficient algorithm to guess passwords using brute force in a shorter period of time. Furthermore, according to Moore's Law, with rising computational power there is an increased chance of using stronger algorithms for cryptanalytic attacks on the standard encryption algorithm employed by the UNIX system (Kaliski B., 2011).

Moreover, it is imperative to note that in most cases when users are given an unrestricted choice to choose any password to their liking, they will adhere to a very short and simple one. Thus, an attacker might not need to employ a rigorous brute force approach to guessing a password. Instead, browsing through a dictionary of guessable passwords can expedite the process of cracking one and gaining unauthorised access to a system. No matter how and where the password is securely stored, if it is easily guessable then the entire security mechanisms are compromised. To circumvent this issue, the users might be forced by the system to use a longer and stronger password by incorporating special characters. Furthermore, a more efficient algorithm can be used that is much faster than the one currently in use in UNIX and as mentioned earlier access to the password file must be restricted to only a privileged user, thus maintaining the confidentiality and integrity of the password (Stallings, 2011).

Somewhat similar steps can be taken into account when storing the password in a database management system, however, there is a drawback in regards to storing the salt securely on the user's machine. Therefore, the password should be stored as a stronger one-way hash in the database on the server and the communication is carried out on a TLS channel which will also prevent eavesdropping when the password is in transit. Moreover, the verification can be carried out as in the system file, where the input password is converted to hash and the server compares it to the one stored with the corresponding user ID and provide access to services or data if there is a match. However, to possibly prevent an attacker from using information of previous sessions to gain access to the system later on, a unique session ID might be used that it binds to a specific user requesting a service at a given moment which eventually expires after a certain time. Thus, rendering any communication from previous sessions useless.

Finally, storing the password in a public file without using any secure means would make it easily accessible to unauthorised users. Therefore, mitigating this issue would require following the same principles as aforementioned. In addition to hashing, the usage of unique cryptographic salt with each password prevents it from being exposed to cracking with rainbow tables (Gillis, 2022).

Design and Implementation

As mentioned previously in this report, the storage of a password as a plaintext file compromises its confidentiality and integrity. This assertion holds true regardless of choosing the platform to store the password, thus, making it susceptible to unauthorised access and myriad possible attacks. Therefore, to ensure that the principle components of security are not violated in this respect, the use of cryptography is employed. However, this comes at a cost of complicating the process of authentication.

Nonetheless, when storing a password securely in either a system file or a public file would require taking extra steps to make sure that access to the file itself is restricted since it might be possible to gain entry to the platform that hosts these files. This might pave the way in allocating more resources.

Hence, a database management system can be used by the remote print server for managing the users robustly with adherence to security mechanisms and will be implemented in this project. The data flow diagram of the system can be seen in the figure 3 below, which visually represents the flow of data portrayed in terms of its components and looks at how the system is connected with emphasis on the authentication process (Lucidchart, 2023).

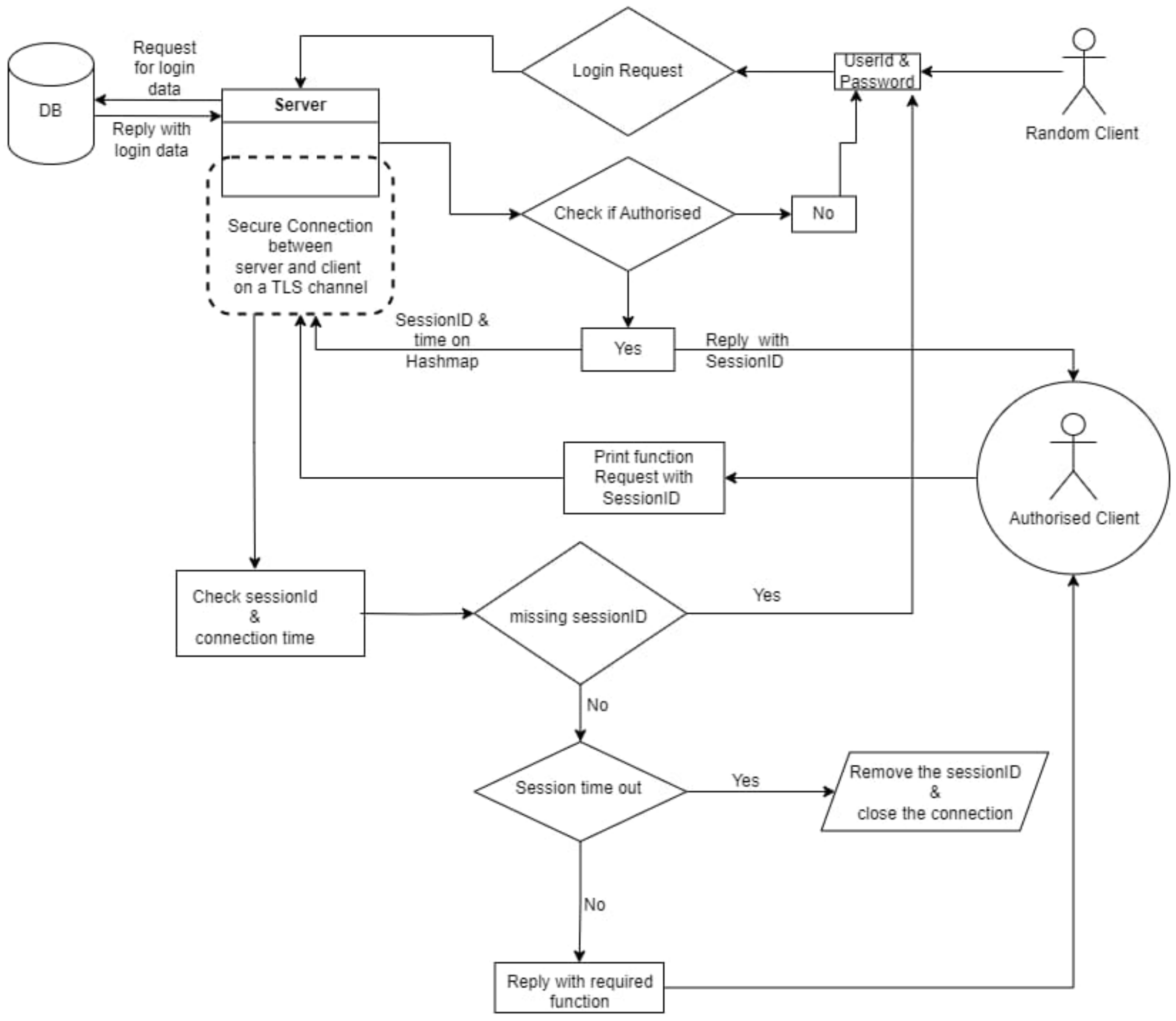
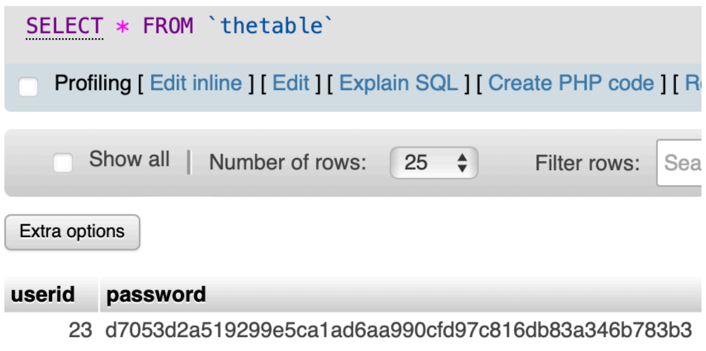

DB
Request for login data
Reply with login data
Server
Login Request
UserId & Password
Random Client
Check if Authorised
No
Secure Connection between server and client on a TLS channel
SessionID & time on Hashmap
Yes
Reply with SessionID
Print function Request with SessionID
Authorised Client
Check sessionId & connection time
missing sessionID
Yes
No
Session time out
Yes
Remove the sessionID & close the connection
No
Reply with required function

Figure 3 - Data flow diagram of the system

It is important to note that the entire communication between every agent must be hosted on a TLS channel and the communicating entities are indeed honest. However, the protocol itself will not be implemented in this project. Furthermore, as seen in the figure, a user must provide some credentials, which can be a user ID and a password, in order to log in and gain access to the services they are entitled to.

However, to ensure a secure storage of the password, it must be stored as a one-way cryptographic hash in the database. Therefore, upon connecting to the server, the plaintext password from the client side is computed to a hash which is then compared to the one stored in the database to verify if there is a match. Upon confirmation is the client able to gain access to the services and in this case, invoke methods from the remote server using Java Remote Method Invocation. Moreover, An open-source web server solution, XAMPP, is used as the database management system for this project (Apache, 2022).

The enrolment of users in the database is done manually and the passwords corresponding to an ID is first converted to SHA-256. The users table in the database can be seen in figure 4 below:


```
SELECT * FROM `thetable`
```
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [R
Show all | Number of rows: 25 Filter rows: Sea
Extra options

userid	password
23	d7053d2a519299e5ca1ad6aa990cfd97c816db83a346b783b3

Figure 4 - User’s table in the database

Similar steps are followed when the user is logging-in to the system, where the plaintext password is once again converted to the similar strong cryptographic hash and then compared with the one retrieved from the database corresponding to the user ID, which is used as an index. The log-in process can be seen in figure 5 below:

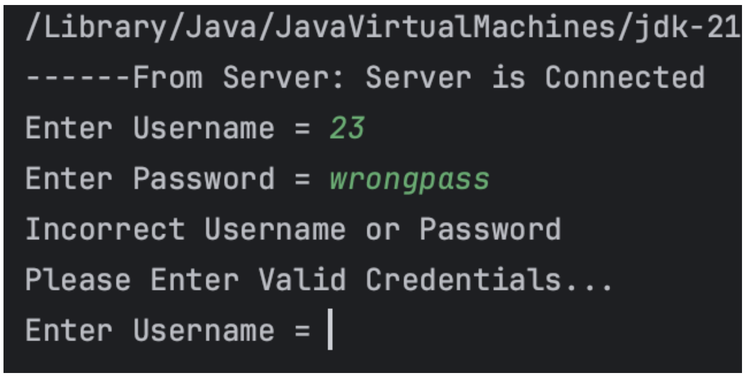
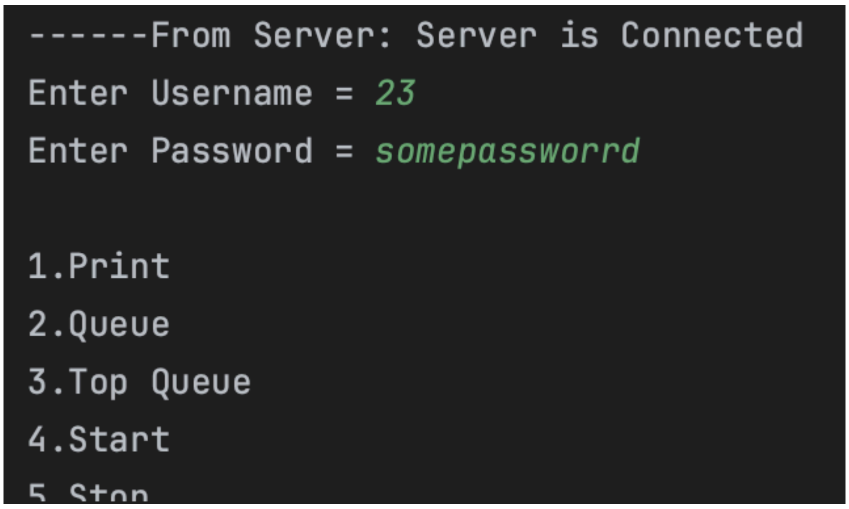

```
/Library/Java/JavaVirtualMachines/jdk-21
------From Server: Server is Connected
Enter Username = 23
Enter Password = wrongpass
Incorrect Username or Password
Please Enter Valid Credentials...
Enter Username = 
```

Figure 5 - Client login (incorrect password)

As seen in the figure above, when the client enters an invalid credential they are denied access to the system. However, if the credentials are indeed correct, i.e. a valid match in the database, then the client is authenticated. This can be seen in figure 6 below:

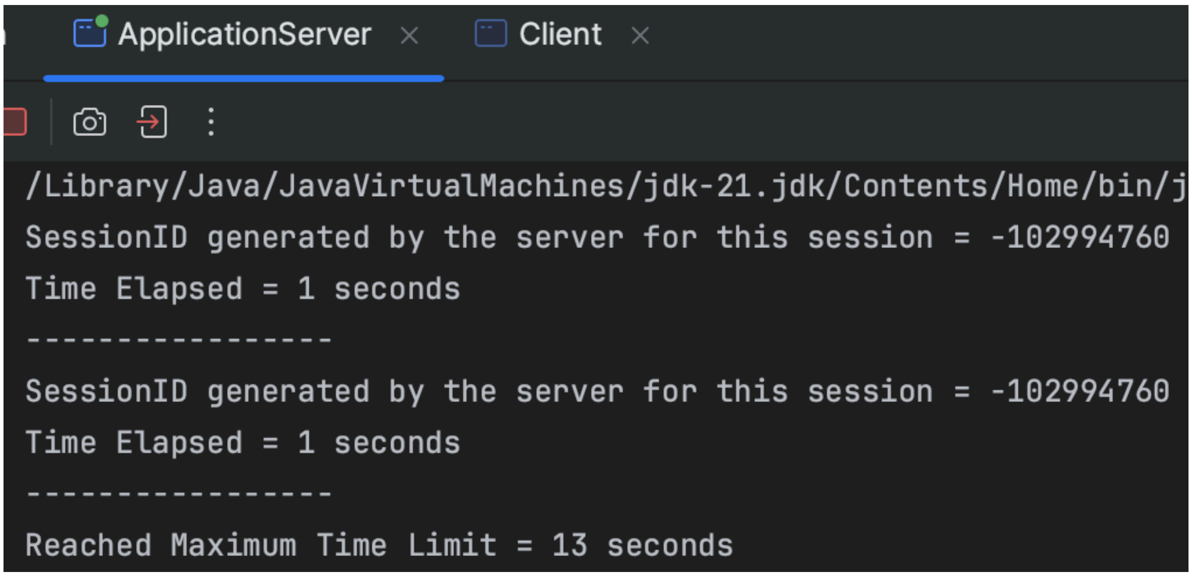


```
------From Server: Server is Connected
Enter Username = 23
Enter Password = somepassworrd

1.Print
2.Queue
3.Top Queue
4.Start
5.Stop
```

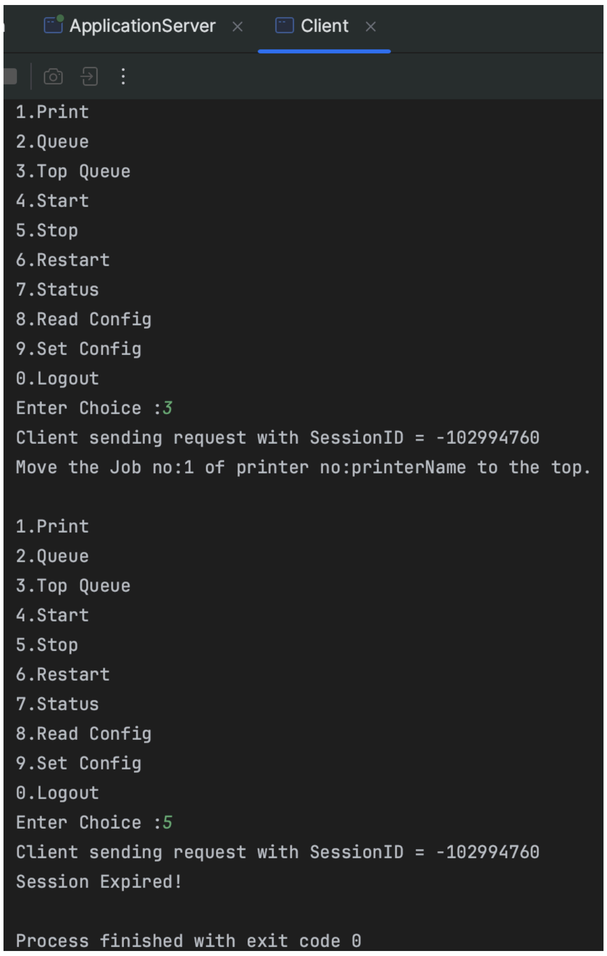
Figure 6 - Valid credentials log-in

Subsequent to authentication, the client is able to access all the operations that the print server provides. Furthermore, to ensure that the server keeps track of the session established by the client throughout the communication, a unique session ID is generated by the former which it binds to the requesting client and after a certain amount of time (ten seconds in this case) the session ID expires and the communication is terminated, as depicted in figure 3. This step may help in preventing an intruder to store a communication in a given session and replay it to the server in the future. This process can be seen in figures 7 and 8 below:



```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/j
SessionID generated by the server for this session = -102994760
Time Elapsed = 1 seconds
-----------------
SessionID generated by the server for this session = -102994760
Time Elapsed = 1 seconds
-----------------
Reached Maximum Time Limit = 13 seconds
```

Figure 7 - Server generating unique session ID and keeping track of time



```
1.Print
2.Queue
3.Top Queue
4.Start
5.Stop
6.Restart
7.Status
8.Read Config
9.Set Config
0.Logout
Enter Choice :3
Client sending request with SessionID = -102994760
Move the Job no:1 of printer no:printerName to the top.

1.Print
2.Queue
3.Top Queue
4.Start
5.Stop
6.Restart
7.Status
8.Read Config
9.Set Config
0.Logout
Enter Choice :5
Client sending request with SessionID = -102994760
Session Expired!

Process finished with exit code 0
```

Figure 8 - Client responding with the same session ID when invoking a method

Whenever the client is invoking a method on the print server, the former responds with the session ID it was assigned to for this specific communication which is always checked by the latter when a service is being requested. Thus, making sure that the client is indeed authentic throughout the communication.

Evaluation

The design and implementation of the system covers a significant number of requirements laid forth in the second section of this report, which touched upon the broader problems revolving around password based authentication and some possible existing solutions. It is worth taking into consideration that no system is completely secure and there will always be ever increasing threats to it, hence, making it vulnerable to attacks.

Nonetheless, the most crucial requirement of essentially authenticating the client prior to invoking any service on the remote server is indeed fulfilled. The client must have an identification and a password of which the latter is stored as a one-way hash in a database management system. Every time the client is trying to gain access to the services, they must provide valid credentials. Furthermore, the server creates a random and unique session ID for each session which has a time limit and the client must always respond with the same ID when requesting a service. Therefore, the client is authenticated each time when invoking an operation and they must be quick in their actions.

Subsequently, after authenticating the client with valid credentials, any operation that is chosen is simply printed out on the console along with the correct session ID to prove the invocation of a method by the client.

Moreover, it is important to assume that the entire communication between both entities is hosted on a TLS channel and this includes the initial exchange of transporting the hashed password to the database. However, as mentioned previously in the third section of the report, the protocol itself is not implemented. In addition, the mechanisms and processes to prove the authenticity of the server and making sure to maintain the availability of the services are also not covered by the software.

Conclusion

This report commenced with addressing the process of authentication in a distributed system comprised of remote entities communicating with each other. The key idea behind this concept is to try to make sure that an entity is only able to request and access some data and or service if authorised.

Further emphasis was laid on one of the main and simple, yet efficient, facet of authentication which is based on using password and what challenges might be faced. Finally a solution was proposed and implemented to circumvent some of these potential issues by incorporating the core features.

Essentially, a simple client and server application was developed using the Java RMI API and a database management system was used to store the password securely as a cryptographically strong one-way hash. The verification process required the server to compute a hash from the password provided by the client and then compare it with the corresponding ID of the latter. The server also kept track of time of a given communication session and allocated a unique token with a certain expiration time and binding it to the requesting client.

However, apart from using a TLS channel for communication, it is important to assume that the communicating entities are in fact honest and not malicious.

Further improvements in the system may comprise of a sign-up feature that forces a client to use a cryptographically strong password. On top of that, enhancing the security may also include a multi-factor authentication scheme which would require a client to take more steps in order to verify its identity. Finally, the system may also incorporate the use of single sign-on (SSO) authentication method that would employ the use of an identity provider authenticating the client to the print server (One Login, 2023).

References

1. William Stallings. NETWORK SECURITY ESSENTIALS: APPLICATIONS AND STANDARDS FOURTH EDITION 2011

2. DOI: 10.1109/HICSS.2004.1265412

3. Alexander S. Gillis, Rainbow Table (2022): https://www.techtarget.com/whatis/definition/rainbow-table#:~:text=A rainbow table is a crack passwords in a database. [Last accessed: 25/10/2023]

4. Ben Lutkevich (2022). https://www.techtarget.com/searchsecurity/definition/access-control#:~:text=Access control is a security,access control: physical and logical.

5. Lucidchart (2023). *What is a Data Flow Diagram*: https://www.lucidchart.com/pages/data-flow-diagram [Last accessed: 30/10/2023]

6. Apache (2022). *XAMPP Apache + MariaDB + PHP + PERL | What is XAMPP?* Apache Friends: https://www.apachefriends.org/index.html [Last accessed: 28/10/2023]

7. One Login (2023) *How Does Single Sign-On Work?*: https://www.onelogin.com/learn/how-single-sign-on-works [Last accessed: 31/10/2023]