ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

A Survey on Security of NoSQL Databases

Dadapeer[1], N.M.Indravasan[2], Adarsh.G[3]

Assistant Professor, Dept. of Computer Science and Engineering, BITM, Ballari, Karnataka, India [1]

Bachelor of Engineering, Dept. of Information Science and Engineering, BITM, Ballari, Karnataka, India. [2] [3]

ABSTRACT: The increasing popularity of Big Data and Cloud computing is the moving power for the organizations to migrate from Relational Databases towards Non-Relational databases referring as NoSQL popularly called as "NOT ONLY SQL". NoSQL databases have been designed for utilizing distributed, collaborating hosts for storing and retrieving the data. Large amounts of user related sensitive information stored in these databases raise the concern for confidentiality, availability, integrity, authenticity of data and the security provided by these databases. In this paper we compare and analyze the security features of few popular NoSQL databases such as Cassandra, MongoDB, CouchDB, Redis and HBase. Finally we suggest few techniques to mitigate the attacks on NoSql Databases.

KEYWORDS: Big Data, NoSQL, confidentiality, availability, integrity, JSON, DAST, JSONP, authentication, authorization, hashing.

I. INTRODUCTION

Over the past 20 years, data has increased in a large scale in various fields. According to a report from International Data Corporation (IDC), in 2011, the overall created and copied data volume in the world was 1.8ZB ($\approx 10^{21}$B), which increased by nearly nine times within five years. This figure will double at least every other two years in the near future. Under the explosive increase of global data, the term of big data is mainly used to describe enormous datasets [1]. Compared with traditional datasets, big data typically includes masses of unstructured data that need more realtime analysis. The term "Big Data" refers to the massive amounts of digital information companies and governments collect about human beings and our environment.

For the definition of the Big Data, there are various different explanations from 3Vs to 4Vs to 5Vs. Doug Laney used volume, velocity and variety, known as 3Vs, to characterize the concept of Big Data. The term volume is the size of the data set, velocity indicates the speed of data in and out, and variety describes the range of data types and sources. Sometimes, people extend another V according to their special requirements. The fourth V can be value, variability, or virtual [2]. More commonly, Big Data is a collection of very huge data sets with a great diversity of types so that it becomes difficult to process by using state-of-the-art data processing approaches or traditional data processing platforms.

The field of databases has emerged in last decades of years. New architectures try to meet the need to store more and more various kinds of diverse data [3]. Big Data and Data Intensive technologies are becoming a new technology trend in science, industry and business. Big Data are becoming related to almost all aspects of human activity from just recording events to research, design, production and digital services or products delivery to the final consumer. Current technologies such as Cloud Computing and ubiquitous network connectivity provide a platform for automation of all processes in data collection, storing, processing and visualization. Consequently, emerging data intensive technologies impose new challenges to traditional security technologies that may require rethinking and refactoring currently used security models and tools [4]

Security and privacy issues are magnified by the volume, variety, and velocity, veracity and value of Big Data. Large-scale cloud infrastructures, diversity of data sources and formats, the streaming nature of data acquisition and high volume intercloud migration all create unique security vulnerabilities. It is not merely the existence of large amounts of data that is creating new security challenges Software infrastructures such as Hadoop enable developers to easily leverage thousands of computing nodes to perform data parallel computing. Combined with the ability to buy computing power on demand from public cloud providers, such developments greatly accelerate the adoption of Big Data mining methodologies. As a result, new security challenges have arisen from the coupling of Big Data with public

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

cloud environments characterized by heterogeneous compositions of commodity hardware with commodity operating systems, and commodity software infrastructures for storing and computing on data.

As Big Data expands through streaming cloud technology, traditional security mechanisms tailored to securing small-scale, static data on firewalled and semi-isolated networks are inadequate. Database security has been and will continue to be one of the more critical aspects of information security. Access to enterprise database grants an attacker a great control over the most critical data. When choosing a database today the problem is much more complex to decide the best architecture for data storage and retrieval of data. The huge volume of data is getting collected and processed today, and it becomes mandatory to collect various kinds of structured and unstructured data and its use became an integral part and it adds richness to applications, popularly known as big data. Today, with the emergence of cloud, applications use a three-tier internet architecture that run in a public or private cloud that support big users and big data. Dealing with big users and big data using relational database technology becomes more and more difficult. The main reason is that relational databases depend on static schemas, and a rigid approach toward modeling the data.

Google, Amazon, Facebook, and Linkedln are among the first companies to discover the serious limitations of relational database technology for supporting big data and big user’s requirements. To overcome these limitations, these companies brought up with new data management techniques, their initiatives results in producing a large interest among several developing companies that were facing the related problems. As a result, a new database is designed with novel data management model called as NoSQL. Today, the NoSQL databases are rapidly growing and deployed in many internet companies and other enterprises. It’s considered as a viable choice when compared to relational databases, especially the performance and scalability requirements of big users and big data on a cloud environment can be successfully achieved by using NoSQL databases.

The NoSQL popularly used today is Open Source NoSQL. In this regard, many related organization and websites have ranked the NoSQL databases by its popularity and found similar result that the top five open source NoSQL databases are MongoDB, Cassandra, CouchDB, Hypertable, and Redis.

Based on the data storage model, NoSQL databases generally can be categorized into the following four groups [5]

Key-Values Databases: Store uninterpreted arbitrary data values into a system that can be recalled later using a key (hash). This schema less data model allows for easy scaling and very simple APIs for implementations.

Column Databases: Store data in a similar key-value fashion, except the key is a combination of column, row, and/or timestamp, which points to one or multiple columns (Column Family). The column family used here is like a table commonly found in a relational database.

Document Databases: Store documents that consist of one or more self-contained named fields in each document, like JSON or BSON format. The structure of documents is dynamic that allows for free modification with the ability to add or remove fields of existing documents. Indexing on the named fields enables fast data retrieval.

Graph Databases: Store data in a flexible graph model that scales across multiple machines. This model is suitable for data with relations that are best represented as a graph (elements interconnected with an undetermined number of relations between them), such as social relations, public transport links, road maps or network topologies.

Table I lists few examples of NoSQL categories used by the companies

TABLE I. NoSQL STORAGE TYPE USED BY DIFFERENT COMPANIES

Company Name	*NoSQL Name*	*NoSQL storage type*
eBay	Cassandra | MongoDB	Column | Document
Facebook	Cassandra	Column
LotsOfWords	CouchDB	Document
MongoHQ	MongoDB	Document
Mozilla	HBase	Column
Netflix	HBase | Cassandra	Column | Column
Twitter	Cassandra	Column

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

II. General Security Issues of NoSQL Databases

A. NoSQL databases only have a very thin security layer, compared to traditional RDBs. In general, the security philosophy of NoSQL databases relies on external enforcement mechanisms.
B. Clustering aspects of NoSQL databases pose additional challenges to the robustness of such security practices.
C. With current NoSQL security mechanisms, it is virtually impossible to segregate sensitive data pertaining to different cloud users sharing the framework's internal NoSQL database.
D. Lack of security standards has caused vendors to develop bottom-up NoSQL solutions and address security issues on an ad-hoc basis.
E. NoSQL architecture fails in its primary objective of attaining better performance and scalability when complex integrity constraints are introduced in NoSQL architecture.
F. NoSQL uses weak authentication techniques and weak password storage mechanisms. This exposes NoSQL to replay attacks and password brute force attacks, resulting in information leakage.
G. NoSQL uses HTTP Basic or Digest-based authentication which are prone to replay or man-in-the-middle attack.
H. NoSQL does not support integrating third party pluggable modules to enforce authentication. By manipulating the RESTful connection definition, it is possible to get access to the handles and configuration parameters of the underlying database, thereby gaining access to the file system. REST is a preferred communication protocol based on HTTP and is prone to cross-site scripting, cross-site request forgery, injection attacks.
I. Although some of the existing NoSQL databases offer authentication at the local node level, they fail to enforce authentication across all the cluster nodes.
J. NoSQL databases have inefficient authorization mechanisms. Authorization techniques differ from one NoSQL solution to another. Most of the popular solutions apply authorization at higher layers rather than enforcing authorization at lower layers. More specifically, authorization is enforced on a per-database level rather than at the collection level.
K. NoSQL databases are susceptible to Injection Attacks. Since NoSQL architecture employs lightweight protocols and mechanisms that are loosely coupled, it is susceptible to various injection attacks like JSON injection, array injection, view injection, REST injection, GQL injection, schema injection, etc. For example, an attacker can utilize schema injection to inject thousands of columns onto the database with data of the attacker's choice. The impact of such an attack can range from a database with corrupted data to a DoS attack resulting in total unavailability of the database.
L. The inability to simultaneously enforce all three elements of the CAP theorem (consistency, availability, and partition tolerance) while in distributed mode undermines the trustworthiness of the churned results. As a result, users are not guaranteed consistent results at any given time, as each participating node may not be entirely synchronized with the node holding the latest image. Current hashing algorithms entrusted to replicate data across the cluster nodes crumple in the event of a single node failure, resulting in load imbalance among the cluster nodes.
M. NoSQL databases are prone to Insider Attacks. Lenient security mechanisms can be leveraged to achieve insider attacks. These attacks could remain unnoticed because of poor logging and log analysis methods, along with other rudimentary security mechanisms. As critical data is stowed away under a thin security layer, it is difficult to ensure that the data owners maintain control.

III. Sequrity Features of NoSql Databases

Cassandra

i) The data files in Cassandra are kept unencrypted and Cassandra does not provide a mechanism to automatically encrypt the data in storage. This leads to any attacker with access to the file system can directly extract the information from the files[6].

In order to mitigate this, the application must explicitly encrypt any sensitive information before writing it to the data base. In addition operating system level mechanisms (file system permissions, file system level encryption, etc.) should be used to prevent access to the files by unauthorized users.

ii) All communications between the database and its clients is unencrypted. An attacker that is capable of monitoring the database traffic will be able to see all of the data as the clients see it. The client interface supports a login() operation, but both user name and password are sent across the network as clear text.

iii) Nodes on a cluster can communicate freely and no encryption or authentication is used. No protection or authentication is provided for fat client (interface used to load bulk data into Cassandra from different nodes in the cluster).

iv) Cassandra Query Language (CQL) is a parsed language it is vulnerable to injection attacks, exactly like SQL.

v) An attacker can prevent the Cassandra server from accepting new client connections by the causing the Cassandra server to allocate all its resources to fake connection attempts. The current implementation does not time out inactive connections, so any connection that is opened without actually passing data consumes a thread and a file descriptor that are never released.

vi) Authentication: Cassandra provides an IAuthenticate interface with two example implementations. The default implementation is one that turns off the requirement to authenticate to the database. The other provided implementation is SimpleAuthenticator.

SimpleAuthenticator allows setting up of users and passwords via a flat Java properties file. The file is formatted as sets of password properties, and the password can be specified either in plain text, or as an unsalted MD5 hash.

The client interface will always transmit the password as plain-text even if it is kept as an MD5 hash in the password file. This means that any attacker that is capable of sniffing the communication between a legitimate client and the database will be trivially discover the password.

MD5 is a hashing algorithm that is not cryptographically secure and it is easy to find appropriate plain-text to match a given MD5 hash using pre-calculated lists and rainbow tables found online. Another issue with the SimpleAuthenticator implementation is that the administrator must make sure that the flat password file found on each cluster member is synchronized with the rest of the cluster. Otherwise, the passwords used by each cluster member may not be the same for the same user.

vii) Authorization: Cassandra provides an IAuthority interface which is used in the Cassandra's codebase when a key space is being modified and one each column family access(read or write). Cassandra provides two implementation of IAuthority. The first is a pass-through implementation that always allows full permissions, regardless of the user, and the second called SimpleAuthority uses a flat Java properties file to allow matching permissions to user names.

The weakness in the SimpleAuthority implementation is that it depends on a flat file , and not on consistent file that is maintained across the cluster. This means that the effective permissions granted to a user are the permissions listed in the file located on the cluster member to which the connection was established. Another issue with the provided SimpleAuthority implementation is that it does not reload the file on every access. This means that the effective permissions cannot be changed without restarting the Cassandra process. Authorization in Cassandra is specified only on the existing column families, and therefore no protection is available on newly added column families and columns, and also protection of the raw (object) level is not available.

viii) Auditing: Cassandra doesn't support inline auditing[6].

MongoDB

i) MongoDB datafiles are unencrypted, and MongoDB does not provide a method to automatically encrypt these files. This means that any attacker with access to file system can directly extract the information from these files[6].

In order to mitigate this, the application must explicitly encrypt any sensitive information before writing it to the database. Operating system mechanisms should also be used to prevent access to the files by unauthorized users.

ii) The binary wire-level protocol supported by MongoDB for client interfaces is neither encrypted nor compressed and the internal HTTP server doesn't support TLS or SSL in any way.

The internal HTTP server can be hidden behind a HTTP proxy server like Apache HTTPD with the mod_proxy module and then the Apache HTTPD's robust authentication and authorization support can be used in addition to robust SSL encryption for the connection

iii) MongoDB heavily utilizes JavaScript as an internal scripting language. Because the JavaScript is the interpreted language there is a potential for injestion attacks

iv) When running in Shared mode, MongDB doesn't support authentication and therefore has no support for authorization.

vi) MogoDB doesn't provide any facilities for auditing actions performed in the database[6].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Redis

i) Redis provides password based authentication to its clients. These passwords are stored in plain text format and set by the system administrators[8].

ii) Redis does not provide authentication by default and listens on all IP addresses on port 6739.

iii) Redis does not ensure any kind of access control mechanisms and rely on third party SSL implementations for the security of data transmissions over untrusted networks.

iv) Redis does not provide support for configuration security, data encryption and auditing mechanisms[8].

CouchDB

i) CouchDB supports basic password based authentication as well as cookie based authentication for its users. Passwords in CouchDB are hashed using PBKDF2 hashing algorithm and are sent over the network using SSL for the security of data transmission[8].

ii) Access control in CouchDB only supports a single role i.e. admin party, for its users. Moreover, authorization is only implemented at database level.

iii) A very medium level auditing is provided to log views and events in log files. However, CouchDB does not provide automatic logging thus the configuration of logs is the responsibility of the database administrators. Automatic backups of database logs and replicas are also not supported in CouchDB database[8].

Hbase

i) HBase not only supports token based authentication for map-reduce tasks but user authentication is also supported by HBase. The user authentication is done by using SASL (Simple Authentication and Security Layer) with Kerberos on per connection basis. Additionally, authorizations in HBase are managed by ACL (Access Control Lists) or Coprocessors with column family level granularity and on per user basis[8].

ii) HBase also provides logging support up to data node level.

iii) Other high level auditing and monitoring features are absent in HBase along with configuration security and encryption of data-at-rest[8].

TABLE II. TECHNEQUES TO MITIGATE THE ATTACKS ON NOSQL DATABASES

Sl. No.	*Type of attack on NoSql Database*	*Mitigating Technique*
1	Injection attack	1. JSON queries 2. DAST and Static code Analysis
2	REST API exposure and CSRF attacks	1. Accept only JSON in the context type 2. Limit the HTML forms to URL encoded content type 3. AJAX requests for CSRF attacks 4. Disable JSONP and CORS in the server API to make sure that no actions can be made directly from a browser

IV. CONCLUSION

This paper in a nut shell can be stated as to assist in fulfilling the following objectives in regardance with the security of NoSQL databases

i. To address critical security flaws in NoSQL.
ii. To formalize the threat model for security of NoSQL and to find tractable solutions based on the threat model.
iii. To design and develop appropriate security mechanism for securing NoSQL databases.
iv. To implement the solution in existing infrastructures.
v. To verify the developed scheme using analytical and implementation models or simulation models for securing NoSQL.
vi. To identify security challenges those are expected in future for securing NoSQL.

REFERENCES

1. Min Chen, Shiwen Mao, Yunhao Liu, “Big Data: A Survey” Published online: 22 January 2014 © Springer Science+Business Media New York 2014.
2. C.L. Phili p Che n, C.-Y. Zhang, Data -intensive applications, challenge s, techniques and technologies: A survey on Big Data, Inform. Sci.(2014), http://dx. doi.org/10.1 016/j.ins .2014.01.01 5
3. Asadulla Khan Zaki, “NoSQL DATABASES: NEW MILLENNIUM DATABASE FOR BIG DATA”, International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308
4. Yuri Demchenko, Canh Ngo, Cees de Laat, Peter Membrey and Daniil Gordijenko, “Big Security for Big Data: Addressing Security Challenges for the Big Data Infrastructure”, Springer International Publishing Switzerland 2014, pp. 76–94,
5. Fidels Cybersecurity, “Current Data Security Issues of NoSQL Databases”, January 2014
6. L. Okman, N. Gal-Oz, Y Gonen, E. Gudes and J Abramov, “Security Issues in NoSQL Databases” in TrustCom IEEE Conference on International Conference on Trust, Security and Privacy in Computing and Communications, pp 541-547, 2011.
7. CSA, “Expanded Top Ten Big Data Security and Privacy Challenges”, CSAReport,16June2013https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf
8. Anam Zahid, Rahat Masood, Muhammad Awais Shibli, “Security of Sharded NoSQL Databases: A Comparative Analysis” 2014 Conference on Information Assurance and Cyber Security (CIACS).
9. Avin Ron, Alandra Sulmen-Peleg, Emanuel Bronshtein,No SQL, No Injection? Examining NoSQL Security.
10. HTTP://en.wikipedia.org/wiki/NoSQ

BIOGRAPHY

DADAPEER is a Asst. professor in the Dept. of Computer Science and Engineering at Ballari Institute of Technology and Management, Ballari, Karnataka, India. He is working on the smart security system for various databases.

N M INDRAVASAN is a student at Ballari Institute of Technology and Management, Ballari, Karnataka, India. pursuing his Bachelors’ Degree in the Dept. of Information Science and Engineering. His field of interest is in Data maintenance and security.

ADARSH G is a student at Ballari Institute of Technology and Management, Ballari, Karnataka, India. pursuing his Bachelors’ Degree in the Dept. of Information Science and Engineering. His field of interest is in Database Analysis and Design.