Proceedings of the 11th INDIACom; INDIACom-2017; IEEE Conference ID: 40353
2017 4th International Conference on "Computing for Sustainable Global Development", 01st - 03rd March, 2017
Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi (INDIA)

Critical Review on Threat Model of Various NoSQL Databases

Prof. (Dr.) Mohammad Ubaidullah Bokhari
Department of Computer Science, Aligarh Muslim University, Aligarh
Email ID: mubokhari@gmail.com

Afreen Khan
Department of Computer Science, Aligarh Muslim University, Aligarh
Email ID: khanafreen2010@gmail.com

***Abstract* - The present era of Big Data has revolutionized the entire computing scenario. The recent advancement in digital data has shifted its core working from traditional databases to NoSQL databases. As the data is growing at a fast pace, there arises an immense need to store them securely. Security and privacy issues are not being validated in NoSQL data stores thoroughly. There are many loopholes that need to be focused upon. With growing amount of data, at every nanosecond, there's an immediate requirement to have knowledge related to all the challenges, loopholes and plug-ins that are exposed to attacks in account of NoSQL databases. The aim of this paper is to analyze and present the threat model related to various NoSQL databases. It further assesses the challenges that need to be covered so as to develop more secure NoSQL data stores. A comparative study has been presented with respect to top ten NoSQL databases commonly used in present time.**

Keywords – NoSQL; Key-Value Stores; Column-Family Stores; Document Databases; Graph Databases; Security

I. INTRODUCTION

The rapid increase in the bulk of data relating to web technologies, mobile applications, and social media sites has increased the production of unstructured data from terabytes to petabytes. The word "unstructured" relates to the very famous terminology of today's times "NoSQL". NoSQL is a database that deals with the unstructured data very efficiently. This is in contrast with the classic relational databases that don't fit well for the unstructured data rather they are meant for dealing with the storage of the structured data. NoSQL databases are not meant to degrade relational databases. According to the problem, specific database is used. Clearly, it is nothing but problem and need specific.

While assessing the needs of an application, a question that arises is whether to make use of NoSQL engines or relational databases. It chiefly depends on the type of application being written, the nature of queries that are acknowledged, and the constancy vs. unpredictability of the data's structure [1].

To deal with the tons of unstructured data, an issue of 'security' is of more importance than other issues and challenges. The security breaches in the various NoSQL data stores are not validated well. There lies in them certain loopholes, which need an immediate attention so as to nullify them. To secure our data especially the one which is on-line and is in unstructured form is of foremost importance. To safeguard it from the evil hands is the need of the hour.

II. OVERVIEW OF NOSQL MODELS

In the make-up process of NoSQL, this datastore is categorized into four branches. They are: Key-Value Stores, Document Databases, Column-Family Stores, and Graph Databases. Each database has its own primary use, advantages, and disadvantages.

Key-Value Stores are well suited when one needs to deal with the applications relating to the session management in web applications, shopping cart transactions, managing user profiles. It is the simplest NoSQL datastore. The entire working lies on the two fundamental elements i.e. key and a value. This is said to be a key-value pair. The "key" is represented in the form of a string while "value" can be any kind of data and is stored as a BLOB. Some examples of Key-Value Stores are DynamoDB, Oracle NoSQL, Riak and Redis. The document databases are the databases which are most suitable for the storage, retrieval and management of document related information in the form of semistructured data such as email messages, text documents, XML, JSON, BSON documents, etc. The functioning of document databases consists of a key-value pair. Unlike Key-Value Stores, the key is paired with an intricate data structure called as "document". Examples of Document databases are: MongoDB, CouchDB etc. Column-Family Stores are mostly used for applications involving distributed data storage. This NoSQL category is suitable if one needs to handle a large amount of data scattered over many servers. As is the name, so is the functioning i.e. these databases are column-oriented. The data is stored in multiple columns together instead of rows of data. Google

Copy Right © INDIACom-2017; ISSN 0973-7529; ISBN 978-93-80544-24-3

BigTable, HBase, Cassandra are some of the examples of Column-Family Stores. The graph databases are an excellent choice when one has to tackle with the management of relationships among the objects likewise social networks, pattern detection, etc. Graph databases works on the norms of graph theory and consist of edges (relationships), nodes (entities) and properties (attributes). The data is retrieved through the pointers which is typically stored in each element. An example of Graph databases are: Neo4j. Key-value datastores, document databases and column-family datastores are appropriate for wide range of applications [1]. Whereas graph databases are an ideal fit to a particular kind of problem [1].

III. MAJOR SCENARIO OF THREAT REPRESENTATION IN NOSQL DATA STORES

As documented by Cloud Security Alliance, the threat representation of NoSQL data stores has six chief settings [2]. They are discussed below:

A. Transactional Integrity

The majority of the NoSQL systems are unsuitable replacements for the traditional databases in transaction processing applications as they are deficient in full ACID (Atomicity, Consistency, Isolation, Durability) properties for assuring transactional integrity and data consistency [3]. Complicated integrity constraints obstruct NoSQL's performance and scalability; which is indeed the greatest security risk. As an alternative, they are BASE (Basically Available, Soft state, Eventually consistent) compliant as they do not adhere to the ACID compliancy.

B. Weak Authentication Mechanisms

Authentication is a process which involves the verification of the identity of the client or any device. It is usually achieved through the password mechanisms. But NoSQL data stores uses weaker authentication methods and feeble password storage techniques [2]. This in turn results in information leakage while exposing NoSQL to password brute force attacks as well as to replay attacks [2].

C. Insufficient Authorization Techniques

If the credentials provided during the authentication phase matches to those stored in some database file of authorized users' information, a next phase called as, authorization, is granted to the clients for further access. Certain NoSQL data stores employ very simple authorization methodologies without support for the RBAC (Role Based Access Control) mechanism or fine-grained control. It is applied to the higher layers than being enforced at lower layers [2].

D. Susceptibility to Injection Attacks

In the injection attack, the attacker injects data into a web application so as to transform the data which aids in the execution of malevolent data in an unexpected way. The injection attacks are the most widespread attacks because they are complex and exist in many forms. Therefore, getting rid of them and securing our own data becomes difficult. NoSQL makes use of lightweight protocols and techniques that are not highly intact between the client and server and also for communication across the participating cluster nodes [2]. NoSQL is comparatively more susceptible to a variety of injection attacks such as, array injection, view injection, REST injection, SQL injection, etc. Furthermore, certain NoSQL databases are prone to DoS (Denial of Service) attacks, thereby, resulting in the complete unavailability of the datastore [2].

E. Lack of Consistency

When the term consistency comes in with regard to the NoSQL databases, typically it is referred to the CAP-theorem (Consistency, Availability, Partition tolerance). It is one of the characteristic features of NoSQL databases that these data stores do not adhere strictly to the three elements of CAP-theorem simultaneously. Its core property is that only two of the three different aspects can be completely achieved simultaneously. Hence, users are not assured consistent output at any known time, because every participating node may not be completely coordinated with the joint holding the most recent data [2].

F. Insider Attacks

An insider attack refers to any sort of malicious attack that is committed on the computer or the network system. It is performed by a known and authorized person who has been given the authoritative credential in order to access the system. These people usually have the knowledge of network infrastructure and other network policies. Generally less security is employed against the insider attacks because the organization targets to safeguard itself from the external attacks. Many NoSQL databases employ poor security mechanisms which make it vulnerable to the insider attacks. Such attacks could stay ignored because of poor logging and log analysis mechanisms, all along with the other basic security techniques [2]. Since, critical data is kept under a thin security layer, it is very hard to make sure that the data owners sustain control [2].

IV. THREAT ISSUES OF VARIOUS NOSQL DATABASES

The NoSQL movement has led the organizations to organize their unstructured or, semistructured data in a more profound way. The ultimate aim of any datastore is to provide security to the organizations, its users, clients and vendors. As NoSQL's popularity is increasing exponentially, and at the same time it hoards up huge amounts of user sensitive data, it poses an immense challenge to look upon the security and privacy of the various NoSQL databases in order to better safeguard them and protect them from a variety of threats.

The threat issues related to top ten commonly used NoSQL databases are categorized in this section. The ten NoSQL databases considered are: Amazon DynamoDB, Google

Copy Right © INDIACom-2017; ISSN 0973-7529; ISBN 978-93-80544-24-3

BigTable, Apache HBase, MongoDB, Oracle NoSQL, Neo4j, Apache CouchDB, Riak KV, Apache Cassandra, and Redis. These databases are further discussed on the six key points mentioned in the Section III.

A. Amazon DynamoDB

It is a completely managed Key-Value Store NoSQL database. It is a product of Amazon. DynamoDB offers fast and conventional performance along with seamless scalability. The functioning of DynamoDB relating to the transactional integrity is that they provide substantially infinite scalable read-write I/O successively running on IOPS (Input/Output Operations Per Second) optimized solid state drives and possess predictable performance [4]. DynamoDB supports transactions. Every write operation is atomic to an item; all of the item's attributes are successfully updated by a write operation or none at all and hence, there don't exist any multi-operation transactions [5]. For authentication, it provides triggers along with the integration of AWS Lambda that allow building the applications which thereby automatically react to any data changes [6]. It articulates with the AWS Identity and Access Management (IAM) so as to provide fine-grained access controls to the clients within the organization [6]. Also, it can allocate unique security authorization to each and every client and thus can control each client's access to the resources and services [6]. It is comparatively less susceptible to SQL injection attacks; the whole lot is an API query parameterized [6]. It possesses eventual consistency and delivers fast performance and consistent state, at any scale for all of the applications . DynamoDB make use of SSD technologies and automatic partitioning, as the data volume grows and application performance load amplify, so as to meet the throughput needs and offer low latencies at any degree [6]. A cryptographic protocol is developed to guard against damaging, intruding, and message counterfeit through a connection made to an AWS access point via HTTPS or HTTP using SSL [7]. It uses a method of anti-entropy by means of the Merkel trees so as to recover from permanent failures; failure detection and Gossip-based membership protocol is also used to tackle with the insider attacks [8].

B. Google BigTable

It is a Google's NoSQL database service. Google BigTable is a Column-Family Store NoSQL database that can handle huge amount of workloads at consistent small latency and elevated throughput. It is best suitable for both analytical and operational services. With respect to the transactional integrity, it is massively scalable, fast performant, scales to hundreds of petabytes without human intervention, can handle millions of operations per second, possess an advanced performance under high load than other products, changes made to the deployment configuration are instantaneous thus, there is no downtime throughout the reconfiguration, supports transactions on a big entity group in principle to the numerous continuous rows lock, doesn't support SQL joins or queries, and doesn't own any support for multi-row transactions [9]. In order to provide authentication, it dynamically add or remove the cluster nodes and OAuth 2.0 is used for API authentication and authorization [9]. To employ BigTable, HDFS cluster or some other file system must be executed. Granular access controls and OAuth offer strong and configurable security, which is available by means of the extensions through the HBase 1.0 API and as a result, it is compatible with a bunch of the Hadoop network as well as with many other big data tools [9]. BigTable employs GQL (Google Query Language) which utilizes parameters to create the query instead of string concatenation and hence, one can't inject anything as it is a read-only language [10]. Thus, BigTable is less susceptible to SQL injection attacks. Google BigTable is developed on the norms of GFS (Google File System), which was basically developed for consistency and it offers both consistency and replication in the best possible form. It is devised to handle enormous workloads at consistently lesser latency and higher throughput, thereby encompassing strong consistency [9]. To protect from the insider attacks, 'gsutil' is used that executes every operation by means of transport layer encryption (HTTPS) [9]. 'gsutil' make use of bearer tokens for resumable upload identifiers and for authentication (OAuth2); therefore, such kinds of tokens should be protected from being reused and eavesdropped [9]. Data is encrypted both, at-rest and in-flight and thus, it possesses a full control over who accesses the data that is being stored in BigTable.

C. Apache HBase

HBase is a Column-Family datastore. It is an open source and distributed database built after Google's BigTable. It possesses modular scalability and is basically available, in regard to the BASE property [11]. It is not an ACID compliant database but still, it does guarantee some specific properties with respect to the transactional integrity. It supports SASL authentication of the clients and requires secure HDFS and ZooKeeper; as a result the customers cannot access and/or alter the data form and metadata under HBase [12]. The ZooKeeper provides a pluggable authentication method; the Access Control Layers (ACLs) are provided per znode so as to restrict the access to znodes. HBase based daemons usually authenticate to ZooKeeper via Kerberos and SASL [12]. Role-base controls are provided, which clients or certain groups can read or write only to a given HBase resource. The visibility labels permit to label cells and thus, control access to the labelled cells. HBase are less susceptible to SQL injection attacks as it possesses automatic failover support between the RegionServers. REST-ful web service and Thrift gateway supports Protobuf, XML, and binary data encoding alternatives thereby, supporting Bloom Filters and Block Cache for large quantity of query optimization [12]. It is strongly consistent and makes it appropriate for jobs like high-speed counter aggregation. It possesses transparent encryption of the data at-rest on the fundamental file system, both- in WAL and HFiles which in turn guards data at-rest from an attacker who has access to the file system [13]. It also protects against the data leakage from inappropriately disposed disks

Copy Right © INDIACom-2017; ISSN 0973-7529; ISBN 978-93-80544-24-3

[13]. If SASL is not being in use, then by looking only at RDC's on the wire is insignificant to perform replay attack [14]. Having extra layers such as REST or Thrift, there are still limitations that must be considered when live traffic is exposed to HBase so as to avoid DDoS or many other attacks .

D. MongoDB

It is an open-source and free cross-platform Document NoSQL datastore. This datastore is considered as the most accepted and popular NoSQL datastore, thereby challenging today many other NoSQL databases. MongoDB employ a flexible schema and it's very well known document structure in a grouping called as, Collection, may differ and common fields of different documents in a collection may have unlike types of the data [15]. It is scalable and possesses high availability (through involuntary failover in replica sets). It has risen from single server deployments to clusters with more than 1,000 nodes, supplying millions of actions per second on more than 100 billion records and petabytes of data [17]. The applications that need scan-oriented applications and complex transactions that use huge subsets of the data, may not fit well for MongoDB. It supports ACID transactions at the document level but doesn't hold up multi-document transactions [18]. To enable access control, it involves authentication of each client. MongoDB supports numerous authentication methods such as, SCRAM-SHA-1 and MongoDB Challenge and Response (MONGODB-CR) [16]. It uses RBAC to manage access to MongoDB system. Basically, a client is granted with one/more roles that decide the client's access to database operations and resources and outside of roles that are assigned, the client has no further access to the system [16]. As a client program gathers a query in MongoDB, it creates a BSON object and not a string because MongoDB represents the queries as BSON objects [16]. Therefore, client libraries give injection free and convenient method that help in building these objects. MongoDB adopts a locking system to assure data set consistency thereby, offering high availability and low latency [16]. It is robustly consistent by default i.e. if one carries out a write operation and after that a read operation, considering the write was actually successful, then the individual will forever be capable to read the output of the write he/she just read. It employs transparent data-at-rest encryption, possess protection against a malevolent privileged user, and non-repudiation is through the dual-logging mechanism. MongoDB supports robust key management in which the encryption keys can be stored within cloud key lockers or in an on-premises machine [19]. The Server General data security engines such as key management, encryption, log management, and advanced access controls, defeat illegal privileged access, theft of media, insider attacks, and log tampering.

E. Oracle NoSQL

This database is a Key-Value NoSQL datastore developed by Oracle Corporation. It offers transactional semantics for horizontal scalability, data manipulation, data administration and monitoring [20]. It is scalable (automatic query and load balancing, dynamic distribution and data partitioning) and distributed [21].It is developed to provide flexible, reliable and available management of data across a configurable deposit of storage nodes. It possesses high availability with local and isolated failover and synchronization. It has easy programming models along with the ACID transactions, JSON support, and tabular data models [21].The support for client authentication has been added by means of Kerberos since it allows for Oracle NoSQL datastore to be simply integrated with client's active applications that are genuinely protected by Kerberos. Oracle Wallet integration, and cluster wide password-based client authentication provides better protection from illicit access to sensitive data [21]. The role-based authorization has been improved so as to support the security operations in the DDL, user-defined roles, and table-level authorization [22]. As it is a Java based Key-Value Store database, it supports a value abstraction layer that implements JSON and BSON types. Oracle NoSQL is less susceptible to SQL injection attacks and there are certain approaches that are used to avoid SQL injection vulnerabilities such as, wherever possible, try to avoid utilizing string-building methods to generate SQL and make use of bind variables wherever feasible enough instead of string concatenation [23]. It own bounded latency, scalable throughput, high volume random reads and writes, and configurable consistency. Oracle NoSQL is secured with the session-level SSL encryption and application security along with the authentication. The network port restrictions and session-level encryption offers better security from network intrusion thereby, providing better mechanisms to protect from the insider attacks [23].

F. Neo4j

Neo4j is a Graph database that is developed by Neo Technology, Inc. Presently, Neo4j is the world's most leading graph datastore and is the largely accepted graph database as conceived by db-engines.com [24]. It is highly scalable, have high performance and reliability, fully ACID compliant to guarantee predictability of relationship-based queries, delivers speedy read and write performance while shielding the data integrity, and extend across key dimensions of scale, such as volume, reads and writes; all while giving reliable query response times, blazing fast-queries, and steadfast data integrity [25]. The transactions are atomic, durable, and consistent in nature; but ultimately propagate to other slaves. It scales up and down, providing support to billions of nodes and relationships, and numerous thousands of ACID transactions per second [26].Data replication and clustering are demanded by operational and transactional applications.The REST API provides support for authentication and it requires customers to provide authentication credentials while accessing the REST API [25]. If the credentials provided do not match then, access to the database will be prohibited. The query language that is used for Neo4j is 'Cypher'. For writing conventional special-purpose extensions, it employs native Java API. It provides support for exporting of query data to XLS and JSON format.

Copy Right © INDIACom-2017; ISSN 0973-7529; ISBN 978-93-80544-24-3

The REST API supports authorization too and offers support for chained SSL certificates [25]. If a user is handcrafting the Cypher queries, then he/she is at risk to SQL injections. It is eventually consistent and index-free adjacency lessens read time and supplies extremely high and parallelized throughput even when the data increases. It exposes difficult-to-detect patterns that far exceed traditional design such as tables. It provisions interconnected data that is neither linear nor wholly hierarchical, building an easy interface to identify rings instead of the shape or depth of the data. Neo4j's inhabitant Graph Processing Engine wires high-performance graph queries on vast datasets to facilitate real-time fraud detection. An integrated support for SSL encrypted communication over HTTPS is provided in Neo4j in which a self-signed SSL certificate and a confidential key are produced without any human intervention on start of the server. 'CyGraph' is a tool that is used for cyber warfare visualization, analytics, and knowledge management [25]. The REST web services in CyGraph offers interface for analytics, data ingest, and graph visualization.

G. Apache CouchDB

Apache CouchDB is a Document NoSQL datastore. It is an open-source database and focus on ease of use. It is highly available, offer a RESTful API for reading and updating (add, delete, edit) datastore's documents, its commitment system and file layout contains all ACID properties. The authentication is provided through the Basic Authentication (RFC 2617) which is a fast and easy way to authenticate with CouchDB [27]. A major disadvantage with this Basic Authentication is the requirement to send client credentials with every request which may be not secure and may injure operation performance [27]. SSL is basically used in combination with HTTP so as to protect the web traffic i.e. HTTPS. CouchDB supports Cookie Authentication, OAuth 1.0 Authentication, and Proxy Authentication. It has the capability to support OAuth credentials within the client documents in lieu of config file (OAuth Configuration and HTTP OAuth Configuration). The CouchDB API is developed to supply a suitable but thin covering around the database core. In place of locks, this datastore make use of MVCC (Multi-Version Concurrency Control) to handle simultaneous access to the database [28]. To store data, it employs JSON, HTTP for an API, and JavaScript as its query language by means of MapReduce [29]. In earlier versions of CouchDB, the injection attacks included: Cross Site Request Forgery Attack Vulnerability (0.11.1) and Timing Attacks Vulnerability (0.11.0) [27]. While in the newer version, it is more likely to be less susceptible to the injection attacks. CouchDB is eventually consistent so as to provide both partition tolerance and availability, possess fault tolerant storage engine that places the security of the data first, and has a simple update validation illustration and easy reader access that can be further extended to execute custom security models [30]. With regard to the insider attacks, it implements document encryption on per client basis but after this, queries become a problem which in turn acts as a constraint.

H. Riak KV

It is a distributed Key-Value database. Riak KV is an open-source version. Apart from this, it also comes in a cloud storage version and supported enterprise version. It makes use of the principles from Amazon's DynamoDB with deep influence from the CAP-theorem [32]. It offers operational simplicity, high availability, scalability, fault tolerance, and CAP-theorem compliant. With respect to the scalability, it elastically grows and shrinks the cluster nodes while consistently balancing the weight on each machine. It supports HTTP basic authentication which consists of two primary interfaces: Protocol Buffers and HTTP [34]. It uses Erlang distribution technique for the inter-node communication and offers RESTful API through the Protocol Buffers and HTTP, for fundamental functions. Each and every request to Riak will fail, if a user enables security devoid of having created a functioning SSL connection. It queries data with MapReduce and doesn't have fine-grained role-based security [34]. Riak is a binary storage. If an injection attack is performed, it will be made against the Ruby part, not against the Riak. The earlier version of Riak had a susceptibility of SSL Version 3 which was revealed as an insecure through an attack on POODLE (Padding Oracle On Downgraded Legacy Encryption); however in Riak 2.0.5, it is made fixed [35]. It has a predictable latency, eventual consistency, and remains extremely available in the face of network partitions, server crashes, or other unavoidable disasters. It requires a safe and sound SSL connection; need to produce suitable SSL certificates, facilitate SSL and create a certification configuration on every node [34]. As soon as security is facilitated, all user connections should be encrypted and by default, all permissions will be denied [34]. Riak CS doesn't presently support encryption of data-at-rest [33].

I. Apache Cassandra

It is a Column-Family database. It is free, open-source and distributed NoSQL database. It is an ideal choice when one needs to have scalability and elevated availability devoid of compromising performance. It offers linear scalability, proven fault tolerance, and CAP-theorem compliancy [36]. The authentication is based on internally guarded login accounts/ password. CQL (Cassandra Query Language) is employed which is a SQL-like option to the conventional RPC interface [37]. CQL possess a very simple API and an abstraction layer is added that hides the implementation information of the structure and hence, offers native syntaxes for common encodings and other collections. It provides a support for replication, and for multi data center replication for failover, redundancy, and disaster recovery and to support all the authorization-related CQL queries, it stores permissions in the system_auth.permissions table. Injection attacks are possible in CQL [38]. If Java driver is employed, then preference to the PreparedStatements is given rather than Statements and it is

must to execute input validation in the application. PreparedStatements, introduced in Cassandra v1.1.0, assist to avoid such attacks [37]. It offers lower latency, and it consistently outperforms i.e. tunable consistency. No single point of failure, and no network barriers are present because each node in the cluster is the same, hence it is decentralized. It contains user to server SSL which assures that the data in-flight is not negotiated and is securely moved back/forth from user machines i.e. Node-to-Node encryption and Client-to-Node encryption [37].

J. Redis

It is a Key-Value Store NoSQL database. It is an open source datastore and is freely available. It is helpful for read scalability (but not write) and data redundancy, a CAP-theorem compliant, and is not well suited for utmost security; but is optimized for highest performance and simplicity [41]. In requisites of ACID properties, atomicity is guaranteed, consistency is guaranteed, isolation is always guaranteed but at command level, and durability- can only be guaranteed when AOF (Append Only File) is active [39]. It doesn't implement access control; instead it offers a small layer of authentication which optionally turns on while editing the redis.config file. The purpose of the authentication layer is to freely provide a layer of redundancy. If any other system or firewalling that is implemented to guard Redis from the external attackers fail, then an outside user won't be able to access the Redis instance without the mere information of the authentication password. If any untrusted access to Redis is made, then it must always be mediated by a coating that implements ACLs [40]. It employs a cluster design; but the cluster trait is presently in Beta stage. As soon as the authorization layer is activated, Redis doesn't further take any query by unauthenticated users and there is no rollback method. Injection attack is quite possible under usual circumstances by means of a normal client library because Redis protocol has no notion of string escaping [40]. 'Lua' scripts are generally executed through the EVALSHA and EVAL commands which are safe enough to implement [41]. The Redis protocol makes use of prefixed-length strings and is totally binary safe. It provides data consistency in the terms of CAP-theorem; not in the sense of ACID [40]. The AUTH command that Redis employs, is usually sent unencrypted, therefore it doesn't protect in opposition to an attacker that has sufficient access to the network to execute eavesdropping. 'Spiped' is a service for producing authenticated and symmetrically encrypted pipes among the socket addresses or, an added layer of security be implemented, likewise an SSL proxy. In order to prevent from a particular attack, Redis makes use of a pre-execution pseudo-random kernel to the hash function. A single FLUSHALL command may be used by an outside attacker to erase the entire dataset, thereby exposing Redis to insider attacks to a certain level but comparatively less [40].

V. KEY RESEARCH FINDINGS

This section includes the key findings of the study that's been done. It's been laid down in the form of a comparison chart in between the above discussed ten NoSQL databases, as shown in Table I.

From Table I it can be seen that Apache HBase, MongoDB, Riak KV, Apache Cassandra and Redis are not fully ACID compliant rather they are best suitable for the BASE feature. There is still a weak authentication in Cassandra and Redis while all other NoSQL datastores employ secure authentication techniques. With respect to authorization, Riak KV employs REST API but doesn't have fine-grained role-based security. Also, Cassandra doesn't have any support for RBAC or fine-grained authorization while Redis has moderate authorization facility. Apache Cassandra is susceptible to injection attacks while all other are less susceptible. It depends what level and what kind of injection attack is being used. These ten NoSQL datastores work consistently better in a given environment. CouchDB, Cassandra, and Redis are more likely prone to insider attacks. They are not fully secure regarding this particular attribute.

VI. RESULT

The study being done suggests that how to further improve different NoSQL databases and what advanced security mechanisms need to be implemented. The lacking of various important features in most of the NoSQL data stores (as mentioned in Section V), depicts and hence forces us to realize and employ with more sophisticated methods so as to better safeguard these databases. The transactional integrity needs to be imposed through an application or middleware layer, so as to achieve better scalability and performance. The finest approach is to implement security in the middleware layer than on the cluster level. Majority of the middleware software possess convenient support for authentication, authorization and various access controls. In case if Java is used, JAAS (Java Authentication and Authorization Services), Spring Security frameworks, ORACLE Corp. J2EE or SpringSource are offered for authentication, authorization and access control. Hence, it becomes a necessity to encrypt susceptible database fields or any other sensitive information. The unencrypted data must be kept in a sandboxed environment from where it can't be accessed by unauthorized individuals. Strong input validation must be used so as to make sure wherever the data is going, it is valuable and worth. Strong and sufficient authentication strategies must be employed such as LDAP and Kerberos methodologies in order to thwart the intruders. In contrast to Kerberos authentication, NTLM (Microsoft Windows NT LAN Manager) is also used for user authentication. Certain tools like web application security scanners and vulnerability scanners are used to combat with SQL injections and web security attacks.

TABLE I
NoSQL THREAT MODELCONCLUSION

NoSQL Datastores										
Traits		***Amazon DynamoDB***	***Google BigTable***	***Apache HBase***	***MongoDB***	***Oracle NoSQL***	***Neo4j***	***Apache CouchDB***	***Riak KV***	***Apache Cassandra***	***Redis***
Database Model		Key-Value	Column	Column	Document	Key-Value	Graph	Document	Key-Value	Column	Key-Value
Transactional Integrity	**ACID**	Yes	Yes	Partial	Partial	Yes	Yes	Yes	Partial	Partial	Partial
	BASE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Transactions	Yes	Yes	No	No	Partial	Yes	No	No	No	Yes
Authentication		Integrates AWS Lambda	OAuth 2.0 for API authentication	-SASL authentication -Requires secure ZooKeeper and HDFS	Supports multiple authentication mechanism	Using Kerberos	REST API	Cookie, Proxy OAuth 1.0	HTTP and Protocol Buffers	Weak	Weak
Authorization		Integrates AWS IAM	OAuth and granular access controls	RBAC control	RBAC control	Role-based	REST API	HTTP OAuth Configuration and OAuth Configuration	REST API but doesn't have fine-grained role-based security	No support for RBAC or fine-grained authorization	Moderate
Susceptibility to Injection Attacks		Less susceptible	Less	Less	Less	Less	Moderate	Less	Moderate	Vulnerable	Moderate
Consistency		Eventual	Strong	Strong	Strong	Configurable	Eventual	Eventual	Eventual	Consistently outperforms	Eventual
Insider Attack		Strong security mechanism	Strong encryption	Transparent encryption	Strong security mechanism	Session-level SSL encryption	Strong security mechanism	Less	Moderate	Less	Less

VII. CONCLUSION

Big data evolved when the data growth reached heights. It came up with the very idea of not going anywhere. In this paper, the security problems are highlighted which need to be addressed so as to make Big Data processing and computing framework more secure. Components specific to Big Data that arise from the usage of numerous infrastructure tiers (both computing and storage) for dealing out Big Data; the use of new compute framework such as NoSQL databases (for speedy throughput impelled by big data volumes) that have not been thoroughly vetted for security concerns; the non-scalability of encryption for bulky data sets; the non-scalability of real-time monitoring techniques that might be realistic for smaller volumes of data; the heterogeneity of devices that generate the data; and the misunderstanding of the varied legal and strategy restrictions that direct to ad hoc approaches for ensuring privacy and security [2]. Loads of items in this list serve to elucidate particular aspects of the

Copy Right © INDIACom-2017; ISSN 0973-7529; ISBN 978-93-80544-24-3

attack facade of the total Big Data processing framework which must be analysed for these threats**.** The aim of this paper was to give a thorough overview to the NoSQL database movement which appeared in the latest years to present alternatives to the chief RDBMSs.

The idea behind the present work was to throw the light on the security loopholes in various NoSQL data models. The security breaches in the various NoSQL data stores are still not thoroughly validated. They need to be focused upon so as to provide a secure and safe environment to the organizations and the users. This will hopefully spur action in the research and development area to collaboratively focus on the barriers to greater security and privacy in big data platforms.

ABOUT THE AUTHORS

Prof. (Dr.) Mohammad Ubaidullah Bokhari is a Chairman at Department of Computer Science, Aligarh Muslim University, Aligarh, India. Afreen Khan completed her MCA from Department of Computer Science, Aligarh Muslim University in November'2016 and is a budding researcher in the field of Data Science at Department of Computer Science, Aligarh Muslim University, Aligarh, India.

REFERENCES

[1] M.U. Bokhari and Afreen Khan, "The NoSQL Movement", IOSR Journal of Computer Engineering 2016;18(6): pp 06-12. Available: http://www.iosrjournals.org/iosr-jce/papers/Vol18-issue6/Version-4/B1806040612.pdf

[2] A Cloud Security Alliance Collaborative research, "Expanded Top Ten Big Data security and Privacy Challenges", April 2013, https://cloudsecurityalliance.org/research/big-data/

[3] (2016) Guide to NoSQL databases [Online]. Available: http://searchdatamanagement.techtarget.com/essentialguide/Guide-to-NoSQL-databases-How-they-can-help-users-meet-big-data-needs

[4] (2015) Amazon DynamoDB: ten things you really should know [Online]. Available: cloudacademy.com/blog/amazon-dynamodb-ten-things/

[5] The AWS Documentation website. [Online]. Available: docs.aws.amazon.com/amazondynamodb/.../developerguide/.html

[6] The AWS Lambda website. [Online]. Available: https://aws.amazon.com/lambda/faqs/

[7] Amazon's White Paper, "Overview of Security Processes", https://d0.awsstatic.com/whitepapers/aws-security-whitepaper.pdf

[8] The Amazon's Dynamo website. [Online]. Available: www.allthingsdistributed.com/2007/10/amazons_dynamo.html

[9] The Cloud Bigtable Documentation website. [Online]. Available: https://cloud.google.com/bigtable/docs/

[10] The GQL Reference website. [Online]. Available: https://cloud.google.com/datastore/docs/apis/gql/gql_reference

[11] The Apache HBase website. [Online]. Available: https://hbase.apache.org/acid-semantics.html

[12] The Apache HBase Reference Guide. website. [Online]. Available: https://hbase.apache.org/0.94/book.html

[13] Shui Yu and Song Guo, *Big Data Concepts, Theories and Applications*, Springer International Publishing Switzerland 2016

[14] (2014) HBase User- hbase attack scenarios? [Online]. Available: apache-hbase.679495.n3.nabble.com/hbase-attack-scenarios-td4062360.html

[15] The MongoDB on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/MongoDB

[16] The MongoDB Documentation website. [Online]. Available: https://docs.mongodb.com/

[17] The MongoDB at Scale website. [Online]. Available: https://www.mongodb.com/mongodb-scale

[18] The How ACID is MongoDB website. [Online]. Available: https://dzone.com/articles/how-acid-mongodb

[19] The MongoDB Encryption website. [Online]. Available: https://www.servergeneral.com/solutions/mongodb/

[20] The Oracle NoSQL on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Oracle_NoSQL_Database

[21] "Data sheet: Oracle NoSQL Database," www.oracle.com/technetwork/products/.../nosql-database-data-sheet-498054.pdf

[22] The Configuring Privilege and Role Authorization website. [Online]. Available: https://docs.oracle.com/cd/B28359_01/network.111/b28531/authorization.htm

[23] The Preventing SQL injection website. [Online]. Available: https://docs.oracle.com/cd/E58500_01/.../task_PreventingSQLInjection-0749b7.html

[24] The Neo4j on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Neo4j

[25] The Neo4j website. [Online]. Available: https://neo4j.com/

[26] The Neo4j Graph Database Solutions website. [Online]. Available: www.infoa.com/neo4j-graph-database-solutions

[27] The Apache CouchDB 2.0 Documentation website. [Online]. Available: http://docs.couchdb.org/en/stable/

[28] The CouchDB: The Definitive Guide website. [Online]. Available: guide.couchdb.org/draft/consistency.html

[29] The Apache CouchDB website. [Online]. Available: couchdb.apache.org/

[30] The Apache CouchDB: Technical Overview website. [Online]. Available: http://people.apache.org/~jan/couchdb.org.new/couch-site/htdocs/docs/overview.html

[31] (2015) CouchDB Authentication [Online]. Available: stackoverflow.com/.../how-do-i-make-private-docs-alongside-public-docs-in-a-couch.html

[32] The Riak on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Riak

[33] The Riak KV Basho website. [Online]. Available: http://basho.com/products/riak-kv/

[34] The Security Basics website. [Online]. Available: docs.basho.com/riak/kv/2.1.4/using/security/basics/

[35] (2014) Do developers need to sanitize JSON input before sending to Riak Client? [Online]. Available: lists.basho.com/pipermail/riak-users_lists.basho.com/2014.../016402.html

[36] The Apache Cassandra website. [Online]. Available: http://cassandra.apache.org/

[37] The DataStax Docs website. [Online]. Available: https://docs.datastax.com/en/

[38] (2015) DataStax Java Cassandra Driver [Online]. Available: stackoverflow.com/questions/.../is-datastax-java-driver-vulnerable-to-injection-attack

[39] The Redis Consistency Features on MediaWiki. [Online]. Available: https://quabase.sei.cmu.edu/mediawiki/index.php/Redis_Consistency_Features

[40] The Redis website. [Online]. Available: http://redis.io/

Copy Right © INDIACom-2017; ISSN 0973-7529; ISBN 978-93-80544-24-3