

**INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH
TECHNOLOGY****A NOVEL REDIS SECURITY BEST PRACTICES FOR NOSQL DATABASES****Jeelani Ahmed**Guest Faculty, Computer Science & Engineering, Polytechnic, Maulana Azad National Urdu University,
Hyderabad

DOI: 10.5281/zenodo.48348

ABSTRACT

In last decades of years the field of databases has emerged. The organizations are migrating towards Non-Relational databases from Relational Databases due to the current trend of Big Data, Big Users and Cloud Computing. Business data processing is the main market of Relational Databases. It turns out to be harder to managing Big Clients and Big information on a cloud domain. To modeling the data these databases uses a rigid and schema based approach and are designed to run on a single machine and NoSQL (Not Only SQL) database provides, flexible data model, dynamic schemas, efficient big data storage, scale-out architecture and access requirement hence these are considering as new Era of database [11][12]. We have taken Redis as a new NoSQL database in this work. Redis in its least difficult structure is a key value pair based information framework. It bolsters all the information structures like arrays, variables, strings and linked list. In SQL database like SQL Server, MySQL and Oracle the data in Redis is structured data unlike the other databases where the data is relational data [6]. However Redis provides minimum to virtually no security for the data. As all the information is put away in key value pair, anyone can get the value if the key is known. Accordingly such a database is not suitable for big enterprise and most useful application information. So in this work we add security to a Redis framework.

KEYWORDS: RDBMS, NoSQL, Encryption, Security and Big Data.**INTRODUCTION**

The NoSQL or Not Only SQL database gives a segment to limit and recuperation of data that is shown exceptionally rather than the even relations used as a piece of relation databases. Horizontal scaling, finer control over availability and simplicity of design are the motivations for this approach. There are numerous distinctions accessible. The huge investments in existing SQL, the lack of standardized interfaces and the utilization of low-level inquiry dialects are the boundaries to the more prominent appropriation of NoSQL stores [3][4][5]. A large portion of the NoSQL stores need genuine ACID transactions; however a couple of late frameworks, for example, c-treeACE, FairCom FoundationDB and Google Spanner have made them key to their plans [1].

Redis is a NoSql database of new generation. As we know that SQL database like SQL Server, MySQL and Oracle contains the data in the form of relational data whereas the data in Redis is structured data. Redis in its simplest form define as a key value pair based data system. All the data structures like variables, arrays, strings, queues and Linked list are supported in Redis. However Redis provides minimum to virtually no security for the data unlike the relational databases. Such a database is unsuitable for most practical application data and enterprise because anybody who knows the key can get the value as the data is stored in key value pair [10].

The systems and tools utilized for questioning information and persevering have developed at a unimaginable pace over the last couple of years. While we can say that transitional databases aren't going anyplace, we can likewise say that the biological community of information is never going to be the same. Redis has been the most exciting among all the new software and arrangements, for me, why? Firstly it takes care of a particular arrangement of issues while

in the meantime being truly non specific Secondly; it's unbelievably mostly easy to learn. It will take a few hours of time to get familiar with Redis.

Using Redis only you can assemble a complete framework, I think a great many people will find that whether that may be a conventional relational database, an archive arranged framework, or something else it supplements their more generic data solution. It is the one sort of arrangement that you use to execute the features. It's like an indexing motor. On Lucene you wouldn't fabricate your whole application. In any case, it's a vastly improved affair for both you and your clients when you need good search. So, the similarities between indexing motor and Redis end there [2].

REDIS GENERAL SECURITY MODEL

Untrusted access to Redis should reliably be intervened by accepting client enter, a layer actualizing ACLs and choosing what functions to perform against the Redis case. When all is said in done, Redis is advanced for most extreme execution and effortlessness however not for maximum security [10].

A. Network security

Computers implementing the application using Redis should be directly access only by the servers running redis. Redis port ought to be firewalled to shield from the outside access on account of a solitary PC that is presented to the web, similar to a virtualized Linux. Redis can still be access by the clients using the loopback interface. Redis can be bind to a solitary interface by adding a line to the redis.conf record:

```
bind 127.0.0.1
```

Because of the nature of Redis a big security impact will occur if neglecting to shield the Redis port all things considered. For instance, an external attacker can use a single FLUSHALL command to erase the entire information set.

B. Authentication feature

A tiny layer of authentication is provided by Redis that is alternatively turned on altering the redis.conf document. While Access Control does implement Redis. Any query by unauthenticated clients will be refuse by Redis when the authorization layer is turn on.

The authentication layer's main objective is to alternatively give a layer of excess. The outside customer can't at present have the capacity to get to the Redis example without illuminating the verification secret word, if and only if firewall fail to protect Redis from external attackers. The AUTH command is very important Redis command, is likely to be forwarded unencrypted, so it is unable to secure against pernicious aggressor that is sufficient fit for the system to perform listening stealthily [10].

C. Data encryption support

The encryption is does not supported by Redis. The additional layer's such as an SSL proxy should be implemented in order to setup trusted Redis parties over trusted and untrusted network. The function is mainly for protection.

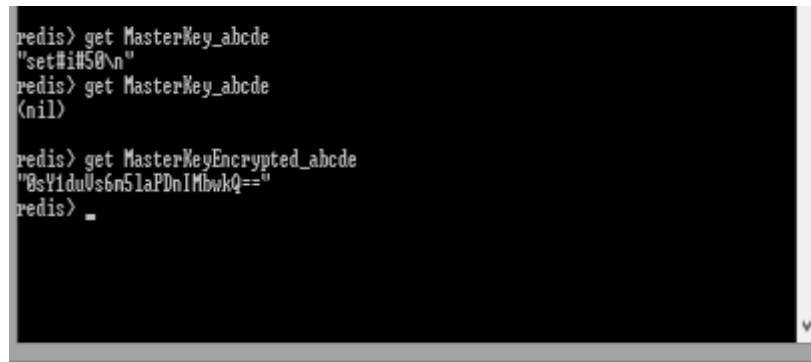
D. Disabling of specific commands

Redis command can be easily disable in order to rename them into an inconceivable name, so that typical customers will just have an arrangement of restricted, determined orders. Case in point, the virtual server supplier just gives an arrangement of Redis occasions. In that case a unauthorized user may not have the capacity to call Redis CONFIG summon to change design but only authorized user can do so.

E. Attacks triggered by carefully selected inputs from external clients

There is a gathering of assaults exists, even without an outside access to the instance an assailant get activated. A sample of such assaults is insertion of information into Redis that triggers most pessimistic scenario algorithm inside Redis Internals.

For example, an arrangement of strings known to hash to the same pail into a hash table supplied by an aggressor by means of web with a specific end goal to change the normal time $O(1)$ into the most pessimistic scenario time $O(N)$ by expending more CPU time and eventually creating a denial of service(DoS). To keep this particular assault, Redis utilizes a for every – execution pseudo-irregular seed to the hash function.



```
redis> get MasterKey_abcde
"set#i#50\n"
redis> get MasterKey_abcde
(nil)

redis> get MasterKeyEncrypted_abcde
"0sYidu0s6n5laPDnIMbwkQ=="
redis> _
```

Fig.1 Redis command line interface

F. Code Security

In an established redis setup, customers are permitted full access to the command set, yet getting to the instance ought to never bring about the capacity to control the framework where Redis is running.

Inside, Redis utilizes all the surely understood practices for composing secure code, to anticipate buffer overflows, organization bugs and other memory debasement issues. Be that as it may, the capacity to control the server configuration utilizing the CONFIG command rolls out the client ready to improvement the working dir of the project and the name of the landfill record. This permits client to compose RDB Redis records aimlessly ways, which is a security issue that may effortlessly prompt the capacity to run untrusted code as the same client as Redis is running.

Redis does not require root benefits to run. It is prescribed to run it as an unprivileged redis client that is utilized for this reason. The Redis creators are right now examining the likelihood of adding another design parameter to forestall CONFIG SET/GET dir and other comparative run-time configuration directives. This would keep clients from constraining the server to compose Redis landfill records at subjective areas [3][10].

SYSTEM DESIGN AND IMPLEMENTATION

It is clean and clear from the data provided in present system that Redis does not Promptly bolster a) Authentication b) Encryption and c) Restrictions to commands which are most key and regular parts of security augmentation of Redis framework. In this way in the proposed framework we give the previously stated Redis escape clauses to think of a totally safe and secured Redis framework.

A. Authentication and Authorization

We first make a widespread key which can hold all username secret key sets. Any client needing to get to the framework needs to first enroll with the framework. Amid enlistment transform, the Users key is unloaded and framework checks if the client is as of now present or not, if present it prompts for an option username. Else it produces a salt from the gave watchword and packs the username and secret key salt in the Users key [7].

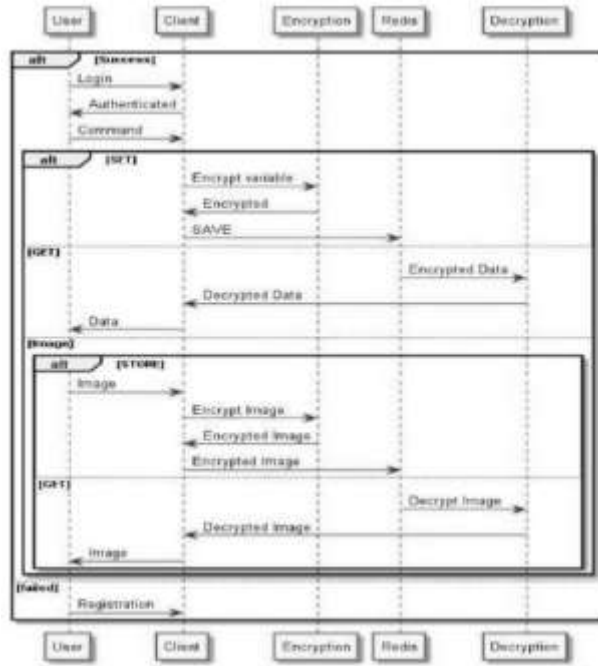


Fig. 2 System Sequence Diagram

Amid Login process, first the framework creates the secret key salt from Rinjdeal system from the gave watchword and joins the username secret key as a solitary string. At that point the framework hunt down this string in the current key, if discovered client is confirmed.

B. Encryption Services

The key is formed by encrypting from each variable and its value which is gotten from client name. Along these lines each client has a solitary key in the Redis database that contains every single other key in the encoded structure. Encryption service is performed by symmetric key cryptography with AES. Redis database stores the encrypted images in binary form.

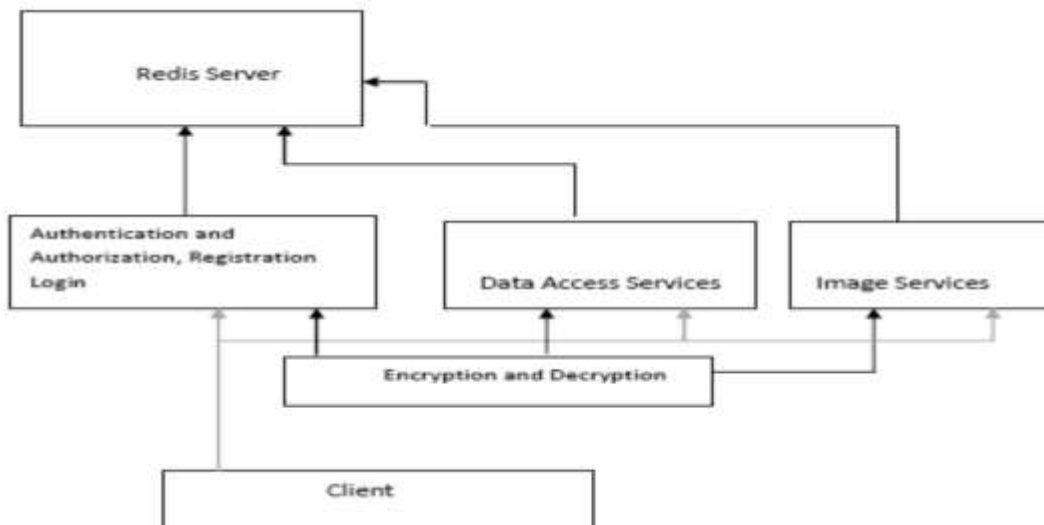


Fig.3 System architecture

C. Command Restriction

The framework does not bolster whatever other commands accept the Get, Set commands and. This is to demonstrate the ability of limiting order sets [8]. Additionally the abnormal state charges for putting away and recovering pictures are acknowledged. The pictures are additionally secured.

D. Network Security

Cryptographic techniques like Encryption and Decryption are continuously performed at the client side. Hence the propagated information in the network is secured data. Subsequently risks of the information being hacked by gatecrasher by system bundle hacking is minimized.

G. Data Persistence and Persistent data security

Redis database generally stores the persistent data. The availability of this information ought to be available all the time when client returns [9]. This is finished by pressing the client variable which contains variable of present and past sessions and their qualities being stuffed in the client information hub in the scrambled structure where the encryption system embraced is as clarified previously.

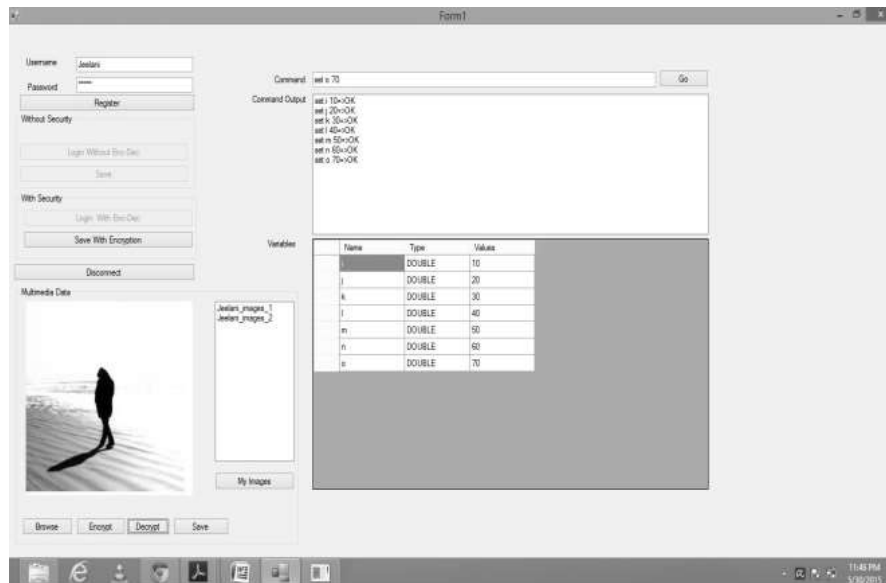


Fig. 4 Developed user interface

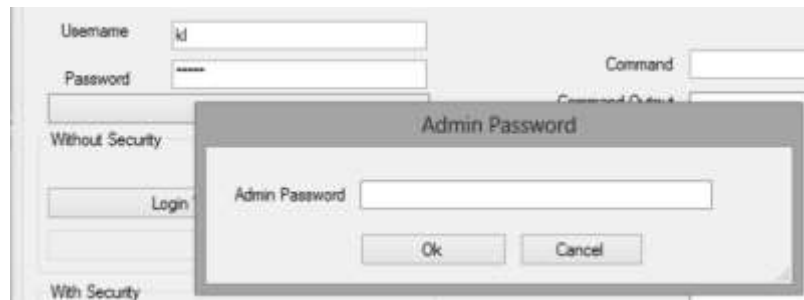


Fig. 5 Client Registration

```

redis> get MasterKey_abcd
"set#i#50\n"
redis> get MasterKey_abcd
(nil)

redis> get MasterKeyEncrypted_abcd
"0sYidu0s6n51aPDnIMbwkQ=="
redis> _
    
```

Fig.6 Encrypting data

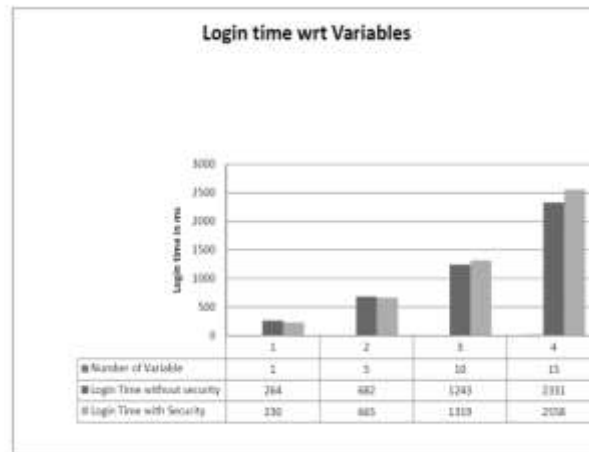
RESULT

To implement Redis as a standalone application comparison between Redis and our developed security and user interface module provides the important information to many organizations. Trusted clients accessed the Redis inside trusted situations.

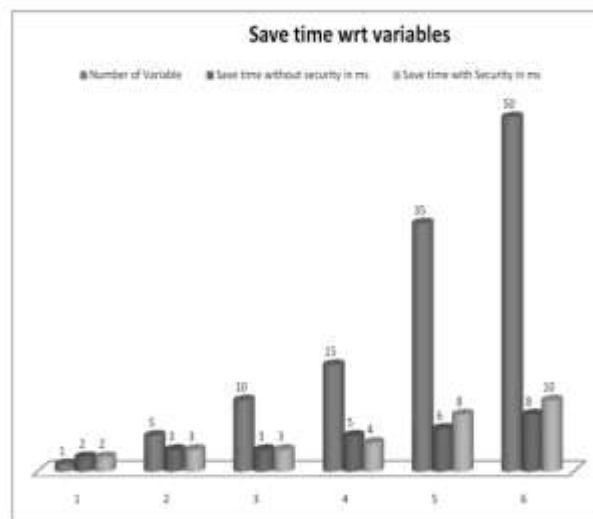
This implies that more often than not it is not a smart thought to Exposing the Redis example straight forwardly to the web it is not a good idea or to a situation where untrusted customers can straight forwardly get to the Redis TCP port or UNIX attachment. The following graphs show the login time of client with respect to number of users and variables and client data saving time with respect to variables.



a. Client login time wrt users



b. Client login time wrt variables



c. Client data saving time wrt variables

CONCLUSION

The Limitation of Redis-NoSQL database system is not providing the required security necessary for real time systems. The fundamental point of interest of the work is that it succeeds above limitation. Generally without any security Redis framework is quick for most normal organized information. Using the above proposed techniques required security is provided to redis system.

- In this manner Redis now bolsters both client particular sessions and in addition client particular information. Variables with same name available for different users.
- Both network and persistent data are secured.
- Symmetric key cryptography which dispenses with the need of any key exchange protocol is used for providing the security.
- Thus the framework can likewise give multimedia administration.

Thus the Redis system's capabilities extended by the proposed system to enable the Redis system to complete secured database and multiuser adaptable which can be utilized for ongoing applications.

FUTURE WORK

For every operation data need to be encrypted and decrypted so this security extension increases additional computational overhead to the summons in this manner the fundamental conceivable future work is decreasing this computational overhead. For secured information trade for Redis environment more bandwidth is needed as AES increases the data length. To support proposed security support the server needs more memory and bandwidth.

REFERENCES

1. Stonebraker, Michael; Madden, Samuel; Abadi, Daniel J.; Harizopoulos, Stavros, "The end of an architectural era: (it's time for a complete rewrite)," Proceedings of the 33rd international conference on Very large data bases, VLDB, p. 1150–1160, 2007.
2. N. G.-O. Y. G. E. G. J. A. Lior Okman, "Security Issues in NoSQL Databases," in 2011 International Joint Conference of IEEE TrustCom-11/IEEE ICSS- 11/FCST-11, 2011.
3. Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastva, Adam Silberstein and others, "PNUTS:Yahoo!'s Hosted Data Serving Platform," ACM, no. 08, 2008.
4. P. W. Kriha, "NoSQL Databases," [Online]. Available: www.christof-strauch.de/nosql dbs.pdf. [Accessed 2 2013].
5. "NoSQL databases," [Online]. Available: nosql-database.org. [Accessed 10 6 2013].
6. Mohamed A Mohamed, Mohammed O Ismail, "Relational vs. NoSQL Databases: A Survey", International Journal of Computer and Information Technology, volume 03, May 2014.
7. Okman, L.; Gal-Oz, N.; Gonen, Y.; Gudes, E.; Abramov, J.; , "Security Issues in NoSQL Databases," Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on , vol., no., pp.541-547, 16-18 Nov. 2011 doi: 10.1109/TrustCom.2011.70
8. Neal Leavitt "Will NoSQL Databases Live Up to Their Promise?" IEEE Computer Society 0018-9162/10/\$26.00 © 2010 IEEE.
9. Srinipenichkala, —Virtual Panel: Security Considerations in Accessing NoSQL Databases, Nov. 2011. <http://www.infoq.com/articles/nosql-data-security-virtual-panel>.
10. [Redishttp://redis.io/](http://redis.io/)
11. Jeelani Ahmed, Raafiya Gulmeher "Nosql databases: New trend of databases,emerging reasons, classification and security issues" International journal of engineering sciences & research technology, Volume 4, Special issue 6, June 2015.
12. Asadulla khan zaki "NoSQL databases: new millenium databases for big data. Big users, cloud computing and its security challenges" International journal of research in engineering and technology, volume 3, special issue 3, may 2013.

AUTHOR BIBLIOGRAPHY

