Array 15 (2022) 100236

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

Mechanisms and techniques to enhance the security of big data analytic framework with MongoDB and Linux Containers

Akalanka Mailewa [a,*], Susan Mengel [b], Lisa Gittner [c], Hafiz Khan [c]

[a] St. Cloud State University, Department of Computer Science and Information Technology, St. Cloud, Minnesota, USA
[b] Texas Tech University, Department of Computer Science, Lubbock, TX, USA
[c] Texas Tech University, Department of Public Health, Lubbock, TX, USA

ARTICLE INFO

Keywords:
Vulnerabilities
Threats
Big-data
MongoDB
Data security
Docker & Singularity
Linux Containers
Sensitive data
HIPAA
Privacy

ABSTRACT

The frequency and scale of unauthorized access cases and misuses of data access privileges are a growing concern of many organizations. The protection of confidential data, such as social security numbers, financial information, etc., of the customers and/or employees is among the key responsibilities of any organization, and damage to such sensitive data can easily pose a threat to the future of a business and the security of the customers. Therefore, this paper proposes and implements some security mechanisms and techniques, such as secure authentication, secure authorization, and encryption, to assure the overall security of a big data analytic framework with MongoDB free community edition. This paper presents the fourth phase of our continuous research where in the first phase we proposed a data analytic framework with MongoDB and Linux Containers (LXCs) with basic security requirements. Next, in the second phase we proposed a vulnerability analysis testbed to find vulnerabilities associated with the system. Finally, in the third phase we discussed in detail root causes and some prevention techniques of vulnerabilities found in the system. In addition, this paper introduces a new security mechanism for privacy preserving data handling with MongoDB to ensure the privacy of the data before being processed. Our results show, with our initial model of the analytic framework, how well our newly introduced security mechanisms work and how these security mechanisms and techniques can be used to assure the confidentiality, integrity, and availability (CIA) of any data science project conducted on our proposed analytic framework. In addition, these security mechanisms and techniques help us to strengthen the current system against zero-day attacks where attacks on vulnerabilities that have not been patched or made public yet. Therefore, our vulnerability analysis testbed which is proposed in the second phase of this research will not be able to finds vulnerabilities related to zero-day attacks.

1. Introduction

Modern data analytic frameworks often operate in highly dynamic and uncertain environments. Therefore, providing security for such frameworks is a huge challenge since in a networked environment, the risks to valuable and sensitive data are greater than ever before [1]. Understanding and leveraging the inter play of these security areas dimensions can help design more secure systems with more effective policies. This paper focuses on how to improve the security of the evolutionary development of a high-performance big data analytic framework by introducing some security mechanisms and techniques. The ultimate goal is to systematically design a secure data analytic framework that is capable of providing guaranteed security to its users such that it fulfills regulatory compliance mandates, such as Health Insurance Portability and Accountability Act of 1996 (HIPAA), if research focuses on healthcare data.

One of the specific cases that influenced this study is the data security requirements of the Texas Tech University (TTU) EXPOSOME big data project, funded by NIH and NSF. This project is designed to pinpoint the exposures and their impact on the health of affected people within their lifespan. However, data analysts have been struggling to gain insight from raw data because of the differences in data formats. Furthermore, data must be secured and privacy-assured despite the continuous addition of more data including those that are de-identified but still protected personal health records [2]. Currently, a transdisciplinary team is in the process of implementing a secure system to convert these data and

* Corresponding author.
E-mail address: amailewa@stcloudstate.edu (A. Mailewa).

https://doi.org/10.1016/j.array.2022.100236
Received 20 December 2021; Received in revised form 17 July 2022; Accepted 18 July 2022
Available online 4 August 2022

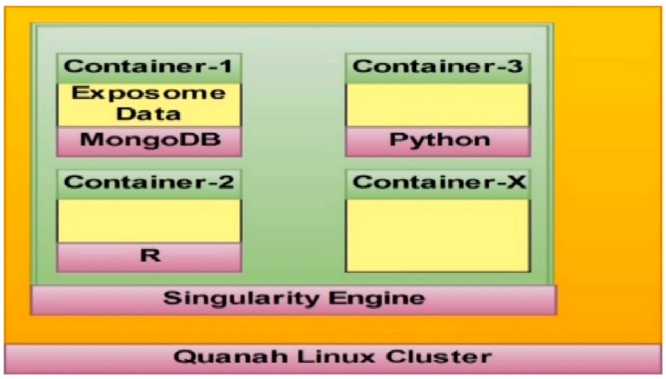


Fig. 1. Data analytic framework with MongoDB and Singularity.

store these various data formats in the MongoDB database for further analysis of the TTU EXPOSOME Project data. Four main areas are addressed in this project in order to protect the data and provide unimpaired access to the authenticated researchers. These four areas include database security (Server with LXCs), network security, web application security, and physical security. As mentioned earlier, the ultimate goal is to provide a secure analytic framework. This research, therefore, investigates ways in which protected data and access to the EXPOSOME data can be secured while avoiding performance degradation during access [3] even when insecure tools may be used.

Akalanka et al. [4] proposed a conceptual secure data analytic framework with MongoDB and LXCs in the first phase of the research. This framework, illustrated in Fig. 1, shows how the current MongoDB database server is implemented with EXPOSOME data on a Singularity Linux container. In the meantime, the remaining containers facilitate other services to the entire data analytic framework, where the host operating system is a Linux node (Quanah Linux Cluster) attached to the TTU network. In the second phase of the research, Akalanka et al. [5] presented a dynamic and portable vulnerability assessment system testbed made with Singularity Linux containers (Fig. 2). Equipped with seven testing tools, the testbed is designed to find vulnerabilities in the data analytic framework proposed in phase 1. The tools facilitate vulnerability testing in the container network, inside the host machine, and outside the network. The implementation of the testbed can be done in three steps: first, the MongoDB server and clients are deployed. Then, "OpenVAS," "Dagda," "PortSpider," "MongoAudit," "NMap," "Metasploit Framework," and "Nessus" vulnerability assessment tools as Docker images are added to the seven separate Singularity containers to complete the testbed with seven testing tools. Finally, in order to identify the root causes of vulnerabilities and remove the available vulnerabilities found in phase two, "Vulnerability Prioritization, Root Cause Analysis, and Mitigation" are performed. In the third phase of the research Akalanka et al. [6] discussed multiple different ways to prioritize vulnerabilities found in the second phase of the research, discovered root causes of those vulnerabilities, and proposed and implemented some mitigation techniques against those vulnerabilities found so far in the Secure Data Analytic Framework implemented with MongoDB on Singularity Linux Containers. In summary, the third phase of the research answered the question, "How do we mitigate vulnerabilities found in the current system by analyzing their root causes?" research question by involving three sub-steps: 1. Vulnerability prioritization, 2. Root causes analysis, and 3. Prevention techniques. This phase (fourth phase) of the research answers the research question, "RQ: How does the data analytic framework with MongoDB and LXCs is designed with enhanced security features?" by introducing some security mechanisms and techniques to the current system, such as authentication, authorization, privacy preserving data handling, and encryption.

Initially the paper discusses in brief the background with the related work. Next, it presents the methodology of security mechanism implementations one by one for authentication, authorization, privacy preserving data handling, and encryption. Thereafter, paper presents results related to authentication, authorization, privacy preserving data handling, and encryption in detail. Finally, the paper concludes with the power of the current system and some future suggestions to improve the current system security, such as block-chain based security [7,8] and how human behaviors will affect the system security.

2. Background and related work

We In this section, we introduce a security model for the stored data in the MongoDB database server within Singularity Linux containers. During transmission, security mechanisms are used to authenticate, authorize, and encrypt data while data-at-rest is used to prevent vulnerabilities. Furthermore, a mechanism is proposed and implemented for privacy preserving data extractions, transmissions, and loadings, by converting the actual data format to numerical data representation.

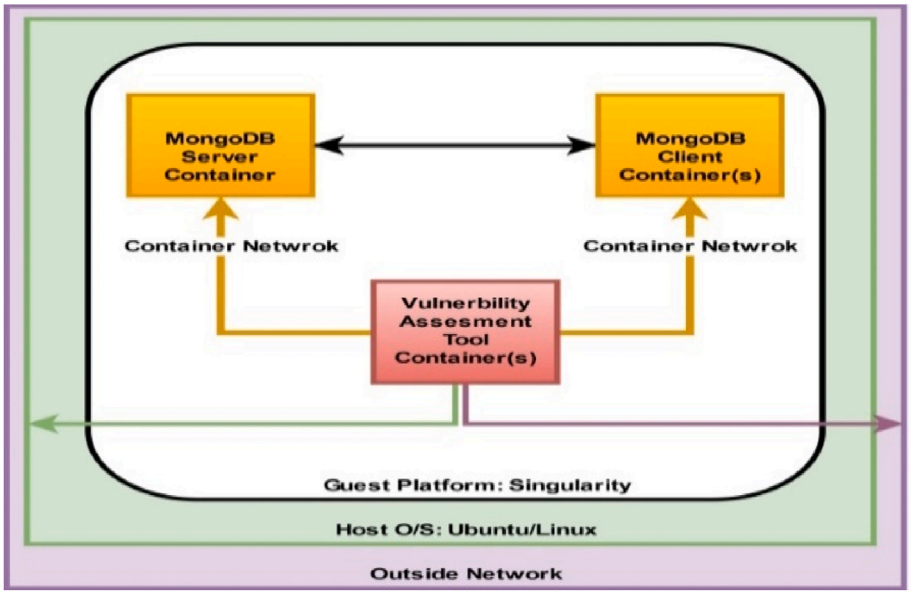


Fig. 2. LXCs based vulnerability assessment testbed.

Subsequently, we introduce several new techniques that we intend to implement to meet our future research goals as we create a more secure platform to deal with sensitive data and to comply with policies, such as HIPAA.

2.1. Authentication

Even though the MongoDB community edition supports multiple authentication mechanisms, such as SCRAM-SHA-1, MongoDB-CR, and x.509 Certificate authentication [9], this edition does not have an authentication mechanism by default. Therefore, we intend to use built-in technologies to implement a password-based authentication mechanism, until we propose a more secure, stronger mechanism using Mongoose [10–12] and bcrypt module with Node. js [13]. Mongoose is an elegant MongoDB object data modeling library for node. js, as well as an object data mapping solution that is capable of organizing and structuring data [14,15]. There is also the option to use the proxy authentication mechanisms, such as Kerberos and Lightweight Directory Access Protocol (LDAP) separately, which is a more complicated process [16–18]. Thus, we plan to encrypt passwords (or other data) through Mongoose before saving them in the MongoDB database server, and this method can also be used to encrypt data-at-rest.

2.2. Authorization

Since MongoDB does not enable access control by default, the authorization mechanism decides the database resources and operations accessible to the verified users [19,20]. After installation, nine control roles defined by the system are available with MongoDB. This research specifies the basic database policy, which guides the steps of access control implementation. The initial focus is on one regular user and the system administrator who has full permission. Complying with the principal of minimal authority in setting roles that determine the amount of access per user, the other users have lesser permission levels.

2.3. Privacy preserving data extractions, transmissions, and loadings

Our proposed privacy preserving data handling mechanism converts the data header and file name into a numerical data header and file name representation when loading data into MongoDB; thus, reducing the ability to determine what a particular column's values and/or file name represent or mean. For every document or file loaded into MongoDB, MongoDB creates a key-value pair for each row inserted, usually of the format 'Header: it's Value'. These headers are usually of the type 'String' of varying lengths. In this phase, the 'Header Formatter' assigns for every unique header a new integer value and this integer value acts as the identifier for that header in the system. Conflict resolution becomes essential at this stage since many cases occur in which the same header is used to identify different parameters. Keeping such headers as they are would only lead to ambiguity and confusion during data selection; multiple header use must be resolved. The system solves this multiple header use with header formatting. The header formatter assigns a unique integer ID to every unique header. If the header is not unique, the description associated with the header is checked. If the description of the recently read header matches the description of any of the similar non-unique headers, the same value that is assigned for the other matching header is assigned to the recently read header. If the description of the recently read header is different from the descriptions of all the other matching headers, then the new header is a conflicting header and a new integer ID is assigned to it. The system also maintains a separate metadata collection for all conflicting headers. During the Header mapping stage, the data processing stage defines and populates a collection to keep track of the headers and the files in which they are present while the data loading process is building the data dictionary. This metadata is particularly important during the retrieval stage [2, 21–24].

2.4. Encryption

Data encryption is essential in both transmission and storage [25, 26]. In this research, using available encrypting techniques and limiting network exposure have ensured data protection during transmission. All MongoDB network traffic is accessible only to the intended users as it is encrypted with Transport Layer Security and the Secure Sockets Layer (TLS/SSL) [27]. The TSL/SSL encryption mechanism is used to encrypt the communication between "mongod' and "mongos" and MongoDB with other applications, such as python and R. The server is carefully set to run on a trusted network environment with limited interfaces of MongoDB instances waiting or listening for incoming connections [28]. The Linux-based server in which the database is located, limits its access to authorized users by SSH secure connection through a VPN tunnel [29–31]. The data-at-rest at the storage layer can be used with any/all of the formats below:

- Encrypt the entire drive
- Encrypt individual files or databases on the disk
- Encrypt entire documents (rows in SQL-land) or individual attributes (columns in SQL-land) at the application level

The encryption is started at the application-level as per the research security requirements of this research. Even though it is conceptually unchallenging to encrypt data-at-rest given that MongoDB replaces plaintext data in a document with encrypted data, and its flexible schema, it can be complicated during the implementation. This is due to several reasons: application layer at this level is in full control and encryption is independent of the server and the network stack; the keys are in the application layer, separate from the data layer and the plaintext information is never stored or transmitted. Therefore, the data layer is unable to reveal the plaintext values for an attack. The common application vulnerabilities, such as cross-site scripting (XSS) or SQL injection [32–34] provide the attack vector for encryption at this level. Our system requirements are supported by application level encryption, which is one of the many aspects of solid security. Moreover, for the encryption and data storage in MongoDB, we use mongoose and crypto module with Node. js [35], similar to its use in authentication in encrypting passwords. Crypto module's AES-256-CBC algorithm with a random unique initialization vector for each operation is used for encryption.

2.5. Related works

Daniel Huluka et al. [36] identified eleven root causes of session management and seven causes of broken authentication vulnerabilities in their investigation on the possibility of using Root Cause Analysis (RCA) to improve web application security. Their investigation and review of vulnerabilities paved the way for developing ways to prevent recurrent attacks on web applications.

Priya Anand [37] emphasized the importance of taking the technical and social aspects into consideration to secure a software system. Focusing on six interconnected social and technical components – hardware, software, networks, data, people, and procedures – the author identifies the nature of information systems as much more than a technical matter. Furthermore, the paper stresses the need to consider different perspectives during the security protocol implementation and highlights several significant root causes of vulnerabilities of the investigated software system. The comprehensive analysis of the related literature provides a number of information system vulnerabilities and the strategies to reduce attack surfaces.

Zhiqiang Lin et al. [38] introduce AutoPag, a solution they have developed for the out-of-bound vulnerability that contains buffer overflows and general boundary condition errors. AutoPag is designed for time-efficient software patch generation, and it has the ability to analyze the program source code through data flow analysis automatically. This

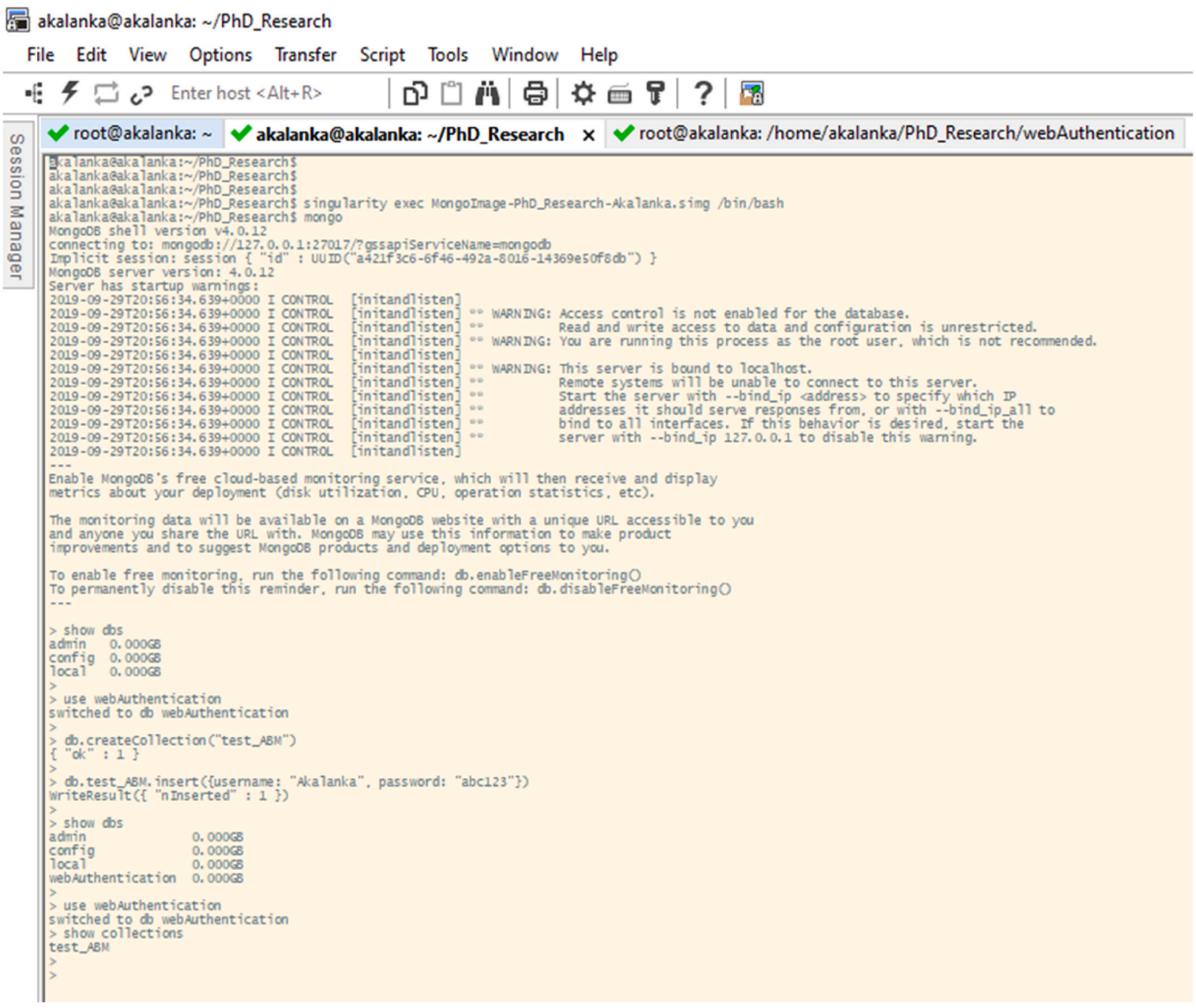


Fig. 3. MongoDB client is started and "webAuthentication" database is created.

process allows it to find the vulnerable source-level program statements of any given out-of-bound exploit. In every vulnerability test performed in the Wilander's buffer overflow benchmark test-suite, AutoPag is able to successfully create its own fine-grained source code patch as a temporary solution within seconds of finding the root-cause. Moreover, the developers have built a proof-of-concept system in Linux with promising preliminary results, and the evaluations with real-world out-of-bound exploits have established AutoPag's practicality and effectiveness.

According to Fernandez [39], patches are not the best solution to fix the security vulnerabilities in software. He points out how local fixes are not architectural solutions and how a patch that is not implemented in an architectural mode can create reconfigured or redundant patterns in the system. Thus, highlighting the possibility of patches opening doors for new attacks, Fernandez proposes two solutions to improve the security of software design: the first is to examine the final production code and search for possible problems; the second is to plan for security from the beginning of the software development life cycle. Even though there are some issues in the approach, Fernandez supports the method of implementing security principles from the beginning of the life cycle to effectively attend to system security.

Youssef [40] proposes a secure healthcare information system that eases the access of sensitive patient information for all relevant parties across platforms and geographical locations. The system gathers all scattered information of each patient and creates an Electronic Health Record in cloud storage. In order to facilitate efficient decision-making, the system sorts out and provides only the necessary, relevant data for the involved parties, instead of letting all parties access all the information in the cloud or in clinical information groups. The proposed system includes a data security model that uses encryption algorithms (AES, RCA), authentication techniques (OTP/One Time Password), Two Factor Authentication (2FA), and clinician authorization to prevent data theft or unauthorized data usage.

Chakravorty, Wlodarczyk, and Rong [41] have developed a data analytics framework aimed at maintaining security and preserving the privacy for analysis of sensor data in smart homes. The heuristic-based, k-anonymization algorithms used in the framework provide security and protect the privacy of the data throughout the data lifecycle. In order to prevent the access of certain users while allowing some other users to access the data, the framework proposes to re-identify data processing results through a separately encrypted identifier dictionary with hashed values and actual values of all unique sets of identifiers.

Abraham and Nair [42] have created a model that exploits a set of

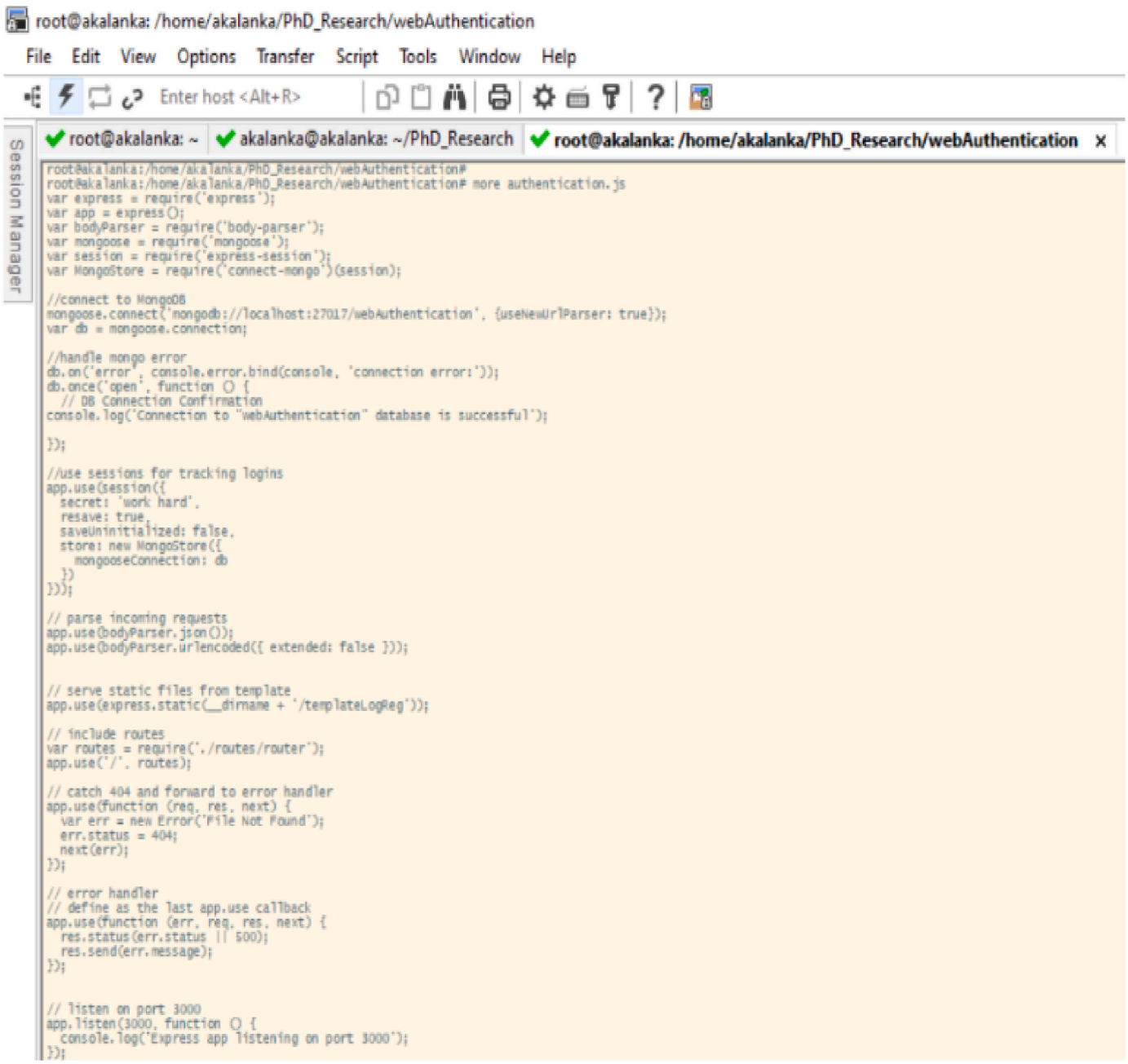


Fig. 4. "Authentication.js" Java-Script file to set up initial configurations.

```
root@akalanka:/home/akalanka/PhD_Research/webAuthentication#
root@akalanka:/home/akalanka/PhD_Research/webAuthentication#
root@akalanka:/home/akalanka/PhD_Research/webAuthentication# node authentication.js
Express app listening on port 3000
```

Fig. 5. Application is stared and listening on port 3000 for web connection.

complementary metrics to provide an accurate evaluation in the process of determining the level of security of a network that oversees the security system status during current and potential future threats. The project considers security as a multidimensional aspect and uses a stochastic modeling technique with attack graphs. Meanwhile, the framework ensures a better understanding, along with a thorough system analysis through the use of Absorbing Markov Chains and the Common Vulnerability Scoring System (CVSS) framework.

Previous work focuses on specific aspects of security and privacy problems in a particular domain and how to resolve them. This research investigates a general approach to prevent most of the attacks due to 1. Week authentication, 2. No authorization policy or mechanism, 3. No encryption mechanism, and 4. No mechanism to foster data privacy. Moreover, this research proposes a generic security model for any container-based virtual application that helps satisfy its security requirements and protects the application against various malicious behaviors.

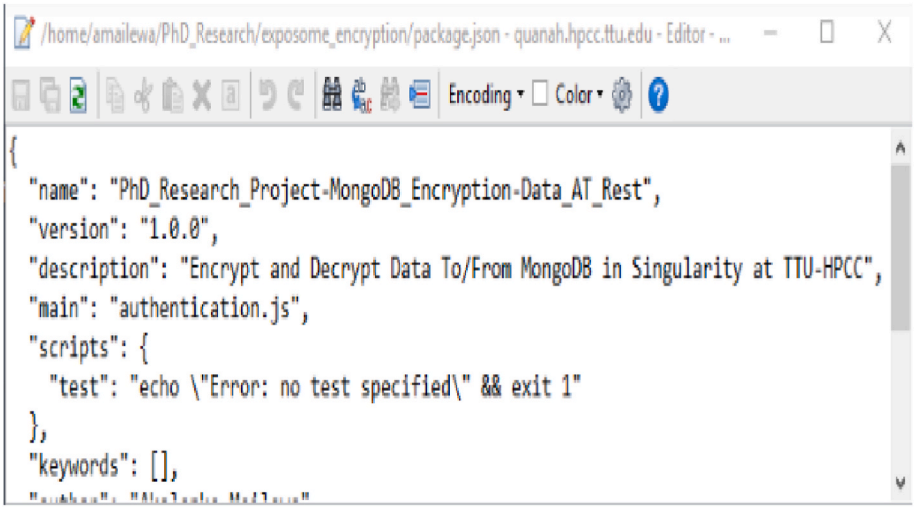


Fig. 6. Initial "package.json" file.

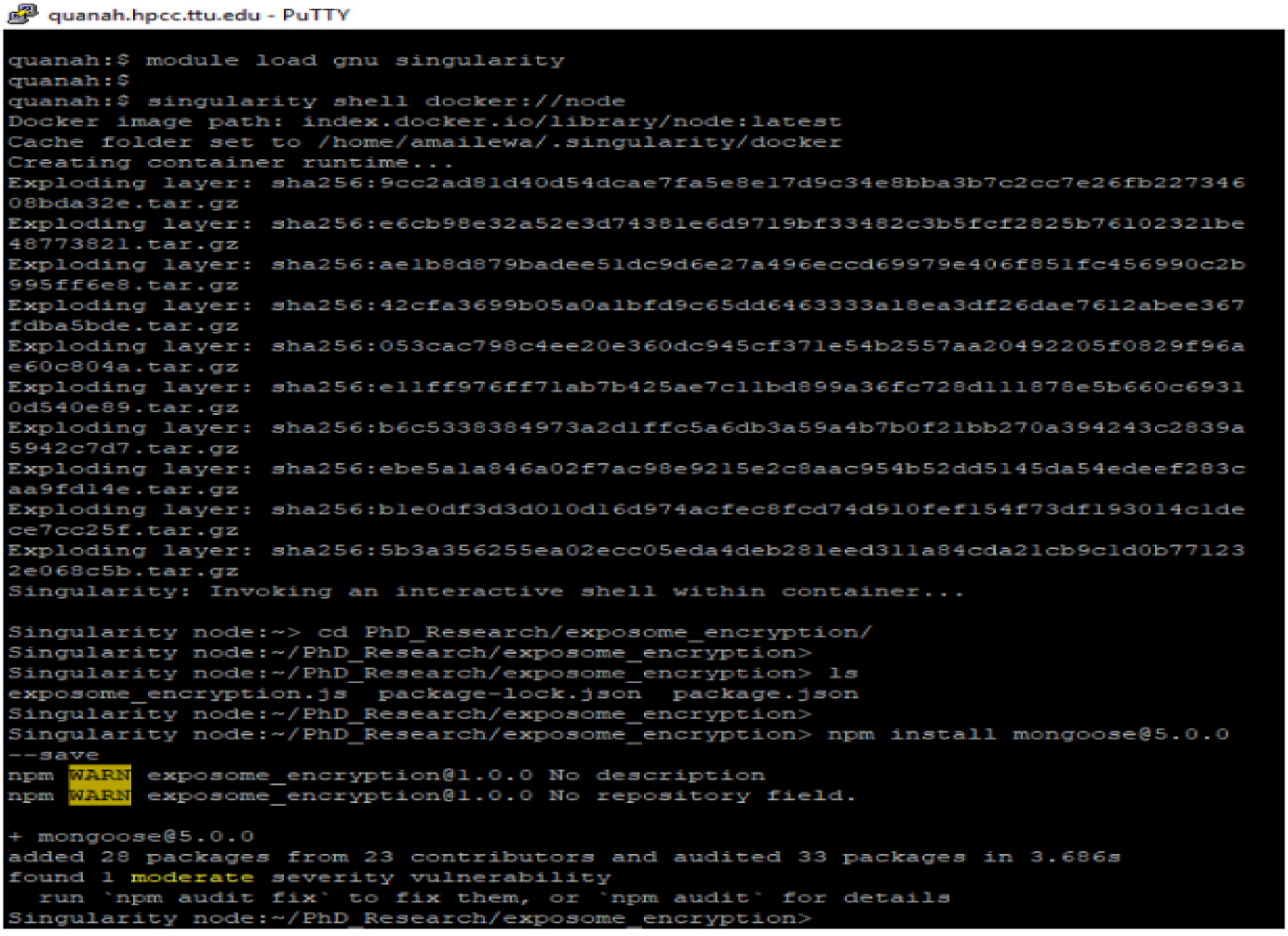


Fig. 7. Load Singularity and NODE package as an image and installation of Mongoose and other packages as a Node Module in Singularity.

3. Methodology and experiments

This section discusses and proposes the techniques that are appropriate to remove vulnerabilities to improve the security of the data analytic framework with MongoDB in Singularity Linux containers mainly in four areas, such as authentication, authorization, privacy preserving data handling, and data encryption.

/home/amailewa/PhD_Research/exposome_encryption/exposome_encryption.js - quanah.hpcc.ttu.edu ...

Encoding Color

```
var patientSchema = new mongoose.Schema({
  fName: { type: String, required: true, max: 100},
  lName: {type: String, required: true, max: 100},
  address: {type: String, required: true, max: 100},
  decease: {type: String, required: true, max: 100},
  ssn: {type: String, required: true, max: 12},
});

var encKey = 'SmXjyenxPZO6pOCEYBUJFjMs8Kdo+RNzS7ooC04JFaw='; //32BYTE_BASE64_STRING
var sigKey = 'KNhMS0neWBUqMWIP9zgTtcq8nSaIzbpE/23mrYsCdDfKr5+vpucZ9JkHxlj01/iOJwKbxfNA8H

patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey }); //By defau
//patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey, decryptPost
//patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey, excludeFrom
//patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey, encryptedFi

Patient= mongoose.model('Patient', patientSchema);
//module.exports = Patient;

Akalanka = new Patient({ fName: 'Akalanka', lName: 'Mailewa', address: 'MN-SCSU', deceas
Akalanka.save(function(err){ // encrypted when sent to the database
```

Fig. 8. "Exposome_encryption.js" Java-Script to create the data base model.

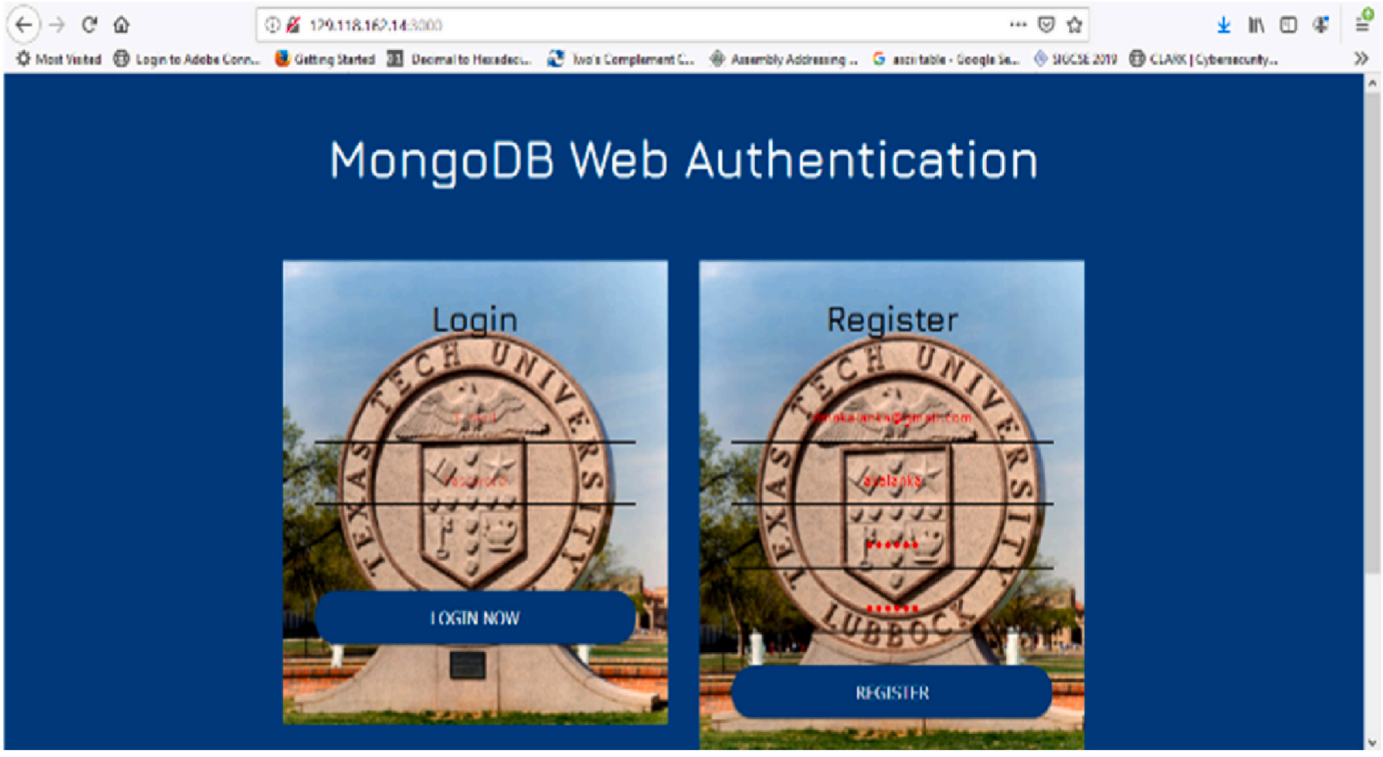


Fig. 9. User registration and login portal for MongoDB.

3.1. Authentication

To authenticate as a user, MongoDB server must be provided a username, password, and the authentication database associated with that user. To authenticate using the mongo shell, either use the mongo command-line authentication options (–username, –password, and

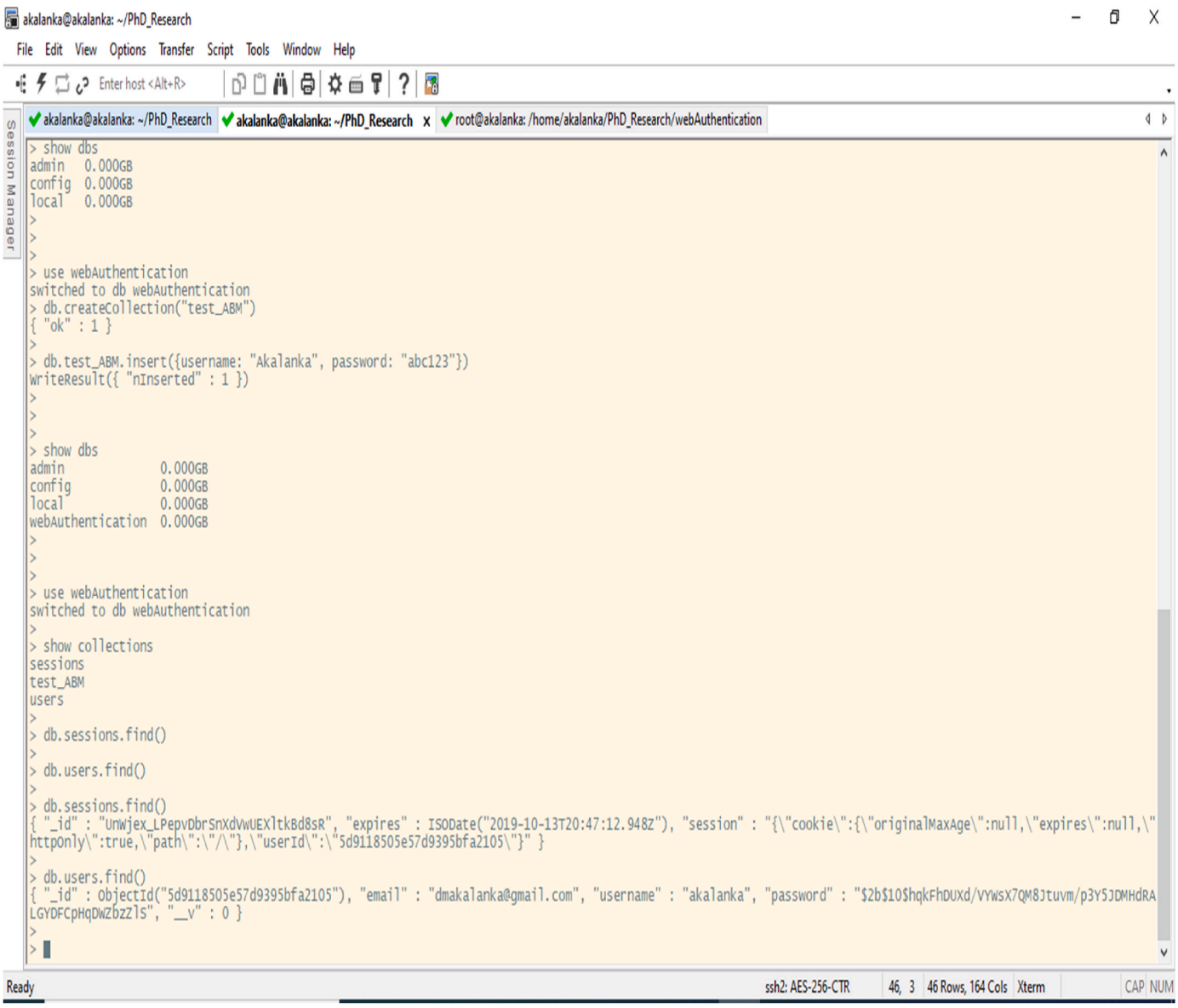


Fig. 10. After creating a user, user passwords are hashed and stored in the "users" collection of the "webAuthentication" database.

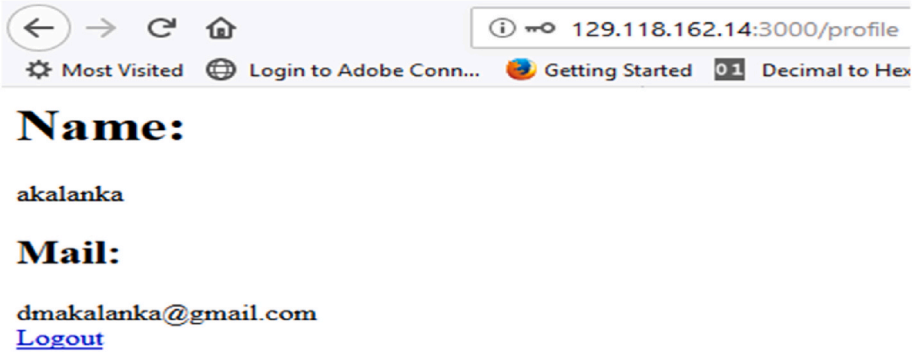


Fig. 11. Sample web page that indicate the successful login.

–authenticationDatabase) when connecting to the mongod or mongos instance, or connect first to the mongod or mongos instance, and then run the authenticate command or the db. auth() method against the authentication database [43,44]. As we discussed before, by default MongoDB has no any authentication mechanism is activated. Therefore, in the first stage we simply use Salted Challenge Response Authentication Mechanism (SCRAM) as the default authentication mechanism for MongoDB. Thereafter as we proposed before, use "Mongoose" with

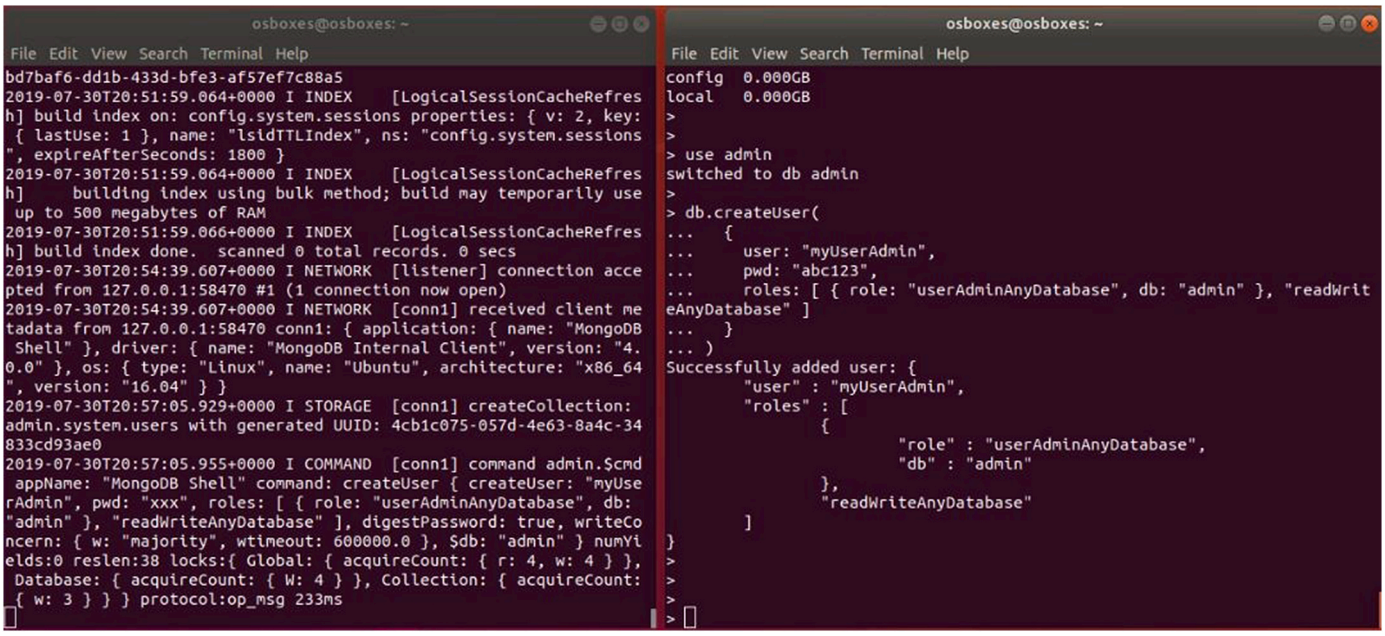


Fig. 12. While server is listing at left side, Client at right side adds the administrator.

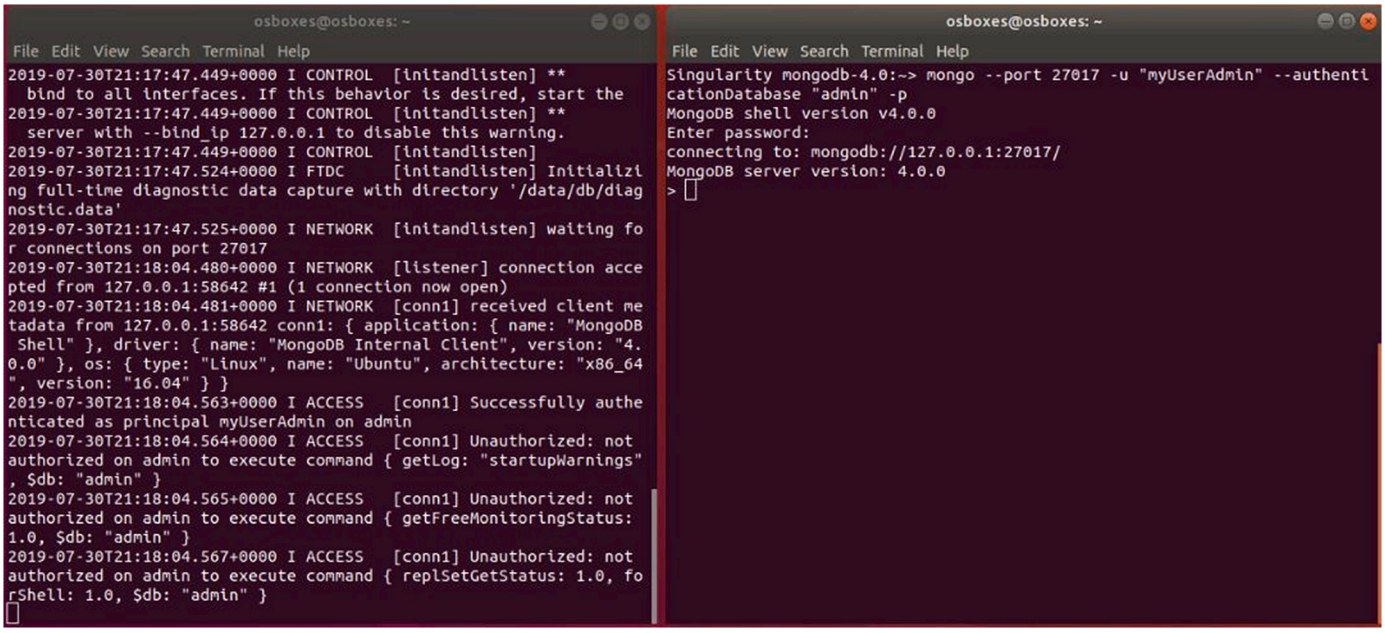


Fig. 13. While server is listing at left side, Client at right is authenticated as admin.

"Express.js", and "bcrypt" to implement a web-based authentication process. Mongoose is an Object Data Modeling library for MongoDB and Express. js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB [45]. Express. js, which is another form of the JavaScript runtime based upon the Node. js platform. This enables us to execute javascript code when interacting with our MongoDB application [46]. Finally, bcrypt is used in order to protect the password information within our database, such that bcrypt is a password hashing function based on the Blowfish cipher [47]. This security mechanism involves following three steps to gain access to a database.

Step 1: Deploy an instance of MongoDB server and client on two Singularity containers. Then, created the "webAutehntication" database. After that the test_ABM data collection is inserted into the database to manage web user's authentication as shown in Fig. 3.

Step 2: Create a Representational State Transfer - Application Programming Interface (REST - API) and hence a web user can be authenticated securely with user credentials, proved via the frontend web interface. Provided user passwords are hashed by using "bcrypt" and stored instead of storing text format in the "webAuthentication" database in the MongoDB server. Sample code of this process is shown in Fig. 4.

Step 3: Starting the Web Authentication Mechanism. After implementing a user schema and configuring the frontend web template, we can start the authentication mechanism by running the authentication. js file with Node. js as shown below in Fig. 5.

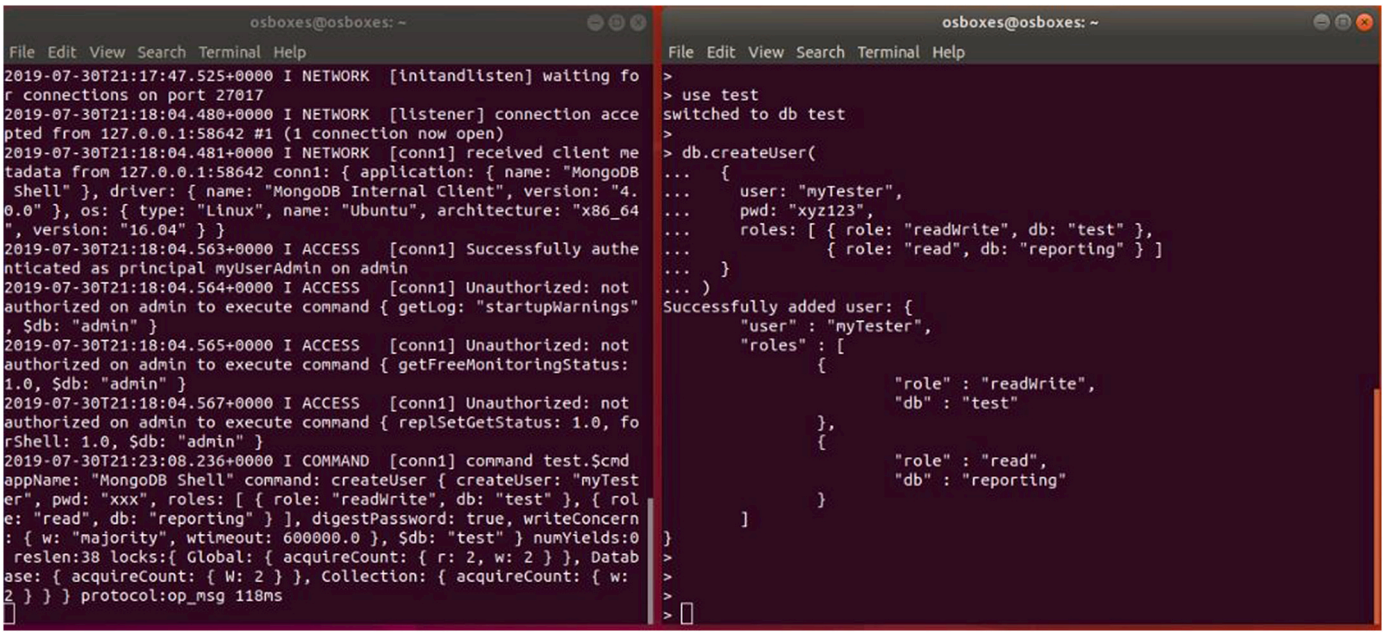


Fig. 14. While server is listing at left side, Client at right add the test user.

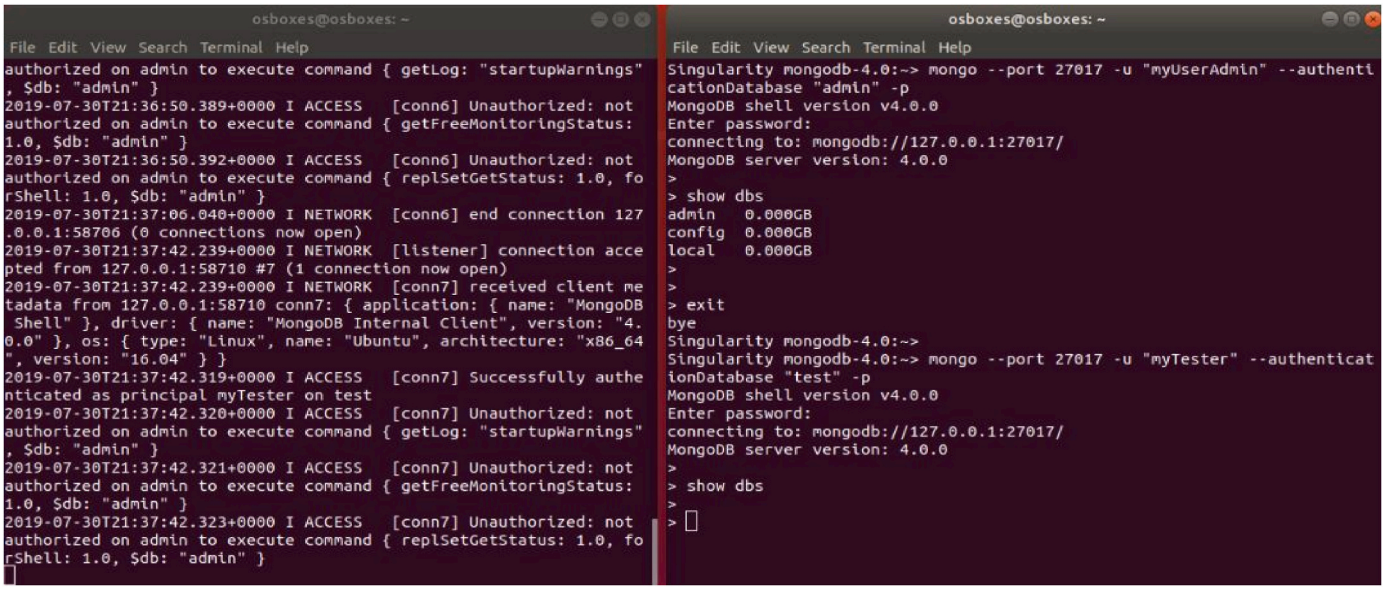


Fig. 15. Client at right tests different accessibility levels with two different users.

3.2. Authorization

This section shows how to add different privilege levels to different types of users by using access control roles. To add a user, MongoDB provides the db. createUser() method. When adding a user, roles may be assigned roles to the user in order to grant privileges, such as root, admin, and normal user. The first user created in the database should be a user administrator who has privileges to manage other users.

3.3. Privacy preserving data extractions, transmissions, and loadings

As discussed previously in section 2.3, privacy preserving data handling is another mechanism to protect the privacy of data in this research, especially with sensitive data. This security mechanism basically converts, an actual data/file header to a numerical data/file header representation. Therefore, it is hard to guess what a particular column's values represent or to which file is being referred because, headers are just numerical values that can only be identified by accessing the data dictionary.

3.4. Encryption

As we discussed before, we have to make sure data is encrypted during transmission and at storage. We already addressed how to enable TLS/SSL encryption to ensure the data is encrypted while transmission under vulnerability prevention techniques in root cause analysis section. Thus, this section shows, how our encryption mechanism works to encrypt our sensitive data and store in MongoDB by using mongoose, crypto module with Node. js to fulfill the "encrypt data at rest" requirement of the data analytic framework as shown below in following steps.

Step 1: After installing node package manager (npm), initiate the

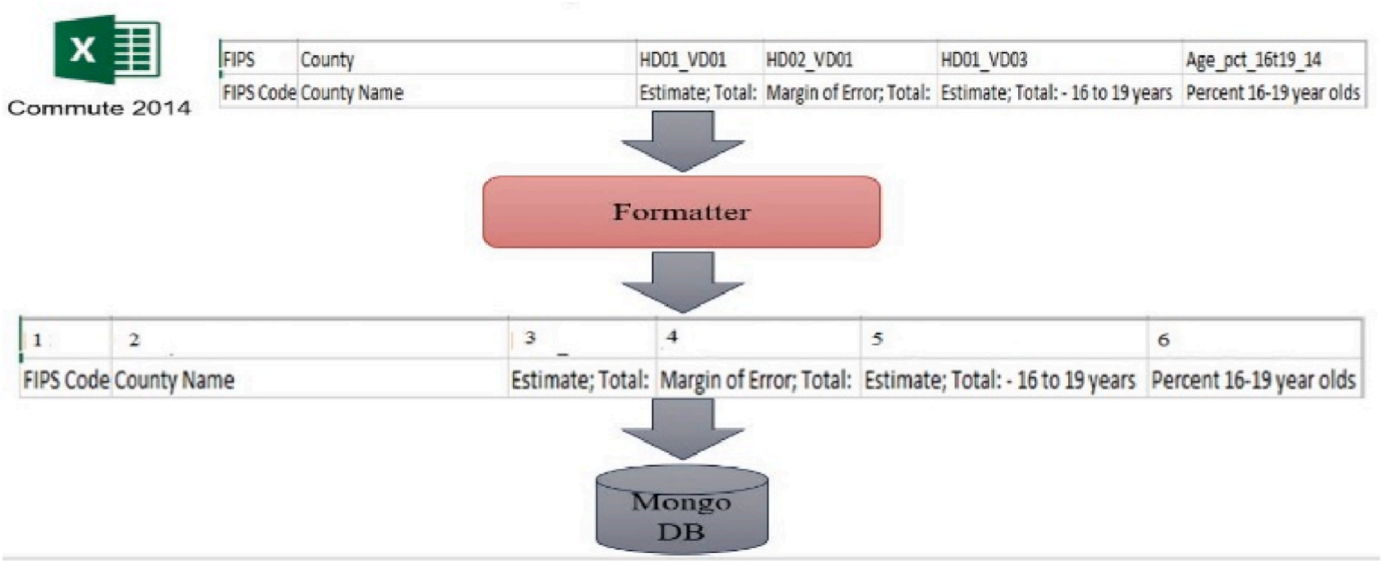


Fig. 16. Actual header names maps to numerical values.

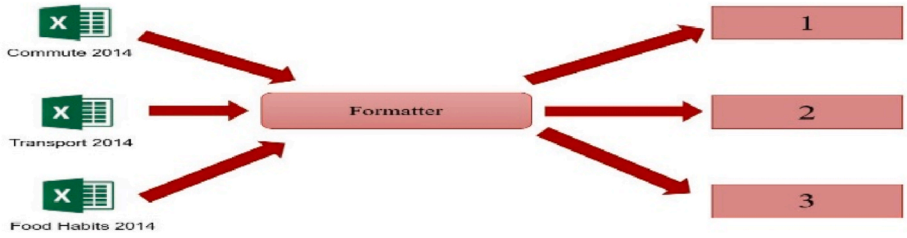


Fig. 17. Actual file names maps to numerical values.

Identifier	Description
_id	Unique ID assigned by the system
inCollections	List (Array containing collection names)

```
> db.collections.find().pretty()
{ "_id" : 1, "inCollections" : [ 1, 2, 4, 6 ] }
{ "_id" : 2, "inCollections" : [ 1 ] }
{ "_id" : 3, "inCollections" : [ 1, 2 ] }
{ "_id" : 4, "inCollections" : [ 2 ] }
{ "_id" : 5, "inCollections" : [ 3 ] }
{ "_id" : 6, "inCollections" : [ 3, 4, 6 ] }
{ "_id" : 7, "inCollections" : [ 3, 4, 6 ] }
```

Fig. 18. Numerical representation of each header in multiple collections.

project environment by creating "package.json" file as shown in Fig. 6. Later on we will add more dependencies based on our requirement.

Step 2: In order to execute this mechanism, first we need to load Node. js as a singularity container running in the High-Performance Computing Center (HPCC) cluster. Thereafter, we have to load all the required packages into package. json file created in the previous step as shown in Fig. 7.

Step 3: In order to store encrypted data in the database, we have created an initial data base model by using "exposome_encryption.js" Java-Script file as shown in Fig. 8. Initially, we have used simple 32 bits base-64 key for the encryption to test the system. But later on, we can use stronger key such as 128-bits or 256-bits in length based on our security requirement. Thereafter, we created a "patients" collection to store a particular patient's data in an encrypted format. With this mechanism, we can encrypt either all the fields or selected fields of a particular collection.

4. Results

This section verifies the accuracy of all the security mechanisms introduced in this research to enhance the overall security of the data analytic framework implemented with MongoDB and LXCs.

Identifier	Description
_id	Unique ID assigned by the system
header	header name (variable name)
description	Basic description of the header (variable name)

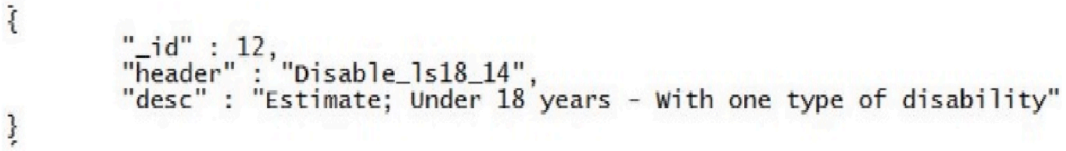


Fig. 19. Format of header ID, Actual name, and Description.

Identifier	Description
_id	Unique ID assigned by the system
file	Name of the File.

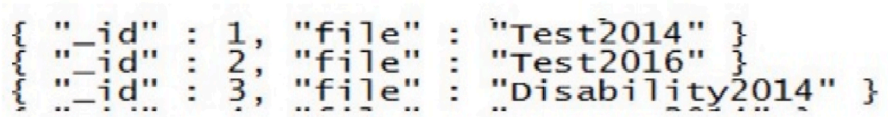


Fig. 20. File names that are mapped to the unique headers.

4.1. Testing the authentication mechanism

To test this initial version of web based authentication mechanism, first, we create a user "akalanka" and a password "abc 123" via frontend web template's registration portal as shown below in Fig. 9.

After the user is created, the user's credentials are stored in the "webAuthentication" database as shown in Fig. 10. Please note that the password is hashed and stored in the "users" collection of the "webAuthentication" database.

Finally, Fig. 11 shows, in order to access the system, a particular user, applies his/her correct logging credentials through the frontend shown in Fig. 9.

4.2. Testing the authorization mechanism

- **Create the user administrator**: After starting two MongoDB instances on Singularity containers, set one as server and the other as client. On the client, add the admin user as shown in Fig. 12. After that restart the MongoDB instance with access control and connect again to authenticate as the user administrator as shown in Fig. 13.
- **Create additional users as needed**: Once authenticated as the user administrator, again use db. createUser() to create additional users as shown in Fig. 14 where it adds a user "myTester" to the test database who has a "readWrite" role in the "test" database as well as "read" role in the "reporting" database.

The database "test" where the user "myTester" is created is called the user's authentication database. Although the user would authenticate to this database, the user can have roles in other databases; i.e., the user's authentication database does not limit the user's privileges. In Fig. 15, the MongoDB client shows that admin can see and test all the databases available in the server, but the user "myTester" cannot see databases available in the server after implementing the access control..

4.3. Testing the privacy preserving data handling mechanism

Fig. 16 shows, each column header of "Commute 2014" file, such as "FIPS", "County", and "HD01_VD01" etc., converts into a numerical value such as 1, 2, and 3, etc. After that the formatted file loads into the data the MongoDB database [2].

Fig. 17 shows, each file name, such as "Commute 2014", Transport 2014″, and "Food Habits 2014" converts into numerical values such as 1, 2, and 3. After that, the formatted file loads into the data the MongoDB database [2].

Fig. 18 shows, how we achieved the privacy preservation of data only by using the numerical representation of each header and file, without displaying the actual header name or file name [2].

Fig. 19 shows, the format of the "Data Dictionary" where it contains all unique header names so that any researcher can easily identify which ID refers to which header [2].

Fig. 20 shows, the file names that are mapped to the unique headers so that any researcher can easily identify which ID refers to which file name [2].

4.4. Testing the encryption mechanism

Fig. 21 shows the results, after executing the "exposome_encryption.js" script with "patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey}); " to encrypt the entire collection. The left side console clearly indicates that all the fields of the "patient" collection's data is encrypted. The right side console shows the current connection status to the "encryption" database form the client and decrypted data of the "patient" collection.

Fig. 22 shows the results, after executing the "exposome_encryption.

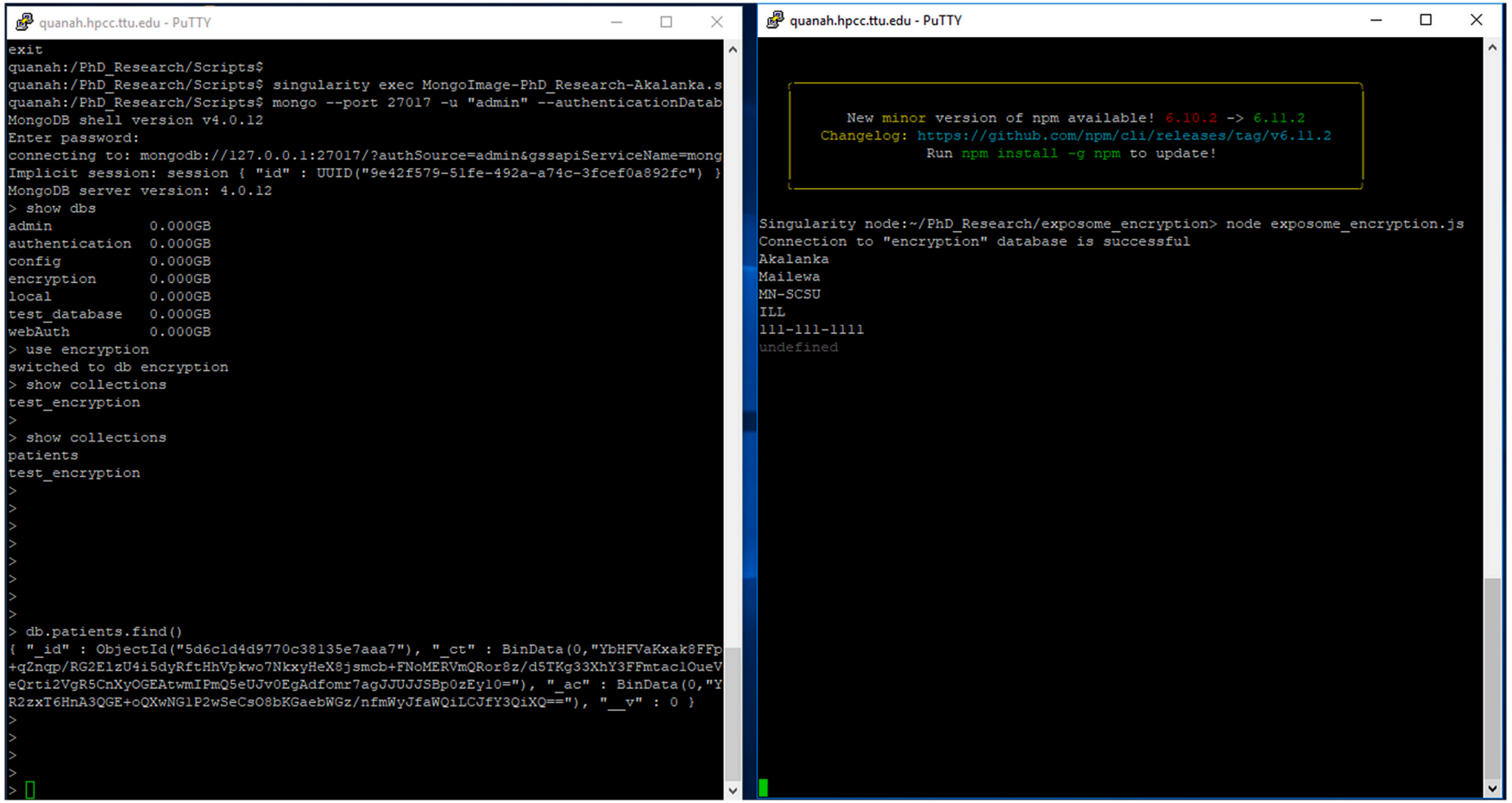


Fig. 21. Encrypted data of entire collection at left console and display of the decrypted results on the console at right side.

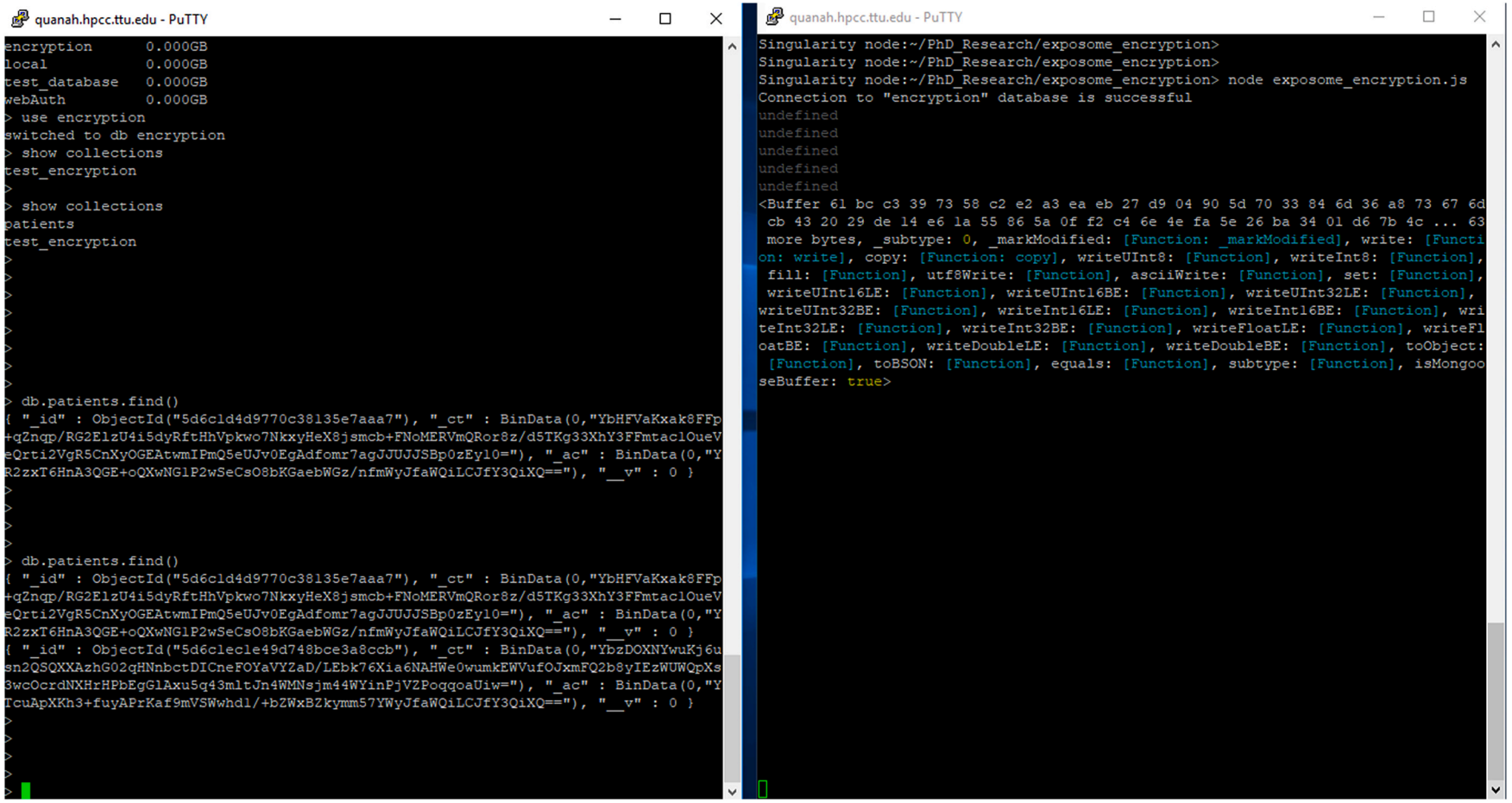


Fig. 22. Encrypted data of entire collection at 2nd object of the left console and at right no meaningful data is displayed.

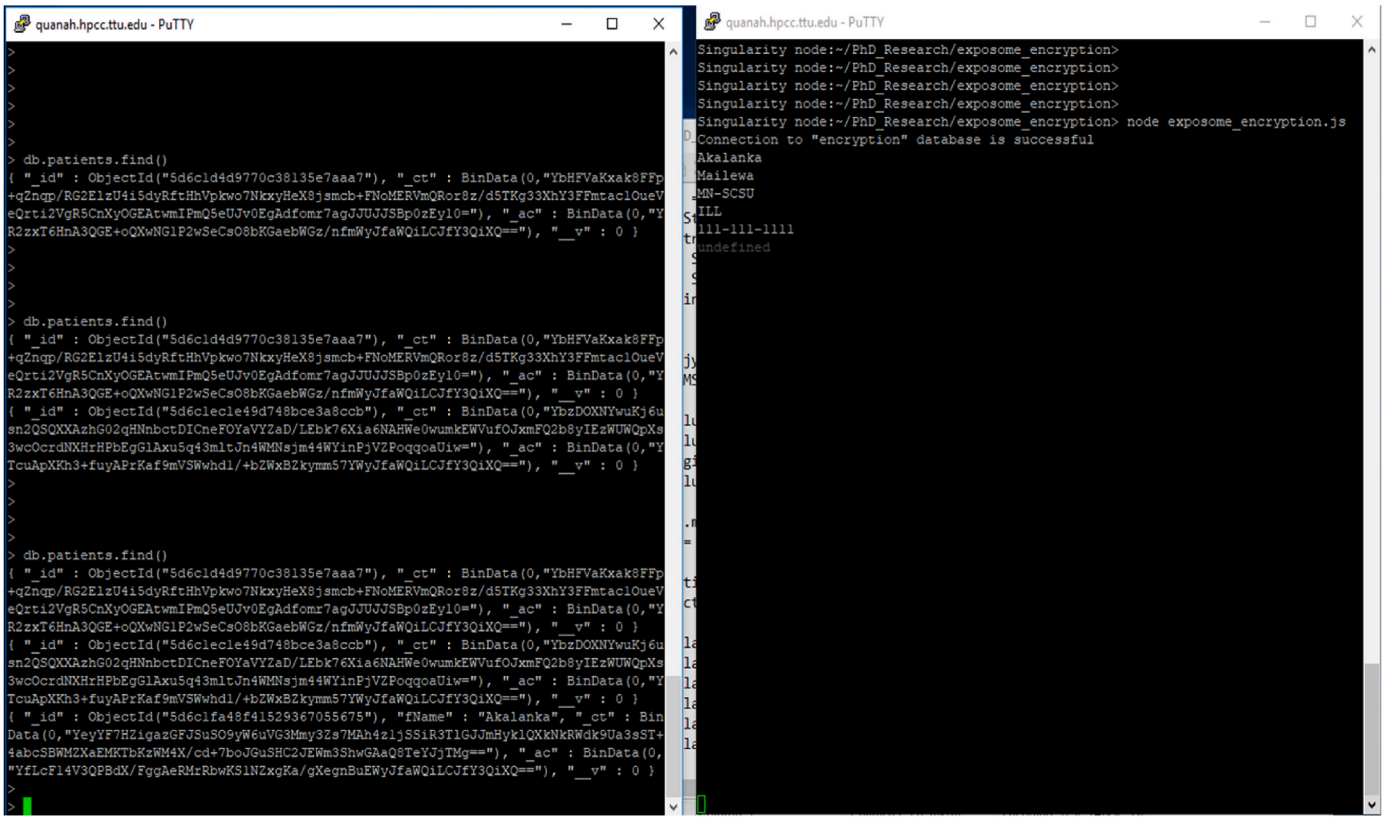


Fig. 23. Encrypted data of entire collection except the "fName" field at 3rd object of the left console and display the decrypted results on the console at right side.

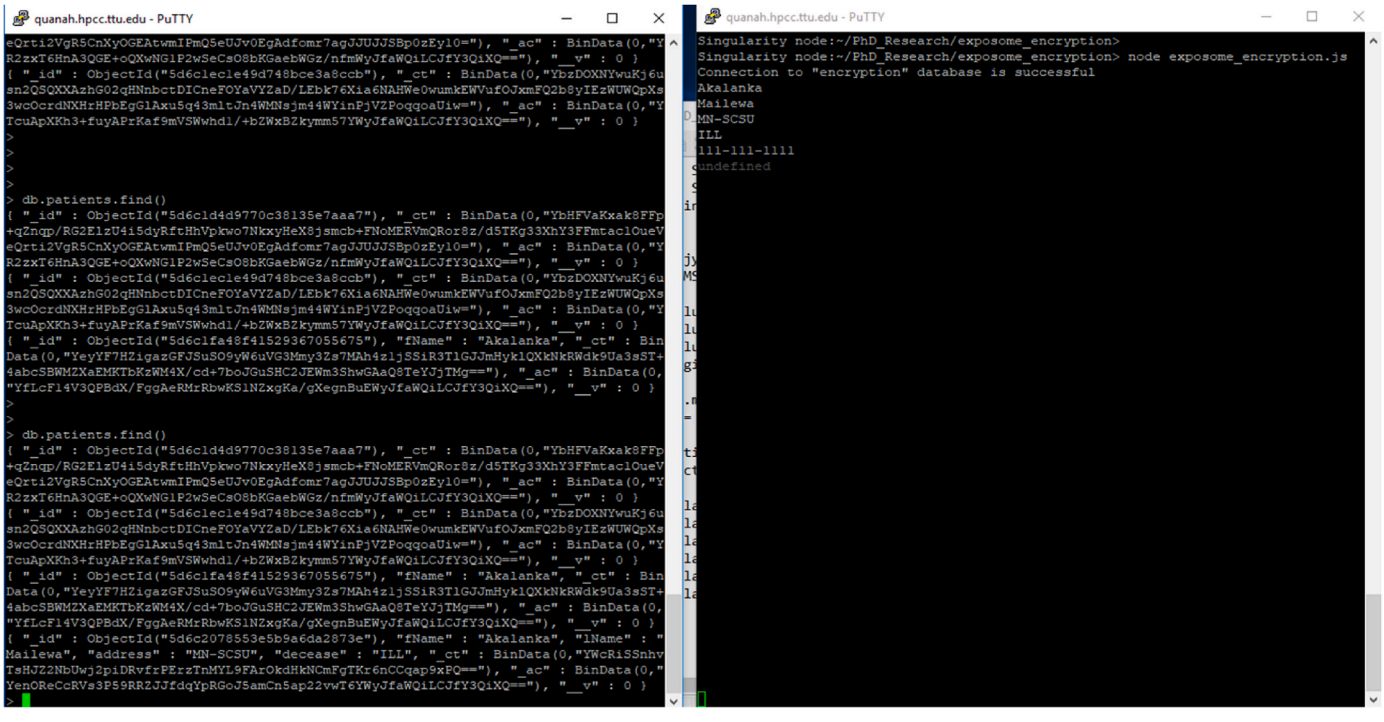


Fig. 24. Encrypted data only available in the "SSN" at the left console and display the decrypted results on the console at right side.

js" script with "patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey, decryptPostSave: false}); " to encrypt the entire collection, but not to decrypt and display on right side console. Left side console's 2nd object clearly indicates that all the fields of the "patient" collection's data is encrypted. Right side console shows the current connection status to the "encryption" database form the client and some scrambled data.

Fig. 23 shows the results, after executing the "exposome_encryption.

js" script with "patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey, excludeFromEncryption: ['fName']}); " to encrypt the entire collection except the "fName" field. Left side console's 3rd object clearly indicates that all the fields of the "patient" collection's data is excerpted except the "fName" field. Right side console shows the current connection status to the "encryption" database from the client and decrypted data of the "patient" collection.

Fig. 24 shows the results, after executing the "exposome_encryption.js" script with "patientSchema.plugin(encrypt, { encryptionKey: encKey, signingKey: sigKey, encryptedFields: ['SSN']}); " to encrypt only the "SSN" field but not to encrypt all the other fields. Left side console's 4th object clearly indicates that only the "SSN" field is encrypted and all the other fields' data is saved in plain text format of the "patient" collection. Right side console shows the current connection status to the "encryption" database from the client and decrypted data of the "patient" collection.

5. Conclusions

By using all of the aforementioned security mechanisms, we were able to add additional security to our data analytic framework proposed in the previous three phases of the research as shown through above results were in the first phase we proposed a data analytic framework with MongoDB and LXCs with basic security requirements. Next, in the second phase we proposed a vulnerability analysis testbed to find vulnerabilities associated with the system. Finally, in the third phase we discussed in detail root causes and some prevention techniques of vulnerabilities found in the system. Second, the paper clearly presented that it has answered the research question "RQ: How does the data analytic framework with MongoDB and LXCs is designed with enhanced security features?" through our research methodologies by proposing the aforementioned security mechanisms and techniques, related to the research question. Third, overall security requirements are dependent on a particular organization's or person's security policies and available budget. Therefore, to have a maximum expected security level of this proposed data analytic framework, it is essential to tune the framework in order to meet with the organization's or person's security policies in order to add much more security. Finally, it is noteworthy that the overall security of a software system is a shared responsibility of the technical experts as well as its users/clients. Especially, the vulnerabilities that may arise through the users play a crucial role, because the security of the system can be compromised if the authenticated users are misguided to exploit the vulnerabilities of the system. Therefore, it is important to bear in mind that the root causes of system vulnerabilities may not always be due to the technical aspects of the security mechanism.

6. Future works

It has been proven time and again that there is certainly a positive correlation between the progress of the cybersecurity world and the technological improvements designed to prevent cyber-criminals from exploiting data security loopholes. Each step of this research emphasized the fact that security must be implemented in a multi-layered, evolutionary development process, which requires constant investigations to find vulnerabilities in the system using the vulnerability testbed proposed in the second phase of the research. It can also be used to test the real data analytic system containing sensitive data. Moreover, adding some other tools such as "OWASP Zed Attack Proxy" and "OpenSCAP" are also compatible with this testbed to verify the detected vulnerabilities and to investigate the undetected vulnerabilities, are associated with all four areas: "images with containers," "host," "network," and "MongoDB application." Thus, adding more tools to the current testbed will enable experts to perform more tests and re-do the experiments to find previously undetected vulnerabilities. Based on these new vulnerabilities, a new data analytic framework will be proposed. This framework would include security features, such as advanced encryption and authentication mechanisms as well as enriched auditing methods to improve the data retrieval performance in any given system. Apart from the traditional security mechanisms, several sophisticated techniques, such as "Auditing Mechanism" and "Blockchain", will also be used in the framework. A blockchain consists of a chain of blocks with data/information. Each block has a cryptographic hash of the previous block, a timestamp, and transaction data [48–50]. Despite its simple design, a blockchain is well-secured in the face of date exploitation and the versatile technology behind it has a broad implication in the future of data security. Innovative uses of blockchain technology are already becoming a part of other fields beyond cryptocurrencies and can be especially useful to boost the security of the cyber world [51–53]. The rigorous encryption and the data distribution of the network ensure that the data remains intact and safe from attackers. Thus, the authors have identified a blockchain to have the most suitable technology to enhance the security of the proposed system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Akintaro Mojolaoluwa, Pare Teddy, Mailewa Dissanayaka Akalanka. DARKNET and black market activities against the cybersecurity: a survey. In: The Midwest Instruction and Computing Symposium. (MICS). Fargo, ND: North Dakota State University; April .

[2] Ramprasad Shetty Roshan, Mailewa Dissanayaka Akalanka, Mengel Susan, Gittner Lisa, Vadapalli Ravi, Khan Hafiz. Secure NoSQL based medical data processing and retrieval: the exposome project. In: Companion Proceedings of the10th International Conference on Utility and Cloud Computing (UCC '17 Companion). New York, NY, USA: ACM; 2017. p. 99–105.

[3] Gittner LS, Kilbourne BJ, Vadapalli R, Khan HM, Langston MA. A multifactorial obesity model developed from nationwide public health exposome data and modern computational analyses. Obesity Research & Clinical Practice; 2017.

[4] Mailewa Dissanayaka Akalanka, Ramprasad Shetty Roshan, Kothari Samip, Mengel Susan, Gittner Lisa, Vadapalli Ravi. A review of MongoDB and singularity container security in regards to HIPAA regulations. In: Companion Proceedings of the10th International Conference on Utility and Cloud Computing (UCC '17 Companion). New York, NY, USA: ACM; 2017. p. 91–7.

[5] Akalanka Mailewa Dissanayaka, Susan Mengel, Lisa Gittner, and Hafiz Khan. Dynamic & portable vulnerability assessment testbed with Linux containers to ensure the security of MongoDB in Singularity LXCs. In Companion Conference of the Supercomputing-2018 (SC18).

[6] Akalanka Mailewa Dissanayaka, Susan Mengel, Lisa Gittner, and Hafiz Khan. Vulnerability prioritization, root cause analysis, and mitigation of secure data analytic framework implemented with MongoDB on singularity Linux containers. In The 4th International Conference on Compute and Data Analysis -2020 (ICCDA-2020). [San Jose, CA].

[7] Jiye Wang, Gao Lingchao, Dong Aiqiang, Guo Shaoyong, Hui Chen, Xin Wei. Block chain based data security sharing network architecture research. J Comput Res Dev 2017;54(4):742.

[8] Jeon Jin Hyeong, Kim Ki-Hyung, Kim Jai-Hoon. Block chain based data security enhanced IoT server platform. In: 2018 International Conference on Information Networking (ICOIN). IEEE; 2018. p. 941–4.

[9] Zugaj Wilhelm, Stefanie Beichler Anita. Towards a NoSQL security map. 2018.

[10] Mardan Azat. Boosting node. Js and MongoDB with mongoose. In: Practical Node. js. Berkeley, CA: Apress; 2018. p. 239–76.

[11] Doglio Fernando. Reactive programming on the back-end. In: Reactive Programming with Node. js. Berkeley, CA: Apress; 2016. p. 47–66.

[12] de Araujo Zanella, Rettore Angelita, da Silva Eduardo, Albini Luiz Carlos Pessoa. Security challenges to smart agriculture: current state, key issues, and future directions. Array 2020:100048. https://doi.org/10.1016/j.array.2020.100048.

[13] Kulshreshta Akhil, Rawat Neelam, Saxena Krati, Agrawal Prashant. Web based accounting integrated management system (AIMS) over cloud using mean stack. In: 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). vol. 1. IEEE; 2019. p. 1–5.

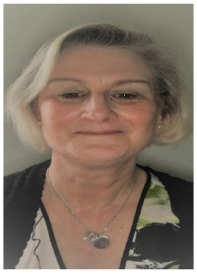
[14] Cayres, Ungari Leandro, Bruno Santos de Lima, Garcia Rogério Eduardo, Ronaldo Celso Messias Correia. Analysis of node. Js application performance using MongoDB drivers. In: International Conference on Information Technology & Systems. Cham: Springer; 2020. p. 213–22.

[15] Mailewa Akalanka, Herath Jayantha, Herath Susantha. A survey of effective and efficient software testing. In: The Midwest Instruction and Computing Symposium. (MICS); April 10-11 2015. Grand Forks, ND.

[16] Vijayakumar Pandi, Chang Victor, Jegatha Deborah L, Bharat S, Kshatriya Rawal. Key management and key distribution for secure group communication in mobile and cloud network. 2018. p. 123–5. https://doi.org/10.1016/j.future.2018.03.027.
[17] Jena Swagat Kumar, Krishna Tripathy Bata, Gupta Prachi, Das Satyabrata. A Kerberos based secure communication system in smart (internet of things) environment. J Comput Theor Nanosci 2019;16(5–6):2381–8.
[18] Matotek Dennis, James Turnbull, Peter Lieverdink. Directory services. In: Pro Linux System Administration. Berkeley, CA: Apress; 2017. p. 733–97.
[19] Wen Shuo, Yuan Xue, Xu Jing, Yang Hongji, Li Xiaohong, Song Wenli, Si Guannan. Toward exploiting access control vulnerabilities within mongodb backend web applications. In: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). vol. 1. IEEE; 2016. p. 143–53.
[20] Yerlikaya Özlem, Dalkılıç Gökhan. Authentication and authorization mechanism on message queue telemetry transport protocol. In: 2018 3rd International conference on computer science and engineering (UBMK). IEEE; 2018. p. 145–50.
[21] Feng Qi, He Debiao, Zeadally Sherali, Liang Kaitai. BPAS: blockchain-assisted privacy-preserving authentication system for vehicular ad-hoc networks. IEEE Transactions on Industrial Informatics 2020;16(6):4146–55. https://doi.org/10.1109/TII.2019.2948053.
[22] Abosaif Aghabi N, Haitham S Hamza. Quality of service-aware service selection algorithms for the internet of things environment: a review paper. Array 2020;8:100041. https://doi.org/10.1016/j.array.2020.100041.
[23] Jose Benymol, Abraham Sajimon. Exploring the merits of nosql: a study based on mongodb. In: 2017 International Conference on Networks & Advances in Computational Technologies (NetACT). IEEE; 2017. p. 266–71.
[24] Shih Dong-Her, Huang Feng-Chuan, Lai Wei-Hao, Shih Ming-Hung. Privacy preserving and performance analysis on not only SQL database aggregation in bigdata era. In: 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). IEEE; 2017. p. 56–60.
[25] Tian Xingbang, Huang Baohua, Wu Min. A transparent middleware for encrypting data in MongoDB. In: 2014 IEEE Workshop on Electronics, Computer and Applications. IEEE; 2014. p. 906–9.
[26] Simkhada Emerald, Shrestha Elisha, Pandit Sujan, Sherchand Upasana, Mailewa Dissanayaka Akalanka. Security threats/attacks via BOTNETS and botnet detection & prevention techniques in computer networks: a review. In: The Midwest Instruction and Computing Symposium. (MICS). Fargo, ND: North Dakota State University; April .
[27] Humphrey M, Steele J, Kim IK, Kahn MG, Bondy J, Ames M. CloudDRN: a Lightweight, end-to-end system for sharing distributed research data in the cloud. In: eScience (eScience), 2013 IEEE 9th International Conference on; 2013. p. 254–61.
[28] Angrish Atin, Starly Binil, Lee Yuan-Shin, Cohen Paul H. A flexible data schema and system architecture for the virtualization of manufacturing machines (VMM). J Manuf Syst 2017;45:236–47.
[29] Makowski Łukasz, Grosso Paola. Evaluation of virtualization and traffic filtering methods for container networks. Future Generat Comput Syst 2019;93:345–57. https://doi.org/10.1016/j.future.2018.08.012.
[30] Mailewa Akalanka, Herath Jayantha. Operating systems learning environment with VMware. In: The Midwest Instruction and Computing Symposium. Verona, WI: MICS); April .
[31] Zhipeng Zhang, Chandel Sonali, Sun Jingyao, Yan Shilin, Yu Yunnan, Zang Jingji. Vpn: a boon or trap?: a comparative study of mpls, ipsec, and ssl virtual private networks. In: 2018 Second International Conference on Computing Methodologies and Communication (ICCMC). IEEE; 2018. p. 510–5.
[32] Chokhawala Kirit I, Chuabay Vinit Kumar, Patel AR. Secure web application: preventing application injections. 2016.
[33] Bhathal Gurjit Singh, Singh Amardeep. Big data: hadoop framework vulnerabilities, security issues and attacks. Array 2019;1:100002. https://doi.org/10.1016/j.array.2019.100002.
[34] Saxena Upaang, Sachdeva Shelly. An insightful view on security and performance of NoSQL databases. In: International Conference on Recent Developments in Science, Engineering and Technology. Singapore: Springer; 2017. p. 643–53.
[35] Dayley Brad, Dayley Brendan, Dayley Caleb. Node. js, MongoDB and Angular Web Development: the definitive guide to using the MEAN stack to build web applications. Addison-Wesley Professional; 2017.
[36] Huluka Daniel, Popov Oliver. Root cause analysis of session management and broken authentication vulnerabilities. In: World Congress on Internet Security (WorldCIS-2012). IEEE; 2012. p. 82–6.
[37] Kritikos Kyriakos, Magoutis Kostas, Papoutsakis Manos, Ioannidis Sotiris. A survey on vulnerability assessment tools and databases for cloud-based web applications. Array 2019;3:100011. https://doi.org/10.1016/j.array.2019.100011.
[38] Lin Zhiqiang, Jiang Xuxian, Xu Dongyan, Mao Bing, Xie Li. AutoPaG: towards automated software patch generation with source code root cause identification and repair. In: Proceedings of the 2nd ACM symposium on Information, computer and communications security. ACM; 2007. p. 329–40.
[39] Fernandez Eduardo B. A methodology for secure software design. Software Engineering Researchand Practice; 2004.
[40] Youssef Ahmed E. A framework for secure healthcare systems based on big data analytics in mobile cloud computing environments. Int J Ambient Syst Appl 2014;2 (no. 2):1–11.
[41] Chakravorty Antorweep, Wlodarczyk Tomasz, Rong Chunming. Privacy preserving data analytics for smart homes. In: 2013 IEEE Security and Privacy Workshops. IEEE; 2013. p. 23–7.
[42] Abraham Subil, Nair Suku. Cyber security analytics: a stochastic model for security quantification using absorbing Markov chains. J Commun 2014;9(12):899–907.
[43] Chellappan Subhashini, Ganesan Dharanitharan. MongoDB security. In: MongoDB Recipes. Berkeley, CA: Apress; 2020. p. 215–41.
[44] Plugge Eelco, Membrey Peter, Hawkins Tim. Database administration." the definitive guide to MongoDB: the NoSQL database for cloud and desktop computing. 2010. p. 193–223.
[45] Pandian Parri. Building node. Js REST API with TDD approach: 10 steps complete guide for node. Js, express. Js & MongoDB RESTful service with test-driven development. 2018.
[46] Zhang Qi. Medical data visual synchronization and information interaction using Internet-based graphics rendering and message-oriented streaming. Inform Med Unlocked 2019;17:100253.
[47] Sriramya P, Karthika RA. Providing password security by salted password hashing using Bcrypt algorithm. ARPN J Eng Appl Sci 2015;10(13):5551–6.
[48] Puthal Deepak, Malik Nisha, Mohanty Saraju P, Elias Kougianos, Das Gautam. Everything you wanted to know about the blockchain: its promise, components, processes, and problems. IEEE Consum Electron Mag 2018;7(4):6–14.
[49] Singh Madhusudan, Kim Shiho. Branch based blockchain technology in intelligent vehicle. Comput Network 2018;145:219–31.
[50] Peters Gareth W, Panayi Efstathios. Understanding modern banking ledgers through blockchain technologies: future of transaction processing and smart contracts on the internet of money. In: Banking beyond banks and money. Cham: Springer; 2016. p. 239–78.
[51] Lin Chao, He Debiao, Huang Xinyi, Choo Kim-Kwang Raymond, Athanasios V. Vasilakos. "BSeIn: a blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. J Netw Comput Appl 2018;116:42–52.
[52] Yuan Yong, Wang Fei-Yue. Blockchain and cryptocurrencies: model, techniques, and applications. IEEE Trans Syst Man Cybernet: Systems 2018;48(9):1421–8.
[53] Kearney Joseph J, Perez-Delgado Carlos A. Vulnerability of blockchain technologies to quantum attacks. Array 2021;10:100065. https://doi.org/10.1016/j.array.2021.100065.

Dr. Akalanka B. Mailewa (Dissanayaka Mohottalalage) earned his Ph.D. in Computer Science from Texas Tech University, Lubbock, Texas, USA in Spring 2020 and his PhD dissertation research focuses on implementation of Secure Big Data Analytic Framework with MongoDB and Linux Containers in high performance clusters (HPC) in regards to HIPAA regulations. He received his M.Sc. in Computer Science from Saint Cloud State University, Saint Cloud, Minnesota, USA and earned his B·Sc. Engineering (Hons.) in Computer Engineering from University of Peradeniya, Kandy, Sri Lanka. In industry, he worked as several fulltime roles such as; Network Engineer, System Administrator, Software Developer, and Software Test Automation Engineer. He is certified with several Microsoft certifications such as MCP, MCTS, MCSA, MCITP, and MCSE-Security while he has followed RHCT and CCNA courses. In addition, he is a Palo-Alto Networks Authorized Cybersecurity Academy Instructor with CIC and CPC certifications. His research interest includes Big-data security, IoT security, cryptography, ethical hacking, penetration testing, and software test automation. On his research area, he has published several scientific research journal articles and conference proceedings by presenting in several well-known conferences. Also, he is serving as editorial board member and reviewer of several journals and conferences. In addition, he has received fellowships, awards, and travel grants from various institutes. Overall, about 15 years of teaching experiences in higher education, he is currently working as a tenure-track assistant professor in the department of computer science and information technology at Saint Cloud State University, Saint Cloud, Minnesota, USA.

Dr. Susan Mengel received her Ph.D. from Texas A&M University in 1990. She is an Associate professor at Texas Tech University. She has played strategic leadership roles in numerous transdisciplinary projects involving the delivery of innovative software and data models in sleep management, student retention and advising, computer education, nutrition, speech therapy, cardiovascular disease, and cybersecurity. She has been funded by the National Science Foundation and has over 60 papers including reports, journal publications, conference papers, and chapters. She helped to establish the Master's in Software Engineering degree program at Texas Tech University, served as Vice-President for the Texas Tech Faculty Senate, served on the Steering Committee of the ACM/IEEE Computing Curriculum, served as an Associate Editor for Computing for the IEEE Transactions on Education, served on the IEEE Computer Society Board of Governors, and served as the Outreach Chair FY19 of the SWE Outreach Committee. She became one of the inaugural E-Learning Faculty Fellows for Texas Tech University eLearning & Academic Partnerships in 2022. She is on the Texas Tech Institutional Review Board for the Protection of Human Subjects and is a faculty co-advisor for the TTU Collegiate Chapter of SWE.



Dr. Lisa Gittner is an Associate Professor in the Department of Public Health at Texas Tech University Health Sciences Center. She has a Bachelor and Master of Science in Biochemistry/ Toxicology from Wright State University and received her Doctorate in Public Administration and Health Policy from University of Akron. Dr. Gittner's post-doc was at Case Western Reserve University as a Program Director for a National Institute of Health/ National Institute of Minority Health and Health Disparities grant (1RC2-MD004760-01) studying mechanisms to activate medically and socially disenfranchised patients to advocate for their own health. She is an Associate Member of the Laura W. Bush Institute for Women's Health and a Member of The American Society of Public Administrators. Dr. Gittner is an expert on peer social support to improve health outcomes. Her research focuses on the management of life course disease through social networks. Dr. Gittner's prior research has been funded by the Robert Wood Johnson Foundation, Garfield Foundation, Eastbay Foundation and Komen Foundation. Previously, Dr. Gittner was Director of Research for Kaiser Permanente and Director of novel dose development for Watson Pharmaceutical Company, Inc.

Dr. Hafiz Khan is a Tenured Full Professor and Professional Biostatistician in the Department of Public Health at Texas Tech University Health Sciences Center. He has nearly 30 years of higher education work experience as a lead biostatistician at Texas Tech University, Florida International University, Rutgers University, New Jersey Institute of Technology, University of Western Ontario, Chittagong University, and Chittagong Cantonment Public College. He has significantly contributed as a single, lead, and joint authorship in over 100 scholarly peer-reviewed scientific research publications in statistics, biostatistics, biomedical fields, and 120+ peer-reviewed scientific research abstracts in several conference proceedings. He has presented 130+ scientific research works in several biostatistics and biomedical research communities. As a biostatistical methodologist, he has advised 100+ graduate students on their research projects, theses, and dissertations. He is also the lead Biostatistical Advisor at the NSF Cloud and Autonomic Computing Center at TTU and Garrison Institute on Aging at TTUHSC. He has contributed to numerous funded projects and is presently working on three NSF funded projects. Dr. Khan serves as an Editorial Board Member for many reputable journals including Computational Statistics and Statistical Papers.