Security Challenges and Solutions in MongoDB

Alankrit Chaturvedi

Carnegie Mellon University, Pittsburgh, PA, 15213

Abstract: ***The data present in this world is in bulk and in order to provide flawless performance along with two of the major things in the world of database, i.e. consistency and availability, it is difficult to achieve in RDBMS. To tackle this increase in data and work towards getting better performance in applications these days, NoSQL databases were introduced. There are different types of NoSQL databases that are built upon the concepts of documents, key-value pair, graphs etc. This paper talks about one such NoSQL database called MongoDB which is a based on a document store principle. It is widely used document store database in industries [11]. Despite being performance rich, there are some security flaws when adapting the database and those flaws are discussed in this paper in detail and are compared with traditional RDBMS system. Previous attacks on MongoDB, along with how to avoid such attacks are explained in detail. Some suggestions to work upon in order to improve the security in MongoDB is also suggested in this paper.***

Keywords: MongoDB; RDBMS; Security; MongoDB Security Features; Attack; Best Practices

1. Introduction

With the increase in technology and people being exposed to web, the volume of data has increased. The data varies from being about users themselves, their preferences, products for a retail industry, events, locations and the list goes on. The use of data has significantly increased with the accessibility of applications, systems, mobile phones, computers etc. Data is accessed every second of the day by millions of users worldwide and in order to accommodate that, processing has increased intensively. It is not always the work of the processor to increase the performance, but performance can be enhanced by the way data is stored and retrieved. Efficient systems worldwide are adopting new ways to discover performance enhancement and provide users with a fast experience.

Most widely used databases these days are NoSQL databases that have redefined the way data was stored in traditional RDBMS systems. Their primary advantage is that, the way they handle data is unstructured i.e. documents, key-value pairs, multimedia etc. NoSQL databases such as MongoDB that stores data as a form of documents, stores data which is highly scalable, provides better performance and is designed to process significant amount of unstructured data at a speed 10 times faster than traditional RDBMS has high availability too and strong fail over capabilities.

However, when we talk about security, MongoDB is very weak. Authentication and Encryption does not exist and is very weak when it is implemented.

Security with data, is the most essential part of any system, as there is a lot of user related sensitive information stored in the databases and raises the concern of confidentiality and privacy of the data and security provided by these systems.

In this paper, we review the main security features of MongoDB, brief overview of the database functionality and discuss how these security flaws can be improved

2. Overview of MongoDB

2.1 What is MongoDB?

MongoDB is a schema less database which is built in C++ programming language and is widely based on the concept of document-oriented databases. By document like database, it means that it manages data (also called collections) in JSON like documents. The advantage of this is data can be stored in the same schema without having to traverse through varied tables as done in RDMBS or simply using less "JOIN" operations. Complex data can be stored in nested hierarchies and still be query-able and index-able. Every collection has attributes pertaining to the requirement and user specific. [2]

2.2 Features of MongoDB

Following are the features of MongoDB: [2]

1) Data Model: A collection is equivalent to a table if we compare to traditional RDBMS and a collection stores sets of documents. Documents are equivalent to set of fields. Every document can be attributed to a row in a collection. Any document can store static string data or complex data structures like lists or even other documents which makes it faster to lookup in case of extended design and embedded design. Every document has a system generated '_id' and can be queried using that or any attribute within a document.
2) API: MongoDB uses a RESTful API which means it uses HTTP requests to post, read data and delete data. To retrieve certain documents from a database collection, a *Mongo Query language* is used. For example, to retrieve certain documents query like *{name: {first: Alankrit, last: Chaturvedi}}*. It is understood that if we are querying like the above example, we have the fields 'name', 'first', and 'last'.
3) Architecture: A MongoDB cluster is made of one or more shards. By shards it means that a part that holds portion of the total data. Sharding is managed automatically and is backed with a replica set which holds the data set. If the primary server goes down, it is backed up with a secondary server and thus provides consistency. To dig deeper, all writes, and consistent reads go to the primary server, and all consistent reads are distributed among the secondary servers. The data is distributed, and the cluster

contains a group of servers called configuration servers. Each server holds a copy of the meta-data indicating the data that lives on each shard.[1]

4) There are two types of replication functionality in MongoDB, namely Master-Slave and Replica-set. "In both types of replication, the write operations are performed for a single server (Master or Primary). Replica-Sets are known to provide better flexibility, allowing automatic primary promotion (if enough of the secondary servers are available), automatic fail-over and better support for rolling upgrades [1]. Both techniques provide data-redundancy and read-scaling (where data can be read from any of the servers in the cluster), however, in Master-Slave configuration, if a slave is too far behind from the master, then the administrator has to manually fix this, usually by fixing or rebuilding the slave instance".

"The last piece of the MongoDB puzzle is its ability to automatically shard the data between multiple hosts. This effectively allows Mongo to scale horizontally to thousands of servers. When sharding is combined with replica-sets, the end-result is a highly scalable, redundant cluster, with no single point of failure" [1].

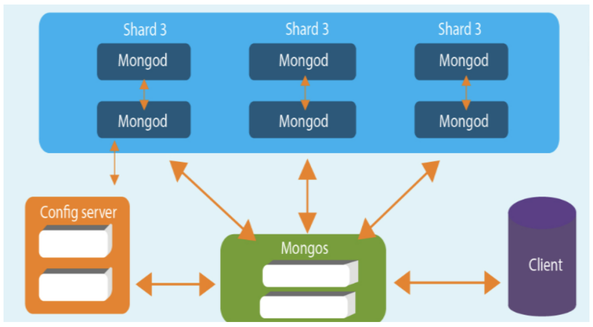


Figure 1: MongoDB Architecture [6]

2.3 Comparison Between RDBMS and MongoDB

The major difference between a traditional RDBMS and MongoDB system deals with normalization of data, In RDBMS, due to flexibility of JOIN operations, we have normalized data i.e. ACID transactions support. This is done for better performance, removal of duplicates and consistency of data. However, when we talk about a document store, MongoDB database keeps denormalized data in the form of embedded documents, which in turn helps in performance, how? that's because it stores the data as required by the end user. [5]

A detailed comparison between traditional RDBMS system (MySQL) and MongoDB is described below.

Table I: Comparing MySQL(RDBMS) and MongoDB [4]

Date	MySQL	MongoDB
Written in	C++, C	C++, C and JavaScript
Type	RDBMS	Document-oriented
Main points	- Table - Row - Column	- Collection - Document - Field
License	GPL v2 / Commercial licenses available OD	GNU AGPL v3.0 / Commercial licenses available OD
Schemas	Strict	Dynamic
Scaling	Vertically	Horizontally
Key features	- Full-text searching and indexing - Integrated replication support - Triggers - SubSELECTs - Query caching - SSL support - Unicode support - Different storage engines with various performance characteristics	- Auto-sharding - Native replication - In-memory speed - Embedded data models support - Comprehensive secondary indexes - Rich query language support - Various storage engines support
Best used for	- Data structure fits for tables and rows - Strong dependence on multi-row transactions - Frequent updates and modifications of large volume of records - Relatively small datasets	- High write loads - Unstable schema - Your DB is set to grow big - Data is location based - HA (high availability) in unstable environment is required - No database administrators (DBAs)
Examples	NASA, US Navy, Bank of Finland, UCR, Walmart, Sony, S2 Security Corporation, Telenor, Italtel, iStock, Uber, Zappos, Booking.com, Twitter, Facebook, others.	Expedia, Bosch, Otto, eBay, Gap, Forbes, Foursquare, Adobe, Intuit, Metlife, BuzzFeed, Crittercism, CitiGroup, the City of Chicago, others.

2.4 MongoDB Security Features

The official website of MongoDB says "One valid way to run the Mongo database is in a trusted environment, with no security and authentication. ... Of course, in such a configuration, one must be sure only trusted machines can access database TCP ports." [1]

Security was not a main concern when the developers were building MongoDB. As discussed in the introduction, MongoDB is still open to attacks related to user data. Attacks like insecure connections (not https) and not enough authentication support. Currently, most systems build the security phase in the middleware or interaction layer and no security on Mongo cluster level.

MongoDB security can be challenged in many areas. Use of default ports, lack of authentication control, broad access to users in authentication, lack of use of LDAP, not using SSL and not limiting database access to known network devices.

An overview of the categories is mentioned below and in detail in the following read: [3]

Table II: Security Overview of MongoDB with Recommendations

Category	Status	Recommendation
Data at rest	Unencrypted	Protect with OS level mechanisms.
Authentication for native connections	Available only in unsharded configurations.	Enable if possible.
Authorization for native connections	READ/READ-WRITE/Admin levels, only in unsharded configurations.	Enable if possible, requires enabled authentication.
Auditing	Not available in MongoDB	
AAA (authentication, authorization auditing) for RESTful connections	Users and permissions are maintained externally.	Available if configured on a reverse proxy
Database Communication	Encryption is not available	
Injection attacks	Possible, via JavaScript or string concatenation.	Verify that the application does reasonable input validation.

a) MongoDB Data Files

There is no automatic encryption when it comes to MongoDB data files. Attackers can access the files/information directly once they are exposed to the database. In order to mitigate this, it is up to the application to encrypt sensitive information, for example, encoding user password with base64 encoding or encoding using MD5 for certain sensitive data.

Network encryption should be enforced in order to use encoding. Configuring TLS layer will encrypt the communication to and from the database. With operating system security in place i.e. firewalls, even after that, all the communication should still use TLS because if some attacker is able to breach the firewall, they can breach the database too. Encryption of data at rest is only available in Enterprise edition of MongoDB.

b) Client Interfaces

By default, MongoDB does not support SSL client node communication. This leads to security breach in the network. “To use SSL, it is required recompile whole MongoDB with the “-sl" option or deploy MongoDB enterprise version. Additional steps to generate keys are needed for configuring client/server for SSL communication”.[1]

Mongo supports a binary wire-level protocol, using TCP port 27017 by default. This protocol is used by various drivers and is considered the most efficient way of communicating with Mongo. In addition to the drivers for application, MongoDB uses this port and protocol in order to perform replications (both variants). Also, the port number that is 1000 more than the binary client port is used as a HTTP server (TCP port 28017). The HTTP server provides management level statistics and can also be configured to provide a RESTful interface to the database, by adding *rest=true* to the database configuration file or using command line. [7]

The binary wire-level protocol is neither compressed nor encrypted, and the HTTP server doesn’t support SSL or TLS in any way. The internal HTTP server can be hidden behind a HTTP proxy server, like Apache HTTPD with the reverse proxy mechanism, and then the Apache HTTPD’s robust authentication and authorization support can be used, in addition to SSL encryption for the connection.

c) Injection attacks potential

The main utility language used by MongoDB is JavaScript. JavaScript can be used to store data in the MongoDB database and are available to the database users. JavaScript being a scripted language, has a potential for injection attacks.

For example, the following statements can be all used to perform the same query in MongoDB:

```
db.collection.find(
{ index : { $gt: 2 } } );
db.collection.find(
{ $where: "this.index > 2" } );
db.collection.find(
"this.index > 2" );
db.collection.find(
{ $where: function() {
return this.index > 2 ;} } );
```

The above statements are prone to injection attacks especially the where clause. It evaluates each records in the collection. So, any manipulation in this type of query can lead to injection attacks.

d) Authentication

When sharding happens, authentication is not supported in MongoDB. Authentication can only be enabled in Mongo in standalone or replica-set mode. Thus, basic MongoDB does provide support for authentication on a single database level. On the other hand, MongoDB enterprise version (the paid version) adds an additional Kerberos service for authentication.

e) Authorization

There are limited roles in MongoDB namely, read, readWrite, readAnyDatabase, readWriteAnyDatabase, userAdmin, clusterAdmin, userAdminAnyDatabase, dbAdmin, and dbaAdminAnyDatabase. [1]

In MongoDB, since there is no support for authentication, there is no support for authorization.

f) Auditing

Auditing in MongoDB only happens when a new database(namespace) is created. There is a line in the log about the creation but there will not be logs about subsequent operations like updates or queries.

There is an HTTP console for information about systems and clients for each MongoDB instance. If there is no security i.e. there is no authorization, there is a potential threat for an attacker to leak the data. This can be rectified by implementing authorization feature using Kerberos of MongoDB enterprise.

2.5 Previous Attacks On MongoDB

The following attacks have been reported in high frequency by majority of the organizations that use MongoDB as a service. [1]

a) *Injection Attack:* A potential attack as discussed earlier was JavaScript injection attacks. Turns out, organizations have been victim of these attacks. MongoDB API works with BSON (Binary JSON) calls and includes a BSON query assembly tool. However, JavaScript expressions and un-serialized JSON are allowed in several query parameters that has made it vulnerable.
b) *DoS Attack:* For any attacker to fetch data from MongoDB, if they get their hands-on valid user credentials, they don't have to be an admin to carry out attacks, since there is no authentication and authorization.
c) *XSS Attacks:* If attackers insert random JavaScript code, they have been successful in attacking and stealing sensitive information. This is because MongoDB allows scripting using JavaScript for the database layer and for the client to fetch data.

The following is a way to explain how attacks incur in a MongoDB database:

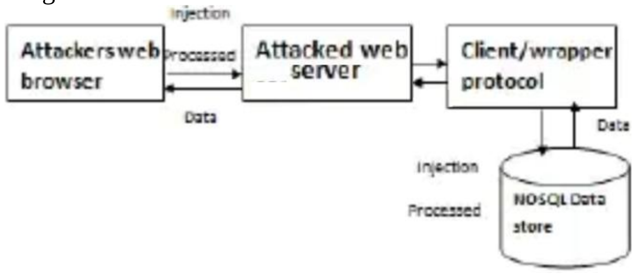


Figure 2: Attack on MongoDB [7]

Few of the companies have been exposed to MongoDB breach such as:

Shodan [8] – On 10th May 2019, personal data of 275m users including name, gender, email, address, employment history, current salary, current employer, phone number etc. were exposed. This happened due to public facing database and lack of authentication.

Pipl [9] – On June 18th 2019, 188m records of users were compromised. This was again due to lack of authentication.

In addition to this, about 20 percent of the data on MongoDB was wiped out and held for ransom by the Unistellar hacking group. About 12564 databases were sabotaged. [10]

All the issues can be rectified in any organization and many practices have been introduced to protect data.

2.6 Solutions to MongoDB Security and Best Practices

With the following steps in place, MongoDB security can be enhanced, without switching to the Enterprise version. The below mentioned steps are employed by most of the companies and works in reducing most of the above-mentioned attacks. [7]

a) Authentication:

By either using SCRAM-SHA-1 or MongoDB-CR. SCRAM-SHA-1 is *Salted Challenge Authentication Mechanism* [7] that uses a simple text-based user credentials transmitted over a channel and is layered by transport layer security (TLS). Like SCRAM, MongoDB-CR verifies user credentials against an authentication database.

External protocols can be employed to MongoDB as well like LDAP and Kerberos. LDAP uses the concept of providing centralized passwords and is designed to help anyone to locate information needed in a public or private network. Kerberos is a secret key authentication protocol in which a shared key is used for encryption for server-client interactions.

b) Authorization/Role based security:

In addition to defining roles that covers most of the users, custom roles can be created in MongoDB based on requirements. Enabling authorization is done using the '–auth' command which controls a user's access to the database and upon every access, the identities are verified.

c) TLS/SSL encryption:

After Mongo 2.6, SSL and TLS are supported by x.509 certificates [5]. Clients can use the certificates to authenticate users instead of using usernames and passwords. While MongoDB can use a valid certificate from a trusted server, self-signed certificates are best not considered because it may lead in not proper verification of the server identity and also in order to avoid man-in-the-middle attacks.

d) Hardening the MongoDB database:

Hardening means adding security layer by layer. There can be many hardening ways in MongoDB. The most essential is network hardening with firewalls and VPNs.

Few of the best practices for MongoDB are [4] –

- Enabling access control and using any of the authentication mechanisms mentioned. Each instance of the cluster should be individually configured.
- Administrator user should always be created first. Additional users can be added as per usage.
- All communications between mongos and mongod instances should be encrypted using TLS/SSL.
- MongoDB should be run in a trusted network. Database should not be allowed to be routable to a public network even if it is residing inside a private network. Interfaces should be limited as this does not allow a bad actor to move the data.
- Track data movement and changes using auditing.

- Understanding roles and assigning correct privileges.
- Creating a user specific to the use case of the application. For example – for a user to run an application, the user should be provided with least privileges and for a user to run analytics, a user with read only access should be given. There is a clear isolation and separation of roles in this case.
- Use IP filtering to provide better access to the people using the environment.

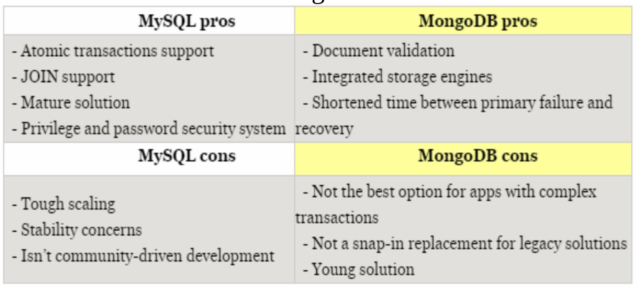
2.7 MongoDB vs RDBMS Security

For any database to be secure, it needs to provide confidentiality, availability and integrity (CIA). Relational Databases has security that includes integrated features like role-based security, access control by user-level permissions on stored procedures. RDBMS is built on ACID (Atomicity, Consistency, Isolation, Durability) transaction properties that means that database transactions are processed with data integrity, data logging and data consistency. Replication in RDBMS ensures durability and data integrity. However, to use these features, users have to pay a cost, which is for license and speed to access data [5].

Giants like Google and Facebook, which deal with continuous large data sets, availability and scalability are the key requirements they look for. In order to distribute the systems across hundreds of servers, they have adopted the use of MongoDB database, however their security is implemented in the middleware layers.

Basically, security in NoSQL database is nowhere as robust as relational databases.

Table III: Pros and Cons of MySQL(RDBMS) and MongoDB



MySQL pros	MongoDB pros
- Atomic transactions support - JOIN support - Mature solution - Privilege and password security system	- Document validation - Integrated storage engines - Shortened time between primary failure and recovery
MySQL cons	**MongoDB cons**
- Tough scaling - Stability concerns - Isn't community-driven development	- Not the best option for apps with complex transactions - Not a snap-in replacement for legacy solutions - Young solution

2.8 Which Database to Choose?

RDBMS has been the go-to solution for most of the companies worldwide. Users have reportedly complained about the less collaborative, helpful community of MySQL (RDBMS) compared to MongoDB, after being acquired by Oracle. [9] Another issue with RDBMS is owner's direction towards MariaDB development and therefore ignoring patches and providing a sustainability plan.

Comparing MongoDB speed vs MySQL, developers note that the MySQL or any RDBMS system lacks speed and experience difficulties with large data volumes[7], so it'll be a better choice for companies with fixed and concrete schema with small scale data or small amount of data, basically companies that are looking for more general solution. MongoDB has one big advantage over RDBMS i.e. the ability to cope with large, unstructured and varied amounts of data [6].

The main question is "when to use MongoDB instead of RDBMS?". In order to answer this, developers need to take into account the requirements of the project, the company and further goals. A traditional RDBMS system is recognized for its flexibility, high performance, data protection, high availability, and ease of management. The concept of indexing can even help in performance issues, help in interaction and can ensure robustness. MongoDB is a better option when the data is complex and unstructured, the schema is not pre-defined. Also, if there is a need to store large volumes of data and in a document format, MongoDB will help in storing and retrieving the data in a much efficient manner.

3. Conclusion

We reviewed the problems for MongoDB database security that includes lack of encryption for data files, weak support for authentication between client and servers, weak authorization (except the Enterprise version), vulnerability to injection attacks etc. Currently, with the middleware layer in place and encryption, authentication and authorization happening at the middleware layer, this problem is under control however, there is a need for considerable hardening and development in order to provide an environment that is secure to store sensitive data without relying on middleware layers to perform the task and also to get the robustness like that of relational databases. Many options mentioned in the paper like use of Kerberos, LDAP, role-based authorization, use of SSL etc. are being used in the industry and there is scope to explore many more options with the enhancement of technology.

References

[1] Hossain Shahriar, and Hisham M. Haddad, "Security Vulnerabilities of NoSQL and SQL Databases for MOOC Applications," International Journal of Digital Society (IJDS), Volume 8, Issue 1, March 2017
[2] MongoDB Security Concept. [Online]. Accessed from http://docs.mongodb.org/master/core/security/
[3] "Security Checklist." Security Checklist - MongoDB Manual. Accessed from https://docs.mongodb.com/manual/administration/security-checklist/.
[4] David Murphy. "The Essential Guide to MongoDB Security." InfoWorld. InfoWorld, February 2, 2017. https://www.infoworld.com/article/3164504/the-essential-guide-to-mongodb-security.html
[5] Cobb, Michael. "Comparing Relational Database Security and NoSQL Security." SearchSecurity. Accessed November 18, 2019. https://searchsecurity.techtarget.com/answer/Comparing-relational-database-security-and-NoSQL-security.
[6] "MongoDB vs MySQL Comparison: Which Database Is Better?" By Eugeniya. Accessed November 30, 2019. https://hackernoon.com/mongodb-vs-mysql-comparison-which-database-is-better-e714b699c38b.
[7] International Journal of Scientific Research. "Analysis on Database Security Model Against NOSQL

Injection." Academia.edu. https://www.academia.edu/33112210/Analysis_on_Database_Security_Model_Against_NOSQL_Injection.

[8] Bradbury, Danny, Danny Bradbury, Hydra, Paul Ducklin, Mahhn, John E Dunn, and Danny Bradbury. "275m Personal Records Swiped from Exposed MongoDB Database." Naked Security, May 10, 2019. https://nakedsecurity.sophos.com/2019/05/10/275m-indian-citizens-records-exposed-by-insecure-mongodb-database/.

[9] Asokan, Akshaya, and Ron Ross. "MongoDB Database Exposed 188 Million Records: Researchers." Bank Information Security. Accessed November 30, 2019. https://www.bankinfosecurity.com/mongodb-database-exposed-188-million-records-researchers-a-12769

[10] Barth, Bradley. "Hacking Group Wipes Content from over 12,000 Open MongoDB Databases." SC Media, May 20, 2019. https://www.scmagazine.com/home/security-news/cybercrime/report-hacking-group-wipes-content-from-over-12000-open-mongodb-databases/.

[11] "DB-Engines Ranking." historical trend of document stores popularity. http://db-engines.com/en/ranking_trend/document store.