International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

Challenges and Security Issues of NOsql Databases

Dr. Deepak Chahal[1] , Dr.Latika Kharb[2] , Manhar Gupta[3]
[1,2]Associate Professor (IT), Jagan Institute of Management Studies, New Delhi. India
[3]MCA, Department of IT, Jagan Institute of Management Studies, New Delhi. India

ABSTRACT

In this paper we will be discussing about what are the security problems in NOSQL, what concerns has to be taken care to solve the security issues in NOSQL while the data is in moving or in rest, during authentication and authorization. This paper also includes the vulnerabilities in NOSQL as well as Injections in NOSQL from which un-authorized person can breach the security. Multi model Heterogeneous problems explain the detail concept of the security issue when types of databases merge with each other. It includes the explanation of different layers exist in NOSQL with some particular databases issue. In last some currents issues or breaches in NOSQL has been explained.

Keywords : Middleware,Validation, Database, Security, Authentication, Key-Value.

I. INTRODUCTION

In today's World, security is the main concern to be taken care as the data is being increasing; more the data breaching is taking place. As the technology increasing day by day, hacking skills are also increasing with respect to it. Security is very much important in today's scenario. In past, it would say that we live in the world of people, but now days we live in the world of data. Data is somewhere like a diamond in our industry which is very precious. So to protect this diamond, security is the key role has to be taken care[1].

Data is shared or transfer between thousands of nodes in NOSQL Database, so the entry point of each node will be multiple. Un-authorized access can be made from any of these nodes. So we have to protect all the nodes to protect the data. Security is very much complex in NOSQL Database. So to protect the sensitive data, security model should be properly managed and maintained as it can become a challenge.

Third party software or companies are providing the security layer to protect the data breach. PCI-DSS (The Payment Card Industry Data Security Standard) is a standard that is used to provide some standards protocols or rules and regulation to organization for major card (Visa, Master card etc) transaction. Datastax is an organization which is providing a layer for protection in a NOSQL database. So via this, some NOSQL databases are allowed to achieve PCI-DSS standards[2].

NOSQL does not provide in built security itself, even the documentation of all the NOSQL databases says to use our database in a "Trusted Environment" which is not possible. For some security, it depends on a middleware or we can say a security layer which is used to provide some security, like a firewall. Almost in all NOSQL database, by default administrator user authentication is turned off, very weak password storage etc. are some security problems which we will discuss later on. That is why; NOSQL is on the most wanted list in Hackers World.

Data is shared or transfer between thousands of nodes in NOSQL Database, so the entry point of each node will be multiple. Un-authorized access can be made from any of these nodes. So we have to protect all the nodes to protect the data. Security is very much complex in NOSQL Database. So to protect the sensitive data, security model should be properly managed and maintained as it can become a challenge.

II. Security Concerns In Nosql Databases

Before selecting a correct NOSQL database for your organization, you have to focus on the security

concerns. It's a good practice to gather security concerns before implementing it. There are several security concerns which decide how strong security exists in a database.

2.1 Database

- **Data At Rest**

It means data is in the database and is not using. We have to implement security when the data is in database to protect it, as we cannot store the data without any security. It can be stolen by some loop holes in the database. Security can be provided by the physical layer itself or by using any third party software.
So to protect the data while it is in rest, we have to implement some kind of encryption to the data or hierarchal password protection. So no unauthorized person can see the data without the decryption with the help of the key which only user knows.

- **Data In Motion**

It means data is in the flow, moving from database to the application layer or the system. Like in financial banking, out account details fetch form the bank database and displays in ATM screen. That data is very precious and security is the key role. If any of the data stolen while moving from bank database to ATM screen, then your account can be hacked very easily[3].
So data should be encrypted while data is moving back and forth. We can also use tagging, we can tag the data package weather it is confidential, high level, medium level or low level. So while sending the package, we will tag the package so that it is ease to know which kind of data is being transferring and what kind of security has to be taken care.

- **Data In Use**

It means that data is using in the application, data is displaying in the user screen. As the data is in the encrypted format, it will not be in readable form. So to read the data, decryption has to be done with the help of key, so that the data is visible on the screen to the user.

2.2 Authentication

It is another concern which is required to protect your database. It is the process of validating the authorized person only to access the database. A simple login which is used to access the database, if the login fails then that person cannot access the database controls.

In some companies, having large databases, sometimes need to communicate with the centralized service that used to validate the credentials. Generally, the service is designed such a way so that all the company databases can use it, which is known as Single Sign-on (SSO). If SSO interface is not being used, then by using directory access API, database validates users. Lightweight Directory Access Protocol (LDAP) and Stored Lightweight Directory Access Protocol (SLDAP) are commonly version used to accomplish it.

There are six types of authentication[4]:

- Basic access
- Digest access
- Public key
- Multifactor
- Kerberos
- Simple Authentication and Security Layer (SASL)

- **Basic Access Authentication**

This is the Basic level for authenticating the database user to access. The credentials are transferred in the http header in a simple plain-text format that can be fetched easily by a hacker. For providing some security, Basic Access Authentication should always be used with Secure Socket Layer (SSL) or Transport Layer Security (TLS). For the connection, there is no need for Web browser cookies or Handshaking.

- **Digest Access Authentication**

It is much better than Basic authentication but it is quite complicated to use as it required some additional handshaking for connection between the client and the database, we can use it over un-encrypted SSL/TLS layer. It is not recommended as highly secure authentication as uses MD5 hash function for transferring the data. For increasing password security, Digest Access Authentication is a good way to implement.

- **Public Key Authentication**

Public key authentication basically works on Asymmetric Cryptography. Basic concept of Asymmetric Cryptography is that it uses a pair of keys which are dependent on each other as one key is used to encrypt the sender's data which is known as Private Key, while other key is used to decrypt the data on the receiver's end which is known as Public Key. The basic drawback of this authentication is that if a hacker has access to your private key, then he can gain access to your database completely. So the security is only depends on the private keys.

- **Multifactor Authentication**

Multifactor authentication is not a single way for access as it requires two or more modes for authenticating the user which provide more security and reliability.

Example: If you allow your GMAIL account multifactor authentication, first you have to provide your credentials to login, then a pattern of words and number will be sent to your verified mobile number, which you have to type in the form to gain access to your account. If any of the two means login password or PIN you received on mobile phone is incorrect, then you will not be able to access your GMAIL account.

- **Kerbos Protocol Authentication**

Kerbos Protocol authentication is used in an insecure network for a secure connection. For authentication, it uses some third party application and uses cryptography for data transfer. After establishing the trusted Connection, database transfer the data to the server for the verification of the credentials. Each session access policy is controlled by Central Authority.

- **Simple Authentication And Security Layer**

Simple Authentication and Security Layer (SASL) is a framework that is used to provide certain set of protocols for trusted Communication. It defines some set of rule and regulation that can be used by any NOSQL database to authenticate and establishing a connection.

2.3 Authorization

After authenticating the credentials, next step is to give the permission to access the data according to their specific role. In authentication, it occurs only once per connection, when credentials are passed. But authorization is a complex process as one mistake can totally impact the overall security and performance of the database. Permissions have to be applied in many data items that are why, it is much more complex[5].
Authorization is dependent on level of abstraction and this level depends on categorization of the data which is needed to be authorized.

As you move downward, the level of security increases and performance decreases. Let us discuss on each of the level.

- **Database:**

It is the basic level to apply roles and permission after credentials verified. Security is low as it is the basic level but performance of the database is very high as no further verification will be there. It consists of number of collections. Example: While entering to your office building, your ID card is checked and if you have permission according to your role to enter the building, then only you are accessed else you cannot enter to the building[6].

- **Collection:**

It is much more secure than database as it is 2^{nd} level of abstraction. Security is higher than basic level but performance is also less than basic level. It is the collection of many documents. Example: in your office building, you have entered. Now to enter in a particular floor, you have to get permissions. Only if you have permission according to your role, then only you can get access to enter in a particular floor of the building.

- **Document:**

Security increased as it is 3^{rd} level of abstraction but performance is also decreased. It is the collection of many elements. Now the permissions are applied on

the document level, so it will affect the performance. Example: in your office building, you have entered your desired floor, now to access the particular room, you have to get permissions. If you have desired permission, then only you are accessed to the room.

- **Element:**

It is the most secure level of abstraction but also performance impact is very large. Element is nothing but the data which you have to access. Example: in your office building, you have entered the floor and the desired room. Now to work on your project, you need to have the permission. According to your role and permission, you will be granted to work on the project else not.

2.4 Data Encryption

While the data is in moving, or in resting, data has to be in an encrypted form to protect from a hacker. So with the help of digital signature or encryption we can verify the data that weather it has been modified or not while it was in the move from database to the application. This verification can be done either at the application level or at the database level or both sides. NOSQL does not provide any built in feature to achieve this kind of security, so by the use of third party application, we can achieve this.

There are many tools by which we can encrypt the data. Even we can combine the encryption and the digital signature for much more security. We can use private and public keys combining with the digitally signed certificates. Main problem is to detect where the encryption has to be applied, weather on the application layer or the database layer. Providing the encryption on both side will affect the performance as the time will increase.

If encryption process is provided in Database layer, then data accessing and storing method will have a centralized control but if the encryption process is in the application layer, then the control will be on each module. The main drawback of this process is that any changes done in the encryption algorithm by the external un-authorized person will completely affect the security issue and performance.

In some NOSQL Databases, self created encryption algorithm is used so that it will be very tough to change the algorithm by any external un-authorized person. This will surely increase the security if any major project is there. The US National Institute of Standard and Technology (NIST) have specified some standard that consist of multiple level of certification for cryptography libraries. If your database holds the entire standard then you can easily transmit the data by encryption.

III. Vulnerabilities In Nosql

Vulnerabilities are the weakness or loop holes from which a hacker or un-authorized person can enter the database and get the illegal authorization to access.

Some of the NOSQL vulnerabilities are [8]:

- Connection Pooling
- Key Brute-forcing
- HTTP REST API
- Denial of Services (DOS) Attack

3.1 Connection Pooling

Let us understand this concept using a NOSQL Database. CouchDB can be implemented with the help of RESTful API. Many concepts like Cross-Database, Pool-Access, Configuration etc. can be implemented directly by RESTful API. Even CouchDB Handlers are also implemented via RESTful API, which are very easy to use. As it is easy, the attack vectors are also easier than before. If a hacker is successful to manipulate the connector string, then he can even restart the database.

Connector example: NOSQL.connect (http://couchDB/**_restart**);

There are many CouchDB handlers which allow us to execute many database commands by just manipulating the connector like _sleep, _session, _utils, _stats etc. If a hacker is able to change the configuration by RESTful Interface, then he can even change the JavaScript interpreter to its custom file.

Cross Database can be easily implemented in Traditional SQL System. Cross database in nothing just jumping from one database to another when an SQL Injection performed. But in NOSQL it is harder to

implement. But not impossible, we can implement with the help of CouchDB connector by passing the variables in it. As connector is established, then you can have full access to that particular context only or you can say limited to the context.

Ex: NOSQL.connect (http://.$PoolDatabase."/nosql18/")

3.2 Key Brute-forcing

To understand this concept, we are using NOSQL Key-Value Databases. In a key-value database, key is the most important thing that needs to be secured. Key management has to be very strong in this type of database. There are some companies who offer key management services or we can create our own key also. If any un-authorized person gets the key, then your data will be in a big problem.

Key Brute-forcing is an attack that is used for this type of databases to search the key. This attack is very strong to decrypt any key or find any key. If a strong encryption is not there to protect the key, then this attack can easily breaks down the security and a hacker can get access. Performance issues are also there as if the key has so many encryption and decryption, then it will directly affect the database performance. So it depends upon the architecture and client library implementation.

Key-Value database are schema free, so there is no need to find any schema. This factor consider as the drawback of this type of database as security can be breached up easily. Smart work can be done by a hacker, by statically analyzing the Key before sending it in the attack. According to the judgment or analyzing, process of Key Brute-forcing can be speed up and easily fetch the key.

For securing the data from this attack on the application level, we need to be very careful. As this attack can breach the database security and our data can be fetched. As there is no schema, if we are storing all the confidential records in a single key value database and the database is brute forced, then all the data can be manipulated of fetched by un-authorized person. So database model should be good, your application logics should be strong, data has to be segmented by application itself. That is called modeling the data. Some other factors need to be considered also, like key space and key size as key has to be protected. Algorithms which are generating unpredictable key with a good algorithm that should be used in this database type for storing data. Key should be challenged based like some captchas can be used so that it will prevent the Brute force attack.

3.3 Http Rest Api

The client applications can be used to query the database from which the speed of the application can be enhanced. The REST API allows querying the database from the client application directly whether it is a HTML based application or any other. The data can be fetched or manipulated by running the particular queries the application provides. Mostly these applications provide the HTTP queries to be performed on the database. There is no mediate driver included in the structure to query the data and very fast way to execute the query related to the database. This although, increases the threats to the database because the attacker can attack the database using the skills to hamper the HTML pages or the application pages and inserting the codes between to access the database. The APIs, behind the firewalls, allows CSRF attacks to attack the database by exposing it to the client application. This will allow the user to go into the environment of the application and once the client embedded with the arbitrary queries of the hacker to enter the database, the injection will be performed easily. This will also allow the user to enter the HTML page and run the POST command from the form of the website to attack the database. Thus, this hinders the security of the database a lot.

3.4 Denial of Services (Dos) Attack

As we have discussed there are many attacks which hinders the functionality of the user or the database. Denial of Service Attack targets the database server and prevents the database from the access of the valid users such that it can slow down or shut down the system and maliciously add the code into a file which is in the input form of the website form to directly take over the control of the database. This gives the attacker a power to manipulate access or change the data according to his wish which can be a big threat to the security of the database. The attack is being done by sending the queries or the data in the form of queries repeatedly to the database within an infinite loop which hinders the

usage of valid users and unexpectedly shuts down the system.

To prevent your database from the DOS Attack, we can perform a pre define check for repeated rapid request coming from the same IP Address. If we found any external query fired repeatedly from the same system or the IP Address very frequently, we can abort the system from attending any more queries from the same IP Address.

IV. Current Nosql Data Security Issues

Exposure of 600 TB of MongoDB database on the internet

As we all know, MongoDB is very famous and very efficient multi model database structure providing a very high level of security in the market. But hazards are never hazards if they do not come with information and security. Almost 600 TB of the data of MongoDB database was exposed to the open source internet system reportedly as every application builds its versions and prototypes before booting and finally deploying the application or API online.

This has been one of the most popular databases used by many of the famous companies and organizations such as, Sourceforge to The New York Times, and LinkedIn. Few said that around 20-30 thousand GBs of the information of MongoDB database is openly available on the internet without any permission seeks or authentication and authorization.

The MongoDB didn't have any of the problems or the flaws but still by using the older version which was unable to bind itself from the local host, made the huge failure and the leakage of the data.

During the investigation of the matter, Matherly's main focus was growing popularity of the MongoDB database. It came to know that the versions the older version till 2.4.14, which was the last version to follow this local host, rule. Roman Shtylman told about the problem and reported in Feb, 2012. But this accidentally occurred flaw became so big that it took about 2 year and even little more to be recovered by MongoDB developers.

It's after effects were even worse. It affected many further versions of the MongoDB. Mainly 2.4.9, 2.4.10 and 2.3.7 were affected a lot. Many of the instances of the database were exposed publicly. This also led to many further problems and security issues faced by MongoDB.

V. CONCLUSION

In NOSQL, there are lot of security issues that need to be handled very carefully otherwise data can be hacked. As we have seen there are many attacks that can be performed on NOSQL Databases and security are very minimal to prevent these attacks. While comparing to the most 2 popular NOSQL Databases, MongoDB and Cassandra, we get to know that both databases lack encryption for the files, authentication is very weak and authorization is very simple and access to all the vulnerabilities like SQL Injection or DOS Attack. NOSQL is weak in terms of security. So in future, more security features should be included in NOSQL to make it stronger.

Key Terminology & Definitions

Middleware - For connecting any 2 layers, here in the document it is usually database and the application. A middleware is needed that acts as a bridge between the 2 layers. This bridge is mostly used for loading a driver and conversion of environment variable. Sometimes the middleware is not created by the application provider and handle by default but it is better to improvise the middleware efficient enough for the performance.

Validation - An application is very prone to error which can occur due to the wrong input of processing made by the mechanism. This can be handled by implementing form the default checking. This mechanism is called mechanism. Validation is generally done for the empty arguments, invalid argument etc.

VI. REFERENCES

[1]. List of NoSQL Databases [currently 225]. (n.d.). Retrieved May 17, 2016, from http://nosql-database.org

[2]. Main Page. (n.d.). Retrieved May 16, 2016, from http://en.wikipedia.org/wiki/Main_Page

[3]. NoSQL, no security? (n.d.). Retrieved May 19, 2016, from http://www.slideshare.net/wurbanski/nosql-no-security
[4]. Cassandra. (n.d.). Retrieved May 15, 2016, from http://cassandra.apache.org
[5]. MongoDB for GIANT Ideas. (n.d.). Retrieved May 18, 2016, from http://mongodb.com
[6]. Retrieved May 19, 2016, from http://www.youtube.com/?app=desktop
[7]. Ron, A., Shulman-Peleg, A., & Bronshtein, E. (2015). No SQL, No Injections. 9th Workshop on Web 2.0 Security and Privacy (W2SP) 2015, doi:10.1109/msp.2015.06
[8]. Ron, A., Shulman-Peleg, A., & Puzanov, A. (2016). Analysis and Mitigation of NoSQL Injections. IEEE Security & Privacy IEEE Secur. Privacy, 14(2), 30-39. doi:10.1109/msp.2016.36