RESEARCHERID
THOMSON REUTERS

[Siriah* *et al.*, 7(5): May, 2018]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

PRIVACY PROTECTION OVER ENCRYPTED MONGODB

Swetha Siriah, B.A. Deshpande
Computer Science, KITS, India

DOI: 10.5281/zenodo.1252862

ABSTRACT

The Larger data storage Space, Time and Privacy has become a key requirement for data management systems. The, NoSQL data stores, namely highly compress data on non relational database management systems, which often support data management of Internet applications, still do not provide support. It consists of the enhancement of the Mongo DB level based access control model with privacy keys for security and monitor. The proposed monitor is easily used into any MongoDB deployment control with high protection for data security.

KEYWORDS: Purpose-based access control, Privacy, NoSQL, datastores, MongoDB.

I. INTRODUCTION

NoSQL datastores are emerging non relational databases committed to provide high security for database operations over several servers. These platforms are getting increasing attention by companies and organizations for the ease and efficiency of handling high volumes of heterogeneous and even unstructured data. Although No SQL data stores can handle high volumes of personal and sensitive information, up to now the majority of these systems provide poor privacy and security protection. Initial research contributions started to studying these issues, but they have mainly targeted security aspects. To the best of our knowledge, we are not aware of any work targeting privacy-aware access control for No SQL systems, but we believe that, similar to what has been for privacy policies. With this work, we begin to solve this issue, by proposing an approach for the secured data purpose-based policy capabilities into MongoDB, one of the most popular NoSQL data store proposed for relational DBMSs, privacy-aware access control is an urgency for NoSQL data stores as well. However, different from relational databases, where all existent systems refer to the same data model and query language, No SQL data stores operate with various languages and data models. This variety makes the definition of a general approach to have of privacy-aware access control into NoSQL datastores a very important goal. We believe that a stepwise approach is necessary to define such a general solution. As such, in this, we start focusing on: 1) a single datastore, and 2) selected rules for privacy policies. We approach the problem by focusing on MongoDB, which, according to the DB-Engines Ranking, 2 ranks, by far, as the most popular NoSQL datastore. MongoDB uses a document-oriented data model. Data are modeled as documents, namely records, possibly images collections that are stored into a database [1]. We analyzed several privacy-aware access control models proposed for relational DBMSs to identify the characteristics of privacy policies to be supported. In all the analyzed models [2] privacy policies require rule based and enforcement mechanisms, as different data owners can have different privacy requirements on their data. The purposes for which data should be accessed with those for which they are stored is considered as the key required condition to grant the access is thus the important of any privacy policy. As such, fine grained purpose-based policies have been selected as the target policy type for our proposal. MongoDB integrates a role-based access control (RBAC) model which supports user and role management, and enforces access control at collection level. However, no support is provided for purpose-based policies. As such, in this work we extend MongoDB RBAC with the support for purpose-based policy specification and enforcement at document level. More precisely, the rule level at which the MongoDB RBAC model operates, integrating the required support for purpose related concepts. On top of this enhanced model we have developed an efficient enforcement monitor, called Mem (MongoDB enforcement monitor), which has been designed to operate in any MongoDB deployment. Within the client/server architecture of a MongoDB deployment, a MongoDB server front-end interacts, through message exchange,

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

with multiple MongoDB clients. Mem operates as a proxy in between a MongoDB server and its clients, monitoring and possibly altering the flow of messages that are exchanged by the counterparts [3].

Access control is enforced by means of MongoDB message rewriting. More precisely, either Mem simply forwards the intercepted message to the respective destination, or injects additional messages that encode commands or queries. In case the intercepted message encodes a query, Meme writes it in such a way that it can only access documents for which the specified policies are satisfied. The integration of Mem into a MongoDB deployment is straightforward and only requires a simple configuration. No programming activity is required to system administrators. Additionally, Meme has been designed to operate with any MongoDB driver and different MongoDB versions. First experiments conducted on a MongoDB dataset of realistic size have shown a low Mem enforcement overhead which has never compromised query usability.

II. ANALYSIS AND DESIGN

Recommendation of index type for proposed indexes. Using frequent itemset as a method to build a certain order of combined indexes out of fields of each frequent query. Use of query optimizer to select the final recommended indexes. Our approach to create virtual indexes which removes any modification in the database. Applying the approach to a document-based NoSQL database. A typical setting involves two user: one that gets information from the other that is either to share the requested information. Consequently, there is a tension between information sharing and privacy. On the one hand, sensitive data needs to be kept confidential; on the other hand, data owners may be willing, or forced, to share information.

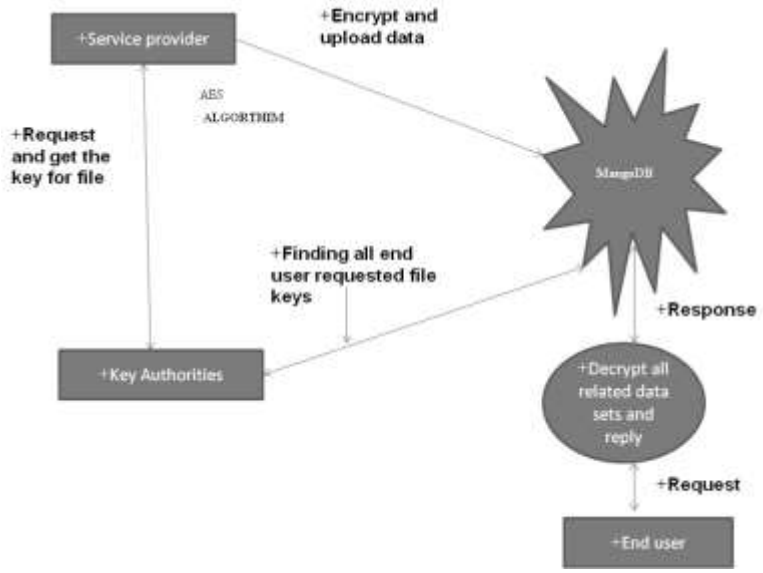

Figure 1. Data Encrypted Key System

The general approach to the rule of privacy-aware access control into NoSQL data stores a very important goal. Users are only allowed to execute for access purposes for which they have a proper authorization. Purpose authorizations are granted to users as well as to roles. The data storage and network transfer format for documents, simple and fast. Most regulatory requirements mandate that the encryption keys must be rotated and replaced with a new key at least once annually. MongoDB can achieve key rotation without incurring downtime by performing rolling restarts of the replica set. When using a appliance, the database files themselves do not need to be re-encrypted, thereby avoiding the significant performance overhead imposed by key rotation in other databases. Only the master key is rotated, and the internal database key store is re-encrypted.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

III. RESULTS AND DISCUSSION

Mongo DB stores its data in BSON. The server larger has a number of databases, and each database has a number of collections. Collections like you think of tables in a relational store, we only need a single collection to model our data.. If we were to query the Post collection from the shell, we'd see BSON come back representing our data.

1. Create Data, Create sample data for Departments and Employees collections (keep remember, at first time you query any particular collection, mongo Db automatically create these collections if not already created).
2. Get Departments, this will display the sample data created for departments collection, in the grid placed on the form.
3. Get Employees, this will display the sample data created for employees collection, in the grid placed on the form.
4. Delete Departments, this will delete all the departments available in the department collection.
5. Delete Employees, this will remove all the records available in the employees collection

Triple DES (**3DES**) is the common name for the **Triple Data Encryption Algorithm** (**TDEA** or **Triple DEA**) symmetric-key block cipher, which applies the Data Encryption Standard (DES) cipher algorithm three times to each data block. The DES cipher's key size of 56 bits sufficient when that algorithm was designed, but the availability of increasing computational power made by brute-force attacks feasible. Triple DES provides a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm.

Triple DES uses a "key bundle" that has three DES keys, K_1, K_2 and K_3, each of 56 bits. The encryption algorithm is: ciphertext = $E_{K3}(D_{K2}(E_{K1}(\text{plaintext})))$

Each triple encryption encrypts one block of 64 bits of data. In each case the middle operation is the reverse of the first and last. This improves the strength of the algorithm when using keying option 2, and provides backward compatibility with DES with keying option 3. The standards define three keying options:

Keying option 1

All three keys are independent.

Keying option 2

K_1 and K_2 are independent, and $K_3 = K_1$.

Keying option 3

All three keys are identical, i.e. $K_1 = K_2 = K_3$.

This function returns a list of the Mongo Picture Model objects retrieved from the database. The thing we did different here is use the Set Fields function to reduce the fields we bring back. For the gallery page we will only need the filename and ids of the pictures and not the data. First, _id is our identifier. While the probably figured that out, you may not know some of the ins and outs. _id will be automatically generated if you don't provide one. Since this record, a type of object called an Object Id was used. This type is documented on Mongo DB's site, and is fully supported by all the client-side implementations, including MongoDB in C Sharp. MongoDB in C Sharp also lets you specify either your own identifier, or you can use other types of auto-generated identifiers like a GUID. Second, notice how comments are stored as an array and embedded right within the Post document. As mentioned before, To have no need of performing a join to get all the information to need about a post, it is already a part of the document. Recommendation of index type for proposed indexes. Using frequent itemset as a method to build a certain order of combined indexes out of fields of each frequent query. Use of query optimizer to select the final recommended indexes. The approach to create virtual indexes which removes any modification in the database. Applying the approach to a document-based NoSQL database. The typical setting involves two user: one that gets information from the other that is either to share the requested

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

Figure 2. Image Key encryption

information. Consequently, there is a tension between information sharing and privacy. On the one hand, sensitive data needs to be kept confidential; on the other hand, data owners may be willing, or forced, to share information. The general approach to the rule of privacy-aware access control into NoSQL data stores a very important goal. Users are only allowed to execute for access purposes for which they have a proper authorization. Purpose authorizations are granted to users as well as to roles. The data storage and network transfer format for documents, simple and fast. Sign and Rotate Encryption Keys**.** Encryption keys for network and disk encryption should be periodically rotated. Encryption channels should use signed certificates to ensure that clients can certify the credentials they receive from server components. By default, the encrypted parts of documents are authenticated along with the id to prevent copy/paste attacks by an attacker with database write access. If you use one of the above options such that only part of your document is encrypted, you might want to authenticate the fields kept in clear text to prevent tampering. In particular, consider authenticating any fields used for authorization. The purposes for which data should be accessed with those for which they are stored is considered as the key required condition to grant the access, thus the important of any privacy policy. As such, fine grained purpose-based policies have been selected as the target policy type for our proposal. MongoDB integrates based access control model which supports user and role management, and enforces access control at collection level. However, no support is provided for purpose-based policies. As such, in this work we extend MongoDB RBAC with the support for purpose-based policy specification and enforcement at document level. More precisely, the rule level at which the MongoDB RBAC model operates, integrating the required support for purpose related concepts. On top of this enhanced model we have developed an efficient enforcement monitor, which has been designed to operate in any MongoDB deployment. Within the client/server architecture of a MongoDB deployment, a MongoDB server front-end interacts, through message exchange, with multiple MongoDB clients. Mem operates as a proxy in between a MongoDB server and its clients, monitoring and possibly altering the flow of messages that are exchanged by the counterparts. Access control is enforced by means of MongoDB message rewriting. More precisely, either simply forwards the intercepted message to the respective destination, or injects additional messages that encode commands or queries. In case the intercepted message encodes a query, it writes it in such a way that it can only access documents for which the specified policies are satisfied. The integration of data into a MongoDB deployment is straightforward and only requires a simple configuration. No programming activity is required to system administrators. Additionally, Meme has been designed to operate with any MongoDB driver and different MongoDB versions. First experiments conducted on a MongoDB dataset of realistic size have shown a low enforcement overhead which has never compromised query usability.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

Figure 3. Privacy Key Management

This shows that algorithm takes less time as compared to the data storage in mongo DB. The blue line indicates the amount data to be encrypted on the mongo server and the line in red gives the time take by the encrypted the data and generates the encrypted keys with encryption. This is used to conceal small blocks of data such as encryption keys and hash function Values which are used in Digital Signatures asymmetric cryptography, is any cryptographic system that uses pairs of keys: public keys that may be disseminated widely paired with private keys, which are known only to the owner. There are two functions that can be achieved: using a public key to authenticate that a message originated with a holder of the paired private key; or encrypting a message with a public key to ensure that only the holder of the paired private key can decrypt it. In a public-key encryption system, any person can encrypt a message using the public key of the receiver, but such a message can be decrypted only with the receiver's private key. For this to work it must be computationally easy for a user to generate a public and private key-pair to be used for encryption and decryption. The strength of a public-key cryptography system relies on the degree of difficulty computational impracticality for a properly generated private key to be determined from its corresponding public key.

IV. CONCLUSION

The Purpose concepts and related give mechanisms to regulate the access at document level on the basis of purpose and key based policies. An enforcement monitor, has been designed to implement the proposed security. It operates as a between MongoDB user and a MongoDB server, and enforces access control by monitoring and possibly manipulating the flow of exchanged messages. Furthermore, we plan to generalize the presented approach to the support for multiple NoSQL datastores.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

V. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocraticdatabases. In28th International Conference on Very Large Data Bases(VLDB), 2002.
[2] K. Browder and M. A. Davidson. The Virtual Private Database inOracle9iR2. Technical report, 2002. Oracle Technical White Paper.
[3] J. Byun and N. Li. Purpose based access control for privacy protection in relational database systems.The VLDB Journal, 17(4), 2008.
[4]]R. Cattell. Scalable SQL and NoSQL Data Stores. SIGMOD Rec., 39(4):12–27, May 2011.
[5] A. Cavoukian. Privacy by Design: leadership, methods, and results. In S. Gutwirth, R. Leenes, P. de Hert, and Y. Poullet, editors, European Data Protection: Coming of Age. Springer, 2013.
[6] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.
[7] P. Colombo and E. Ferrari. Enforcement of purpose based access control within relational database management systems. IEEE Transactions on Knowledge and Data Engineering, 2014.
[8] P. Colombo and E. Ferrari. Enforcing obligations within relational database management systems. IEEE Transactions on Dependable and Secure Computing (TDSC), 11(4), 2014.
[9] P. Colombo and E. Ferrari. Efficient enforcement of actionaware purpose-based access control within relational database management systems. IEEE Transactions on Knowledge and Data Engineering, 27(8), 2015.
[10] P. Colombo and E. Ferrari. Privacy aware access control for big data: A research roadmap. Big Data Research, 2015.