Self-Protection against Insider Threats in DBMS through Policies Implementation

Farukh Zaman, Basit Raza
Department of Computer Science
COMSATS Institute of Information Technology
Islamabad, Pakistan

Ahmad Kamran Malik, Adeel Anjum
Department of Computer Science
COMSATS Institute of Information Technology
Islamabad, Pakistan

***Abstract*—In today's world, information security of an organization has become a major challenge as well as a critical business issue. Managing and mitigating these internal or external security related issues, organizations hire highly knowledgeable security expert persons. Insider threats in database management system (DBMS) are inherently a very hard problem to address. Employees within the organization carry out or harm organization data in a professional manner. To protect and monitor organization information from insider user in DBMS, the organization used different techniques, but these techniques are insufficient to secure their data. We offer an autonomous approach to self-protection architecture based on policy implementation in DBMS. This research proposes an autonomic model for protection that will enforce Access Control policies, Database Auditing policies, Encryption policies, user authentication policies, and database configuration setting policies in DBMS. The purpose of these policies to restrict insider user or Database Administrator (DBA) from malicious activities to protect data.**

Keywords—autonomic; self-protection; insider threats; policies; DBMS

I. Introduction

Data is probably most important and valuable asset on which entire organization depends. However, it's difficult to memorize some data so these data should be kept in an organized way in a special storage location called databases. It's necessary to build a trustworthy relationship with an organization and its clients by protecting its data from possible threats. Data should protect by imposing CIA (Confidentiality, Integrity, and Availability) security model which should be guaranteed in any kind of security system [5] [34] [35] [36] [37] [38]. Without CIA security model data can be lost or destroyed. Some security threat against database management systems are:

- Misuse of sensitive data by the authenticated user
- Malware infection causing damage to data or programs
- Physical damage of database server
- Weak parameter setting or design flaws causing vulnerabilities in DBMS
- Unauthorized access of DBMS

Database threat may have initiated either in an external way or from within an organization. The external threat can be detected by imposing software tools and technologies such as Firewall, network traffic monitoring, enforcing password mechanism and penetration testing [4]. However, it's difficult to monitor insider's intent. According to CERT survey, more than 700 cases were caused by the insider threats [6]. To protect against these threats database should have some extra features of Autonomic Computing like self-protection. We first provide an introduction to Autonomic computing and its components.

Autonomic computing has the ability to self-manage its system [39] [40]. It controls all the functionality of computer systems or applications without any user involvement. Autonomic computing concept is taken from human body's autonomic nervous system, which controls human body functions such as heart rate, respiratory rate, pupillary response parts and Digestive system without the conscious input of an individual [2]. How the human body mechanisms manage itself without external involvement in many cases? The main objective of autonomic computing is to build a system that has a self-managed characteristic and make a decision on its own by using high-level functionalities when any unpredictable problem occurs. Autonomic computing framework based on autonomic components that interact with each other. The autonomic computing system has the ability to respond to any problems occur and make the system precise and available to the user. Instead of directly user input in the system, User defines general procedures and policies that guide the self-management process. IBM defines four main self-* components [7] [41] [42] [43] [44] [45].

- Self-optimization
- Self-healing
- Self-configuration
- Self-protection.

Some other extended self-* features are defined as in [8] are Self-Adaption, Self-regulation, Self-learning, Self-awareness, Self-organization, Self-creation, Self-management and Self-descriptive. When all these self-* features of self-managed apply to any system that system has the ability to protect any external or internal threats and heal itself when it is needed without any user input [9][3]. Autonomic functions and their management are automated in a control loop task called MAPE. Self-optimization consists of the system's automatic ability to configure and optimize itself to achieve top level performance against current settings, workload, and resources [9]. In DBMS environment different features are used to achieve the best optimization. The query optimizer is used to optimize and execute the query execution plan. The Database statistic manager is used to collect statistics of database objects. Such features are already configured to obtain self-optimization in DBMS.

Self-healing is to recover the damaged part or data automatically without any human intervention in order to remain active and operating correctly [9] [45]. Self-healing is a grand-challenge to an autonomic system which first detects a problem in the system, diagnoses it, and then repairs it automatically. Self-Healing deals with lacking precision in the uncontrolled situation and recovers it according to the dynamics. Healing the system is a serious problematic situation when the information is being corrupted by a malicious attack or any insider's malicious intent or by mistake as this could lead to disastrous decisions when it comes to Military or Health database. For this, the system must be smart enough that it can detect the problem, prepare a plan against it and execute it to bring the database to a normal state.

An autonomic computing system configures its components automatically to achieve its goal [9]. In this environment, the system automatically detects changes and configures, reconfigures its components accordingly [48]. Since the adaptation needs to achieve optimal performance, the category of self-configuring is close to self-optimize. Following features provide self-configuration in autonomic DBMS: Memory components, dynamic parameter configuration, supporting objects for performance purpose, such as indexes, materialized views, partitions, etc. are all components which are used to provide self-configuration ability in the Database. Self-protection is a key component of self-managed systems capable of automatically defend against malicious attacks at runtime. A self-protecting system or application proactively identifies malicious threats and triggers necessary actions to stop them [9] [46] [47]. Security professionals used different tools and skills such as (protection filters, detectors of suspicious activity, logging mechanism & backtracking tools) to protect their systems [1].

The organization of this research paper comprises of the following sections. Section 2 discusses autonomic computing in Database Management system that mainly focuses on the self-protecting perspective. Section 3 discusses current approaches to database protection and section 4 present proposed autonomic model w.r.t self-protection. provides analysis and discussion of database protection and section 5 concludes the research and provide future directions.

II. AUTONOMIC COMPUTING IN DBMS

In today's era Complex Databases and their manageability have become a serious concern for organizations nowadays. These databases need to be easily accessible and available to their clients. For this purpose, it requires expert Database Administrators (DBA) for their continuous monitoring, evaluation, and availability. Keeping in view the scarcity of such expert Database Administrators in the market and the cost of their hiring, the concept of the Autonomic Database Management System is introduced which is capable of managing and maintaining such databases without any human intervention [2].

A. Self-Protection in DBMS

Self-protection of the database is to protect your data from both external threats and internal threats and make available 24/7 to their clients. Experienced DBAs are being hired by organizations for continuous monitoring and availability of complex databases. As a DBA has full access to the database so he or she can easily carry out or harm organization data. The organization uses different techniques and methods to protect their information or data from the internal user, but these techniques and methods are insufficient or not enough. In this regard, the database should have some extra ability or features of autonomic computing, i.e. Self-healing, Self-protection, Self-configuration and Self-optimization to protect and manage its information without any human interventions. The autonomic computing system has the capability to respond automatically to any issue occurred and to make the system precise and available to the user.

A number of authors use different techniques and approaches to achieve database security. Data is an important asset for any organization and its security is critical for maintaining the relationship between an organization and its end users. Different techniques such as access control, encryption scheme, auditing policies, and inference control are used in database management system by a different researcher. While combining autonomic properties such as self-healing and self-protection with database security features such as access control, encryption, database auditing features, we can get the more secure DBMS without the involvement or intervention of any DBA or security engineer. Such autonomic properties are very useful for insider threat or monitoring DBA activities. Table I, presents protection techniques against different attacks and Self-protection of external threat is mostly implemented by configuring the firewall and network traffic monitoring. On the other hand, self-protection against internal threat or insider's malicious intent should achieve by obtaining best security policies [2]. Implementing these policies within a database block every attempt to compromise the state of the database. Database security achieved by user access control mechanism and by using stored procedures to manage the internal database threat. When the attacker attempts a request to change security configuration request carried to the stored procedure for verification. Fig 1 shows some critical areas need to be considered in Database Security [5] and how different researcher use different techniques and methods to mitigate these risks.

TABLE. I. PROTECTION TECHNIQUES AGAINST DIFFERENT ATTACKS

Protection Techniques	**Attack type**	**Reference**
Access Control Policies	Used for both insider and outsider attacks	[11, 12, 16, 19, 22]
Mixed Cryptographic Database	Used for both insider and outsider attacks	[13]
RSA Encryption Technique	Insider attack	[15,17]
Attributes Based Encryption	Used for both insider and outsider attacks	[3]
Hash-Based Encryption	Used for both insider and outsider attacks	[18, 28, 29]
Data Centric Approach	Insider attack	[23]
SQL Injection and insider misuse detection system	Used for both insider and outsider attacks	[20]
Auditing Method	Used for both insider and outsider attacks	[24, 27]

Hackers exploit these critical areas and security holes in a database application to gain database administrator (DBA) level grants and privileges to access sensitive data and cause a denial of service (DOS) attacks. Following are the security threats that need attention [10].

- Excessive and unused privileges: granted extra privileges to user that exceed the requirement of their job function
- Privilege abuse: authenticated user misuse authentic database privileges for illegal purposes
- SQL injunction: targets traditional database and big database [NoSQL]. Inserting malicious statement into the input field of web application and big data components.
- Malware: an advance attack that uses multiple approaches to stealing organization data. these approaches are phishing emails and malware.
- Weak audit trail and misconfigured database
- Storage media Disclosure such as backup media needs for special protection.

III. CURRENT PROTECTION APPROACHES

Database security has the main concern of computer security or information security. Security Analyst uses different security controls, i.e. (physical, procedural and technical) to protect their organization data. Protecting databases on multiple hosts and securing information within the database are done with these controls. It's all required deeper research to protect the database from malicious activities. Researcher used different method and techniques such as Access control [4] [11] [12] [14] [16], Encryption technique [3] [13] [15], Audit Trail [19] [24] [27] mechanism for Database security purposes. The Summary of these methods and techniques are as follows.

A. *User Identification*

User identification means to verify any user or application identity who use information or data. User identification is based on password management system and password should keep secret all times. Password management system control through the user profile. Self-protection is a key component of self-managed systems capable of automatically defend against the malicious user, attacks at runtime. A self-protecting system or application proactively identifies users, malicious threats and triggers necessary actions to stop them [9] [46] [47]. Security professionals used different tools and skills such as (protection filters, detectors of suspicious activity, logging mechanism & backtracking tools) to protect their systems [1].

B. *Access Control*

Jabbour, et al. [4] presents Insider threat security architecture (ITSA), of self-protection in databases against insider threats. In this architecture privileged user compromised the database state where ITSA can protect. ITSA framework consists of security policy and defense mechanism managed by the super system owner. Security policy contains system parameter and their values while built-in logic is embedded in defense mechanism in the form of stored procedures and triggers and this logic is used to protect the system parameters. Three main components of ITSA are Autonomic Access Control Enforcement (AACE), Integrated Self-Protection Capability (ISPC), and Integrated Business Intelligence Capability (IBIC). The author discussed how the same scenario can be moderated under the Insider threat security architecture framework.

Jabbour, et al. [11] present notion based self-protection framework within the database by using the policy based approach. These policies are created by the system owner and block every attempt that compromises the Database state. Each action in the database is verified by the system owner before it applied to the database. Protection is achieved by implementing stored procedures, functions and triggers that have the built-in logic of checking insider user request. When an insider or attacker wants to change database security parameters, its request for changing parameters goes through a verification process through stored procedures before the following change can be applied to the database. If the change request truly verifies set of policies, then it can be applied to the database and its audit trail is maintained in the database. If the request is not verified from stored procedures, then change request is blocked and system owner is alerted through email and audit trail is maintained. Author present four types of policies, i.e. verifying and controlling user actions, monitoring database resources, changing security policy conditions and their parameters.

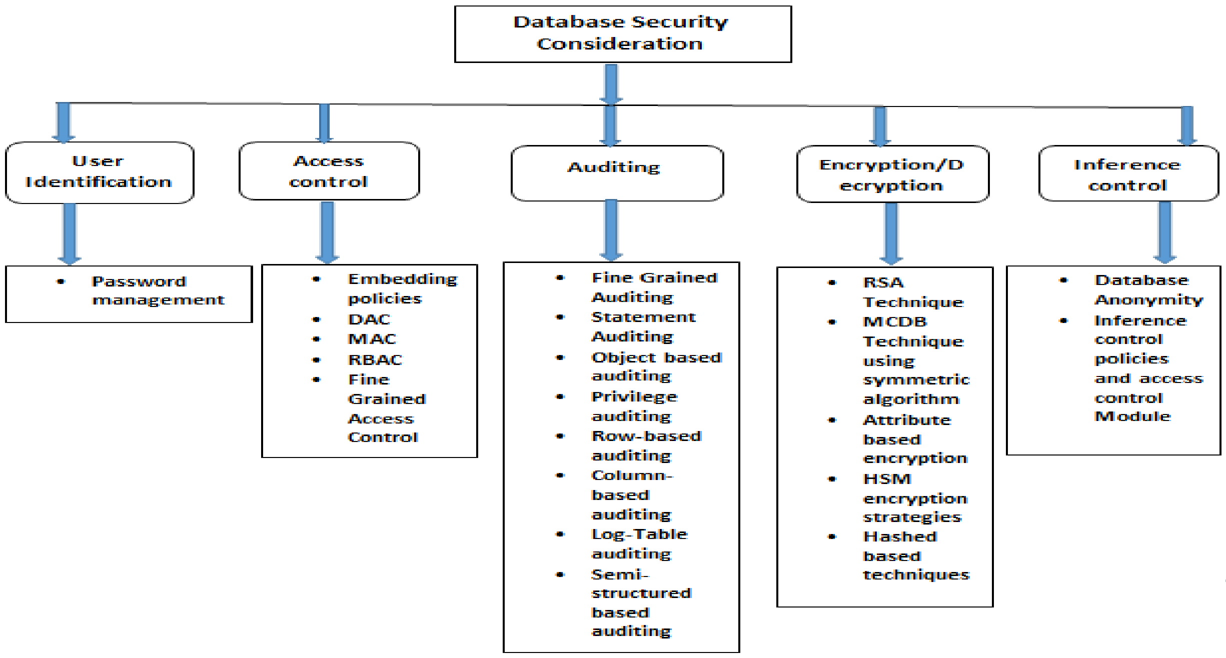

Fig. 1. Critical areas need to be considered in DBMS Security

Jabbour, et al. [12] addresses a protective framework for securing autonomic system policies. The author used two types of methodology in this framework. The first type is to partition security policies, blocks into numerous levels and then adding complexity to the entire architecture of the policies. This assists the purpose by adding alleged obscurity, which denies the potential attackers from decoding the policy's contents and directives. The second method is to insert false sense or false elements to different partitions of the policies (parameters and their values). Whose purpose is too confusing an attacker and giving a false sense of accomplishing his/her goal. K. Ahmed, et al. [14] addressed different types of a security layer, i.e. Database administrator (DBA), the System administrator (SA), Security officer (SO), Database developers and client or end user. These security layers are applied at almost all DBMS i.e. (Oracle, SQL Server, DB2, Teradata) environment. Theses security layers are responsible for implementing some well-defined security policies. The purpose of implementing these policies to ensure security features such as Confidentiality, integrity, efficiency, access control and privacy within the database.

A. Patil, et al. [16] presented Access control policy mechanism is used to secure a database against insider user. Three types of AC policies are mainly used, i.e. discretionary access control policy (DAC), Mandatory access control policy (MAC) and Role Base access control policy (RBAC). DAC based on the discretion of information creator or owner of the data. DAC used to restrict access of user on the basis of user identity and authentication. In MAC all users follow the same rule created by the Database administrator. RBAC used in a large organization where turnover rate of the employee is high. RBAC model built on the notion of role where role signifies a specific function within the organization. Each user performs a specific action which is granted to the specific role associated with it.

C. Auditing

Auditing is one of the important components in Database security infrastructure. In the database production environment in various database operations such as user login, Data manipulation language statements (DML), Data definition language statements (DDL) are needed to obtain an audit trail. Different methods and techniques are used by Researcher for auditing. The Database auditing purpose is to monitor and record user actions what he or she performs on the database.

Olumuyiwa O. Matthew et al. [24] discussed several already existing database auditing techniques such as statement auditing, privilege auditing, schema object auditing and fine-grained auditing etc. at various database environments. The author also discussed issues concerning about handling of audit trails against different database environment. According to author Database Auditing performs level by level. At first level logging (login and logoff) activities are a monitor, second level privileges check are an audit. In third level changes made to database schema are monitored, fourth level database DML activities are monitored and fifth level concerned with auditing changes made to a stored procedure, function and other codes. In next level database error is an audit and in the last level auditing any changes made to the definition of what is to be audited.

Li Yang [25] developed to extend auditing concept and technique by applying practical lab experience on security and auditing of a relational table that comprising an audit log of all commands and causes data changes on the target table. Some Common techniques of database auditing for monitoring database access control attempts, user login and logoff attempts, Data Control Language (DCL) activities, Data Definition Language (DDL) activities, and Data Manipulation Language (DML) activities. Erroneous queries should also be logged and monitored. Database auditing is implemented through log files and audit tables. According to author security and auditing should be applied with integrated way.

Liu and Huang [26] present a framework of network-based database auditing that offers zero-impact of database performance. An agent is configured in passive mode to capture traffic flowing from the Database system and extract the audit log data which is beneficial for audit log analysis and then store this log information on another server. The author used Berkeley Packet Filter (BPF) filtering mechanisms to capture traffic and compare them against given conditions. They divided their methodology into three steps: packet filtering, the packet analyzing and data storage. Then alarm will be generated against any database anomaly or upon detection of malfunction of security regulation.

Narongrit Waraporn [27] suggested four methods implement database auditing for historical data. These methods are row-based auditing, column-based auditing, log-table auditing and semi-structure-based auditing. In row-based auditing, a separate audit table was created against each relational table. The operational table contains the last updated value while auditing table contains both static and historic data, two timestamps (start time and end time), operation type (update, delete, insert) and username. Row-based auditing caused data redundancy because the same record exists in two tables. To remove data redundancy column-based auditing is used. Column-based auditing does not contain the static data in auditing table. Column-based auditing caused null value in auditing table. The author suggests two approaches using log-table mechanisms. In the first approach extra table creates against each auditing column, while in second approach the only single audit table will be created against all operational tables. Semi-structure-based auditing also categories in two ways, i.e. Object-relational type, and XML type.

D. Fabbri et al. [31] proposed the idea of select triggers which are executed implicitly when a select query takes place on a specific object on which it is defined. Mostly none of the database management systems are implementing such a select trigger. Only Microsoft, however, is working on select query trigger and its researchers have presented their work earlier. Mostly triggers are based on the insert, update or delete commands, but the author also extends trigger in select command. It is also important to understand the action which is performed during trigger execution. The major issue of integrating select triggers in the DBMS is to handling a low overhead mechanism while ensuring the semantics are richly adequate to capture the modification of data access using SQL queries.

D. Encryption/ Decryption

In [3], Akinyele et al. present a flexible approach using attribute-based encryption (ABE) to generate self-protecting electronic medical records (EMRs), when health data is transferred on cloud servers or cell phones which are outside the trust boundaries of the healthcare organization. The EMR system ensures availability when the provider is offline. In this approach, the patient can encode each node of medical records in XML-based EMR file with attached access policy before it is transferred to the cloud storage. The Policy engine creates these access policies over electronic medical records on the basis of different user types (patient, physician, and insurance agent). Policy engines further define attribute sets, i.e. record type, patient age, and date to encode each record using attribute based encryption.

H. Kadhem, et al. [13] presented Mixed Cryptographic Database MCDB [13], a new data classification framework used to protect the databases by encrypting it in the semi-trusted scenario where data are shared among different parties using different keys. In this technique, database encryption is done over the unsecured network in an altered way that involves keeping numerous keys of different parties. In This scheme encryption is done at the client side, untrusted database and server side and it use symmetric key encryption mechanism. The purpose of keeping numerous keys by different authenticated parties that when the database is attacked by the attacker (insider or outsider) the database is not compromised. The performance of queries and security analysis is affected because of encryption Algorithms.

S. Sachdeva et al. [15] proposed negative database as extra security layers on generic databases. Negative data defined by some database security researchers as a database that contains a large amount of data consisting of bogus data and as well as real data. In this approach, author separated the information into two parts, i.e. sensitive information and non-sensitive information. Non-sensitive information directly stores in the Database while sensitive information first encrypted using RSA encryption algorithm and then convert the cipher text of sensitive information into base 32 shrink its length and then create large amount counterfeit. Now encrypted sensitive data along with counterfeit data stored in the database.

L. Bouganim et al. [18] suggest A new approach which embeds the security server inside the hardware security module (HSM). HSM is used to manage users, privileges, encryption policies and keys. HSM is responsible for all cryptographic operations and encryption keys are not exposed from this technique. Security server cannot modify or altered because it’s fully embedded in the tamper-resistant Hardware Security module. The main limitation with this approach is that the Hardware Security Modules require a complex piece of software to be embedded in it. In this approach, encryption is done at the storage level, database level, and application level.

R. Jena et al. [28] proposed a cryptographic hash based function and digital timestamp technique to prevent from silently corrupting audit log files from both insider and outsider malicious user. Proposed technique will be implemented for the database system and trusted timestamp is efficiently used if logs are compromised or corrupted. The author implements their results in a high-performance engine. Audit log files comprise of log entries and each entry contains an element in a hash chain which authenticates the value of previous log entries. Two additional columns such as HashCode and Chain_ID and an additional table for digital timestamp is added. Chain_ID contain at most recent digital timestamp and it is generated by timestamp authority. Hash code based on previous values or data. If any audit log entry is tempered then database forensic analysis algorithm identifies the tempering and regulate who, when, where and what components of audit log are tempered.

Kyriacos E. Pavlou et al. [29] developed a prototype DRAGOON to monitor the audit logs of the database and then detect malicious activities and perform forensic analysis against both insider and outsider users. They added some additional properties in DRAGOON to support information accountability in a cloud computing environment. The author used a cryptographic one-way hash function to protect silently corrupting audit log from an insider or an outsider or an unknown error in DBMS. Analyst used a series of algorithms which were designed for the forensic purpose to detect malicious activities. Extending some more features in DRAGOON architecture in database management systems increase scalability and it supports multiple databases and DBMSs. Extended DRAGOON architecture isolates four different areas of control. The first area is user application and GUIs controlled by the company itself. The second area is monitored by cloud provider where the monitored database resides (CLOUD A). The third area is monitored by cloud provider where DRAGOON resides (CLOUD B). The final area is END, which should not use cloud services. The extended DRAGOON architecture is scalable and customizable for providing a level of security and forensic analysis.

Kyriacos E. Pavlou et al. [30] highlighted the deep relations between time and the definition, Temper detection, forensic analysis of temper detection, and characterized different level of a database exploitation within the context of information accountability. Time in the context of applying information accountability and identifying time-security interactions. They categorized their audit system in three phases. The first phase is audit system execution phase second is their sub-phases and the third phase is an action performed during each phase. Transactions are hashed and cumulative associated with a cryptographically strong hash function in the first phase and the results of its digitally notarized with an external digital notarization service. In the second phase hash values are again extracted and matched from previously notarized. If the hash values are not matched from previously notarized, then these values are detected. The author introduces different forensic algorithms to detect when malicious activities occurred and what type of data has been corrupt.

E. Inference control

Inference control is a data mining technique used to attack databases where malicious user or attacker infers data from complex databases at a high level. The inference is used to find information hidden from common users. Popeea T et al. [32] presented multi-layer security to database anonymity and database security in a data warehouse which contains information of current and past employees of large companies. They mainly focus on securing communication channel, securing operating system and securing the database. They developed an engine based on java, which provides protection of both static and dynamic sensitive data. In this paper, an inference can be classified into six categories, i.e. splits queries, overlapping inferences, subsume inferences, complementary inferences, unique characteristic inferences and functional dependency inferences. To achieve high-level database security, they used mandatory access control layer, secure communication channel SSL, Ubuntu OS enhanced with MAC module and MYSQL as an open source DBMS.

Yang et al. [33] provide a secure inference control model by the trusted computing paradigm. This model entrusts the implementation of inference control to specific users' computer platforms. In this architecture, the database server is liable for the implementation of traditional access control, while the individual user's platform is allowed to handle inference control based on their own query logs in a decentralized manner. This architecture is used for complex and large databases. In traditional architecture, both access control and inference control are imposed at The Database server side. The Access control module (ACM) implements access control functionality, while the inference control module (ICM) performs a designated inference control algorithm. In the new architecture, inference control module resides at user side instead of server side. The user requests a query to inference control module (ICM), ICM transfer this query request to the access control module (ACM). ACM further check user requests against access rules and policies. If the user has granted access, then ACM return a response to ICM together with IC policy. Table II summarized the literature review with respect to protection in DBMSs.

TABLE. II. LITERATURE REVIEW SUMMARIZED

References	Research Contribution	Attack Type	Protection Techniques Used	Limitation
[11]	Policies are enforced for securing database configuration from inside user or DBA	Insider Attack	Embedding policies in DBMS	Policies based on the notion. Database configuration specific policies. No confidentiality provided as DBA can view data.
[12]	Protect security policies of autonomic system	Insider attack	Partitioning and giving a false sense by adding false elements	
[13]	Encryption of database over untrusted networks, data classification is based on data ownership data is confidential if one key compromise	Both for Insider and Outsider attacks	MCDB technique using any symmetric algorithm	Performance of queries and security analysis is affected because of encryption Algorithms.
[4]	ITSA based on security policies and defense mechanism.	Insider attacks	Security policy and defense mechanism	Works only Autonomic Access Control Enforcement, Integrated Self-

	Security policies consist of database parameter and their values Defense mechanism comprises logic encoded in a set of stored procedures.			Protection Capability, and Integrated Business Intelligence Capability
[14]	Define database security layer Each layer has some specified policies for authentication and authorization purpose	Used for both insider and outsider attack	Policies based	
[15]	Proposed extra security layer through negative database techniques Entity, attribute, and value (EAV) model is used	Insider attack	Negative DB and RSA encryption technique	More complex Taking more query execution time More costly and variable k value is fixed
[16]	Review key access control models	Both for Insider and Outsider attack	DAC, MAC, RBAC	
[17]	Protection of real world health databases to restrict access to data from internal user or outsider	Both for Outsider and Insider attack	RSA Technique	Provide application based Database security Not for generic database Security.
[3]	Implementing autonomic property to protect electronic medical records (EMRs) using attribute-based encryption scheme (ABE).	Both for inside and outside attacks	Attributes based encryption access control using RBAC and content based	Encryption and decryption time based on a number of attributes in access policies.
[18]	Review different encryption level, techniques, and methods Key management and their issues.	Both Insider and outsider attacks	HSM Encryption Strategy for key management.	HSM now requires a complicated piece of software to be embedded in it
[19]	DBMS-Layer is a most appropriate layer to protect against insider for exfiltration detection. Virtualization techniques are used to tackle provenance.	Insider attack	Role based access Profiles and threshold Provenance Embedding and virtualization Techniques	Modeling and Specification of Lineage Information Authentication and Authorization Systems and network issues
[20]	Discuss database security threats against both internal and external threats. Proposed SIIMDS to detect both internal and external attacks.	Both internal and external attacks	SQL Injection and Insider Misuse Detection System (SIIMDS)	
[21]	A large number of abnormal queries are running in same specific time caused query-flood attacks. Degrade database performance.	Insider attack	Attack detection algorithms	DB performance slow
[22]	Review all requirements of access control for the context of scalability, granularity, and situation-aware decisions.	Insider attack	RBAC approach Fine Grained Access Control	Implementation is not done.
[23]	Each activity of users is modeled on the basis of SQL commands running and data generated by that user.	Insider Attack	DATA-CENTRIC APPROACH	Performance consideration
[24]	Outlines main auditing techniques and methods Issues relating to handling of audit trail are also discussed and key important impacts of security are also highlighted	Both internal and external attacks	Auditing methods such as FGA, Statement auditing, Privilege auditing, schema object auditing	Discussed already existing auditing technique
[25]	To engage students actively, practical labs are developed to assimilate theories of database security and auditing Use of two major database products (Microsoft SQL Server and Oracle 10g	Paper used for Database Security purpose	Monitoring database access attempts, DCL activities, DDL activities, and DML activities	Some issues regarding terminology and capabilities of DB are not completely discussed in a hands-on lab.
[26]	Monitor network flowing into and of DB system and generate log information about DB Execute audit analysis through event correlation Generate alarm in case of any violation detected	Both internal and external attacks	Agent-based network monitoring Used Berkeley Packet Filter (BPF) filtering to scan packets	Network-based logging has its limitation too if DB has been encrypted, then passive packets capturing method will be invalid
[27]	Discuss different four methods to achieve database auditing. Discuss multiple audit log columns, tables for transaction logs Single audit table for transaction logs	Paper used for Database Security purpose	Row-based auditing Column-based auditing Log-Table auditing Semi-structured based auditing	Row-based auditing caused data redundancy In column-based auditing, null values in the table would lead to problem
[28]	Auditing data integrity themselves is a very serious concern Malicious activities are performed both by authorized user and as well as unauthorized user	Both internal and external attacks	Cryptographic Hash-based technique used for forensic analysis Trusted Timestamping used to prevent the log files from both internal and external	Implement in the only online transactional database. Does not produce tamper resistant audit log

			users	
[29]	Developed a prototype called DRAGOON for information accountability periodically audits database, detect malicious activities and then perform forensic analysis DRAGOON support non-cloud databases Deploy how existing prototype extending within the cloud	Both internal and external attacks	DRAGOON use cryptographic hashing technique	Concurrency issue raised when transaction data is replicated at application level Hashing occur at the application level is open design issue
[30]	Notarization and validation of database exploit the temporal semantics of transaction times database.	Insider attacks	Monochromatic forensic algorithms	
[31]	Triggers are useful to track and log any changes made on data by executing any DML commands Trigger assists row-level auditing of both DML and DDL commands Select trigger fires when a select operation takes place on the object	Internal attacks	Select trigger techniques	Some scenario's large number of false positive occur
[32]	Inference detection is done here with SSL communication channel The Re-identification algorithm is an implementation of k-anonymity	Both internal and external attacks	Split queries	Data anonymity is not fully completed

IV. Proposed Autonomic Model W.R.T Self-Protection

In this proposed model firstly we will explain how an adversary or the attacker can perform what types of malicious actions to compromise database state. In our survey adversary can be internal users or database administrator. He or she can perform the following actions to attack the database. The Attacker can change some configuration parameters of database management system that change the state of the database in a way that Database performance is slow or it's not obvious to its end user. For example, in Oracle database, database administrator changes various system configuration parameters such as disable auditing parameter or run the database in NOARCHIEVE log mode or changes some other security parameter that compromises the database health or its behavior.

In some organization, DBA with full access to the database can run any DML (update, delete or insert) or DDL (create, alter, drop, truncate) commands on sensitive data or information to change it. The DBA can also drop any database or drop any schema in the database. Audit trail used for forensic analysis, provide documentary proof of the sequence of actions that have affected at any time a specific operation. The attacker also Change or remove Audit trail information in the database. If a user is granted database privileges that exceed their job role and requirements, then those privileges can be abused or misused.

Fig 2 shows proposed autonomic model against insider threats in DBMS with respect to self-protection diagram. In this proposed model, we mainly discussed self-protection property in database management system against insider threats. Self-protection against insider threats in DBMS, previously proposed techniques is based on embedding security policies for enforcing database security configuration parameters. In our proposed architecture, we imposed CIA (Confidentiality, Integrity, and Availability) security model for building policies against these three properties.

In this model super user, build security policies for database security. These security policies are related to Access control, Database configuration parameter setting policies, password management policies and encryption policies, etc. When an insider user or DBA attempts to change in the DBMS through SQL command Prompt, the request goes for verification phase. If the request verifies set of policies, then the request will be applied in DBMS and audit trail record will also be saved in a log table, else insider request will be rejected, alert through an email is generated to the super user, notify the insider user and audit trail will be recorded. For monitoring malicious activities against internal threats we used Alert mechanisms. When any malicious activities found alert will be generated.

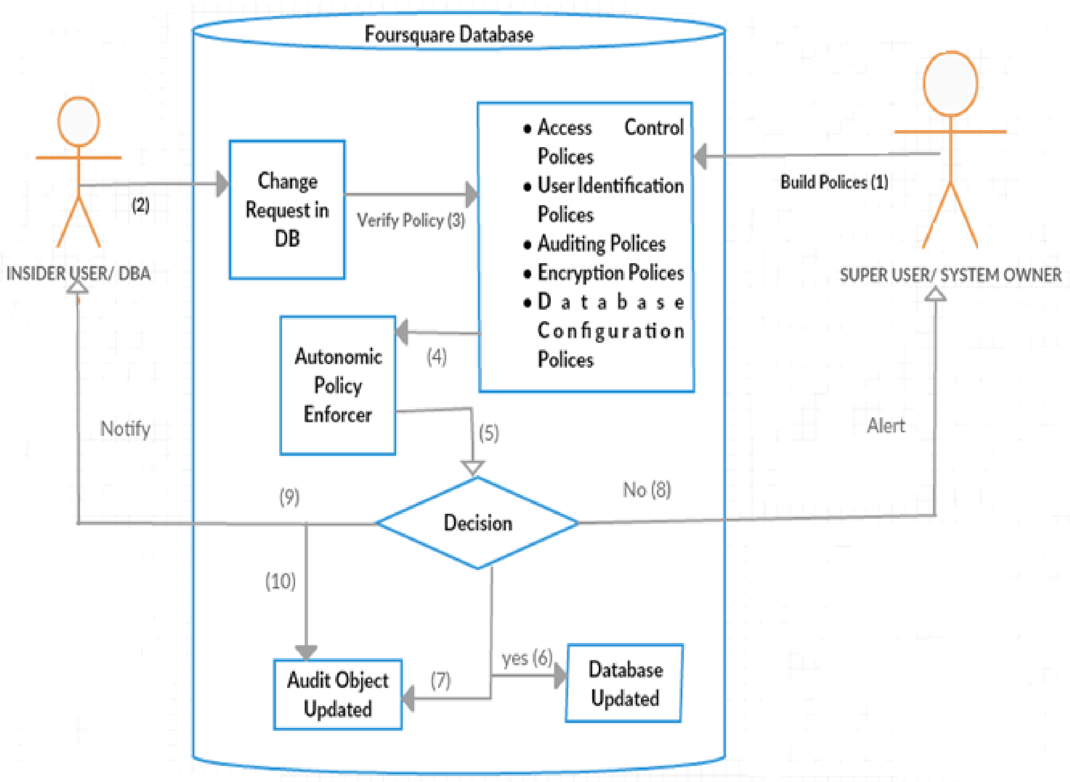

Fig. 2. Autonomic Model against Insider Threats in DBMS with respect to Self-Protection

In data confidentiality insider user or DBA has full access to view sensitive information and non-sensitive information. In our model super user-segregated information into two ways, i.e. sensitive information is non-sensitive information. Sensitive information stored in encrypted form in the database after applying the encryption function and the non-sensitive information is stored as it is in the database. The only superuser can encrypt or decrypt sensitive information. Our purpose is to make sensitive information more confidential from inside user. We evaluated our proposed Architecture with an already existing architecture based on the following criteria:

- Set of Polices is verified using a set of queries.
- Improved Autonomous property of self-protection. (Autonomic Improving Capability)
- Generation of alerts at the time of any attack in DB.

We are expecting our enhanced model provides more secure protection against insider threats in database management systems.

V. Conclusion and Future Work

Data is probably most valuable property on which entire organization depends. Database security is one of the main concerns of the researchers nowadays. This paper addressed the security threats against database management systems and how to mitigate these threats by using autonomic computing properties. The research emphasized some critical areas such as access control, encryption, auditing, accountability and inference control that need to be considered in database security. This study identified, how malicious user exploits these areas and gain DBA level access to the database and causes a denial of service attack. An autonomic model is proposed which protects data against insider threats. A number of security techniques and policies are addressed that should be used in database management system to achieve protection against the insider threats. The premise of our proposed model is to highly enforce the concept of separation of duties in an organization and also brings security. We adapted the concept of building system level policies in such a way that meets the autonomous self-protecting capabilities to defeat privileged insider users and unintentional actions. Organizations owner or super-user builds policies for database security against critical areas. The alerts can also be generated through an email against malicious activities of insider user.

As for future research, we will implement and demonstrate all above mention policies in database management system. We plan to implement access control policies at the database connection level, DML and DDL command level to achieve self-protection in DBMS. Similarly, we will implement database configuration level policies, encryption level policies and auditing level policies, etc. We believe that it would be valid and beneficial attempts to apply and demonstrate these different levels of policies in the DBMS environment to achieve security.

References

[1] Depalma, N.; Claudel, B.; Lachaize, R, “Self-Protected System: an experiment,” In 5th Conference on Security in Network Architectures (SAR). Addison Wesley, Longman, England, 2006.

[2] B. Raza, A. Mateen, M. Sher, M. M. Awais, and T. Hussain, “Autonomicity in Oracle Database Management System,” 2010 Int. Conf. Data Storage Data Eng., pp. 296–300, Feb. 2010.

[3] A. Akinyele, C. U. Lehmann, M. D. Green, M. W. Pagano, Z. N. J. Peterson, and A. D. Rubin, “Self-protecting electronic medical records using attribute-based encryption,” Cryptology ePrint Archive, Report 2010/565, 2010.

[4] Jabbour, G. G., & Menasce, D. A, “The Insider Threat Security Architecture: A Framework for an Integrated, Inseparable, and Uninterrupted Self-Protection Mechanism,” 2009 Int. Conf. Comput. Sci. Eng., pp. 244–251, 2009.

[5] L. Basharat, F. Anam and A. Wahab Muzaffar, “Database Security and Encryption; A Survey Study”, International Journal of Computer Application, vol. 47, (2012), pp. 28-34.

[6] Cert, “cert insider,” 2007. [Online]. Available: http://www.cert.org/insider-threat/research/database.cfm?

[7] P. Horn, “a u t o n o m I c o m p u t i n g : the information technology industry loves to prove the impossible possible. We obliterate barriers and set records with astonishing regularity. But now we face a problem springing from the very core of ou,” 2011.

[8] M. R. Nami and K. Bertels, “A survey of autonomic computing systems,” in ICAS ’07: Proc. Third International Conference on Autonomic and Autonomous Systems. Washington, DC, USA: IEEE Computer Society, 2007, p. 26.

[9] M. C. Huebscher and J. A. McCann, “A survey of autonomic computing - degrees, models, and applications,” ACM Comput. Surv., vol. 40, no. 3, 2008.

[10] P. A. Pe, “Top Ten Database Security Threats The Most Significant Risks of 2015 and How to Mitigate Them Red Flag.”

[11] Jabbour, G. G., & Menasce, D. A, “Policy-Based Enforcement of Database Security Configuration through Autonomic Capabilities,” Fourth Int. Conf. Auton. Auton. Syst., pp. 188–197, Mar. 2008.

[12] I. Technology and C. Science, “Securing Security Policies in Autonomic Computing Systems,” 2008.

[13] H. Kadhem, T. Amagasa, and H. Kitagawa, “A Novel Framework for Database Security Based on Mixed Cryptography,” 2009 Fourth Int. Conf. Internet Web Appl. Serv., pp. 163–170, 2009.

[14] K.Ahmad, “Policy Levels Concerning Database Security,” no. Feb 2016.

[15] S. Sachdeva, “Implementing Security Technique on Generic Database,” 2015.

[16] A. Patil and P. B. B. Meshram, “Database Access Control Policies,” Applications (IJERA) vol. 2, no. 3, pp. 3150–3154, 2012.

[17] N. Batra and Pooja, “Secure Mechanism for Medical Database Using RSA,” IJAIEM vol. 3, no. 7, pp. 320–327, 2014.

[18] L. Bouganim and Y. Guo, “Database encryption,” in Encyclopedia of Cryptography and Security, Springer, 2010, 2nd Edition.

[19] Bertino and G. Ghinita “Towards mechanisms for detection and prevention of data exfiltration by insiders.” In Proc. 6th ACM Symp. on Information, Computer, and Communications Security. pages 10–19, 2011.

[20] Asmawi A, Sidek ZM, Razak SA. " System architecture for SQL injection and insider misuse detection system for DBMS," in International Symposium on Information Technology (ITSim'2008), 2008, pp. 1 -6.

[21] A. C. Squicciarini, I. Paloscia, and E. Bertino, “Protecting databases from query flood attacks,” in ICDE, 2008, pp. 1358–1360.

[22] Park, J. S. & J. Giordano, “Access Control Requirements for Preventing Insider Threats,” Proc. ISI’06 LNCS 3975, pp. 529–534, Springer, 2006.

[23] S. Mathew, M. Petropoulos, H. Q. Ngo, and S. Upadhyaya, "Data-Centric Approach to Insider Attack Detection in Database Systems," Recent Advances in Intrusion Detection, 2010.

[24] O. O. Mathew and C. Dudley. "Critical Assessment of Auditing Contributions to Effective and Efficient Security in Database Systems." Int Conf on CSITA, At Royal Orchid Central Bangalore, India pp. 1-11, March, 2015.

[25] Yang, L., “Teaching Database Security and Auditing,” Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE), Chattanooga TN, March, 2009.

[26] Liu, L. and Huang, Q. "A Framework for Database Auditing". Computer Sciences and Convergence Information Technology, 2009.

[27] Waraporn, N. “Database Auditing Design on Historical Data.” In Proceedings of the Second International Symposium on Networking and Network Security (ISNNS ’10). Jinggangshan, China, April. 2010, pp. 275-281

[28] Jena, R., Aparna, M., Sahu, C., Ranjan, R. and Atmakuri. "Ensuring Audit Log Accountability through Hash Based Techniques." International Journal of Future Computer and Communication 1.4, Dec. 2012.

[29] Kyriacos E. Pavlou and Richard T. Snodgrass, "Achieving Database Information Accountability in the Cloud" Tucson, AZ 85721–0077, USA, 2002.

[30] Pavlou, Kyriacos E., and Richard T. Snodgrass. "Temporal implications of database information accountability." 2012 19th International Symposium on Temporal Representation and Reasoning. IEEE, 2012.

[31] Fabbri, D., Ramamurthy, R. & Kaushik, R. “SELECT triggers for data auditing.” Proceedings of the 29th International Conference on Data Engineering (ICDE). IEEE:1141-1152, 2013.

[32] T. Popeea, A. Constantinescu, L. Gheorghe and N. Țăpuș, "Inference Detection and Database Security for a Business Environment.", International Conference on Intelligent Networking and Collaborative Systems, (2012), pp. 612-617.

[33] Yang, Y., Li, Y., and Deng, R.H., “New paradigm of inference control with trusted computing.”, in Proceedings of the 21st annual IFIP WG 11.3 working conference on Data and applications security, Redondo Beach, CA, USA. 2007, pp. 243-258.

[34] Martin S. Olivier. “Database privacy: balancing confidentiality, integrity and availability.” SIGKDD Explor. Newsl., 4(2):20–27, 2002.

[35] Von Solms R, Van Niekerk J. “From information security to cyber security.” Computers & Security. 2013 Oct 31;38:97-102.

[36] Safa, N. S., Von Solms, R., & Furnell, S. “Information security policy compliance model in organizations.” computers & security, 56, 70-82, (2016).

[37] Chen D, Zhao H. “Data security and privacy protection issues in cloud computing.” InComputer Science and Electronics Engineering (ICCSEE), 2012 International Conference on 2012 Mar 23 (Vol. 1, pp. 647-651). IEEE.

[38] Tianfield, Huaglory. "Security issues in cloud computing." In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on, pp. 1082-1089. IEEE, 2012.

[39] Buyya R, Calheiros RN, Li X. “Autonomic cloud computing: Open challenges and architectural elements.” InEmerging Applications of Information Technology (EAIT), 2012 Third International Conference on 2012 Nov 30 (pp. 3-10). IEEE.

[40] Patel A, Taghavi M, Bakhtiyari K, JúNior JC. “An intrusion detection and prevention system in cloud computing: A systematic review.” Journal of network and computer applications. 2013 Jan 31;36(1):25-41.

[41] Maggio M, Hoffmann H, Papadopoulos AV, Panerati J, Santambrogio MD, Agarwal A, Leva A. “Comparison of decision-making strategies for self-optimization in autonomic computing systems.” ACM Transactions on Autonomous and Adaptive Systems (TAAS). 2012 Dec 1;7(4):36.

[42] De Lemos, Rogério, Holger Giese, Hausi A. Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl et al. "Software engineering for self-adaptive systems: A second research roadmap." In Software Engineering for Self-Adaptive Systems II, pp. 1-32. Springer Berlin Heidelberg, 2013.

[43] Frei, Regina, Richard McWilliam, Benjamin Derrick, Alan Purvis, Asutosh Tiwari, and Giovanna Di Marzo Serugendo. "Self-healing and self-repairing technologies." The International Journal of Advanced Manufacturing Technology 69, no. 5-8 (2013): 1033-1061.

[44] Eze, Thaddeus, Richard Anthony, Chris Walshaw, and Alan Soper. "Autonomic computing in the first decade: trends and direction." In Proceedings of the Eighth International Conference on Autonomic and Autonomous Systems (ICAS). 2012.

[45] Ferreira da Silva, Rafael, Tristan Glatard, and Frédéric Desprez. "Self-healing of operational workflow incidents on distributed computing infrastructures." In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pp. 318-325. IEEE Computer Society, 2012.

[46] Chen, Qian, Sherif Abdelwahed, and Abdelkarim Erradi. "A model-based approach to self-protection in computing system." In Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, p. 16. ACM, 2013.

[47] De Palma, Noel, Daniel Hagimont, Fabienne Boyer, and Laurent Broto. "Self-protection in a clustered distributed system." IEEE Transactions on Parallel and Distributed Systems 23, no. 2 (2012): 330-336.

[48] Ayala, Inmaculada, Mercedes Amor, and Lidia Fuentes. "Self-configuring agents for ambient assisted living applications." Personal and Ubiquitous Computing 17, no. 6 (2013): 1159-1169.