EasyChair Preprint

№ 3792

Prototype of Security Framework System Under Big Data Challenges

Sliman El Abbadi, Abderrahim Marzouk, Adil Maarouf and Amine Benmakhlouf

July 8, 2020

Prototype of security framework system under Big data challenges

Sliman El Abbadi [1] and Abderrahim Marzouk [2]

Adil Maarouf [2] and Amine Benmakhlouf [2]

[1] Computer, Networks, Mobility and Modeling laboratory, FST, Hassan 1st University,
[2] Settat, Morocco
sliman.elabbadi@gmail.com, amarzouk2004@yahoo.fr,
benmakhloufamine@gmail.com
adil.maarouf@gmail.com

Abstract. With the incredible growth of Big data in real life, healthcare and industry, security of data is becoming even more critical layer concern sensitive data.
In this paper we propose a security framework based on the experiences acquired by carrying out vulnerability assessments of critical Big Data services. we summarize how to prevent problems by showing many types of vulnerabilities that occur in real code and what techniques can be used to prevent or mitigate them by use of some data masking techniques and Complying with Security drivers a special use of MongoDB as leader of NoSQL database

Keywords: Big Data, NoSQL, MongoDB, Data masking API, Data privacy.

1 Introduction

The layer of security system information has been kept relatively safe for many years. Now completely changed when the business needs push to integrate the Big Data into your information system (BI, Social Network, e-commerce, online banking, etc.) Go-to-market strategy means that you will have more opportunities to collect heterogeneous data (structured, semi-structured, and unstructured formats) which that reside on-promise, in the cloud environment or from many other sources IOT than ever.

The other side of the coin is that the architecture used to gather, analyse, and store big data involves new vulnerabilities for the security system, this open new doors for cybercriminals activity and malware. In the fact databases is the most valuable information assets of most enterprises is core enterprise.

The security managers instantly feel nervous when the level C directors ask them those questions: “Are we more secure today than we were before?”. “Can you control what you cannot measure?” and “To measure is to know” [1]. The answers might only be obtained if there is some system able to evaluate the security level of the running system and present the assessment clearly, concisely, and most importantly, visually to the administrators or the directors. Currently, there are some security products available

on the market such as Guardium [5], SecureSphere [6], etc. and they do support in measuring the security level of a database management system (DBMS) within their specific standards. However, they require the running system to be equipped with expensive specialized hardware to operate properly. In this paper we propose a prototype of security framework based on prevention concept to protect organizations sensitive data from a complex and evolving threat landscape with assessment of security metric by implementing a security policies and techniques like encryption, data masking. The framework consists of three main modules:

Policy Module. Processes Module. Security metrics Module

The aim capabilities of system are be able to control and measure simultaneously the security issues of different application layers. By auditing and analysing Data/Code on the support and transfer of the data.

Easy way to Config flexibly database security metrics. To be complying with laws, namely data privacy acts and regulations such as GDPR,SOX.etc

prevent or mitigate new vulnerabilities by use of some data masking techniques

This paper is organized as follows: section 2. Provides an overview of MongoDB as NoSQL leader. Section 3. We discuss NoSQL security consideration such vulnerability and threats of NoSQL database and explore some famous security issues present in MongoDB. We Provides an overview of data masking architectures and techniques. Section. 4 We propose the HLD architecture of proposed framework of security with its different layers. Based on Flexible module of configuration, Finally, section 5. presents concluding remarks and the future work.

2 Overview of MongoDB database

Why present MongoDB? According to Q2 2019 report from research firm Forrester: [2]. the firm recognizes that MongoDB is one of the leaders in the database as a service identified in the report, which supports a wider set of use cases, automation, scalability and high-end performance, and Security. Based on the evaluation of 27 criteria on SWOT of DBaaS providers. MongoDB has been recognized as a leader in the market of the NoSQL database as a service. this is one of the main motivations for choosing MongoDB in our use cases and the demos. we list some Strengths later then will focus on the weakness of the security of MongoDB technology. To improve the security module on MongoDB, we need to understand its architecture.

2.1 MongoDB Concepts

The MongoDB belongs to the NoSQL family, developed in C ++. It is based on the concept of a key-value pair. The document is read or written with the key. MongoDB supports dynamic requests on documents. Being a document-oriented database, the data is stored in the form of JSON, BSON style [3].

The MongoDB database is made up of a collection of collections that has no predefined structure. The collection is like the table concept used in relational databases.

The collection in turn is made up of documents that are the records.

2.2 MongoDB architecture

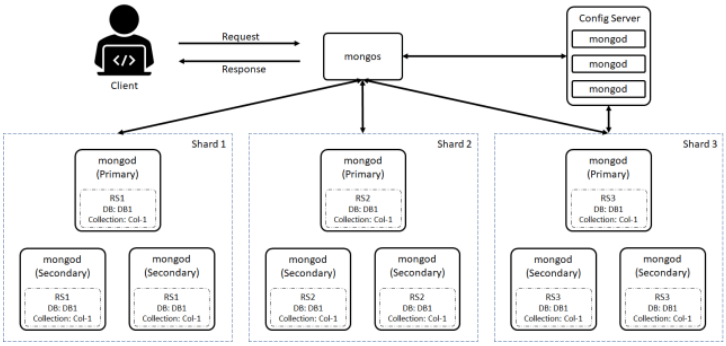

Fig1: MongoDB architecture [11]

A) **Mongos:** is the interface between client and shard clusters. Mongos instance should connect with Config servers to determine which shard should respond to which query.

B) **Config Server :** The role of Config Server is to maintain connecting in a redundant way with every other component of the DB because, it's contains the metadata of chunk of the data in every shard to ensure mangos can respond with requested data at any time.

C) **Shard:** Sharding: is distributing data across multiple achines by Scaling or addressing the system growth of the mongoDB, it has two methods of scalling: **1**-Vertical Scaling: increase the capacity resource of a single server (CPU computing, More RAM, or increasing of storage space). **2**-Horizontal Scaling: distribute over multiple servers, by adding more servers to increase capacity as required.

D) **Replica Set:** Every shard is implemented as Replica set which is the cluster of MongoDB that organised the automated failover and replication.

2.3 Advantages of NoSQL vs RDBMS.

- Scalability: Ease of scale out is the important aspect to consider NoSQL.
- Distributed data over Platform.
- Text search: This feature enables search data into application layer without effort.
- Fast and iterative Development platform support agile project Development.
- Uses internal memory for storing the working set, enabling faster access of data
- Integrated features: Features like Analytical platform, data visualization, event-driven streaming data pipeline, text and geospatial search, graph processing, in-memory performance, and security plugin are helping developers implement without additional efforts of integration.

2.4 Strengths of MongoDB?

- Flexible Data Model: MongoDB supports Dynamic schema you can change your data Model structure at any point (Conversion/mapping of application objects to database objects not needed).
- Document Oriented Storage − Data is stored in the form of JSON documents.
- Index on any attribute
- Replication and high availability
- Auto-Sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB
- Economical and Cost-effective: MongoDB is Cost-effective, and support and maintenance are very economical to Compare to other database systems.

3 NoSQL Vulnerability and security over MongoDB

NoSQL databases may become susceptible to exploits once attackers are able to identify security threats or software unit weaknesses.
Insufficient or ineffective input validation, weak authentication, insecure communication, illegal access to unencrypted data, etc. are some issues of NoSQL vulnerability.

3.1 Authentication

MongoDB apps can use secure method to verify a client's identity by username and password:
SCRAM-SHA-1: Salted Challenge Authentication Mechanism uses simple method text-based usernames and passwords over a channel protected by transport layer security (TLS).
MongoDB-CR: Like SCRAM, this method verifies a username and password against an authentication database.
the passwords are encrypted, and a different hash is generated for each new session.
Enable X.509 certificate-based authentication,

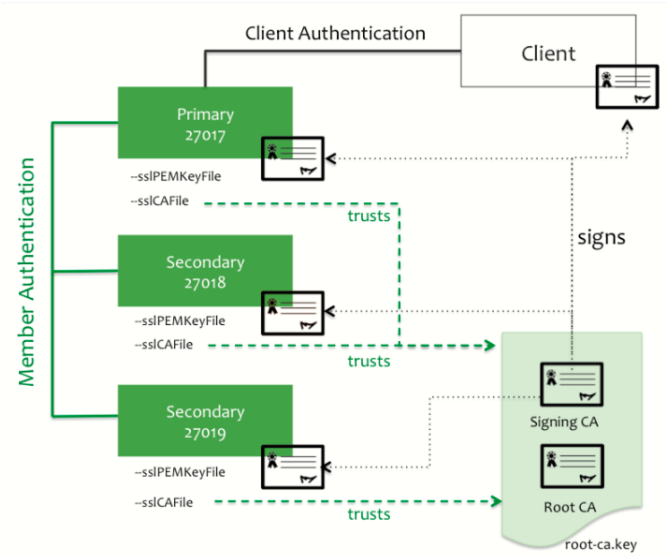

Fig2: Deployment of X.509 certificate-based authentication over 3 node replicas set and a client [2].

MongoDB apps. can employ also external authentication protocols as well:

LDAP: Lightweight Directory Access Protocol using their centralized passwords.
Authentication providers allow users to log in with an email/password, an API Key, or an external OAuth service. If the built-in providers don't cover your use case, you can integrate your own custom provider.
Tools authentication like Kerberos: This is a secret key authentication protocol for server-client interactions. By using an access ticket to log in. But this is enabled on enterprise edition.

3.2 Authorization/role-based security

Role-based access control (RBAC) is use role to control act While you can find well-defined roles within MongoDB that can cover most users, custom roles can be created as well.

A role essentially determines what permissions a user has and what he can access.

Other way there are available roles to use. Strong role limit access to the system beyond it.

3.3 Auditing

MongoDB Enterprise offer the capability to audit mongod and mongos instances. The auditor can track system activity for deployments with multiple users and applications. Administrators can configure auditing to write to the console, syslog, JSON file or BSON file. But we can also use filters to restrict which events are logged.

Audit Events and Filter

Once enabled, the auditing system can record the following operations [4]:

schema (DDL),

replica set and sharded cluster,

authentication and authorization, and

CRUD operations (requires auditAuthorizationSuccess set to true).

3.4 Threats over code and data Consistency

The most Vulnerabilities match with Data/code using MongoDB are:

A- **Injection Attack**: The MongoDB API expect BSON (binary JSON) calls and includes a secure BSON query assembly tool. However, JSON and javaScript un-serialized allow an attacker to manipulate data and execute commands and achieve non expected purposes.

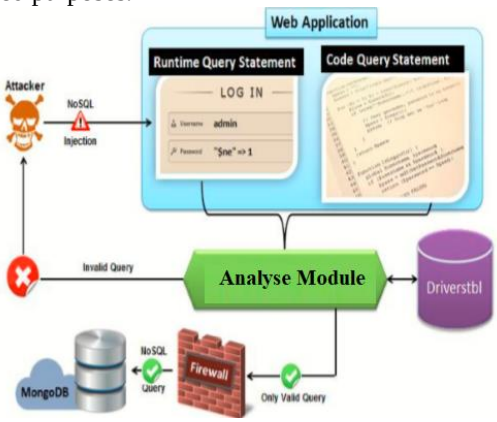

Fig.3 Solution for NoSQL injection in password field/inside query statement.

B- **DoS Attack**: DoS attacks are possible in MongoDB. By default, MongoDB does not require authentication and authorization is not configured to work with this attack. With valid user credentials an attacker does not have to be administrator to carry out the attack.

C- **WSS Attacks** (WebSockets over SSL/TLS) [4]: attacks against WebSockets become impossible if the transport is secured. MongoDB allows developers to write JavaScript code may bypass security authentication or steal sensitive information.

D- **Unencrypted File storage system**: by default, MongoDB has data files are unencrypted. It should automatically encrypt/decrypt these datasets. It means that any active or passive attacker or unauthorized user can easily access the file system and retrieve the valuable or secret information. To reduce this, the application should explicitly encrypt/decrypt any confidential data before updating/retrieving into/from the databases [7].

MongoDB being vulnerable for those attack [8] one approach to prevent these is through careful code dynamic analysis and/or static analysis. But it may have high false positive rates and presents difficulty for full read. While dynamic analysis tools/methods appeared to be extremely valuable for the identification/detection of injection attacks [9], these should be changed as per recognize the specific vulnerabilities of NoSQL.

3.5 Data masking architectures and techniques

Based on prevention concept we should use Some functions to protect organizations sensitive data from a complex and evolving threat landscape with assessment of security metric by implementing a security policies and techniques like data masking. The aim of data masking is to find sensitive data or PII in unstructured sources using multiple search techniques.
Use the search results to provide rectification simultaneously or separately, remove, or fix the PII to be compliant with data privacy

A- Principles of data masking:

There are five principles:

A-1) Masking must not be reversible: whatever you mask the data it should never be possible to reverse it back to its original form.

A-2) Resemblance of the original data even after masking the results of data masked must be representative of the source data should have so that application, can function normally after masking.

A-3) Referential integrity must be maintained for system event data masking

A-4) Only mask non-sensitive data if it can be used to recreate sensitive data because it is not necessary to mask everything in your database

A-5) Masking must be a reproducible process: for a different staging environment: the development, quality assurance and test data must represent the constantly evolving production data as faithfully as possible. It must be an automated, inefficient, costly, and ineffective process.

B- Data masking architectures:

There are two mainly type of data masking used in the design of data masking software:

The fly server to server data masking: the data does not exist in target database server prior to masking. The masking rules are applied during the transfer of data from source to target. This architecture is note suitable for some applications as any error in the process can interrupt the transfer of the data.

In-situ Data masking architecture: clone of database to be masked is created by some other means and the data masking process will simply operate on the cloned target database.

C- The techniques of data masking:

Substitution: this technique consists of randomly replacing the original data in a column with information that looks similar but completely unrelated to the original data.

Shuffling: this technique uses the existing data for masking. Replacing of the existing vales in rows will be done by moving the values between rows in such a way that the no vales are presents in their original rows. Its look like substitution but the main disadvantage, which is reliance on an algorithm can be detrained, then the data can be easily unshuffled.

Number and Date variance this technique varies the existing values by certain range order to mask them

Nulling out or deletion: this technique is simple to removes the sensitive data by deleting a column and replacing them with NULL values.

Masking Out: this technique sanitizes sensible data by replacing certain specified characters with mask characters like a credit card number should be masked test environment.

Hashing: this technique is not a form of encryption; tough it does use cryptography. Hashing replace data and creates a hash string out of it, there are three important way to hash:

The same data will always produce the same hash string.

It is impossible to reverse it back to the original data. Given knowledge of only the hash, it' s infeasible to create another string of data that will create the same hash/ the most important use of hashing is protecting passwords. If system store a password hash instead of a password, it can check an incoming password by hashing that and seeing if hash match. It is not possible to use the hash to authenticate. Another common use of a hash is authentication otherwise clearly transmitted data using a shared key secret.

4 Proposed framework of security under Big data challenges

Any efficient security software must ensure that business needs and IT strategy are aligned. As such, enterprise information security architecture track information traceability from the business strategy down to the underlying technology.

4.1 Security business model

To understand in dept the security solution proposed, we must follow information on business security model

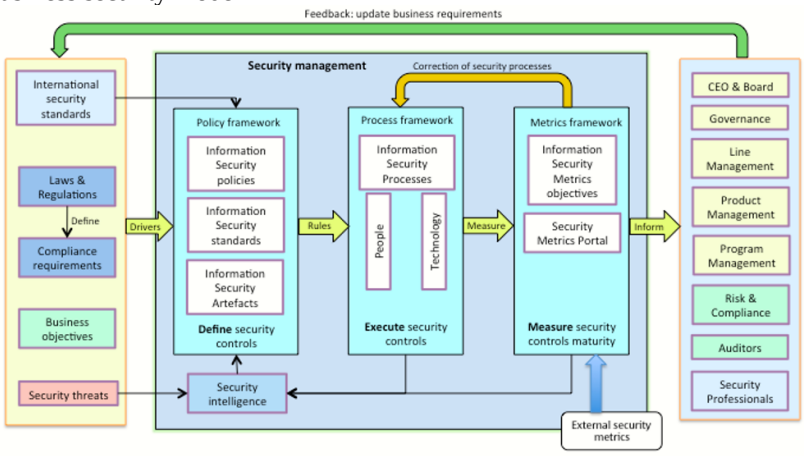

Fig.4 Business security model of framework

In this model there are three main areas:

Security drivers: the three major drivers for security work are:

Laws & regulations - These are something than a company must comply with like GRPD.

Business objectives - Security supports these business objectives by protecting systems and data that is used in the business processes. E.g. Business objective is to have on-line bank 24/7. Security objective is to keep systems up and available, be it by keeping them free of malware that could disrupt or slow IT systems.

Security threats – Act of cyber security against laws & regulations and business objectives.

Security management (the main part)

In this area we have three modules that enable company to achieve objectives defined in the drivers' section.

Policy framework is a set of policies, standards and guidelines that describe how the company addresses information security drivers. All together these define security controls that are available for a company to implement.

a. Policies – describe high level to detail policy statements, Security Control Objectives (typically using words should and must). The key objective of the security policy document is the alignment with the business objectives and drivers.

b. Standards – detail of Security controls that should be implemented to support individual policy statements. With relationship 1:N. One policy statement can be implemented by multiple security controls. Then a link to a policy, the security professionals will hardly justify why the password needs to be 12 characters and change every 45

days. The controls should be selected from an internationally accepted catalogue of controls.

c. Artefacts – It is all very well to have statements like "must be authenticated" but how is that done in practice by an engineer that needs to configure the system? I have learned that architecture standardization is the key to the success of any company. Same applies to security.

Processes framework

This process executes the security control in a policy or standard is a process, no exceptions. This is where people and technology come into play. some process is supported by people and most are supported by technology. However, the efficient company must track a link between any human process and technology corresponding control in the Policy framework up to the business objective.

Security metrics framework

Security professionals should be able to measure the status of security controls, compliance with own policies and effectiveness of security processes. The key metrics here is to take a security policy statements and measure each team against them; output a nice balanced scorecard for security.

Stakeholders

The stakeholders need to assure that what has been promised is being delivered. More importantly the security managers need to show the value of security to business. We should ask them what their concerns are and show them how you are addressing the concerns. Then send a report to them that relates to their area and concerns. Always get them on your side!

4.2 Security logical model

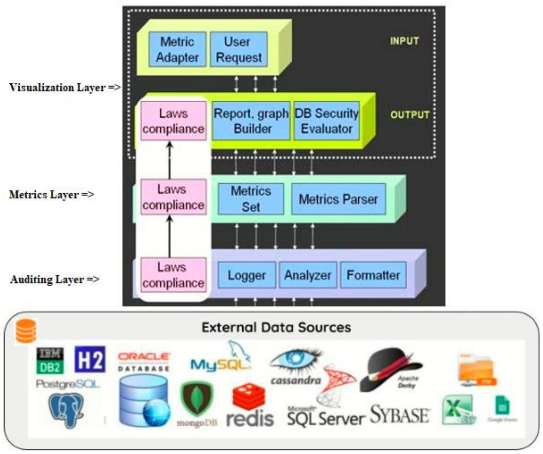

Fig.5 Logical Model of framework

Each component uses an API as interface

There are Four principal layers as mention in figure 5, Each layer of framework is Subsystem IOTT (input output tools and technics). Note that a Laws component in

three layers when laws compliance is enforced this component will ensure that the data transmitted through it to the upper layer will conform to the required laws.

1) Visualization layer: interacts with users, receives user requests as INPUT and selects suitable presentation to show the final score as OUTPUT. in the most visual way, using many graphs and reports. There would be lots of types of graph such as pie chart, bar chart, line chart, etc. and lots of types of reports such as daily report, weekly report, monthly report, or quarterly report and so on.

2)Metric Layer is a concept of a sub system or standard of measurement define how well security service are. So, the metric is the heart of the framework we use techniques and tools and without metrics, the system is defunct or no longer able to measure the level of security attends.

The metrics value is saved into the database. Users can create new metrics To perform calculation a need of a metrics parser which can parse the metric content to get all the essential data such as the input, output data type, where the metric processing cores are stored, etc. Besides, Metrics Parser is not directly in charge of calculating but it will call web services other libraries at run time.

3) Auditing layer communicates with the last Databases layer gets their data, refines and sends them to Metrics layer to carry out metrics calculation. The Logger will communicate with databases to acquire the data which might have to go through Laws to ensure law compliance. When the data reaches Analyzer, it is refined and handed over to Formatter to reformat in some standards for Metrics layer to use later.

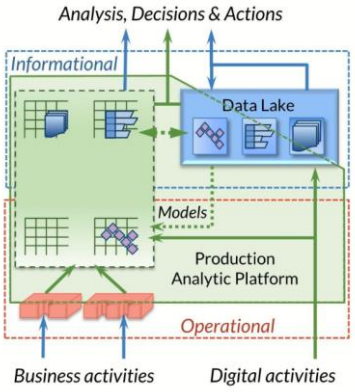

Fig.6 API building block for both Logical/busines Model of framework

The Old operational systems run the business activities already optimized for read-write processing like in DWH (and marts) While ideal for the logical/business model is to have an API building block able to analyse new challenges of business activities, this ongoing process must run continuously in a production environment to offer real-time predictions or prescriptions.

So, the informational area square drive the business process when the data are optimized for write-once / read-many processing.

Be sure that both operational systems and apps are today built on high-performance, production-ready in both RDBMS and Nosql (such as documents, key-value stores and graphs), are the preferred environment to handle the production phase of analytics, as shown in Figure 6 show API building block for both Logical/busines Model with

characteristics of new approach Analysis of these issues must be done in the context of the function and features of existing and anticipated technology in both data storage and processing. Some commercial APIs offer capability key insights into a process-oriented view of this environment and allows us to step back from an overly storage centric assessment. E.G. [12] API CoSort for metadata and advanced resource exploitation speed job design and data delivery. Sort, join, aggregate, and load 10x faster than SQL and 6x faster than ETL tools.

5 Conclusion

This paper shows an architecture of proposed framework of security with its different layers. Based on API building block as flexible module of configuration. The aim is to fix new threats and limit vulnerabilities issues of big data system in real time. We can improve security performance in big data by using machine learning, blockchain system especially with integration of IOT device or by combining these three approach-es in Hadoop Distributed File System which is the base layer in Hadoop, where it contains large number of blocks. These approaches are introduced to overcome certain issues occurs in the name node and in Data node. In Future these approaches are also implemented in other layers of Hadoop Technology.

6 References

1. A. Jaquith: “Security Metrics: Replacing Fear, Uncertainty and Doubt”, ISBN: 0321349989, Addison-Wesley Professional, 2007
2. MongoDB“https://www.mongodb.com/collateral/forrester-wave-database-as-a-service?” (2020)
3. Ciprian Octavian Truică, Alexandru Boicea, Ionut Trifan, “CRUD Operations in MongoDB,” International Conference on Advanced Computer Science and Electronics Information, pp. 347-348, 2013
4. https://docs.mongodb.com/manual/core/auditing/#transactions
5. IBM Guardium web: https://www.ibm.com/fr-fr/marketplace/guardium-data-protection-for-big-data.
6. Securesphere product: http://www.imperva.com/products/securesphere.
7. Lior Okman, Nurit Gal-Oz, Yaron Gonen” Security Issues in NoSQL Databases” 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications 2011.
8. Ron, Aviv, Alexandra Shulman-Peleg, and Emanuel Bronshtein, "No SQL No Injection? Examining NoSQL Security", In Proceedings of the 9th Workshop on Web 2.0 Security and Privacy (W2SP), Vol. 1, 2015.
9. Ron, Aviv, Alexandra Shulman-Peleg,”analyse and mitigation of NoSQL injection” IEEE security and Privacy 14.2, 2016.
10. Charmi “Encrypting Data of MongoDB at Application level” Advances in Computational Sciences and Technology Volume 10, Number 5 (2017) pp. 1199-1205.
11. MongoDB “https://digitalvarys.com/introduction-to-mongodb/” (2020)
12. IRI « https://www.iri.com/products/cosort » (2020)