

Security Analysis of MongoDB and its Comparison with Relational Databases

Sukriti Bharti

Master of Information System Management Carnegie Mellon University

Abstract: *Although the world has been translating towards data focused operations for the past decades, the dependence of all industries on data has exponentially grown in the last five years. The explosion of Big Data and increase in connected devices are the reasons why there has been a shift from Relational databases to NOSQL databases, to handle huge amounts of unstructured data generated. These databases trade consistency and security for performance and scalability. With handling such vast amounts of sensitive data, security issues are a growing concern. This paper focuses on security analysis of MongoDB and draws comparisons of its security features with the traditional relational databases.*

Keywords: security, NOSQL, MongoDB, relational databases

1. Introduction

In the recent years, major companies have started adopting different types of non-relational databases to cater to the needs of the data and applications they serve. Each database has a different data model and some unique selling points and can be put into practice for specific business scenarios. For e.g., if we wish to have persistent data sharing over multiple processes or microservices, key-value store databases are the best. If we wish to perform deep relationship analysis, fraud detection etc., a graph database works the best [1].

NOSQL databases can handle large volumes of structured, semi-structured and unstructured data. They have high scalability and reliability, support flexible schema, mostly do not support ACID transactions like relational databases and provide eventual consistency.

Any organization looking transition to a NOSQL database must perform a detailed analysis of the features as well as the security aspects of the database. Through the course of this paper, we will focus our discussion around the features of MongoDB. MongoDB has a document oriented model and numerous security features, but to be able to make the best use of it and to keep the data well protected, the database administrator must do an intensive research suitable to their use case.

2. Overview of MongoDB

MongoDB is a document-oriented database program, written in C++ language. It is schema-free and works with JSON-like documents. It supports complex hierarchies of data. MongoDB's key features are:

a) Data Model

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are like JSON objects. The values of fields may include other documents, arrays, and arrays of documents [3]. These documents are stored in collections that are analogous to tables in relational databases.

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```



Figure 1: Sample MongoDB JSON-like Document [3]

b) High performance

MongoDB provides high performance data persistence. It provides support for embedded data models which reduces I/O activity on database system. Since MongoDB works with indexes, it results in faster queries [3].

For e.g., for MongoDB to retrieve a whole entity, it takes about $\log(n)+1$ I/O operations and if all indices reside in the memory, it takes 1 operation only. On the other hand, if a relational database has 20 tables, and even if the indices reside in the memory, it would take 20 operations for retrieval [4].

c) Rich Query Language

MongoDB supports a rich query language to support CRUD operations as well as Data Aggregation, Text Search and Geospatial Queries. MongoDB has in-built methods like `insertOne()`, `insertMany()`, `find()`, `updateOne()`, `deleteMany()` etc. It also lets users write the data in bulk using `writeBulk()` method [3].

```
db.users.insertOne(
  {
    name: "sue",
    age: 26,
    status: "pending"
  }
)
```

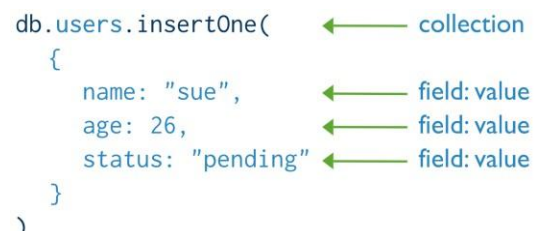


Figure 2: Sample query for inserting data into a document [3]

d) High Availability

MongoDB achieves high availability by using its replication facility called the "Replica Set". This essentially creates data redundancy which in turn ensures high availability of the data. It also provides automatic failover in case the data handling is compensated.

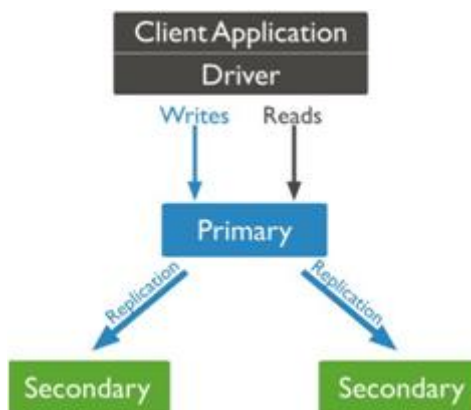


Figure 3: Replication in MongoDB [5]

e) Horizontal Scalability

MongoDB provides horizontal scalability with the key concept of Sharding. Sharding means distributing the data based on one of: location, alphabetical order, hashing etc. It creates zones of data based on the shard key. In a balanced cluster, MongoDB performs reads and writes only to those shards that fall into a zone.

In the following image, ‘X’ represents a shard key.

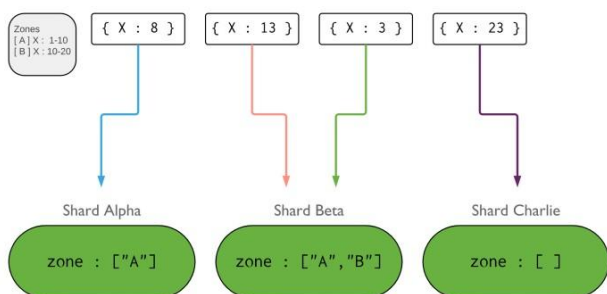


Figure 4: Zones in a sharded cluster [6]

3. Security Features of MongoDB

This section of the paper deals with analyzing the different features that MongoDB provides for the security of data and preventing attacks. MongoDB provides a checklist for developers/database administrators that can be followed to avoid the popular database attacks.

a) Authentication

Authentication means verifying the identity of the client. This is important as we would like for each user to have a personalized view of the data as well as safeguard everyone else’s data. MongoDB supports various authentication mechanism and based on an organization’s existing mechanism, an appropriate mechanism can be chosen for integration [6].

MongoDB’s default authentication method is **SCRAM** (Salted Challenge Response Authentication Mechanism) i.e. SCRAM- SHA-1 (and SCRAM-SHA-256 for version 4.0). SCRAM is based on Internet Engineering Task Force(IETF) that defines best practices for authenticating users with passwords [7]. SCRAM makes use of the provided credentials to match with user’s name, password and authentication database.

Another mechanism supported by MongoDB is **x.509 Certificate Authentication**. This mechanism verifies an organization/user by its valid certificate signed by a single certificate authority. A single Certificate Authority (CA) issues the certificate to both client as well as server. It must contain the following [8]:

```

keyUsage = digitalSignature
extendedKeyUsage = clientAuth
    
```

In this case, each MongoDB user must have a unique certificate. MongoDB also supports LDAP Proxy Authentication and Kerberos Authentication [6].

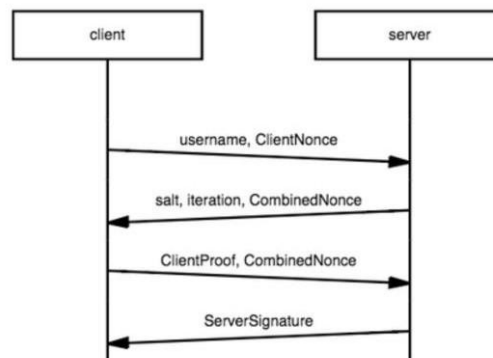


Figure 5: Sample set of messages exchanged during authentication [9]

b) Authorization

Authorization is the next step to authentication. Now that we have the verification of identity of the user in place, each user can be identified with pre-defined roles. Based on these roles, the access and privileges can be assigned. For e.g., a user who needs to only build reports can be given the read access whereas a user who needs the access to read as well as write the data can be given both accesses. Database administrators have the maximum capacity of operations and they can define roles, accesses to those roles, etc. It is a good practice for database admins to keep control of all roles and accesses so that the system has better protection from hack attacks.

There are two types of roles, **built-in** and **user-defined**. Built-in roles include readWrite, dbAdmin, dbOwner, userAdmin etc. and user-defined roles are defined by administrator as mentioned above. Access control is not enabled by default and needs to be enabled by using **security, authorization** setting [10].

c) Encryption

Encryption of data is performed in order to protect from attacks. MongoDB encrypts its **communication** using TLS/SSL protocol for all incoming and outgoing connections, using certificates and public-private key pairs. This is also called **Data- in-motion Encryption**.

For e.g., this encryption can be seen in Payment Credit Card Industry (PCI) requires that credit card numbers be encrypted in storage.

To encrypt and protect **data**, MongoDB uses AES256-CBC symmetric key encryption. This is called **Data-at-rest Encryption**. MongoDB also provides an option to turn encryption on in “FIPS mode” that means that the encryption we use is tested against National Institute of Standards and Technology Federal Information Processing Standard (NIST

FIPS) [11].

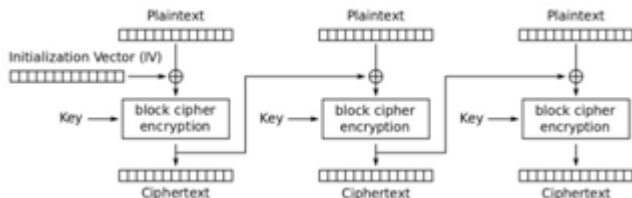


Figure 6: AES256-CBC Encryption [12]

d) Auditing

Auditing refers to the ability to see who did what in the database and is one of the most important pieces of security. MongoDB includes a system auditing facility that keeps a track of events like access, configuration changes, user operations, connection events etc. on an instance. This is really use in capturing suspicious activity.

For e.g., if an attacker tries to log into an instance to impact the data but has made multiple failed authentication attempts, analyzing the system event of failed login can help in identifying this malicious attack.

Another important aspect of auditing is analyzing the performance of the database while carrying out DDL & CRUD operations, authentication attempts, authorization changes and replica set & sharded cluster operations [13]. The way MongoDB does this is by keeping track of the timestamps. By default, it ignores certain operations and completely ignores their logging. Although, it is recommended that all the system events are logged so that if anything goes wrong, we have a complete traceback for the event.

The auditing system writes every audit event to an in-memory buffer and periodically, MongoDB writes this buffer to the disk. For the events of a single connection, if MongoDB writes one event to the disk, it is guaranteed that all the preceding events have also been written to the disk [13]. This is called the **Audit Guarantee**.

A word of caution is that the DB may lose the events if the server shuts down before it commits the events to the audit log. The user might receive a confirmation of successful completion of the operation, but its logs might be lost.

For example, while auditing an aggregation operation, the server might crash after returning the result but before the audit log flushes. To maintain durability, all DDL operations are written to the disk immediately [13].

e) Network exposure

MongoDB's implementation should be done to sure it runs in a trusted network environment. The number of instances available for incoming connections should also be limited and should only allow trusted clients to access the interface and ports [14].

For the earlier versions of MongoDB (2.6 to 3.4), only the binaries from MongoDB RPM and DEB would bind to localhost by default. From version 3.6, MongoDB, mongos and mongod, bind to localhost by default. This is good to start with, but in practical sense, this implementation is not

common as DBs need to be accessed remotely. So before binding to a publicly accessible IP address, security of the cluster against unauthorized access should be ensured.

Once the instance has been hosted, network hardening techniques of Firewall and Virtual Private Networks can be implemented for better security. **Firewall** helps system administrators to have a granular control over network communication which helps in limiting incoming traffic on a specific port and limiting incoming traffic from untrusted hosts. On linux systems, **iptables** interface is used and on windows system, **netsh** command line interface is used.

This goes without saying that for best results, administrator must ensure traffic only from trusted sources and connection to only trusted outputs [15].

VPN is used to link two networks over an encrypted and limited-access trusted network. The protocol typically used for MongoDB is TLS/SSL which has better performance than IPSEC VPNs. VPNs provide certificate validations and choice of encryption protocols, which requires authentication and authorization as well. Since VPN provides a secure tunnel, we can prevent tampering and man-in-the-middle attacks using access control.

4. Comparison of MongoDB with relational databases

a) Integrity

Both of the solutions have integrated complete logging but only in Relational Databases it is activated by default. Transactions and rollbacks maintain the consistency in the relational databases better. MongoDB trades this consistency with higher availability by supporting unacknowledged writes [16].

b) Encryption

Relational Databases natively support encryption using industry standard algorithms such as DES and AES whereas MongoDB supports TLS/SSL and AE 256-CBC as mentioned in the earlier sections.

c) Access Control

Relational Databases have a privilege- based access control whereas MongoDB has role-based access control.

5. MongoDB Security Flaws

a) MongoDB Data Files

One of the most important flaws of MongoDB is that the data files are unencrypted and there is not method to automatically encrypt these files. This means that if any attacker has access to the file system, they can directly extract the information [16].

b) Weak Authentication:

By default, there are no password credentials when the DB is installed. It is left to the developers to build-in the security. This weak authentication between clients and servers had led to a data breach in 2017 which saw about 30,000. MongoDB instances exposed. This had primarily happened due to a

default setting not being changed by the user [17]

c) Authorization

A user, when created, has a read-only access to all the data available in the WHOLE database. This is very insecure as any user will have the access to read the data even when they aren't supposed to.

d) Potential for attacks

One of the attacks that MongoDB is prone to is **JavaScript Injection attacks**. Since MongoDB primarily uses JavaScript for internal scripting, a hacker can hide a JavaScript code into a MongoDB query and corrupting the database by, say, running the query multiple times instead of just once [17]. For e.g., a user performs an operation of a simple addition. But since the code is corrupted with the injection, instead of performing the addition once, it will be performed multiple times which will result in wrong output and log values.

Another attack that is possible is **HTTP trespassing**. This type of attack is executed by making alterations to the source code of a website. Once the hacker has information about the MongoDB database, like DB name, collection name, port, username etc., they can easily attack and corrupt the database [17].

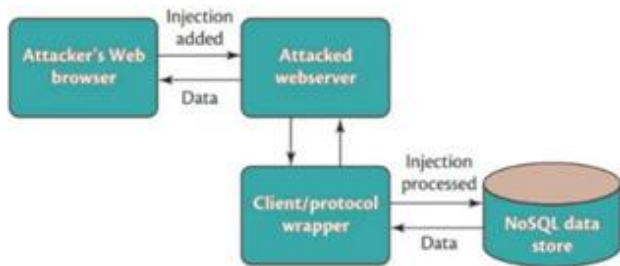


Figure 7: Conceptual design of an injection attack [18]

6. Security Checklist to Address Concerns

MongoDB provides a list of practices that can be followed in order to maintain a good security stature. The recommended actions/best practices are follows:

- Enable access control
- Configure Role-based Access
- Encrypt network traffic
- Use data-at-rest encryption
- Configure system auditing
- Stay up to date with fixes
- Disable JavaScript execution
- Know your framework
- Server hardening practices

Apart from these best practices provided by MongoDB documentation, MongoDB could really benefit from defaulting to an effective and less costly **authentication**. Currently, the authentication uses expensive hashing algorithms

Another important step that can be taken to get rid of the flaw of encryption is to implement **application level encryption**, independent of the instances. This should be a

built-in encryption and shouldn't be in the hands of the developers. If all the fields are encrypted at every single step, it will achieve a considerably high grade of security [18].

7. Conclusion

At a glance, from all the security features provided by MongoDB, it might seem like it has high security. To an extent it does have good security features but the two major areas that it needs to improve upon are the authentication and encryption of in-motion and at-rest data.

MongoDB should promote the best practices of enabling all the security features, as mentioned above in various section, so that an incident like the breach and hacks that occurred in 2017 are not repeated.

Another good way ahead could be enabling all security features by default and let the user "opt-out" of the ones that are not applicable to their organizations. Instead of going for "opting-in", this approach ensures more security. Another advantage of doing this would be that the administrators will become aware of the different security features available to them[18].

Reiterating the fact, MongoDB has security features that can be really beneficial, but it still has a long way to go to strike the right balance.

References

- [1] Violino, Bob. "How to Choose the Right NoSQL Database." InfoWorld. InfoWorld, March 9, 2018. <https://www.infoworld.com/article/3260184/how-to-choose-the-right-nosql-database.html>.
- [2] "(PDF) Security Issues in NoSQL Databases - Researchgate.net." Accessed December 2, 2019. https://www.researchgate.net/publication/254018091_Security_Issues_in_NoSQL_Databases.
- [3] "Introduction to MongoDB." Introduction to MongoDB - MongoDB Manual. Accessed December 2, 2019. <https://docs.mongodb.com/manual/introduction/>.
- [4] Bukhsh, Imran Omar BukhshImran Omar. "MySQL vs MongoDB 1000 Reads." Stack Overflow, May 1, 1962. <https://stackoverflow.com/questions/9702643/mysql-vs-mongodb-1000-reads>.
- [5] "Replication." Replication - MongoDB Manual. Accessed December 2, 2019. <https://docs.mongodb.com/manual/replication/>.
- [6] "Authentication." Authentication - MongoDB Manual. Accessed December 2, 2019. <https://docs.mongodb.com/manual/core/authentication/>.
- [7] "SCRAM." SCRAM - MongoDB Manual. Accessed December 2, 2019. <https://docs.mongodb.com/manual/core/security-scam/>.
- [8] "X.509." x.509 - MongoDB Manual. Accessed December 2, 2019. <https://docs.mongodb.com/manual/core/security-x.509/>.
- [9] Person. "Improved Password-Based Authentication in MongoDB 3.0: SCRAM Explained - Pt. 1: MongoDB Blog." MongoDB. MongoDB, February 2, 2015.

- <https://www.mongodb.com/blog/post/improved-password-based-authentication-mongodb-30-scram-explained-part-1>.
- [10] "Role-Based Access Control." Role-Based Access Control - MongoDB Manual. Accessed December 2, 2019.
<https://docs.mongodb.com/manual/core/authorization/>.
- [11] Townsend Security. "The Definitive Guide to Encryption Key Management Fundamentals." Blog. Accessed December 2, 2019.
<https://info.townsendsecurity.com/definitive-guide-to-encryption-key-management-fundamentals>.
- [12] "Block Cipher Modes of Operation." Wikipedia. Wikimedia Foundation, September 11, 2013.
https://en.wikipedia.org/wiki/Block_cipher_modes_of_operation.
- [13] "Auditing." Auditing - MongoDB Manual. Accessed December 2, 2019.
<https://docs.mongodb.com/manual/core/auditing/>.
- [14] "Security Checklist." Security Checklist - MongoDB Manual. Accessed December 2, 2019.
<https://docs.mongodb.com/manual/administration/security-checklist/>.
- [15] "Network and Configuration Hardening." Network and Configuration Hardening - MongoDB Manual. Accessed December 2, 2019.
<https://docs.mongodb.com/manual/core/security-hardening/>.
- [16] "An Analysis and Overview of MongoDB Security", <https://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/07/Paper.pdf>
- [17] "A Security Comparison between MySQL and MongoDB", https://www.academia.edu/16557343/Mongo_vs_MySQL-Security
- [18] "Possible Mitigation of NoSQL database Injections", <https://www.stjoern.com/menu-db/nosql>