Security of MongoDB in Healthcare

Joshua Randolph

ABSTRACT

MongoDB have many potential benefits for the healthcare; however the default version of MongoDB is also very insecure because that the database authorization is disabled. Its vulnerability led to data breaches. Thus, the main goal of the research is to create more secure MongoDB system.

In one of the authentication methods in free version of MongoDB, x.509 certificate is the most secure. However, the research used the password authentication instead of using x.509 certificate. MongoDB is installed in Docker Toolbox. As for the authorization, MongoDB only have a role-based access control. The role-access based model is enabled in the MongoDB. The administration created two isolated databases - one for the doctors and another for the clinical researchers. Because MongoDB does not have password rules, the user creating system is created to assign the roles more easier and help making more secure passwords.

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference format:
Joshua Randolph. 2019. Security of MongoDB in Healthcare. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 7 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Motivation

The Healthcare use many types of databases to store the patient information. One of the types of the databases used in the Healthcare is MongoDB because of many of its benefits. The benefits from MongoDB draws that fact that MongoDB is a document-stored database. MongoDB is flexible compared to Structured Query Language (SQL). Because MongoDB is a document-stored database, the database is suited for semi-structured or unstructured data. MongoDB is suited for the visits of the patients because the visits includes different activities thus forming semi-structured data. Also, the MongoDB would be good use for the identification of the patient. The doctor need a patient identification to make a correct diagnosis on the patient [3] . Any failure in patient identification would result in medical errors. When using the queries for searches, MongoDB make selection by row, or document, unlike relational database that uses columns. One disadvantage is that MongoDB is not a relational database. However, MongoDB can use left join using an aggregation function called the lookup.

Despite its benefits, one of major drawbacks, however, is that the default version of MongoDB is insecure. In the default version, there is no authorization. In other words, anyone can have unlimited access when the default version is used. A lot of the information from the patient is so private the only their doctor know. Victor Gever, who is an ethical hacker, found that there is large number of MongoDB databases that are exposed and can be easily attacked by hackers. Thousand of healthcare firms have their MongoDB database hacked by cybercriminals. However, this does not means that no one should use MongoDB in healthcare, but rather that MongoDB needs to be more secure.

In 1996, the Health Insurance Portability and Accountability Act was passed. One of two major objectives of HIPAA was to provide coverage to workers who lose their job. Another objective of the act is to decrease the administrative cost by standardizing the electronic healthcare transaction [2] . Under the Privacy Rule of HIPAA, all identifiable information are protected [2] . This identifiable information is known as Private Health Information (PHI). PHI includes name, address, date of birth, and social security. Under the Privacy Rule, one possible consequence for a healthcare data breach is a fine. Under the Security Rule, there are standards for securing the patient data that are stored electronically [2] .

1.2 Goal

The design is to make MongoDB compliant with HIPAA, thus making MongoDB secure. At least two of the security issues will addressed - authorization and authentication. Authentication help identify the subjects. Authorization controls how much access that the subject to the object. The research will explore type of authentication mechanism that can be used in MongoDB. The research will design the access control for two data users - doctor and clinical researcher.

1.3 Minor Goals

1.3.1 Second Phase and Third Phase. The first step is to attempt to find the right ports and the right IP Address. The next goal to use the firewall which will be put on MongoDB to harden the security in the connections. The firewall will be set rules that permit user to go to only two ports. The next goal of the is to find a way to enable the authorization and strengthen the options for authentication. The attempt is to enable the role-based access control. Another goal is to create a user creation system that would assist on strengthen the authorization and authentication.

The third phase would create documentations of the installation and usage. So far, MongoDB Shell, MongoDB Compass Community, and Mongo Altas is installed and at least two of these tools would be expressed. The MongoDB Shell is an interactive shell that permits the user to create, modify, and search for the data. MongoDB Compass Community is a graphical user interface that permits the user to analyze the data without knowing the query

Conference'17, July 2017, Washington, DC, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

language. MongoDB Altas is a cloud database management system for MongoDB. MongoDB Altas permit the user to create clusters. Docker is also installed as well and might come to use as well.

In final phase, we hope to create a more secure system for MongoDB. There is still room for making MongoDB even if the system becomes more secure.

2 DESIGN CONSIDERATIONS

2.1 Port and Host

The ports for each of the database will not be at 27017 and the host might be the changed as well. The default port for MongoDB is 27017. To secure the access from the hacker, the port for each of the databases will not be at the default port.The default host is the localhost, which is 127.0.0.1. To secure access from the hacker, the localhost will not be used for the medical database. The research will find a possible port and host for the medical database.

MongoDB can used in the multiple hosts and ports. One of issue is that the hacker could take the data from port and hide another data in another port. Therefore, a firewall might be good to harden the security.

2.2 Access Model

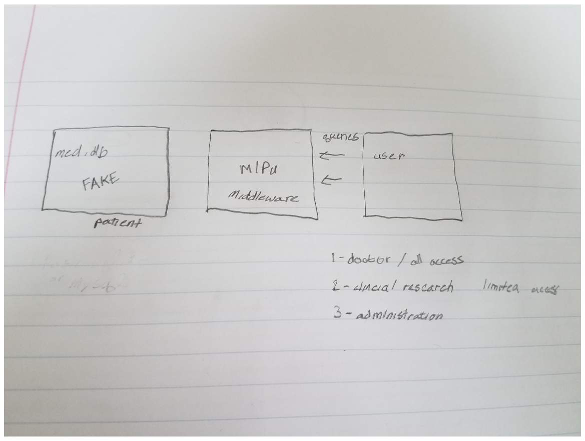


Figure 1: the rough draft of the access model shows the subject(users of the database), the middleware, and the fake representation of the data.

There are at least two type of access controls. The first one is a role-based access control and another is a attribute-access control. In role-based access control, the access are based on the role. In attribute-based access control, the access are based on the attributes based on the subject, environment, and object. Such attributes include the identity. Role-based access control are less secure than attribute-based access control; however, the trade-off is that attribute-based access control makes the access more convoluted thus making it less quicker than role-based access control. The trade-off of the attribute-based access control can have a negative effects on the patient.

In MongoDB, there is an existence of a role-based access control; however the role-based access control is disabled. The research will make effect on enabling the access control.

2.3 Roles

There are roles for the users of the data. In figure 1 shown, the doctor have full access to the patient database; however, the duty of the doctor is not to create the database. The data will have full access in both reading and writing. The clinical researcher have limited access to the data obtains from the patient database. The clinical researcher will not have any writing role. The clinical researcher does have reading goal; however, the clinical researcher cannot have full access to reading all of the health information, because many of the health information is private.

The design is to make two databases - one for the doctor and another for the clinical researcher. The database used by the doctor includes the raw data. This database is used for both reading and writing. The database used by the clinical researcher includes the clean data. The clean data excludes the PHI.

The main administrator may have full access to the database. The main administrator creates both user and assign role to the user. The main administrator reads the raw data and write the clean data.

Because there are two separate databases, the design for the administrator is to create clean the data using a program that is created by the administrator. This address another issue - the code need to be secure.

2.4 Authentication

Once the middle-ware permits authorization, the user will be required an authentication to get access to the database. The research will explore possible mechanisms. One possible mechanism is password. Another possible mechanism is password-less. Password-less mechanisms are more secure than password ones. One of the problems is that it is common for the password to be the same on multiple accounts. Once a password is find out, it is likely that the hacker find out the password on more than one account. However, there are the authentication methods that are harder to be establish.

3 ARCHITECTURE

Use this section to describe the overall architecture of your database engine, and implementation of your project.

3.1 Windows Firewall

The Windows Firewall permits the administrator to control which host can have access to the system [5]. By default, the Windows Firewall allows all outbound connections and block incoming connections. There are rules that the Firewall can be used to allow incoming traffic on specific ports. This does not mean that hacker cannot permits the Thus, the Firewall will be used on the MongoDB.

3.2 MongoDB Mechanisms

3.2.1 Role-Based Access Model. In MongoDB, there is role-access model; however in the default version, the role-based access control is not enabled.

There are two data base user roles in MongoDB - "read" and "readWrite" [5] . The "read" role give the data base user the privilege to read the database. The "readWrite" role give the data base user the privilege to both read and write the data.

The most important of the administration roles are the "userAdmin" role [5]. The "userAdmin" role provides the privilege to create and change users and roles.

3.2.2 Collection-Level Access Control. The principal objective of the collection-level access control is to give the administrator the power to grant privileges restricted to specific collections. The query to grant these privileges is createRole. The privilege is consist of resources, which includes the database and the collection, and actions. The actions are not same as the roles; however, the actions are part of the roles. An example of an action is find. The find method is designed to retrieve one or more documents.

3.2.3 Authentication. There supposed to be two cryptography methods that are used in the authentication in MongoDB 4.0 community version - SCRAM and x.509 [5] . In the Enterprise version, there are other cryptography methods - Kerberos and LDAP. By default, the cryptography methods used for the authentication is SCRAM. Because, the Enterprise is not free, authentication methods to use is either SCRAM or x.509.

SCRAM methods are a password authentication. Using SCRAM, the user is identified based on an authentication database. There are two SCRAM methods - SCRAM-SHA-1 and SCRAM-SHA-256. SCRAM-SHA-1 uses SHA-1, which is a one-way hashing function. Recently, in 2017, a team from Google found the first SHA-1 collision [7]. In a collision, there exist hash function of two messages that are equal to each other. SCRAM-SHA-256 uses the SHA-256 hashing function. So far, no collision is found in the hashing function SHA-256. Therefore, out of the SCRAM authentication methods that is used in MongoDB, SCRAM-SHA-256 is the most secure.

If the "userAdmin" is assigning a password authentication to the user, then writes out the username, the password, and authentication.

Unlike SCRAM, x.509 is not a password authentication. Instead of using user name and password, the x.509 authentication used certificates of authority to verify the user. One of the requirements is to have a secure connection with Transport Layer Security (TLS) or Secure Socket Layer (SSL). Transport Layer Security is a protocol that give both privacy and uncorrupted data between two application [4].

Because password-less authentications have stronger security than password authentications, x.509 authentication is more secure than SCRAM. Therefore, it is wisdom to use x.509 over SCRAM. Although in the MongoDB manual, there supposed to be the existence of the x.509 in the Community Version, in many of the tools that it seem that x.509 authentication is not enabled. Thus, it is decided that the password authentication to be used for this research.

3.3 MongoDB Tools

3.3.1 MongoDB Shell. MongoDB Shell does have SCRAM-SHA-256 as the password authentication. By the default, MongoDB Shell have password authentication. In the MongoDB Shell, the password is visible. The visibility increase a risk from the insider to use the password to get access to another account.

The user can be created via shell. In the shell, in order for the root administrator to create the user, the administrator goes to the administration database, by the command "using admin", and then authenticated by using the command "db.auth(u,p)" such as u is the user name and p is the password. To create an user, the createUser command is used. This raise at least two problems. One of them is the visiblity of the password. The administrator must not know the password of the user and the user must not know the password. The visibility increase a risk from the insider to use the password to get access to another account. Regardless, the root administrator will know the username of the users by using the getUser query. Also, there is no rules concerning in creating a password. The lack of rules would put a risk for the user to create a weak password. For an example, an user could create a short password.

3.3.2 MongoDB Compass. MongoDB Compass Community is a graphical user interface that permits the user to analyze the data without the usage of MongoDB queries; however, the MongoDB Compass permits the user to use the language JSON to create the document. This graphical user interface is an analytical data management system although some of the analytics are limited.

Like the MongoDB Shell, the default version of the MongoDB Compass does not have authorization; however, there are steps to set up a plugin that permits the administrator to set up the user in the interface.

MongoDB Compass Community does have SCRAM-SHA-256 as password authentication. By the default, MongoDB has password authentication. Unlike the MongoDB Shell, the password is invisible with the exception of the length of the password.

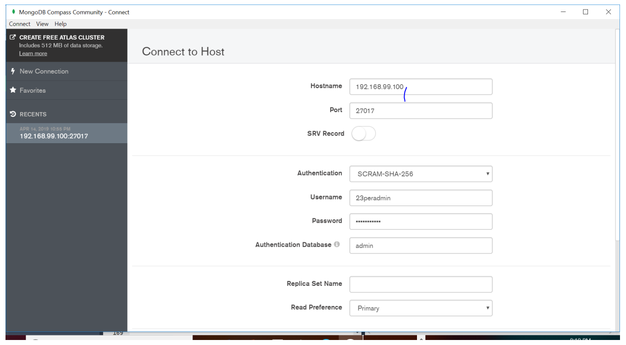
However, unlike the MongoDB Shell, one cannot create a root administrator in the MongoDB Compass Community.

One of the major security flaws of MongoDB Compass Community is that the graphical interface automatically saves the password and the user name as the user logs in. If the user picks any of the logins It sounds good for the user because the user does not have to memorize the user name and password. However, if the user is sharing the same server with another person, then there is a chance is that another user can use login with the password and username. The user can clear all the recent login, erasing the usernames and passwords. To clear the login, the user moves the point next to the login and, as a trash can appear, click on that button. To clear all of the passwords and username, the user moves the pointer next to "recent" and then as "clear all" appears, the user then clicks on the button.

3.4 Docker

Docker is a platform that is design to run applications easier by using containers [1]. Containers are an unit of software that package the code. Containers are lightweight and secure.

One of the tools for Docker is the Docker Toolbox. Docker Toolbox can be run on the Window 10. One of the desktop apps for the Docker Tool is the Docker Quickstart Terminal, which is terminal that uses container technology.



Hostname: 192.168.99.100
Port: 27017
SRV Record
Authentication: SCRAM-SHA-256
Username: 23peradmin
Password: ••••••••••••
Authentication Database: admin
Replica Set Name:
Read Preference: Primary

Figure 2: On the left side, an user can click for the login below the "recent" title.

3.5 Python Language

Python have two important modules that can be used for the authentication of MongoDB - PyMongo and getpass.

3.5.1 PyMongo. The research explores different programming language that can be connected to MongoDB. Out of the languages, there are at least two languages that can easily be connected to MongoDB - Python and node.js. Among these languages, Python is the easiest to comprehend. The module that was added to Python is PyMongo. The MongoClient object permits the client to connect with MongoDB. The parameters in MongoClient includes IP Address, port, authentication mechanism.

The authentication mechanisms that can be for PyMongo are SCRAM-SHA-256 and X.509. The Enterprise version supports Kerberos and LDAP so PyMongo can be used; however for Kerberos, PyMOngo requires another module. Also, the Enterprise version is not free.

3.5.2 getpass. . One of problem is that password is visible in the MongoDB Shell. The method getpass in the getpass module asked the user for the password without making the input visible. The getpass cannot be run on a console. Instead, the getpass can be used in a terminal. One of the drawbacks of getpass is that if the password is printed, then the password is visible. The getpass is useful for making the username invisible.

4 IMPLEMENTATION

Use this section to describe the overall implementation of your project.

4.1 IP Address and Port

Instead of the localhost as the selected IP Address, the IP Address 192.168.99.100 is used. 192.168.99.100 is the IP address of the Docker Quickstart Terminal. Using the MongoDB Community Compass, the

There have have been attempts to try to use different ports for MongoDB, but somehow these attempts did not work. One of the challenges is restarting the MongoDB server on Windows.

4.2 Firewall

The local address used for the firewall is 192.168.99.100. The connection is permitted if the there is data-integrity and authentication in the connection.

4.3 Role-Access Model

4.3.1 Windows Attempt. Before using docker, there was attempt to create the role-access based model via the Window Command Prompt. One of the most important steps is to create a root administrator using these chain of commands and queries. In MongoDB Server, there is set of warnings and one of the warnings is: "WARNING: Access control is not enabled for the database". Before using the query to create the root administrator, there is a query that permits the user to go to the "admin" database. the query is "use admin". To create the root administrator, use the query "db.createUser(user:"<USERNAME>", pwd:"<PASSWORD>",roles: [role: "root", db:"admin"])". "user" is the username, "pwd" is the password, "roles" is the roles of the user, "role" is the specific role, and "db" is the database. The biggest problem of using Windows is restarting the MongoDB server. If the MongoDB server restarts, then the authorization is activated. There were commands for restarting the MongoDB server in Ubuntu; however, there is an appearance of no commands to restart the server. One can follow the MongoDB manual but even following the steps in the manual somehow did not work. There was an attempt to modify the configuration file.

4.3.2 Docker Attempt. In order to create a role-access based model in MongoDB in Docker, there are chain of commands. One significant command was: "docker run –name <name> –restart=always -d -p 27017:27017 mongo mongod –auth" such as <name> is the name of the container. "–auth" is the authorization. "–restart=always" means that the server will always restart as the user exit. In order to go to the MongoDB, one go to the container using "docker exec -i -t <name> bash". Once the person is in the container, to enter the MongoDB server, use "mongo". Like using the Windows, there is a warning say "WARNING: Access control is not enabled for the database". Use the query "use admin" to go to the "admin" database. The root administrator is created using the query "db.createUser(user:"<USERNAME>", pwd:"<PASSWORD>",roles: [role: "root", db:"admin"])". Then exit the MongoDB server using "exit".

Thus, the chain of commands and queries in Docker are:

1. docker run –name <NAME> –restart=always -d -p 27017:27017 mongo mongod –auth
2. docker exec -i -t <NAME> bash
3. mongo
4. use admin
5. db.createUser(user: "<USERNAME>", pwd: "<PASSWORD>", roles: [role: "root", db:"admin"])
6. exit

4.4 Databases

After the role-based access control is enabled, the administrator log-in to Mongo Community Compass via password authentication. The requirements to log-in are the IP Address, port, user name, and password. Once the administrator is log-in, the administrator then clicked on the "create database" button. One of the database is the "patient" database" while the second database is the "clean" database.

4.4.1 Patient Database. The patient database contains three collections - private, data, and visit. In the private collection, much of the variables are private health information (PHI) which helps identify the patient. The data is supposed to not have any private information. One of the factors in the data includes the patient's sex. The visit records the activity of the patient's visit.

4.4.2 Clean Database. The clean database is the database that is designed for the clinical researchers. So far, the database contains the data collection. Unlike the patient database, the database is designed not to contain any private health information.

4.5 User Creation System

The system is designed as a contract between two people - the administrator and the user. The administrator. The administrator first input the user name and password. Then the administrator assigned the role to the user. Then user then creates an username and password.

In the user creation system, there are two files - the main file and the password files.

4.5.1 Main File. In order for the system to work properly, the user creation systems must be run on the terminal. The user creation system used getpass to make the input invisible especially the length. The getpass module cannot be run properly on a console; however, the getpass can be used in a terminal.

The administrator types in the username and the password. Both the username and the password are invisible, especially in length. The getpass is used on the administrator's username and password.

The administrator is given a menu. The following choices are 1. doctor 2. researcher and 3. data cleaner. The program prompts the administrator for the role. The choice is visible so that the user can see the choice.

The users then type in the username and the password. Both the username is visible because the administrator can know the username by using the getUser query. However, the password is invisible, especially in length. The getpass is used on the user's password. Then the user then must confirm the password. If the password follows with rules then the user is confirmed.

While the user is confirmed, the user is assigned to the role that is selected by the main administrator. If the role of the user is a doctor, then the user is given the "readWrite" role on the "patient" database. The doctor is given a rule that he has both reading and writing access to the "patient" database. If the role of the user is a clinical researcher then the user is assigned to be the "read" role. The clinical researcher is informed of the role only permits this user to have reading access to the "clean" database. If the role of the user is a data cleaner, then the user is given the "readNonPHI" role on the "patient" database and "readWrite" database. However, the data cleaner is given a warning not to write private health information on the "clean" database. The data cleaner is also informed that if the main administrator sees any private health information then the data cleaner will be investigated.

Table 1: Occupation and Roles

Occupation	Database	Role
Doctor	Patient	ReadWrite
Cleaner	Patient	ReadNonPHI
Cleaner	Clean	ReadWrite
Researcher	Clean	Read

4.5.2 Password File. There is a second file we called "passwordRules.py". In the second file, there is a class for the username and the password. In the class, there are functions that analyze the password to see if the password is assigned to the rules. Some of the functions are based on traditional password rules. The traditional rules are:

1. The password must be long. (In this case, the password must have a length of 12 or greater.)
2. The password must have a lowercase letter.
3. The password must have an uppercase letter.
4. The password must have a digit.

Other rules are:

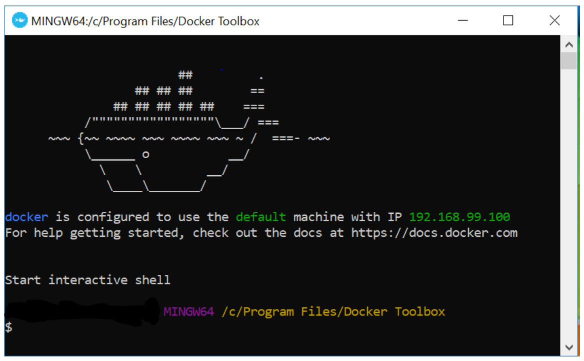
5. The user name must not be equal to the password.
6. Password must have "password".

The result uses these five rules. The first, and most significant, rule of creating a password in this user creating system is that the password must have a length of 12 or more. If the password is too short, then the hacker can find out the password easily. The traditional rules behave like a whitelist while the nontraditional rule behaves like a blacklist.

5 ANALYSIS

5.1 IP Address

If the user attempted to login into the database and fails, this indicates the computer shutdown. To get access to the IP Address of the database that was established, the user have to click on the Docker Quickstart Terminal first. The user then have to wait until Docker is configured to use the default machine with the IP address 192.168.99.100.



```
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \______ o           __/
             \    \         __/
              \____\_______/

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com


Start interactive shell

[redacted] MINGW64 /c/Program Files/Docker Toolbox
$
```

Figure 3: Docker Quickstart Terminal

5.2 Role-Based Access Model

After exiting the MongoDB server on the Docker Terminal, the database is checked for a role access based model. There is no warning that tells that role-based access control is not enabled. A root administrator cannot read or write unless using the password authentication. One way of using the password authentication is to use two queries, "use admin" and "db.auth(<USERNAME>, <PASSWORD>)". There was a test on creating a fake user whose role is "read". The fake user cannot write; however, the fake user can read the data. Therefore, the role-based access control is enabled.

5.2.1 Role Creation. Addition custom roles were established for the Patient database - "readDataOnly", "readVisitOnly", and "readNonPHI". "readDataOnly" is a role designed to only read the data collection from the Patient database. Likewise, the "readVisitOnly" is a role designed to only read the visits collection only. "readNonPHI" is a combination of "readDataOnly" and "readVisitOnly" roles.

5.3 User Creation System

5.3.1 Authentication. There is a testing function in the Password File to see if a username and password aligned to the rules. The testing function prompts the user for the user-name and password. The functions of the class appear to works. Thus, the password authentication rule seems to work.

One of the tests is to use a fake user that is assigned as a doctor. Using the fake user's username and password on the authentication in the MongoDB Compass Community, it is found that the password authentication works for this fake user.

5.3.2 Authorization. One of the tests is to use a fake user that is assigned as a doctor. The fake doctor is assigned the "readWrite" role in the patient database. After logging into the MongoDB Compass Community, the fake doctor is tested on both reading and writing access to the patient database. The database was visible so the fake doctor does have the reading access. When inserting the document, the access is not denied and thus we can conclude that the fake doctor has writing access to the patient database.

Another test was to use a fake user that is assigned as a cleaner. The fake cleaner is assigned with the "readNonPHI" role in the patient database and the "readWrite" role in the clean database. When logging into the patient via MongoDB Compass Community, the fake cleaner can read both the data collection and the visit collection. There is no access to the private collection.

6 LESSONS LEARNED

6.1 Response to Investigating Teams

Discuss how you addressed the issues, if any, that were identified by the investigating teams. Be specific.

6.2 Other Lessons

Use this section to describe mistakes you made and corrected (or did not get a chance to correct including why you didn't). Also describe what all you learned during the course of this effort; this section, like the others, plays a critical component in determining your final grade.

6.2.1 Role of the Data Cleaner. In the third phase, one of the previous mistakes was not to make the greater restriction on the data cleaner and fully trust the data cleaner not to use any of the private health information. Before the submission of the third phase, the user would give access to the readWrite role for all of the collections in the patient database. After the submission, the decision was to create role that restricted access to the non-private information and then change assigned role from "readWrite" to "readNonPHI". This applied to the data cleaner.

6.2.2 USB Access Control. Another mistake is forgetting about the USB Drive Access Control in the design considerations. There was an attempt to using the USB Drive Access Control. The USB Drive Access Control does requires ASP Net framework. But somehow this attempt fails.

7 ETHICAL AND LEGAL ISSUES

Use this section to discuss ethical and legal issues relevant to your project.

7.1 HIPAA

The design of the application was to be compliant with the Health Insurance Portability and Accountability Act (HIPAA). In 1996, United States passed HIPAA, which was an act that regulates the privacy and security. Under the privacy rule, Private Health Inform (PHI) is protected. Under the security rule, there are standards for securing the data.

One of the biggest issue is that healthcare have failed to make the data in MongoDB more secure. In the default version of MongoDB, the authorization is disabled.

8 CURRENT STATUS & FUTURE WORK

Use this section to describe the current status of your work and what else needs to be done. Also, discuss what further directions your work can be taken by others.

8.1 USB Drive Access Control

One of the problems facing this application is the data not to be stolen from the outside. By using a USB Drive Access Control, access control could use a blacklist on all USB. One of the problems of keeping a USB is the risk of losing sensitive data especially private health information. Any loss of a USB contain private information could cause a breach.

8.2 Age Range Generator

Next goal to create the tool for the root administrator to generate or update the age interval in one of the non private health information collections. Age does have factor on the patient's health. One of the problems of putting an exact age in the clean database is that one can easily identify the person with the age of the person although age is not as easy for identification as the birth date. During an internship in the University of Rochester Clinical Research Building, I was working on creating a Java program that computes statistics from the i2b2 database. Three of the areas of interest was age group, race, and sex. In the i2b2 database, one of the columns was on the person's age. The age interval is less likely to help identify the patient than the exact age of the patient. The tool would requires the authentication from the MongoDB to transport that data into the nonprivate health information.

8.3 Data Cleaning Program

The task of the data cleaner, who is one of administrators, is to create a program that clean out the data. The data cleaner does not have any access to the Private collection. It does not always mean that the Private Collection would not have any private health information. There is a chance that a doctor would put private information in the Thus, the data cleaner must remove any form of private health information while cleaning out the data and transports the data to the clean database. However, the data cleaner must also be aware of the differential privacy. Differential privacy is a definition given by Dr. Dwork [6]. In one research, one can name the person through zip code, sex, and date of birth. Of course, the date of birth would be considered private health information. In 2007, Netflix released a data set of the user rating. The researchers were able to breach privacy. 99 percent of personal data was removed from the dataset. The data cleaner could use noise on the dataset in way that the statistical outcome would be about the same. However, the challenge for differential privacy is there could be a trade-off between accuracy and privacy.

9 CONCLUSION

The default version of MongoDB. It is understandable why people in the healthcare using MongoDB while authorization being disabled. It is difficult for enable a role-access model on a Windows operating system, so Docker is a good approach in securing MongoDB. Even with enabling the role-access control and using password authentication, there was still more security concerns, such as lack of rules on the passwords and password visibility. The user creation system was established to make a safe approach to MongoDB.

REFERENCES

[1] [n. d.]. Docker. ([n. d.]). https://www.docker.com/resources/what-container.
[2] [n. d.]. HIPAA (Health Insurance Portability and Accountability Act). ([n. d.]). https://searchhealthit.techtarget.com/definition/HIPAA.
[3] [n. d.]. Identify Your Patients - Infection Prevention and You). ([n. d.]). http://professionals.site.apic.org/patient-identification/.
[4] [n. d.]. Transport Layer Security (TLS). ([n. d.]). https://searchsecurity.techtarget.com/definition/Transport-Layer-Security-TLS.
[5] MongoDB. [n. d.]. The MongoDB 4.0 Manual. ([n. d.]). https://docs.mongodb.com/manual/.
[6] White House Office of Science, Technology Policy, and MIT. 2014. . (March 2014). tp://web.mit.edu/bigdata-priv/agenda.html,.
[7] Pedro Umbelino. 2017. SHAttered - SHA-1 is broken. (February 2017). https://hackaday.com/2017/02/23/shattered-sha-1-is-broken/.