Whitepaper:

A Secure Directory Server for trusted and untrusted networks (RepoSe)

Karthik Bhat | *SecureDB Inc* | *Sept 2014.*

With traditional user repositories being compromised frequently by Information Attackers, there is an acute market need for a secure user repository. This paper introduces 'RepoSe' - a secure user repository that supports LDAP, REST interfaces and provides computationally secure way to store and retrieve user account information. RepoSe depends on a combination of Data Aware Encryption (DAE) and Transparent Data Encryption (TDE) to ensure data is fully protected – in memory and at rest. Thanks to DAE, all queries are performed against encrypted data and thanks to TDE, all data is fully encrypted at the page level. This means that the storage engine of the RepoSe (i.e., RepoSe DB) could be hosted in a trusted, semi-trusted or untrusted environment. Extrapolating this further, RepoSe's design lends itself nicely into various cloud computing paradigms.

I. INTRODUCTION

With more and more business being carried out over the internet, organizations have a need to store user names, passwords and other identity information of their users in databases or directory servers. In vast majority of the scenarios, this information also contains PII or PHI data. These are sought after targets for an Information Attacker (IA) who stands to gain financially, politically, or strategically by getting access to this data. It is estimated that over 1 billion records have been compromised over last decade and affect virtually all industries [1]. Widely reported incidents at Sony, Zappos, LivingSocial, South Carolina Dept. of Revenue, Global Payments, Adobe etc. resulted in millions of user accounts compromised and users' information stolen.

Identity and Access Management products and solutions protect the applications from users by enforcing authentication and access control policies. These products and solutions do little to protect the underlying PII and PHI data stored in the user repositories.

As multiple applications within an enterprise depend on the identity information provided by user repositories, IA has multiple opportunities to find a vulnerable path to get to these repositories. A breach not only affects the privacy of the users, it also has long term legal and financial ramifications to the organization. The financial cost on average of a single record breached in the US is around $199 [10]. It is expected that when customer records are stolen as a result of a breach, the brand value of the corporation suffers by 21 percent [11]. It may be impossible to accurately estimate the economic value of the reputation lost.

This paper contains design for a Secure User Repository ('RepoSe') that protects the user information from two kinds of Information Attackers – (1) external actors and (2) malicious insiders. Additionally, RepoSe can also protect the organization from accidental leakage.

User repositories are usually read-heavy databases where writes happen far in between. In a typical scenario, 80% of the transactions are read transactions. For a given installation, it is not inconceivable to have 90% or more reads as compared to writes. Even among reads, the credential retrievals (aka authentications) make up bulk of the reads. RepoSe is designed for these scenarios and can scale horizontally and vertically to act as a backbone of the enterprise.

RepoSe depends on encryption to protect data. All data saved with RepoSe is encrypted twice using two different keys and using different schemes. RepoSe requires no changes to existing applications and the cryptographic security achieved is transparent to the applications.

RepoSe Proxy and Repose DB only support valid SSL traffic and both come built-in with a host based firewall. In other words, non-SSL traffic or traffic coming from unknown hosts will simply be ignored. RepoSe also comes ready with enterprise grade failover solution to ensure availability.

II. DESIGN

RepoSe solution typically contains two servers – a RepoSe Proxy and a RepoSe DB. RepoSe Proxy acts as a client to the RepoSe DB. In fact, RepoSe DB only accepts traffic from Repose Proxy. It is recommended that RepoSe Proxy and RepoSe DB servers be deployed in two different security domains with each server being administered by a different set of personnel. Figure 1 shows the overall design of RepoSe.

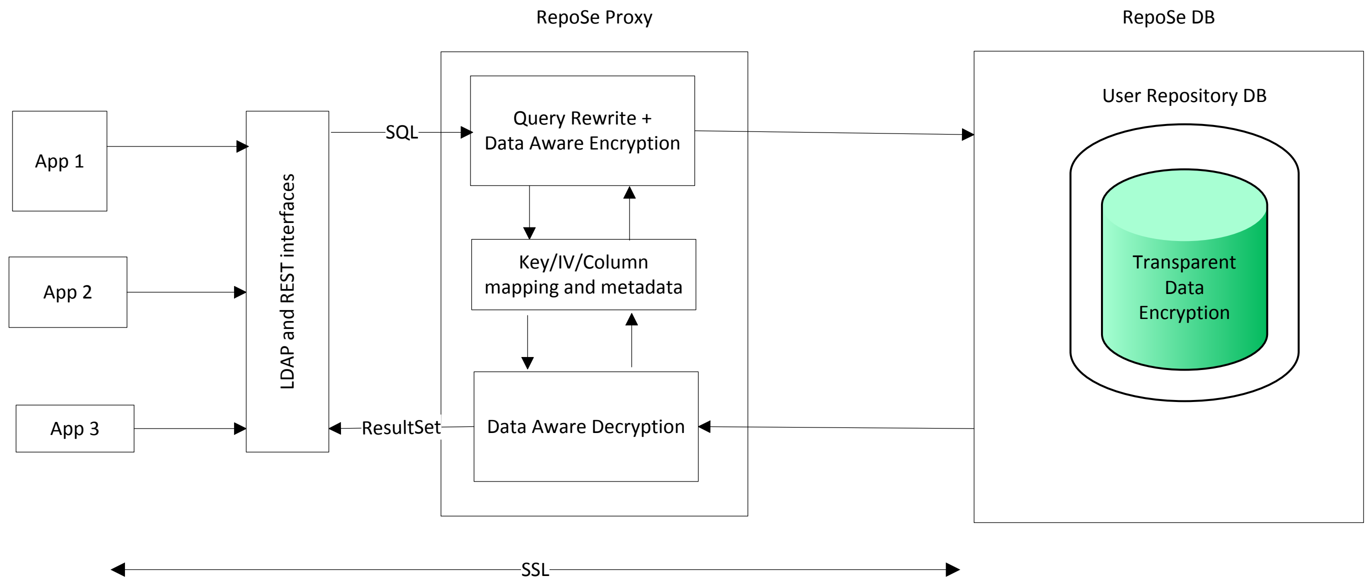

Figure 1 - RepoSe architecture

RepoSe DB is a TDE enabled database. TDE is a well understood and mature technology that ensures cryptographic security for data at rest. Multiple database vendors [2][3] including Open Source databases[4] provide TDE. TDE allows database to transparently encrypt data when data is written to disk and decrypt the data when the data is read back from the disk with no change to schema design or the applications.

RepoSe DB implements TDE at the database page level. Each RepoSe DB page is 8 kilobytes in size and contains the Initialization Vector (IV) and hash of the data (HMAC, to check for tampering) along with encrypted data. When a page is written to the disk from shared buffers (as a result of an insert, delete or update operations), the page gets encrypted. When the page gets read into the database shared buffers (as a result of an incoming query), the data is decrypted. RepoSe DB uses AES in CBC mode for TDE.

Repose DB's TDE encrypts the index pages and log files well. TDE leaks no data when the physical database files are compromised. Unlike Full Disk Encryption based solutions, TDE ensures that the backup of data is encrypted as well.

When database with TDE executes a query, the pages needed to serve the results are decrypted and get loaded into the memory (shared buffers). This information is now available to the Operating System (OS) and DB administrators.

However, in case of RepoSe DB, the data stored in the database is never in clear text – it is encrypted by RepoSe Proxy. That is, the pages themselves contain encrypted data. Moreover, the keys and IV needed to decrypt this data are never sent to the RepoSe DB by RepoSe proxy. Hence a curious OS or RepoSe DB administrator cannot read any user data.

RepoSe Proxy implements a technique known as Data Aware Encryption (DAE) to encrypt the data before sending it over to the RepoSe DB for storage. Each data item is encrypted based on what it represents (i.e., what is the context of the data) rather than what data type is used to store the data. For example, consider a user repository that stores the data items shown in Table 1 for each user.

Although all of these fields would be represented as VARCHAR data type in RepoSe DB, each of these fields have different uses in the enterprise and hence need to be handled differently cryptographically.

Column Name	Description
userId	User ID
password	One way hashed and salted Password
firstName	First Name
lastName	Last Name
email	Email address of the user
ssn	Social Security Number
cardNum	16 digit credit card number
cvv	3 digit CVV number

Table 1 - Fields in a user repository

Consider the field ‘userId’. During authentication, this field will be checked for equality. Hence it needs to be encrypted deterministically (DET scheme). RepoSe Proxy uses AES in CMC with tweak [5] mode in such scenarios. Unlike AES in CBC mode, this mode does not leak prefix equality when two strings with identical 128 bit prefix are in the result set.

DET scheme leaks the size of the result set to the database. That is, for a given query (that is for a given WHERE clause) the database knows how many records match the query. However, this is not a concern since the data itself is never leaked.

Now consider SSN. This field needs to be encrypted using a strong probabilistic (PROB) cryptographic scheme. Here RepoSe uses AES 256 CBC with variable IV. The randomly generated IV is stored along with the encrypted data.

Now consider the field ‘firstName’. It could be checked for equality like ‘userId’, but could also be searched for using a LIKE query (select * from <table> where firstName LIKE ‘JOH%’). In case of RepoSe, the help desk staff could look up the user record using admin console by entering part of the first name. As a result, RepoSe Proxy will encrypt the ‘firstName’ field using two schemes – DET as described above and ENSEARCH. RepoSe Proxy will rewrite the query to store the data encrypted using these two schemes in two different columns. Please see Section III for more details on query rewriting.

Multiple schemes are available [6, 7] to search for keywords in encrypted data. RepoSe uses Goh’s Secure Indexing scheme [7] to implement ENSEARCH. Goh’s scheme allows creation of cryptographic indexes for ‘documents’ at the RepoSe Proxy that are then sent to the RepoSe DB for storage. The word ‘document’ is loosely defined to mean one or more words separated by punctuation. RepoSe’s ENSEARCH scheme supports searching for a word or a prefix of a word in a document along with case insensitive search.

Thanks to DAE at the RepoSe Proxy, the fields in Table 1 would be stored as follows at RepoSe DB.

Logical Column	Physical Columns		
userId	userId_det		
password	password		
	passwordSalt		
firstName	firstName_det	firstName_ensearch	
lastName	lastName_det	lastName_ensearch	
email	email_det		
ssn			ssn_prob
cardNum			cNum_prob
cvv			cvv_prob

Table 2 - Physical columns storing encrypted data; sometimes in more than one scheme based on DAE rules.

As seen in Table 2, the encryption is done based on the security posture that every field requires. Fields like SSN, CVV, email etc. are always accessed in entirety and help desk staff [1] or a compromised application shall not be able to trigger LIKE queries on these fields to fish for data. Significantly different protections are needed for fields such as First Name, Last Name, Credit Card Number etc. These fields need to be searched upon and hence RepoSe encrypts these using DET and ENSEARCH schemes. Encrypting these fields with two schemes allows RepoSe DB to perform both equality checks and searches on these encrypted data items without having to decrypt them.

[[1] Though the help desk staff will be able to see the SSN for all users whose first name starts with ‘JOH’. At that point, it is up to the applications to determine if it is appropriate to display SSN to the someone with the role ‘help desk’ or not.]

III. QUERYING OVER ENCRYPTED DATA

Like most solutions that make use of a proxy to rewrite the query and perform encryption [8, 9], RepoSe Proxy too plays critical role in overall security of the system. RepoSe Proxy is responsible for rewriting incoming queries and managing the Data Aware Encryption. It uses a metadata repository (RepoSe Proxy Store) to store the mappings between logical column names and corresponding physical column names. RepoSe Proxy Store also stores the mappings between physical column name and scheme, IV, key used to encrypt the data in that column. Unlike cryptodb [9], the cryptographic keys and IV are not sent to the database server.

A simplistic and denormalized data model is used by RepoSe Proxy Store for storing these mappings as shown in Table 3. The column level keys could be stored inside RepoSe Proxy Store or could be referenced from a HSM. RepoSe Proxy Store itself is fully encrypted.

TableName	ColName	PhyTableName	PhyColName	Key	IV	scheme

Table 3 - Metadata mapping needed for query re-write and data aware encryption/decryption.

Consider a scenario wherein a new user is being added to the RepoSe system. This will lead to a query arriving at the RepoSe Proxy as follows:

Snippet 1

```
insert into users (userid, password, email,
firstName, lastName, ssn, cardNum, cvv) values
('john.doe', 'joH$122', ' john.doe@gmail.com',
'john', 'doe', '123456789',
'123456789012345','123’);
```

RepoSe Proxy will parse this query and validate it. If it appears to be a valid SQL query, the query rewrite module will rewrite this query based on the DAE principle as follows:

Snippet 2

```
insert into users (userid_det, password, email_det,
firstName_det, first_name_ensearch, lastName_det,
lastName_ensearch, ssn_prob, cardNum_prob,
cvv_prob)

values

(<enc_det('john.doe')>, hash('password'),
<enc_det('john.doe@gmail.com)>,
<enc_det('john')>,
<enc_ensearch('john')>,<enc_det('doe')>, <enc_
ensearch('doe')>,<enc_prob('123456789')<iv>>,
<enc_prob('123456789012345')><iv>,
<enc_prob('123')><iv>);
```

In the snippet above, enc_det() routine encrypts the value using the DET scheme with corresponding key and IV from Table 3. enc_prob() routine encrypts the value using the PROB scheme using the key in Table 3. For every encryption, a random IV is generated and sent to RepoSe DB to be stored along with encrypted data. Similarly, enc_ensearch() routine encrypts the value using ENSEARCH scheme. The values returned by enc_det() and enc_ensearch() are sent to the RepoSe DB for execution.

IV. IMPLEMENTATION

RepoSe DB is a customized version of open source database PostgreSQL. Postgres is an enterprise grade ACID compliant database system written in C. It is modular in design, standards compliant and scales extremely well.

RepoSe DB alters the page structure of Postgres pages to implement TDE. Natively, Postgres deals with 8K sized pages. Each page has a header and some reserved space at the end. RepoSe DB implementation modifies the page structure to include additional space for IV and MAC. The IV is used for AES CBC cryptography, whereas the MAC is used for checking the authenticity of the page contents between reads. Further, RepoSe DB’s TDE implementation encrypts the data when the DB buffer is written to the disk using a database key and page-specific IV. Similarly, when the page is read back to the buffer, it gets decrypted using the database key and IV – thus providing encryption that is completely transparent to RepoSe DB’s clients.

RepoSe Proxy is a customized version of pgpool II. pgpool II is a Postgres middleware that can be run in variety of modes. RepoSe Proxy runs pgpool II in raw mode, allowing for failover of RepoSe DB severs. RepoSe Proxy also makes use of pgpool II’s memcached module to cache results to improve performance.

RepoSe Proxy has a query rewrite module that rewrites the incoming queries. To do so, it analyzes ‘parse tree’ (produced by pgpool II’s query parser module) and builds a ‘walker’. The walker goes through each attribute in the parse tree and does the following: (a) for each column name (for example, firstName), replaces the logical column name with physical column name by looking up the mapping in RepoSe Proxy Store. (b) applies DAE by accessing the appropriate key and IV.

RepoSe system provides two distinct web consoles. The first one, called 'RepoSe Admin' is an administration console/dashboard that provides information around health of the system, performance metrics, etc. This is only available to administrators. 'RepoSe Help Desk' is a help desk web console meant for help desk staff.

The proxy also serves as the system of record for statistics related to RepoSe usage and performance. These metrics are displayed in the 'RepoSe Admin' dashboard. The statistics gathered include, but are not limited to:

1. User authentication: how many users authenticated with the system, how many were successful, along with the timeline..
2. Query performance: Query rewrite time, memcached hits/misses, round trip time for results from RepoSe DB, measure of false positives returned on ENSEARCH columns, performance on filtering false positives.
3. Internal performance: Memory and CPU usage, Disk activity etc.

V. KEY MANAGEMENT

Key management is required both at the RepoSe DB (for TDE) and the RepoSe Proxy (for the DET and ENSEARCH transforms). NIST publishes key management guidelines as part of their Special Publications (800 Series) [12], and will be the selection criteria for choosing a key management solution for RepoSe. The NIST publications address all aspects of key management including types of encryption keys, and recommendations for key rotation periods for storage data encryption schemes. RepoSe will adhere to these guidelines.

RepoSe will also provide integrations with popular commercially available HSM/EKM solutions.

VI. DEPLOYMENT CONSIDERATIONS

RepoSe system lends itself into multiple deployment topologies.

Enterprise Only

In this mode, an enterprise hosts both RepoSe Proxy and RepoSe DB. Though the same organization controls both RepoSe Proxy and RepoSe DB, it is recommended that these two components are deployed in different network segments and isolated administratively. Multiple applications could connect to RepoSe Proxy to access the services provided by RepoSe.

Cloud

Enterprises wishing to outsource databases or directory servers to cloud have a unique problem around key management. This problem is probably the single most important reason why enterprises chose not to use cloud services when it comes to databases. The challenge is to figure out how to encrypt enterprise's data on the cloud without the keys leaving the enterprise boundary? If the enterprise decides to encrypt the data before sending it to (untrusted) cloud provider, then the enterprise has two options for running queries: (a) download the data and decrypt it before running queries or (b) provide cloud provider with the keys. Both of these options are not viable.

RepoSe system provides a simple and elegant solution to this problem by allowing queries over encrypted account data. In this mode, the enterprise outsources RepoSe DB to a secure public cloud such as RepoSe Cloud (SecureDB's cloud offering). The RepoSe Proxy is still maintained within the enterprise. This mode still provides the same security benefits as the 'Enterprise Only' mode since the cryptographic keys for DAE never leave the enterprise boundary.

REFERENCES

[1] Verizon 2013 Data Breach Investigations Report

[2] Transparent Data Encryption (TDE) in Oracle Database. http://www.oracle.com/technetwork/database/options/advanced-security/index-099011.html

[3] Transparent Data Encryption in SQL Server Database. http://technet.microsoft.com/en-us/library/bb934049.aspx

[4] Transparent Data Encryption for SQLLite database. http://sqlcipher.net/

[5] A Tweakable Enciphering Mode, Shai Halevi, Phillip Rogaway, June 2003.

[6] Practical Techniques for Searches on Encrypted Data, Dawn Xiaodong Song, David Wagner, Adrian Perrig

[7] Secure Indexes, Eu-Jin Goh, March 16, 2004

[8] Privacy-Preserving Queries on Encrypted Data, Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright

[9] CryptDB: Protecting Confidentiality with

Encrypted Query Processing, Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan

[10] 2013 Cost of Data Breach Study: Global Analysis, Ponemon Institute LLC. May 2013.

[11] Reputation Impact of a Data Breach, US study of Executives and Managers. Ponemon Institute LLC. November 2011.

[12] Recommendation for Key Management – Part 1: General (Revision 3), NIST. http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf