Security Analysis of MongoDB

Sahib Singh
Heinz College, Carnegie Mellon University
USA

Abstract

NoSQL Databases are a form of non-relational databases whose primary purpose is to store and retrieve data. Due to recent advancements in cloud computing platforms and the emergence of Big Data, NoSQL Databases are more becoming popular than ever. In this paper we are going to understand and analyze the fundamental security features and the vulnerabilities of MongoDB and how it performs compared to relational databases on these fronts.

1. Introduction

The term NoSQL was brought into light by "Carlos Strozzi" who used the term to identify his database which was a lightweight, open relational database which didn't use SQL. The term NoSQL refers to a system which is Not Only SQL rather than the earlier version referring it as a database without SQL using to it is SQL like query support system.

NoSQL models are generally quicker and are able to process large amounts of heterogenous data compared to relational database models hence often being the first choice while working with large and unstructured datasets owing to their speed and flexibility. Not only can they handle unstructured data they are also able to process Big Data quickly hence making them the first choice among top technology companies such as Facebook, Google, Twitter etc.

One type of NoSQL database which is going to be our primary focus for this paper will be MongoDB. MongoDB, written in C++, is an open source database which is currently the most popular NoSQL database according to DB-Engines tracking

The DB-Engines ranks over 340 database systems based on their popularity. This popularity score is generated by taking a number of factors into consideration such as search engine results, Google Trends, Stack Overflow discussions forums, the number of jobs available and profiles present in professional networks such as LinkedIn, and social networks like Twitter.

2. Features of MongoDB

MongoDB is a document-based database developed by 10gen which manages collection of JSON like documents format called BSON or simply "Binary JSON". Documents in MongoDB are stored in a collection which are stored in a database.

Some features of Mongo DB which are worth going over from a security standpoint are given below [5]:

1) Map reduce based Aggregation Framework: This feature of MongoDB is similar to the 'Group By' clause offered in MySQL. MongoDB uses Map-Reduce paradigm to perform aggregation. A map is basically a procedure for filtering and sorting data while reduce procedure performs a summary operation (Eg counting the number of people standing in a queue). MapReduce is generally used to processing large volume of data parallely by distributing it across clusters.

2) Schema Less Database: The schema refers to the structure in which the data should be stored. In the case of relational databases such schema is defined using tables. By schema-less we are referring to dynamically typed schema as opposed to statically typed schemas in Relational Databases. Eg XML allows you to specify XSD if required however BSON can accept a varied type of data. Since there is no constraint on the data and every document in the collection can have different attributes from each other we call it schema less.

3) Ad-hoc Querying: MongoDB supports SQL like complex queries including regex. Similarly, we can also write queries to fetch data less than or greater than a value or use regular expressions for pattern matching.

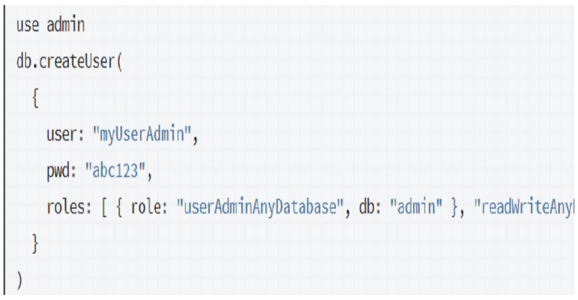
4) Replication and fail-over support: MongoDB supports replication by distributing data over various clusters, this is achieved using replica set which is essentially a group of instances hosting the same data. In a replica set, one node is defined as the primary node while all other nodes classify as secondary. All write operations are assigned to the primary node (i.e. the master node) whereas the secondary nodes may perform read operations.

3. Security Features

Following are some of the Key Security features provided by MongoDB with regards to authentication, encryption and access control. [4]

1) Enabling Access Control: MongoDB requires that all users (clients as well as servers) provide valid credentials before they are able to connect to the system. Enabling access control enforces authentication and requires all users to identify themselves before a connection is made. With Access Control enabled, there is a user administrator role defined which is responsible for creating users, granting and revoking access of other users as well as modifying user roles.

Following is an example on how to set up an admin provided in the official MongoDB documentation:



```
use admin
db.createUser(
  {
    user: "myUserAdmin",
    pwd: "abc123",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAny
  }
)
```

After the admin has been created, you can then go on to create additional users based on exact access principle (giving the least privilege required).

2) Confidential Network: MongoDB only allows users to connect over defined interfaces on a given port in which MongoDB instances are available. This is done to reduce the risk of exposure and ensure only trusted users have access. This network confidentiality can be achieved using the following two routes:

• IP Binding: Starting MongoDB 3.6, MongoDB binaries bind to localhost by default. The binary can additionally bind to other IPv6 addresses by setting them up through the command line interface or through the “net.bindIp” configuration file.

• Network Hardening: One way to achieve this is using Firewalls which limits traffic to only those from trusted sources. The other way is to use Virtual Private Networks which makes it possible to two networks over an encrypted and limited access network. Virtual Private Networks can also be used to prevent tampering and “man in the middle” attacks since they take place over a secure tunnel.

3) System Auditing: The System Auditing facility allows admins and users to track their systems activities during testing and deployment phases.

On enabling the Audit System can record the following information:

- Schema
- Replica Set
- Sharded Clusters
- Authorizations
- CRUD operations

Further the Auditing system writes every audit document to an in-memory buffer of audit events which later get written into disk.

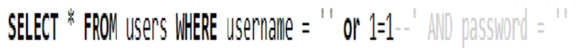
4. Security Flaws And Addressal

Following are some of the Security Issues found in MongoDB and we will always see if there are any ways to address these issues [8]:

1) Lack of Data Encryption
• Currently there is no encryption on data files in MongoDB. This is a cause of concern since anyone with access to file systems can extract the information from these files.
• To prevent any issues arising from such lack of encryption the application should explicitly encrypt all the sensitive information before writing it to their database also file permissions should be adequately put into place to prevent any unauthorized user from accessing them.

2) Vulnerable to Injection Attacks
• Simply because MongoDB does not deal directly with a query language in the form of string does not make it immune to injection attacks (See example below on how injection attacks take place in SQL). Injection attacks are still possible owing to MongoDB’s dependence on JavaScript. MongoDB’s operations allow arbitrary JavaScript expressions to be executed directly on the server.

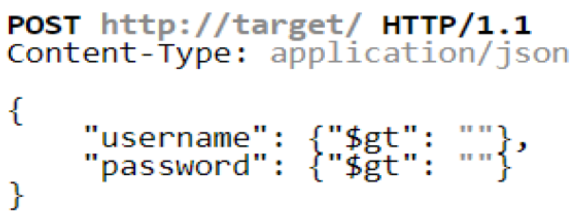
Classic SQL Injection



```
SELECT * FROM users WHERE username = '' or 1=1--' AND password = ''
```

The above example shows how SQL injections take place since 1=1 is always true, the above query will be executed, and attackers might be able to get access to private information (E.g. the above table might contain usernames and passwords.) Hence such statements might be unknowingly be running on your database.

MongoDB Injection



```
POST http://target/ HTTP/1.1
Content-Type: application/json

{
    "username": {"$gt": ""},
    "password": {"$gt": ""}
}
```

In the above example, username and password have not been validated to ensure they are strings hence they may contain any field but strings and manipulate the query structure.

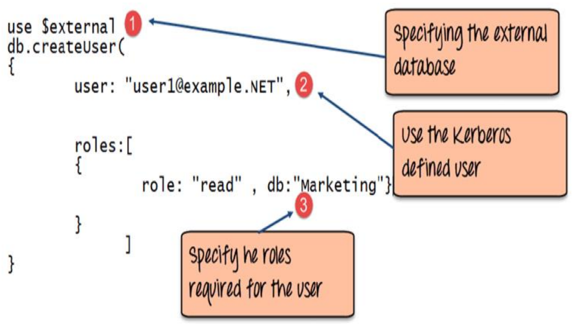
• Based on MongoDB's official documentation [4], user can express most queries in MongoDB without JavaScript and for the queries which do require JavaScript, user can mix JavaScript and Non JavaScript inside a single query by placing all the user-supplied fields directly in a BSON field and passing JavaScript code to the $where field.

3) Authentication & Authorization

• MongoDB does not provide authentication in sharded configuration unless run in standalone or replica set mode also the onus of security lies entirely in the hand of the developer. Any user by default has the permission to access the entire database, moreover any user with administrator access has complete read/write privileges for the complete database.

• A reverse proxy can be configured using REST API's to define fine grained permission adding to authentication.

MongoDB authentication with Kerberos



```
use $external
db.createUser(
{
        user: "user1@example.NET",

        roles:[
        {
                role: "read" , db:"Marketing"}

        }
                ]
}
```

1
Specifying the external database
2
Use the Kerberos defined user
3
Specify he roles required for the user

5. MongoDB Vs MySQL

When talking about Relational Databases, MySQL is one of the first database which usually comes to mind. It's a type of relational database which is currently owned by Oracle and is a part of the LAMP Stack (Linux/Apache/MySQL/php).

1) Structure

• MySQL stores data in the form of tables with rows and columns. It uses schema to defines it's database structure and requires all rows within the table to follow the same structure

• MongoDB stores data in JSON like documents and is schema free i.e. we don't have to define the structure first and different documents can have different types of data as earlier discussed.

2) Replication

• MySQL supports master-slave and master-master replications and allows multi-source replication.

• MongoDB has built in features for replication and sharding i.e. distributing data across multiple machines to support large scale deployments at scale.

3) Scaling

Horizontal scaling refers to adding more machines into our resource pool while Vertical Scaling means adding more power to an existing machine. Achieving horizontal scaling with MySQL often requires significant engineering efforts and resources and vertical scaling is often not possible beyond a limit. MongoDB can be scaled within and across multiple distributed data centers with high throughput and almost no downtime [4]. It also provides support for auto sharding and application unaware scaling.

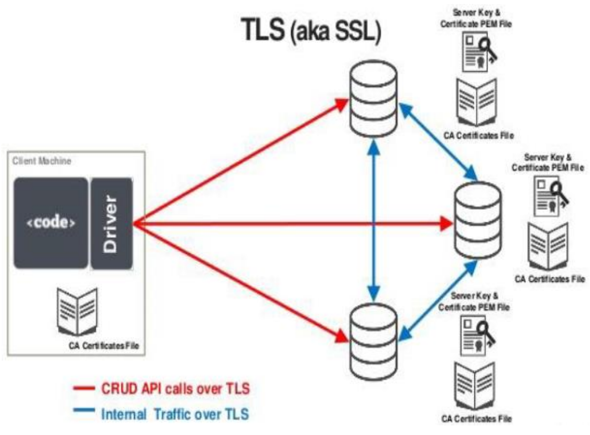
6. Mysql Vs Mongodb Security Comparsion

Following is a comparison done between MySQL and MongoDB from a security standpoint:

1) Security Model

• MySQL provides a privilege-based security model i.e. providing a user which access to only specific commands such as CREATE, UPDATE, DELETE etc. hence based on the user type such privileges can be defined.

• MongoDB supports TLS and SSL for encryption to ensure the data is only accessible to the intended user.



2) Injections
• MySQL is prone to SQL injections which is essentially placing malicious code in SQL statements via web page output.
• While MongoDB is not prone to SQL injections, it is not entirely error prone from injections (as discussed above) owing to the use of an interpretable language such as JavaScript.

3) Logging
• MySQL offers complete logging by default and supporting transaction and rollbacks helps in ensuring data integrity.
• Complete logging is not enabled by default in MongoDB. Additional logging is built into the operating system and application layers. [2]

4) Access Controls
• MySQL provides various types of access control mechanisms like Discretionary Access Control (REVOKE & GRANT) commands, Role Based Access Control etc.
• MongoDB only offers a role-based access control which is not enabled by default. It provides some built-in roles which provide a set of privileges commonly required in a database.

5) Integrity Model
• MySQL follows ACID (Atomic, Consistent, Isolated, Durable) model. A relational database not following any of these four goals is not considered reliable. Database administrators use several strategies to enforce ACID such as write ahead logging (WAL), shadow paging and two-phase commit protocol.
• MongoDB follows the BASE (Basic Availability, Soft state, Eventual consistency) model. With the release of MongoDB 4.0 we now have multi document ACID transaction support. Through snapshot isolation, transactions provide a consistent view of data while enforcing all or nothing execution and maintaining data integrity [4].

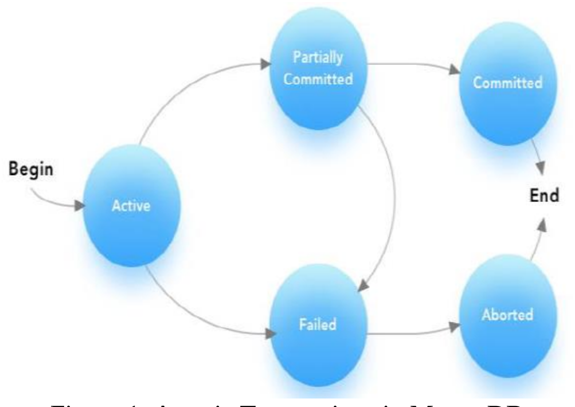


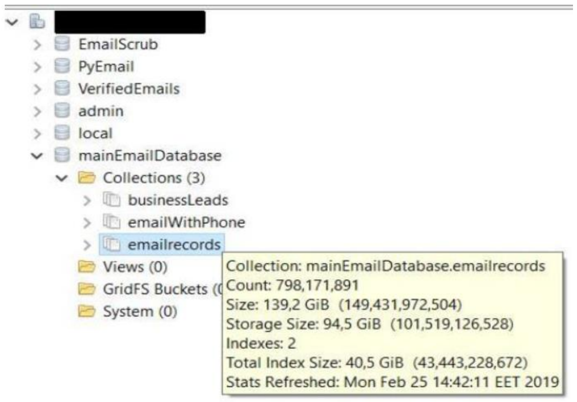
Figure 1. Atomic Transactions in MongoDB (Source: SimForm.com)

7. Data Breach Cases

• A security researcher named "Bob Diachenko" exposed a vulnerability in which was leaking the details of around 11 million users belonging to an email marketing firm based out of California. The dataset contained around 44 Gigabytes of data including full names, email addresses, gender details and physical addresses of 10,999,535 users. Apart from these details DNS details as well as email delivery status information was also found. According to the researcher, Bob, the database had been left exposed since at least 13th September 2019. Based on reports the data belonged to a coupon or discount-based company called "Saverspy", a daily deals website operated by Coupons.com [7]. The database also contained a ransom message demanding bitcoin payment to recover lost data. The message further asked to send along IP address and proof of payment in order to get the data. Similar ransom messages were also reported in China around June this year.

• Earlier in December'18 another leak of data was exposed containing 854 Gigabytes of data without any authentication or password. The data contained the details of more than 200 million Chinese job seekers. The data, which was totally unprotected, was open and available for around a week's timeframe. The data instance was found using a BinaryEdge or Shodan search. Each of the 200 million resumes also contained personal information such as contact number, email address, height, weight, driver license, salary expectations etc. This was particularly dangerous since it could have led to follow on phishing attacks. While the source of the data remained, unknown there were speculations of it being scraped from 3rd party websites. The database was secured once the leak was reported.

• Security researchers discovered over 808 million records including sensitive information such as contact number and email address being exposed on a MongoDB instance. This 150GB exposed data instance was reported around February'19. The data comprised of three folders with about 800 million records in one (emailrecords), 4 million contact information record in another (emailWithPhone) and around 6 million records of business leads in the third folder (businessLeads) which included information such as mortgage details and other corporate information. The database was found out to be belonging to an email validation firm, Verifications.io, a company which approves and verifies email addresses for third parties. The database was aimed at sending out spam emails in bulk. Following Snippet shows some of the leaked details in this regard:



```
[redacted]
  EmailScrub
  PyEmail
  VerifiedEmails
  admin
  local
  mainEmailDatabase
    Collections (3)
      businessLeads
      emailWithPhone
      emailrecords
    Views (0)
    GridFS Buckets (0
    System (0)

Collection: mainEmailDatabase.emailrecords
Count: 798,171,891
Size: 139,2 GiB  (149,431,972,504)
Storage Size: 94,5 GiB  (101,519,126,528)
Indexes: 2
Total Index Size: 40,5 GiB  (43,443,228,672)
Stats Refreshed: Mon Feb 25 14:42:11 EET 2019
```

8. Common Vulnerabilities and Exposures

Some publicly known cybersecurity vulnerabilities, having at least one public reference are described below [5]:

• A privileged escalation was detected in FlintCms (A content management system) allowing takeover due to blind MongoDB injection during password reset.

• IBM API connect getting affected by a NoSQL injection in MongoDB connector for the LoopBack framework.

• The MongoDB Js-Bson module versions before 1.0.5 are vulnerable to Regular expression Denial of Service (ReDoS), the flaw being triggered on calling fromString() function to parse a long untrusted string.

• MongoDB earlier versions of 3.4.x before 3.4.10 has a disabled-by-default configuration setting exposing a vulnerability which when enabled could be exploited to deny service or modify memory by a malicious attacker.

• mongodb-instance before 0.0.3 installs MongoDB locally downloading binary resources over HTTP leaving it vulnerable to MITM attacks.

• The client in MongoDB use world readable permissions on .dbshells history files which risks allowing local users obtaining sensitive information by going through these files.

• Certain MongoDB versions between 2.4 and 2.6 provide vulnerabilities in security to allow a denial of service attack using UTF-8 string in a BSON based request.

• MongoDB earlier versions (between 2.10 and 2.20) do not properly validate requests to native helper function in SpiderMonkey allowing remote authenticated users to cause a denial of service attack or executing arbitrary code through a crafted memory address in the first segment.

• MongoDB dissector in Wireshark before 1.8.2 in 1.8.x is vulnerable to denial of service attacks through a small value for BSON document length.

• The default MongoDB configuration before 2.3.2 does not validate objects exposing remote authenticated users to read system memory through a crafted BSON object in column name inside an insert command, triggering a buffer over read.

9. Conclusion

Data is growing at a rapid pace and is becoming more and more unstructured, such as emails, audio files, and videos. As of today, more than 95% of the data generated is in unstructured format. Having the flexibility for development without any predefined schema is a massive boost for MongoDB however it is essential that it is backed by a robust security mechanism to prevent malicious attacks and unwanted interferences. Based on the findings above we can say that MySQL is usually better when you need reliable data protection and ease in data management. On the other hand, if you have unstructured data or an undefined schema in hand and want to process large amounts of data MongoDB would be a reasonable choice. From a security standpoint, we can see that MySQL being an old player has identified most of its loopholes and even though SQL injections are still present there are some decent workarounds. MongoDB on the other hand does not offer most security configurations by default however, it’s still less prone to injections attacks considering it does not directly deal with a query language in the form of string. However, there are still lots of vulnerabilities in MongoDB specially related to system crashing and denial of service attacks as we have seen in the Common Vulnerabilities and Exposure section. There is an expectation to improve these security flaws in the upcoming MongoDB versions to ensure deployment of robust and large-scale applications.

10. References

[1] Dave, M. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Science and Software Engineering.

[2] DB Engines, Database Management Ranking.

[3] Hou, B., Qian, K., Li, L., Shi, Y., Tao, L., Liu, J. (2016). MongoDB NoSQL Injection Analysis and Detection. 75-78. 10.1109/CSCloud.2016.57.

[4] MongoDB Official documentation.

[5] CVE Mitre official entries containing publicly known cybersecurity vulnerabilities.

[6] ThrearPost, Security News Portal.

[7] ZdNet, Security Website owned by CNET.

[8] Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., Abramov, J. (2011). Security Issues in NoSQL Databases. TRUSTCOM '11 Proceedings of the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, Pages 541-547.

[9] Shahriar, H., Haddad, H.M., Security Vulnerabilities of NoSQL and SQL Databases for MOOC Applications, Department of Information Technology, Department of Computer Science Kennesaw State University, USA.