ISBN 978-93-5156-328-0
International Conference of Advance Research and Innovation (ICARI-2014)

A Security Requirements Perspective towards a Secured NOSQL Database Environment

Prudence Kadebu, Innocent Mapanga
Department of Computer Engineering, Delhi Technological University, Delhi, India

Article Info

Article history:
Received 2 January 2014
Received in revised form
10 January 2014
Accepted 20 January 2014
Available online 1 February 2014

Keywords

NoSQL Database Security,
Database Security,
Security Requirements

Abstract

Security has been and still remains a very pertinent issue to consider in database management systems development especially those handling and storing large volumes of data. Most researches have focused on security at hardware level, Operating system level and application level with only a little or no consideration for security at the database level. With the current Big Data scenario which has recently seen accomplished systems engineers scampering for new solutions to address the challenges, database security has emerged quite a pain to their efforts. The security challenges that were inherent in the previous database systems have not spared the NoSQL databases either. In this paper we evaluate NoSQL database security in line with security requirements proposed by Firesmith [1]. We consider the role played by security policy and security models in achieving adequate security and finally propose security mechanisms for a NoSQL database.

1. Introduction

Data represents the most important asset in any establishment and as a result data has to be protected against unauthorised access, unauthorised change and also should be available only to authorised entities. Recent study has shown that databases are more at risk than before. It is stated in [2] that as the amount of data collected, retained and shared electronically expand, so does the need to understand database security. In recent years internet based companies such as Amazon, Facebook, Twitter, and Google have experienced a mushrooming of data which has resulted in the adoption of new ways to store and scale large amounts of data (termed Big Data) using NoSQL database systems. Security goals or objectives in relation to database systems are established by an organisation in order to protect the data stored. In line with this, security requirements are needed to satisfy the security goals [3]. They express the system's security goals in operational terms. Security mechanisms are implemented to satisfy the security requirements for the NoSQL database system in this context. Security mechanisms provide security services which are related to the security goals. One major problem is that security requirements are often considered as an afterthought in most systems when all architectural components have been put in place. This result in security mechanisms in the systems being rather contrived instead of properly planned prior to design. Firesmith in [1] identified security objectives for addressing the many challenges posed by today's security threats and 12 corresponding security requirements for achieving them. Most security issues that affected RDBMSs were also inherited in the NoSQL databases as well as new ones imposed by their new features and structure. This paper is organised as follows: section 2 will look at an overview of NoSQL database systems, section 3 will look at NoSQL Database Security Issues, section 4 will get into the security requirements for the NoSQL databases, also exploring security policy and various security models and section 5 Recommendations leading to the conclusion and future works.

Corresponding Author,
E-mail address: imapanga@gmail.com

2. NOSQL Database Overview

Not Only Structured Query Language (NoSQL) is a class of "schema-less" database management systems that have been designed with more relaxed data models as compared to RDBMS. These engines usually provide a query language that provides a subset of what SQL can do, plus some additional features [4]. Okman et.al [5] summarized the common features of NoSQL databases as: " high scalability and reliability, very simple data model, very simple

(primitive) query language, lack of mechanism for handling and managing data consistency and integrity constraints maintenance(e.g., foreign keys), and almost no support for security at the database level". There are several categories of NoSQL databases. Four main ones are Key-Value Stores, Column Family, Document Oriented and Graph Databases. These databases generally have lower administration requirements, are cheaper to manage and offer very high performance. Challenges of NoSQL databases include lack of standardised models and expertise for efficient database support. NoSQL databases make use of CAP (Consistency, Availability and Partition tolerance) and BASE (Basically Available, Soft State, Eventual Consistency) theorems in their design as opposed to the RDBMSs which are based on the ACID (Availability, Consistency, Isolation, Durability). In order to achieve high performance and scalability, NoSQL databases generally trade off consistency and security. Owing to the unstructured nature of the data stored in these databases, security may be difficult to enforce making them more vulnerable to security threats.

3. NOSQL Database Security Issues

In this section we will review security built-into the NoSQL databases environment which are entrusted in handling Big Data, we will further evaluate the weaknesses of these systems. Our goal is to uncover security problems inherent in the NoSQL database system environments and explore how best to secure these environments. The amount, diversity and rate of data being generated for processing and storage results in sheer masses of data that need to be safely secured. A survey of the top big data vendors as well as deployments for services extending from server and storage hardware, database software, analytics applications and other associated services have revealed that vast amounts of valuable and sensitive data are being handled through the various applications across many platforms all over the world generated by humans or by machines that routinely access and use Web and mobile applications. This data, in the hands of organizations is highly valuable, and a subject to privacy laws and compliance regulations, and must be protected. It can be established in this research that the various significant ways in how these database systems are deployed have impact on the security of the NoSQL database environment.

3.1 Threats Posed By Distributed Environments

Nodes within the NoSQL database environment are distributed subsequently resulting in massive parallel computation. This creates increased attack surface across several distributed nodes which makes it very complex to secure the NoSQL database system. Another issue is deciding where to grant database system Access, whether at the Clients home locations or at the remote location which increases the probability of unauthorized access.

3.2 Safeguarding Integrity

The protection of integrity is much harder in NoSQL database system because of its heterogeneous nature than in homogeneous environments. There is absence of central control and its schema-less nature makes it difficult to enforce integrity constraints

3.3 Communication between Nodes

All communication protocols as DataNodes talk to the NameNode are layered on top of the TCP/IP mainly relying on RPC over TCP/IP. A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol in the NoSQL distributed environment. NoSQL database Systems with RPC ports exposed to the Distributed environment are especially vulnerable. Security concerns emanate as nodes interact through message passing, because communication is not secure.

3.4 Sharded Data/Fragmented Data

NoSQL databases horizontally segments slices of data and share them across multiple servers. Data from a variety of nodes move to and from in the NoSQL database environment which is distributed across multiple servers, An example is that at one organisation which has clusters with up to 4000 nodes, and about 65 million files and 80 million blocks. Movement of this data to multiple locations is automated for large inter/intra-clustering employing MapReduce parallel copying mechanism in copying portions of the source data into the destination file system. The maintenance of replicated shards of data that includes passwords is computationally expensive, more prone to error and increases the risk of theft. This model is not centralised which poses difficulties in securing data as it gets replicated and moves in many places as needed.

3.5 Compromised Clients

Clients accessing NoSQL databases are in contact with resource managers and various nodes directly. In situations where malicious data gets propagated from a single compromised location, the entire system is compromised. Protecting nodes, name servers and those clients becomes difficult especially when there is no central management security point.

3.6 Protection of Data at Rest

Most NoSQL databases are found wanting when it comes to protecting data in storage, only a few categories of NoSQL databases provide mechanisms to protect data at rest by employing encryption techniques. Encryption is widely regarded as the de-facto standard for safeguarding data in storage. Malicious intruders who intend to then steal from archives or with intention to read directly from the disk will find the data unintelligible. Encrypted data will be accessed by users with decryption keys, but however most industry solutions offering encryption services lack horizontal scaling and transparency required in the NoSQL environment.

3.7 Challenges In Enforcing Access Control

The NoSQL database's schema-less structure makes Role-based access control difficult to enforce. We take for instance the Key-Value store that store data by means of a distributed index for object storage. In this type of database different data are stored in one huge database. This becomes a challenge as heterogeneous data is stored together in one database as opposed to relational models which conform to defined schemas and tables that store only related data.

3.8 Administrative Data Access

NoSQL database systems lack in-built facilities, documentation, and third-party tools to address issues of administrative rights with for instance full access to data and enabling creation of administrative boundaries for the purposes of encryption. Tasking administrators to choose the right security controls that would tighten all the screws on the four corners of our database systems like the proper access controls and the best encryption technologies can cause unwanted direct access to data files or data node processes. It will be better if this is left to the system designer to select controls to close this gap.

3.9 Configuration and Patch Management

Existing configuration management tools work for underlying platforms. Different nodes or clusters of servers may have different patch revisions. Added nodes may have newer patches than existing nodes. This may create challenges in enforcing security uniformly across the NoSQL database environment.

3.10 Firewalls

Firewalls cannot protect data at rest or in-transit within the NoSQL database environment. If a firewall gets breached, the database is immediately exposed to attacks. Firewall breaches emanating from the firewall perimeter cannot be avoided like attackers who get into data centres physically or electronically can get access to data.

3.11 Authentication Clients

Kerberos can be used to authenticate clients, DataNode, NameNode in the NoSQL database environment. Malicious Clients and Nodes can gain unauthorised access to the NoSQL database system

upon stealing or duplicating the Kerberos ticket. These credentials can be obtained from system snapshots as well as virtual images. The situation has worsened in this Big Data environment where exact copies, clones and imposter nodes can be used to generate malicious services into the databases environment.

3.12 Audit And Logging

Audits and logs are performed to aid in discovery of malicious activities in the database system. However without actually looking at the data and developing policies to detect malicious activities, logging is not useful. Also the frequency at which the Audits are carried out can have impact on their effectiveness. If audits are performed say quarterly that means malicious activities can occur which can result in serious problems for the organisation. This may be discovered too late when the damage has already occurred.

3.13 Monitoring, Filtering, and Blocking

Existing NoSQL database monitoring tools lack the capability to identify malicious queries, misuse activities and subsequently block them. Monitoring undertaken by several tools in the NoSQL database environment mostly perform their task at the API. There is an assumption that all access by client connections will pass through the same path that authenticate clients through Kerberos, which results in a performance constrains. Also advanced threats may bypass the central Kerberos authentication.

3.14 API security

APIs can be subjected to several attacks such as Code injection, buffer over flows, command injection as they access the NoSQL databases. The APIs for big data clusters need to be protected from code and command injection, buffer overflow attacks, and every other web services attack. This responsibility often left to the application that uses the cluster.

4. Roadmap Towards A Secured NOSQL Database

Database security is not very easy to achieve as shown in section 3 above. As a step towards achieving adequate security for a NoSQL database,

several elements are discussed which have an impact on the success of this quest. Considering a combination of these elements rather than a single element is the best way towards achieving a secured NoSQL database. We will portray these elements by means of a model for the security of a NoSQL database as shown in Figure 1 below. In this section we look into some of these elements and show what part they play in contributing to achieving a secured NoSQL database.

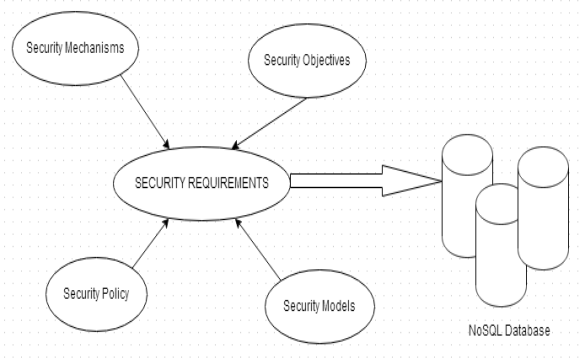

Fig: 1. Security elements for NoSQL Database

4.1 Security Requirements for Nosql Databases

We evaluate the security requirements of the NoSQL databases in this section and draw a summary of the extent to which the security requirements are satisfied using a case study of four widely used NoSQL databases as shown in Table 1.

4.1.1. Identification Requirements

Identification requirement is any security requirement that specifies the extent to which the database system shall describe a user or entity. It is assigning a unique label/identity to a user or entity involved in a database interaction to distinguish it from the rest.

4.1.2 Authentication Requirements

Authentication requirement is any security requirement that specifies the extent to which the database system shall be able to verify and make a confirmation of the identity of an entity wishing to access some resource in the database

4.1.3 Authorization Requirements

Authorization requirement is any security requirement that specifies the extent to which the database system shall verify that the user in question has the correct permissions and rights to access the requested resource.

4.1.4 Immunity Requirements

Immunity requirement is any security requirement that specifies the extent to which the database system shall provide an internal ability to defend itself from corruption and attack by malicious software.

4.1.5 Integrity Requirements

Integrity requirement is any security requirement that specifies the extent to which the database system shall ensure that data is protected from unauthorised modification through insertion, deletion or update of data.

4.1.6 Intrusion Detection Requirements

Intrusion detection requirement is any security requirement that specifies the extent to which the database system shall provide internal ability to monitor and identify attempts to gain unauthorized access to it. If audit trails are properly designed and implemented, they can aid intrusion detection.

4.1.7 Non-repudiation Requirements

Non-repudiation requirement is any security requirement that specifies the extent to which the database system shall be able to record all activities such that there will be no future denial of access to the database by any user or process. This requirement ensures that adequate tamper-proof records are kept to prevent parties to interactions with the database system from denying that these interactions have taken place.

4.1.8 Privacy Requirements

Privacy requirement is any security requirement that specifies the extent to which the database system shall ensure personal control over data stored in the database. This means owner determines who sees what information and use it for what reason according to their discretion.

4.1.9 Security Auditing Requirements

Security auditing requirement is any security requirement that specifies the extent to which the database system shall monitor system for violations of security policy, malicious activities and recording there-of. "It is a systematic, measurable technical assessment of how the organization's security policy is employed at a specific site."[8] Database security audit can help accomplish several security-related objectives, including individual accountability, reconstruction of events, intrusion detection and problem analysis.

4.1.10 Survivability Requirements

Survivability requirement is any security requirement that specifies the extent to which the database system shall make a tradeoff between

integrity and availability in the face of database attack. In [9] it is stated as, "we evaluate the survivability of a database system based mainly on the proportion of data objects that are not altered or destructed by intrusions and are available for user access at the same time." This means integrity and availability may not be achieved 100% when the database system is operating in degraded mode. One will be lower than the other.

3.1.11 Physical Protection Requirements

Physical protection requirement is any security requirement that specifies the extent to which the database system shall protect itself from physical conditions and procedures that could cause serious losses or damage to an organisation. This includes protection from fire, natural disasters, theft and destruction.

3.1.12 System Maintenance Security Requirements

A system maintenance security requirement is any security requirement that specifies the extent to which the database system shall prevent any system modification be it authorised or not from disrupting its deployed security mechanism. It means all security mechanism deployed by the database system should be maintained and reviewed.

4.2 Security Policy

Security policy is a detailed, written document that states how an organization plans to protect the company's assets by giving a definition of goals, purposes, responsibility, overall requirements and identifies the rules that will be followed to maintain security in a system and defines the security vision for the organization. A security policy expresses exactly what the security level should be by setting the goals of what the security mechanisms are to accomplish. (CRISSP)

Data security policy determines which users have access to a specific data item, and the specific types of actions allowed for each user on the data item. This definition can be in terms of query statements a particular user or group of users can issue for instance CREATE, INSERT, DELETE, UPDATE etc. This also depends on the NoSQL Query Language in use. Data security policy should also define the actions, if any, that are audited for each data item. Data security policy is determined primarily by the level of security required for the data in the database and is directly proportional to the sensitivity of data. Security policies that prevent information from flowing from a high security level to a lower security level are called multilevel security policies.

4.3 Security Models

A security model is a statement that outlines the requirements necessary to properly support and implement a certain security policy. It is a symbolic representation of a security policy. A security policy outlines goals with no idea of how they would be accomplished. A model is a framework that gives the policy form and solves security problems for particular situations. Below are some security models:

4.3.1 Bell-LaPadula model

Bell-LaPadula model is a model that protects the confidentiality of the data within a database system. A system that employs the Bell-LaPadula model is called a multilevel security system because users with different clearances use the systems, and the systems process data with different classifications.

4.3.2 Biba model

Biba model protects the integrity of the data within a database system. Biba addresses the integrity of data being threatened when subjects at lower integrity levels are able to write to objects at higher integrity levels and when subjects can read data at lower levels.

4.3.3 Clark-Wilson model

Clark-Wilson model is an integrity model implemented to protect the integrity of data and to ensure that properly formatted transactions take place. Subjects can only access objects through authorized programs (access triple). It enforces separation of duties.

4.3.4 Information flow model

Information flow model enforces that data is restricted in its flow to only go to and from entities in a way that does not negate the security policy.

4.3.5 Non-interference model

Non-interference model enforces that commands and activities performed at one security level should not be seen or affect subjects or objects at a different security level.

4.3.6 Brewer and Nash model

Brewer and Nash model allows for dynamically changing access controls that protect against conflicts of interest.

4.3.7 Graham-Denning model

Graham-Denning model creates rights for subjects, which correlate to the operations that can be executed on objects.

4.3.8 Harrison-Ruzzo-Ullman model

Harrison-Ruzzo-Ullman model allows for access rights to be changed and specifies how subjects and objects should be created and deleted.

To ensure adequate security, the right models for the situation have to be selected. This will be based upon the level and type of security required for the NoSQL database under consideration.

Table: 1. Security requirements of NoSQL databases

Security Requirements	MongoDB	Cassandra	NEO4J	Hadoop/HBASE
Identification & Authentication	Turned off by default (trusted environment), Per connection authentication, Authentication with Sharding (v1.9.1+), Replica Set authentication, User in admin database: super user	Uses IAuthenticator interface, AllowAllAuthenticator (default), SimpleAuthenticator (cassandra.ymal)	Custom Authentication Provider	Enabled by default, Kerberos (v5)based authentication, Server authentication is bi-directional
Authorization	Normal user(Full read and write access) Read only user(read access)(v1.3.2+) Notable level access control	Uses IAuthority interface, AllowAllAuthority, SimpleAuthority	Custom Authorisation Provider	HadoopUser, SecureHadoopUser, User
Immunity	No immunity, Injection attacks are possible via JavaScript or string concatenation	No immunity, Injection attacks are possible in CQL, no input validation	No immunity, Injection attacks possible	No immunity, Injection attacks are possible
Integrity	passwords are hashed using MD5, Makes use of Locks for concurrency control	MD5 Hashing, MVCC Multi Version Concurrency Control		
Intrusion Detection	No internal mechanisms	No internal mechanisms	No internal mechanisms	No internal mechanisms
Non-repudiation	No internal mechanisms	No internal mechanisms	No internal mechanisms	No internal mechanisms
Privacy	Data at rest unencrypted	No encryption available for client communication	No security at the data level, No data encryption,	No encryption on the wire

			Communication with database is not encrypted	
Security Auditing	Only logs change activity, not reads.	Not available	Not available	Not available
Survivability	Distributed, replication and sharding improves fault tolerance hence survivability	Distributed, replication and sharding improves fault tolerance hence survivability	Distributed, replication and sharding for fault tolerance hence survivability	Distributed, replication and sharding improves fault tolerance hence survivability
Physical Protection	Depends on organizational security policy	Depends on organizational security policy	Depends on organizational security policy	Depends on organizational security policy
System Maintenance	If no security updates/maintenance are performed together with System maintenance weaknesses result	If no security updates/maintenanc e are performed together with System maintenance weaknesses result	If no security updates/maintenance are performed together with System maintenance weaknesses result	If no security updates/maintenance are performed together with System maintenance weaknesses result

5. Proposed Security Mechanisms

Database security includes the mechanisms that control the access to and use of the database at the object level. NoSQL Database is founded upon the Hadoop framework which is not a single technology but rather a partnership constituted by several cooperating applications. Each of these applications in the framework needs to have security engraved in them which have the capability to scale with data as the need arises. It is imperative that security mechanism be deployed within the NoSQL database environment rather focussing them on a control point where data and queries enter the environment. The proposed security mechanisms are given as a summary in table 2 below.

Table: 2. Security Mechanisms

Mechanisms	Description
Using Firewalls	• Putting firewalls close to the data store and the data. • Embedding security in the data itself like strengthening user authentication process using RBAC
Audit and logging	• Use of third party open source tools that integrate into most big data environments like Scribe and *LogStash*. • Integrate it with other systems such as log management. • Archive logs to save them for as long as required by the organisation's compliance standards. • Implement as part of the authentication and authorization solutions. • Audits of the database should be performed as frequent as possible to discover mishaps quickly

Authentication	• Use of built-in authentication capability within the NoSQL database or use authentication within the framework. • Passwords should be user-defined.
Input validation	• Filter to remove JavaScript • Set up the NoSQL database not to allow Java - Script within the database environment. • Eliminate JavaScript injection attacks and string concatenation
Access Control	• Impose restrictions on what data a user is allowed access. • Grant authenticated user's access to specific resources based on organisational security policies and the permission level assigned to the user or user group. • Use authentication, which proves the identity of the user or client machine attempting to log in.
Segregation of duties	• Apply least privileges to employees to only have access to the minimum amount of information that is necessary to perform their daily duties.
Encryption	• Protects confidentiality/privacy of data. • Apply an encryption algorithm to sensitive data. • Transformed data into an intelligible format which can only be decrypted by those who are authorised by applying a decryption algorithm.

6. Conclusion

A thorough consideration of security requirements for the NoSQL database solution and implementation of corresponding security mechanisms is key to ensuring adequate security of the NoSQL database system. Major security issues identified in this paper include increased attack surface, difficulty in protection of integrity. Some concerns where noted in inter- node communication, fragmented data, client interaction, data in storage protection, administrative data access, configuration and patch management, firewalls, authenticating clients, audit and logging, monitoring, filtering, blocking and API security. A security policy should be put in place to express exactly what the Database security level requires. Security models and mechanisms are necessary to properly support and implement a security policy. The following recommendations were proposed to address the various security issues highlighted in the paper: making use of third party open source tools for audit and logging, built-in authentication, input validation, access control, Segregation of duties and encryption just to mention but a few. In our future works we shall work on a NoSQL solution able to satisfy as much of Firesmith's security requirements [1] as possible.

References

[1] Donald G Firesmith, Engineering Security Requirements, Journal of Object Technology, 2, 1, 2003, 53-68

[2] Meg Coffin Murray, Database Security: What Students Need to Know, Journal of Information Technology Education: 9, 2010

[3] Charles B. Haley, Robin Laney, Jonathan D.Moffett, Security Requirements Engineering: A Framework for Representation and Analysis, IEEE Transactions On Software Engineering, 34, 1, 2008

[4] A, MongoDB vs Oracle - database comparison, IEEE 2012

[5] Lior Okman, Nurit Gal-Oz, Yaron Gonen, Ehud Gudes, Jenny Abramov, Security Issues in NoSQL Databases, 2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11

[6] Sohail Imran, Irfan Hyder, Security Issues in Databases, 2009 Second International Conference on Future Information Technology and Management Engineering

[7] Fahmida Y. Rashid, http://mobile.eweek.com/c/a/Security/Database-Security-Faces-Greatest-Threats-from Insider Human-Errors-422328

[8] Bill Hayes, Fundamentals of A Security Audit, http://www.certconf.org/presentations/2004/Wednesday/ WA4.pdf

[9] Hai Wang, Peng Liu, and Lunquan Li, Evaluating the Survivability of Intrusion Tolerant Database Systems and the Impact of Intrusion Detection

Deficiencies, Int. J. Accounting, Auditing and Performance Evaluation

[10] Sadalage, P. J., Fowler, M., NoSQL distilled - A brief guide to the emerging world of polyglot persistence, 1, 1-77, 2012

[11] I. Mapanga, P. Kadebu, Database Management Systems: A NoSQL Analysis, International journal of Modern Communication Technologies and Research:1(7),2013