A Novel Survey: Current Trends in Database Management System and Their Impact on Security

Anjali Yadav, Assistant Professor, RKGIT Ghaziabad, India, anjali.yadavskb@gmail.com
Praveen Kumar, Assistant Professor, RKGIT Ghaziabad, India, ranagfcs@rkgit.edu.in
Lalit Kumar Saraswat, Assistant Professor, RKGIT Ghaziabad, India, lalitfcs@rkgit.edu.in
Manish Gupta, Assistant Professor, RKGIT Ghaziabad, India, manihfcs@rkgit.edu.in

Abstract - In today's era, as organizations increase their reliance on information systems for daily business, it becomes more vulnerable to security breaches even though they gain the advantages of productivity and efficiency. Although various techniques are available to secure the data such as encryption and electronic signatures when transmitted across sites, a comprehensive approach for data security must also include mechanisms for enforcing access control policies based on data contents, subject qualifications and characteristics, security issues and many more. In this modern era where everything is digital, the number of reported cases of data breaches, cyber crime, cyber attacks, leak of sensitive information, disclosure of confidential data, data intrusion increases day by day, in that case need for industries dependent on databases which ensure their data security and defend their data from all the security threats is the main concern. The primary goal of database security is to restrict exposure of unnecessary information and updating data while ensuring the availability of the needed services. In this paper, we discuss different security methods that have been created to protect the databases and various different security models have been developed based on different security aspects of the database. Use all these different security methods when the database management system is designed and developed for protecting the database.

Keywords — Cyber Crime, Threat, SQL Injection, Vulnerabilities

I. INTRODUCTION

In today's time the data has become a strategic asset of high-value because of its ability to make new discoveries. Data has become an indispensable part of everyone's life from a person keeping track of his monthly expenses to an IT company trying to boost its revenues through data mining. The collection of data related to each other, referred as a Database contains all the information pertinent to an organization. Database has made it possible for businesses to add to the effectiveness of their operations and enhance its capabilities. Any organization's success and failure probability depends upon the quality, quantity and level of security of their database. However, these advancements come along with various security threats like malicious people targeting data and compromising data integrity, unauthorized access to data and lastly data critical to the industry getting leaked to the outside world. Since data held by the database is of great significance, it is utmost important to secure the database. Database security refers to the process of preventing data from unauthenticated misuse, inadvertent mistakes, data loss and corruption or any unintended activity on the database. Just like every tangible asset of the organization is protected, organization's data in the database is one of the key assets that needs to be secured. Data resides in the database at different levels namely physical, data, network, application and host level. Data security ensures all the levels of the database are protected. As organizations increase their adoption of database systems as the key data management technology for day-to-day operations and decision making, the security of data managed by these systems becomes crucial. Damage and misuse of data affect not only a single user or application, but may have disastrous consequences on the entire organization. The recent rapid proliferation of Web based applications and information systems have further increased the risk exposure of databases and, thus, data protection is today more crucial than ever. It is also important to appreciate that data needs to be protected not only from external threats, but also from insider threats. Security breaches are typically categorized as unauthorized data observation, incorrect data modification, and data unavailability. Unauthorized data observation results in the disclosure of information to users not entitled to gain access to such information. All organizations, ranging from commercial organizations to social organizations, in a variety of domains such as healthcare and homeland protection, may suffer heavy losses from both financial and

 DOI : 10.35291/2454-9150.2021.0349

human points of view as a consequence of unauthorized data observation. Incorrect modifications of data, either intentional or unintentional, result in an incorrect database state. Any use of incorrect data may result in heavy losses for the organization. When data is unavailable, information crucial for the proper functioning of the organization is not readily available when needed. Thus, a complete solution to data security must meet the following three requirements: 1) secrecy or confidentiality refers to the protection of data against unauthorized disclosure, 2) integrity refers to the prevention of unauthorized and improper data modification, and 3) availability refers to the prevention and recovery from hardware and software errors and from malicious data access denials making the database system unavailable. These three requirements arise in practically all application environments. Consider a database that stores payroll information. It is important that salaries of individual employees not be released to unauthorized users, that salaries be modified only by the users that are properly authorized, and that paychecks be printed on time at the end of the pay period. Similarly, consider the Web site of an airline company. Here, it is important that customer reservations only be available to the customers they refer to, that reservations of a customer not be arbitrarily modified, and that information on flights and reservations always be available. In addition to these requirements, privacy requirements are of high relevance today. Though the term privacy is often used as a synonym for confidentiality, the two requirements are quite different. Techniques for information confidentiality may be used to implement privacy; however, assuring privacy requires additional techniques, such as mechanisms for obtaining and recording the consents of users. Also, confidentiality can be achieved by means of withholding data from access, whereas privacy is required even after the data has been disclosed. In other words, the data should be used only for the purposes sanctioned by the user and not misused for other purposes. Data protection is ensured by different components of a database management system (DBMS). In particular, an access control mechanism ensures data confidentiality. Whenever a subject tries to access a data object, the access control mechanism checks the rights of the user against a set of authorizations, stated usually by some security administrator. An authorization states whether a subject can perform a particular action on an object. Authorizations are stated according to the access control policies of the organization. Data confidentiality is further enhanced by the use of encryption techniques, applied to data when being stored on secondary storage or transmitted on a network. Recently, the use of encryption techniques has gained a lot of interest in the context of outsourced data management; in such contexts, the main issue is how to perform operations, such as queries, on encrypted data. Data integrity is jointly ensured by the access control mechanism and by semantic integrity constraints. Whenever a subject tries to modify some data, the access control mechanism verifies that the user has the right to modify the data, and the semantic integrity subsystem verifies that the updated data are semantically correct. Semantic correctness is verified by a set of conditions, or predicates, that must be verified against the database state. To detect tampering, data can be digitally signed. Finally, the recovery subsystem and the concurrency control mechanism ensure that data is available and correct despite hardware and software failures and accesses from concurrent application programs. Data availability, especially for data that are available on the Web, can be further strengthened by the use of techniques protecting against denial-of-service (DoS) attacks, such as the ones based on machine learning techniques. In this paper, we focus mainly on the confidentiality requirement and we discuss access control models and techniques to provide high-assurance confidentiality. Because, however, access control deals with controlling access to the data, the discussion in this paper is also relevant to the access control aspect of integrity, that is, enforcing that no unauthorized modifications to data occur. We also discuss recent work focusing specifically on privacy-preserving database systems. We do not cover transaction management or semantic integrity. We refer the reader to for an extensive discussion on transaction models, recovery and concurrency control, and to any database textbook for details on semantic integrity. It is also important to note that an access control mechanism must rely for its proper functioning on some authentication mechanism. Such a mechanism identifies users and confirms their identities. Moreover, data may be encrypted when transmitted over a network in the case of distributed systems. Both authentication and encryption techniques are widely discussed in the current literature on computer network security and we refer the reader to for details on such topics. We will, however, discuss the use of encryption techniques in the context of secure outsourcing of data, as this is an application of cryptography which is specific to database management.

II. ASPECTS OF DATABSE SECURITY

2.1 Confidentiality, Integrity and Availability (CIA) in Database Management System

As mentioned in [1] a complete solution to data security must fulfilled the following three requirements Confidentiality, Integrity, Availability (CIA): these entire factors can gained in database using following ways:

2.1.1 Confidentiality

Means the protection of data against unauthorized disclosure can be achieved using access control mechanisms. It is already further enhanced by the use of encryption techniques applied to data when being stored on secondary storage or transmitted on a Network.

 DOI : 10.35291/2454-9150.2021.0349

2.1.2 Integrity

Means the prevention of unauthorized and improper data modification and can be achieved in combination with access control mechanisms by semantic integrity constraints.

2.1.3 Availability

Means to the prevention and recovery from hardware and software errors and from malicious data access denials making the database system inaccessible. The data that are available on the Web can be powered by the use of techniques protecting against denial-of-service attacks such as the ones based on machine learning techniques.

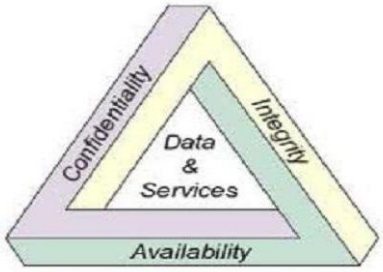

Figure 1: CIA Triad

Latest approaches to protecting databases are illustrated in [2]. All of these approaches are related to the CIA. In these approaches the author proposes that it can be implemented with the help of below listed appropriate techniques:

Authentication of Users

Within this point the author has mentioned public key encryption (PKI) for the databases that require higher levels of safety one-time passwords X.509 digital certificate smart cards can be used. PKI is very useful when contacting over irrelevant networks like the Internet and both on the internal servers.

Access control to objects and authentication of authorized applications

This point means the access control should be defined at the design state. Here main emphasis is given on the roles and based on this access is given to the user.

Administration policies and procedure

Its mean Security and safety policies with plans are required for varying requirements of data security.

Secure initial configuration

This point indicates the Policies and procedures also define auditing requirements for securing initial configuration and managing change regulation.

Auditing

This point refers to auditing the author emphasizes on maintaining logs of the changes to the database management system.

Backup and recovery strategies

This point refers to the backup and strategies, As the backup and recovery there should be three kinds of backup's cold, hot and logical. All these aspects are traditional and there are vulnerabilities in these security methods which may cause threats to the database system. Henceforth this paper gives detailed information about the vulnerabilities, threats and different security methods to avoid them.

2.2 ORIGIN OF SECURITY THREATS

Security threats can have various sources of origination such as *Internal*, *External* and *Partner*.

Internal: These are the security threats sources that exist within the organization like some company executives who have high access and privileges of the database. Internal sources enjoy certain levels of trust and privileges [8].

External: Sources outside the organization pose as external threats to the database. Hackers, cybercrime groups and other government entities are some examples of external sources of threats. No trust or privileges are invested in external sources.

Partners: These are the people outside the organization that share business relationships with them. Customers, vendors, suppliers and contractors are a few examples of partner groups of organizations that can be a source of threat for the database. Since the communication between both the parties are necessary for the functionality of business, moderate levels of trust and privileges are associated with them.

On the basis of Data Breach Investigation Report (DBIR) 2016[10], 2017[11], 2018[12] and 2019[13], where different origins of security threats such as Internal, External, Parties and Multiple parties were considered following chart Figure 1 has been derived. The conclusions made from the following graph [Figure 1] are: There was an increase in the percentage of security threats originating from within the organization from 11% in 2016 to 34% in 2019. Threats originating from external sources decreased over the period of 2015 – 2019 from 86% to 69%.

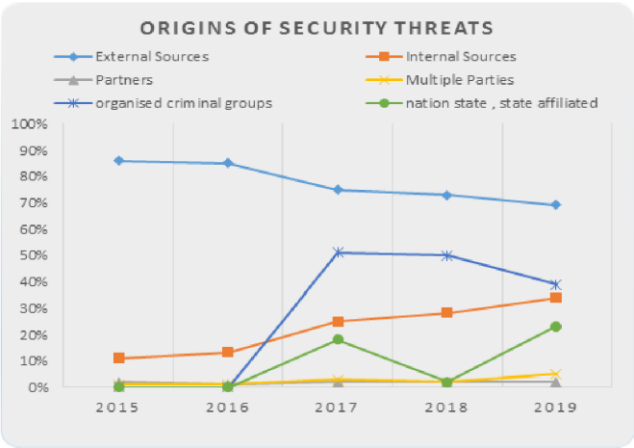

Figure 2: Origins of Security Threats Chart

 DOI : 10.35291/2454-9150.2021.0349

2.3 SECURITY THREATS

According to IBM's 2019 Data Breach Report, Conducted by the Ponemon Institute, the cost of a data breach has increased by 12% over the past 5 years and is now $3.92 million on world average[5]. The financial consequences of the attacks on databases are not only immediate but the cost impact is felt for years after the incident. There are several security threats that can lead to data breach incidents. Top 10 threats over the past decade are:

A. Excessive and Inappropriate Privilege abuse:

Database management systems and their corresponding data structures are complicated which makes administrators granting excessive rights to the users so as to prevent any application failure due to lack of rights. When users are given privileges more than what is required for their job functionality, these privileges can be used maliciously. For example course coordinator for any university is given the right to upload marks of every student. This privilege can be misused to change the marks of any student or any subject. This misuse is the result of granting generic access rights to a certain group of users even when it exceeds their specific job requirements.

B. Legitimate Privilege Abuse:

This happens when a user is given only those privileges which are required by their job functionalities and these legitimate privileges are used for unauthorized purposes. User groups like Database System Administrator (DBA) and Developers have access to the entire database due to their job requirements. If a DBA tries to access the database data directly instead of the application interface, all the application permissions and security mechanisms would be surpassed making the way for privilege abuse clear [4].

C. Privilege Elevation:

Users with low-level privileges may use the vulnerabilities in the database to convert their access rights to high-level privilege. This can lead to the availability of critical information to unauthorized users.

D. Platform Vulnerabilities:

Any vulnerability in the underlying Operating System like Windows 2000, Windows XP, and Linux etc. can lead to privilege escalation, denial of service, data corruption and unauthorized access security threats. For example, A potential security vulnerability in Intel WIFI Drivers and Intel PROSet/Wireless WiFi Software extension DLL with severity rating as High was patched in November 2019 platform update. Memory corruption issues in Intel(R) WIFI Drivers before version 21.40 may allow a privileged user to enable escalation of privilege, denial of service, and information disclosure via local access [8].

E. Weak Audit Trails:

Automated recording of any database transactions involving sensitive data should be a part of every database deployment. Failure to monitor transactions and collect audit details of database activities poses risk to the organization on many levels [2]. Many organizations rely on native audit tools provided by the database but the native audit tools do not record sufficient contextual information necessary to ensure security, detect attacks and provide incident forensics. Another reason native audit tools are not reliable is that users with administrative rights either legitimate or escalated can turn off database auditing to hide malicious activities [12]. Therefore database responsibilities and audit capabilities should be separate from both database server platform and database administrator to ensure strong separation of duties policy [11].

F. Denial of Service (DoS):

This is a general category attack in which the legitimate users like employees, members or account holders are deprived of database services or resources which they require. This is done by shutting down the machine or the network making it inaccessible for its intended users. This can be done in two ways either by flooding the destination with excess traffic or by sending them information that results in a crash [12]. Even though Dos doesn't directly result in data theft, loss or corruption they can cost a significant amount of time and money to handle.

G. Unsecured Storage Media:

Backup storage media is often less secured compared to the other database assets. This resulted in several high profile data breaches involving theft or incidental exposure of database backup tapes and hard disks. Many regulations have made it mandatory to protect backup copies of sensitive data. One of the possible solutions to this is encryption of all the backup data [2] [7].

H. SQL Injection Attack (SQLIA) [14]:

This is an attack which gives a potential attacker complete control over your database through the insertion of unauthorized or malicious SQL code in the database query. There can be multiple types of SQLIA:

- Injection by passing malicious strings in for user input in web forms.
- Through cookies; modifying cookie fields so that they contain attack strings.
- Through server variables where headers are modified to contain attack strings.
- Second Order SQLI; where the attack is designed to run at a later stage and not immediately [12].

 DOI : 10.35291/2454-9150.2021.0349

I. Database Communications Protocol Vulnerabilities:

Proprietary protocols are created by database vendors for the communication between database client and servers through data and commands. Vulnerabilities in these protocols can lead to various fraudulent activities like unauthorized data access, denial of service, data corruption. In addition to these threats, what makes them worse is the fact that no record of these fraud activities will be there in the native audit trail since these protocols are covered by database native audits. Attacks based on protocols can be prevented by using protocol validation which audits and protects against attacks by comparing live protocol to expected protocol structure [15].

J. Weak authentication:

If the authentication procedure of any database is weak, an attacker can acquire the identity of a legitimate database user by using any of the following techniques: Brute force, social engineering and direct credential theft. Two step authentication procedures are a must for database security.

III. SECURITY TECHNIQUES

Database security deals with all issues to guarantee the confidentiality, integrity and availability of data stored within database systems. Access control, as a major aspect of database security, aims at ensuring confidentiality (the protection from unauthorized disclosure) and up to a degree also integrity2 (the protection from unauthorized modification) of data. *Access rights* are used to allow or forbid *subjects* (the active entities of a system, i.e. processes running on behalf of users) to execute a particular *action* (or operation) on a *protection object* (the assets to be protected from unauthorized accesses, e.g. relations, types or classes, tuples or objects, attributes, etc.). *Access control* comprises all system mechanisms that are required to check whether a request, issued by a particular subject, is allowed or not, and all mechanisms that are required to enforce the corresponding decision. It is based on a chosen policy (a set of rules as authoritative regulations or directions determining what should be protected using which principles). The same mechanisms can be used to enforce different policies, and the same policy can be enforced by different mechanisms [16]. Database is the backbone of any organization. Therefore it is important for the organization to implement any security solution. The security technique must ensure the safety of not only the data inside the system but also the database hardware, software and human resources. Database security techniques can be broadly classified into four categories, namely: Access Control, Techniques against SQLIA, Data Encryption and Data Scrambling [5].

A. Access Control (Mechanism):

Data confidentiality can be ensured by using Access Control Mechanism. Most users are assigned or have authorized privileges to specific database resources and every time a user tries to access any data from the database, the access control mechanism will compare the required privileges to assigned privileges. Through this technique users can only access that data object for which they are adequately authorized. For example, for a university database teachers and students can be two categories of users with different access privileges. A student can only read grades and courses offered and the teacher can update grades of students. A student can't make changes in the grades obtained whereas a teacher can't make changes in the courses offered.

Access Control in databases can be maintained in different ways such as:

Discretionary Access Control (DAC):

DAC grants or restricts the access to a data object based on an access policy created by the owner of the data object. It is discretionary because the owner can transfer the authenticated objects and information access to other users. Object's owner group has complete control over the access of the object [7]. *Discretionary access control* (DAC) is based on subject and protection object identities. Access rights are explicitly granted. An access right can be represented as a tuple (only the first three components are mandatory) (grantee, protection object, action, kind of right, grantor, grant option) indicating that the *grantee*3 is allowed (*permission*) or forbidden (*prohibition*) to execute the *action* on the *protection object*. The *grantor* is the user who has granted that access right. If *grant option* is set and the access right is permission, the grantee is allowed to pass on that access right to other subjects. This kind of access control is often combined with an *ownership paradigm*, where each protection object has an owner (a user) who is responsible for granting and revoking access rights concerning this object. Since this happens at her discretion, the policy is called "discretionary". However, the same discretionary mechanisms can be combined with an *administration paradigm*, where only the security administrator is allowed to grant and revoke access rights. In this case, the policy is mandatory.

Depending on the access rights that can be granted, several kinds of discretionary systems can be distinguished:

– positive systems: only permissions can be granted

– negative systems: only prohibitions can be granted

– mixed systems: permissions as well as prohibitions can be granted (in this case, a policy to solve conflicts is required)

Usually, the authorization base (the set of explicitly granted access rights) is not complete, i.e. there are some requests where neither a permission nor a prohibition applies. Hence, a closure assumption is necessary:

– closed world assumption: a request is forbidden unless an appropriate permission exists or can be inferred

 DOI : 10.35291/2454-9150.2021.0349

– open world assumption: a request is allowed unless an appropriate prohibition exists or can be inferred

Meaningful combinations are:

– positive/mixed system with the closed world assumption

– negative/mixed system with the open world assumption

Mandatory Access Control (MAC):

MAC allows a user to access a data object only when the authority level of the user matches the security level of the needed data item. Access control in MAC is based on the following two principles:

No read up: Users can only read a data object when the access class of user is higher than the access class of that object.

No write down: Users can write a data object only if the access class of the object is higher than that of the user.

Mandatory access control (MAC) policies were developed to enforce organization-wide security policies automatically. The most popular one is *multilevel security* (MLS) based on the model of Bell and LaPadula /BeLa 75/. Multilevel security does not rely on explicitly granted access rights; instead access decisions depend on so-called *security classes* (or labels) that are associated with subjects (clearance levels) and protection objects (confidentiality levels). Security classes are organized as partial ordered sets. The main property of those models is that data never flows from higher to lower security classes (unless the initiating subject is "trustworthy"). Thus, this model is inherently unidirectional with respect to data flows. The two main rules for MLS systems are the following:

– simple property: A subject is allowed to read a data item if its clearance level dominates the confidentiality level of the data item.

– *-property: A subject is permitted to write into a data item if its clearance level is dominated by the confidentiality level of the data item.

In recent years, "nonstandard" policies have been proposed, which have some practical relevance outside the military world (for which MAC was originally conceived). The most important ones are the Clark/Wilson model /ClWi 87/ and the Brewer/Nash model /BrNa 89/ (Chinese Walls).

Content Based Access Control:

In this model, the access control decisions are based on the contents of data objects. For example, the Employee table has salary details of all the employees of the organization. So only those employees of the accounts department who are working on the employee salary part should be able to access that data. This approach is implemented using views. Users are presented with the temporary view of the table with only those data they are authorized to access and not the complete table itself.

Fine Grained Access Control (FGAC): General access control for databases is coarse grained, i.e. it grants access to all the rows of the table or none at all. In contrast to this is fine grained access control that implements access control at the tuple level of the database. It enforces access control at the granular level. In this scheme each data object is given its own access control policy. This is implemented using specialization of views [12]. Oracle Virtual Private Database (VPD) is one such database implementing FGAC.

B. Preventing SQLIA - Fighting Techniques

SQL injection attack gives complete control of our database to the attacker and thus it is one of the most dangerous security threats. The detection approaches for SQLIA can be categorized as:

Pre-Generated: Implemented during the testing phase of web application of database.

Post-Generated: Used when the dynamic SQL generated by a web application is analysed.

Post Generated Approaches:

Positive tainting and Syntax Aware Evaluation: In this technique valid input strings are provided to the system initially to detect SQLIA. Positive tainting here means identifying, marking and tracking of trusted SQL queries and differentiating malicious queries from the legitimate ones using taint marking. Syntax aware evaluation allows us to actually use taint marking to identify trusted queries for the database. It allows the use of untrusted input data in a SQL query as long as it does not lead to SQLIA. Syntax evaluation of a query string is performed before the string is sent to the database for execution [8].

Context Sensitive String Evaluation: It works on simple classification of data, User based data is considered as unreliable and data given by application is considered as reliable. Un-reliable data is then sent for syntax evaluation where string and numeric constants are differentiated from each other and all unsafe characters are removed from the strings identified [1].

Parse Tree evaluation based on grammar: This approach defines a predefined grammar which is used to parse all the queries generated from users. A parse tree is a data structure that represents a parsed statement. Parsing a statement requires the grammar of the language it was written in [9]. When a malicious user injects a SQL query into the database, the parse tree of the legitimate query and injected query will not match and this is how the SQLIA would be detected using parse tree.

Pre Generated Approaches:

Pixy: The first open source tool to statically detect cross-site scripting (XSS) [10]. It follows a data flow analysis approach to create information and statistics for each program point. For example, the constant analysis computes

 DOI : 10.35291/2454-9150.2021.0349

for all program points, the values that variables can hold. After data flow analysis parse trees are created and a taint analysis tool is applied to find out all the points in the database which are vulnerable to attacks and malicious data entry.

Program Query Language: It is a language having pre-defined grammar to express patterns of events on data objects. It provided a static and dynamic program analysis to find the sequence of programs as it runs. These are recorded in data logs which provide support for detecting malicious queries.

C. Data Encryption

The technique used to secure any kind of data or information can also be used to protect the data stored in the database. Data encryption is a technique of transforming a plain text to intelligible form. This resulting information is known as encrypted data which can be converted back to its original form using encryption key. This technique can be used to secure the database by saving encrypted data in the database instead of plain text and converting the encrypted data to its original form when it is required for processing purposes. There are two different approaches to data encryption technique:

Symmetric Encryption: One common key is used for both encryption of data and decryption of data as well.

Asymmetric Encryption: Two keys are used, one key for encryption and the other for decryption.

There are three aspects to be considered while encrypting data for database security:

Encryption Algorithm: First aspect of encrypting a database is to identify the algorithm to be used. Various data encryption algorithms supported by DBMS are: AES, DES, Triple DES, RC2, RC4 and DESX. Second aspect is to identify the encryption level of database from the following:

File system Encryption – Encrypting the physical disk where the data is stored.

DBMS level Encryption – Encrypting tables, rows or fields.

Application level Encryption – A middleware is used to translate user query into new queries to work on encrypted data.

Client Side Encryption – It is used when the database is being used as a service and the organization outsources the complete database and data privacy is a major concern [14].

Place of encryption: Second aspect is to identify different levels where Data encryption can be done. The encryption of data can be done either inside the database as its part or outside the database. When the encryption is carried out inside the database then the impact on the database application environment is less. One problem area of this approach is that the encryption keys are stored along with the database itself which can pose a security concern. Another way to encrypt a database is to perform it on separate encryption servers. The liability of encryption and decryption is now not on the database and done on independent servers thus maintaining the database performance [14]. In this approach Encryption keys and Data is stored separately and not on the same database.

Granularity of Encryption: Third aspect is to identify the encryption level of the data to be encrypted. Different granularity levels of encryption can be Cell, Column, Tablespace and File with cell-level being the most specific level and File level being the most general level of encryption for the database. More granular level of encryption can result in performance degradation for the database. Column-level is the most commonly used encryption level since it includes less processing than that required at cell level and still provides encryption at a specific level of database.

D. Data Scrambling

It is a process of deliberately changing or removing the data saved in the database so as to make sensitive data safer for wider visibility. It is also known as Data masking, Data sanitization and Data obfuscation. It is used in the scenario where a user has access to a certain data but still the data needs to be protected from the user. For example, testers and third party developers involved in working on the data in the database [1]. Even though they require working on the data, the actual values of data can be changed to hide the sensitive information. Basically, data is changed but the changed data resembles the actual data. Relationships between the columns in the original data would exist in the scrambled data as well. This way the actual sensitive information would be hidden from third party developers and they can still work with the data

IV. VARIOUS DATABASE MANAGEMENT SYSTEM MODELS WITH THEIR CHARACTERIZATION, SECURITY ISSUES AND EXISTING APPROACHES

4.1 Object-Oriented DBMS

Characterization An *object-oriented DBMS* is a database management system implementing an object oriented data model at the logical level. *Object-orientation* suggests that the universe of discourse is modeled as a collection of cooperating, interrelated and distinguishable units, called objects, which are instances of *types* (or *classes*) that are organized in type hierarchies supporting inheritance /Atki 89/. An *object* has an identity independent of its value, and is an abstraction unit characterized by a *state* (it's – usually structured - value) and *behavior* (the requests an object is able to answer). Usually, objects are *encapsulated*, i.e. the state can only be observed and modified via a well-defined interface. This way, *information hiding* is ensured; other objects only see the behavior of an object, whereas the state is hidden (although there are systems supporting weaker

 DOI : 10.35291/2454-9150.2021.0349

forms of encapsulation). Further, it is possible to define *associations* (interrelations) between objects, in particular so-called *complex object relationships*. Such a relationship is not only a pointer that can be used for "pointer chasing", but also influences the semantics of operations. Complex object relationships allow for treating sets of objects, being declared as one complex object, as a single unit, i.e. operators applied to an object are propagated to all component objects, too. Finally, object-oriented DBMS (as any DBMS) provide for schema information about the structure of objects, and support collections of similar objects, i.e. they provide for a type (or class) concept. Types, including the operations, can also be defined by users. *Super-/subtype relationships* between these types are used to inherit properties from super- to subtypes, supporting the reuse of structure and code. *Polymorphism*, i.e. the ability to send the same request to objects of different types (and getting a semantically meaningful answer), is implemented by overriding, overloading and late binding. Optionally, object-oriented DBMS support features like object versions and query languages [17].

Security Issues

Access control for database systems obviously has to reflect the data units handled by the respective data model. Hence, known database access control features at least have to be adjusted to the notion of "object", i.e. *the unit of access control* - the protection object - *has to coincide with (complex) objects* of the data model. It has turned out that this is not as easy as might seem at first glance, particularly due to object structures and their variety of semantics. Furthermore, if in DAC systems an ownership approach is applied, components of a single complex object may have different owners. Shared components cause additional problems to ensure a consistent authorization state.

In MLS systems, the complexity of the arising problems depends on whether *single-level or multi-level objects* are supported. In both cases, *restrictions have to be identified that must be imposed upon complex object relationships*5 (including the propagation of operators, in particular the delete operator) according to the MLS rules *without introducing covert channels*. Since access control mechanisms have to comply with the logical data model of the DBMS, *encapsulation requires that access control is based on the actions of the data model, i.e. on methods* (and not on lower level read or write operations). Depending on how strict the encapsulation principle is enforced within a concrete DBMS, basic access operations (like reading or writing an attribute) have to be considered, too. Encapsulation and a method-based authorization imply the lack of generic access rights in DAC systems. In a relational DBMS, only select-, insert-, delete- and update-rights can be specified for "basic accesses" to relations. It is thus easy to specify generic rights for sets of protection objects. In object-oriented systems, however, it is simply by chance (or due to intended polymorphism) that a right to execute method XYZ on type A makes also sense for type B (with the exception of generic operators like copying or deleting an object). Access rights are therefore type-specific, and additional concepts are required to define more abstract access rights (e.g. for whole domains of protection objects). Moreover, it is conceivable that there are types providing for hundreds of methods. In this case, authorization becomes a very cumbersome issue if rights can only be granted for individual methods. In order to support security administration, additional abstraction concepts are required. The *access control concept must fit the inheritance paradigm*. It would not be meaningful if the data model supports inheritance of structure and code, but the access control mechanisms require explicit authorizations for inherited methods (or do even prevent inheritance). This problem appears in both areas, DAC and MLS. In case of discretionary mechanisms ownership restrictions may contradict inheritance (if two types in a super-/subtype-relationship have different owners). In MLS systems, constraints have to be imposed upon the classification of sub- and super types, as well as upon their instances. This problem is very complicated, even more since there is no inheritance paradigm. Several different kinds of inheritance can be distinguished along a variety of dimensions:

– single vs. multiple inheritance

– strict vs. default inheritance /MaMM 91/

– inclusion, specialization, substitution or constraint inheritance /Atki 89/

– selective inheritance /MaMM 91/

– type vs. object inheritance6

Existing Approaches

There already are too many papers on access control for object-oriented DBMS to give an overview here. Consequently, we only provide for references in this section. A lot of work has been done to develop - in parts very elaborate - DAC concepts for object oriented database systems (/Ahad 92, BeOS 94, Bert 92a, Brüg 92, FaSp 91, FeGS 89, FeWF94, GaGF 93, GuSF 91, HuDT 94, JoDi 93b, LGSF 90, NyOs 92, NyOs 93, PfHD 88, RaWK 88, RBKW 91, Spoo 88, TiDH 92b, TiDH 92a, HuDT 93/). Some of these concepts have been implemented in ITASCA (the commercial version of the ORION prototypes) and HP's Open OODBMS (based on the IRIS concepts). However, most of the systems which are Now on the market (like ObjectStore or ONTOS) do not provide for any access control mechanism at all beyond the operating system's mechanisms, which are (at best!) inappropriate for database systems. Some issues that are in our opinion still not completely solved concerning DAC (not even in /JoDi 93b/), are the incorporation of weaker forms of

 DOI : 10.35291/2454-9150.2021.0349

encapsulation into the security concepts, and the reconciliation of ownership on the one hand, and complex objects or inheritance on the other. The situation is similar in the MLS area. Examples for existing approaches include /JaKo90, JaKS 92, KeTT 89, KeTs 90, Lunt 90, Lunt 92, MiLu 92, Morg 91, RWHT 94, SaTJ 92, ThSa 93, Thur 89, Thur 92/. It seems that the conceptual security aspects are somewhat easier here compared to DAC concepts (as usual for level-based systems). However, appropriate architectures and protocols (e.g. for object deletion) to prevent covert channels are still under consideration /BeMJ 94/. A first attempt to incorporate MLS into a commercial object-oriented DBMS is presented in /SaWa 94/. A good overview of both research directions is given in /BeJS 93/ (although focusing on the message-filter approach /JaKo 90/ and the ORION concept /RBKW 91/ and its extensions /Bert 92a/). A preliminary attempt to adapt the Clark/Wilson model for object-oriented DBMS is presented in /Hern 94/. Additional ideas on how dedicated methods can be used to support access control can be found in /Ahad 92/.

4.2 Active DBMS

Characterization Active mechanisms are another promising concept that is now being integrated into commercial systems (e.g. Sybase and Oracle 7). Whereas (classical) passive DBMS only react to user requests, active DBMS include a monitoring facility, allowing an automatic reaction of the system in case of well-defined situations (see Fig. 3). *Active DBMS* allow – beyond providing all regular DBMS-features – the recognition of user-defined situations in the database, and the execution of user-defined reactions when such a situation occurs. The most popular specification paradigm for active DBMS are so-called event/condition/action rules (*ECA-rules*) /Chak 89/

ON <event> IF <condition> DO <action>

which can be defined in addition to the regular database schema. When an event is detected by the system, the condition is checked on the database and if it holds, the specified action is executed. The combination of an event and a condition specifies the *situation* when the rule has to be fired. Many details have to be considered in such systems, including, e.g., the kinds of events, conditions and actions that are supported. In particular, the power of active DBMS is highly dependent on the event model, which may include the occurrence of database operations (accesses to data, or transaction events), but also externally raised events, time events, and various combinations thereof. In many research prototypes (e.g. HiPAC /Chak 89/, Ode /GeJS 92/, SA MOS /GaDi 94/ or Sentinel /CKAK 94/) so-called complex events can be specified, which are iteratively built from basic events, using predefined constructors like sequence, conjunction, disjunction, closure or even the negation of events ("non-occurrences" of situations)[18].

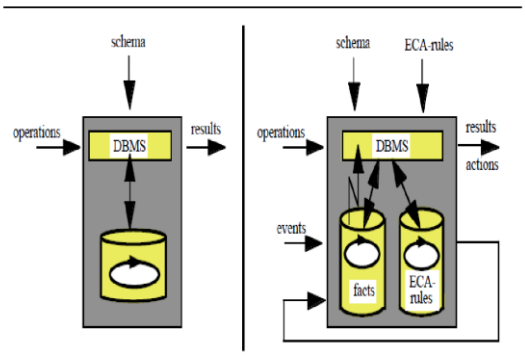

Figure 3: Passive vs. Active DBMS

Furthermore, rule processing has to be incorporated into transaction processing. Nested transactions /Moss 81/ (which are now also available in some commercial database systems, e.g. ObjectStore) have proven to be a viable concept for integrating both technologies into a coherent system. Usually, it is possible to specify coupling modes to declare when a fired rule should be executed /Chak 89/:

– **immediate mode:** The triggering transaction is suspended until the fired rule has been finished (including rules which are transitively fired). Each rule is executed within a sub-transaction of its own.

– **deferred mode:** The triggering transaction is not suspended, but is completely executed until it reaches its commit point. All rules which are fired by this transaction are queued and not executed before the triggering transaction is ready to commit. At this "pre commit point", all fired rules are executed (in sub transactions of their own) and if they have successfully been finished, the triggering transaction is committed.

– **decoupled mode:** The fired rules should neither interrupt the triggering transaction nor be delayed until this transaction reaches its commit point. In this case, rules are processed in top-level transactions of their own, which are concurrently scheduled as any other transaction within the system.

Security Issues

Obviously, active rules and their execution have to be subject to security, too. Two aspects have to be taken into consideration. On the one hand, *rules* as elements of the database *are protection objects*. It thus has to be determined who is permitted to define, modify, enable/disable or delete a rule, as well as to query the set of defined rules. This is a rather simple question, because rules can be represented as "ordinary" database objects. Hence, the usual mechanisms of a DBMS can be used to restrict which user can access a rule. On the other hand, if a rule is fired, it issues several accesses (during the evaluation of the condition and the execution of the action) which have to be checked by the

 DOI : 10.35291/2454-9150.2021.0349

access control system. The problem arises *which access rights should be used to evaluate the accesses issued by a rule*. Three solutions are possible (and combinations thereof):

1. A rule inherits the access rights/clearance level from the subject that has defined the rule.

2. The rule inherits the access rights/clearance level from the subject that has fired the rule.

3. A rule is a subject of its own, and access rights can thus be granted/revoked to/from the rule (or a clearance level can be associated with a rule) as for any other subject within the system.

The second approach is rather odd considering that rules are usually used for change propagation (to enforce integrity constraints), for monitoring purposes, etc. Usually, the subject, having fired the rule, will not even notice that it has done so. Thus, it will usually not have the access rights/clearance level required to execute the rule successfully. The first solution so far is the prevailing one. However, it also has several disadvantages, in particular in DAC environments. If rules are used for change propagation, it is necessary to give the user who has defined the rule the rights to access the affected objects arbitrarily, although only a very restricted (and well-defined) access via the rule would be required. Therefore, we firmly recommend the third solution, which is in our opinion not only the most powerful but also the most natural one. All three schemes require considerable extensions of existing access control concepts. Moreover, the tight connection of rule and transaction processing has to be kept in mind. Supposing, e.g., the coupling mode "immediate" is applied, and a rule does not have the rights of the triggering subject. In this case, the transaction needs to be executed in a different security context (in MLS systems, it has to change its clearance level, and in DAC systems, it needs to get a new access control identifier). Even more important, it has to be ensured that the original context is restored if the rule processing is finished, and the transaction resumes. It obviously has to be investigated *which coupling modes are tolerable in MLS systems* if a rule has to be processed at a higher level than the triggering transaction. In case of the coupling mode "immediate", e.g., it would be necessary to decrease the level of the process executing the transaction when the rule processing is finished, which contradicts the usual MLS philosophy.

Existing Approaches

In the context of DAC systems, very little has been done so far. The approach of Oracle 7 is quite simple. System privileges are required to define rules (called triggers in case of Oracle: "create trigger" to define rules for own relations, "create any trigger" as a wildcard for defining rules for any relation within the system). The user who has defined a rule becomes its owner. Only the owner (or a user having the "alter any trigger"/"drop any trigger" system privilege) can modify/delete a rule. It is not possible to grant access rights concerning rules to other users. An exception concerns the enabling/disabling of rules, because this is also possible for users having the "alter table" privilege for the relation the rule is attached to. Rules are always executed using the access rights which were directly granted to the owner. The Starburst mechanisms /WiCL 91/ are a bit more elaborated. There is an explicit "control"-right for relations which allows for authorizations concerning this relation (which usually is only in the possession of the owner). In order to create a rule, "attach"- and "read"-rights are required for the relation the rule should be attached to. Having a "control"-right for the rule, or an "attach"- as well as a "control"-right for the relation allows for deleting a rule. Modifying a rule requires an explicit "alter"-right for the rule itself (but not for the relation). Analogously, explicit "activate/deactivate"-rights are required to enable/disable a rule. Similar to Oracle, a rule always has the access rights of its owner. Much more has been done in the MLS area. In /SmWi 92a/ and /Smit 92/, a successful combination of the active data model (of Starburst) with MLS has been found, addressing questions like which events are visible to a rule. These concepts have been extended in /Smit 94/ to consider the integration of multi-level secure rules into a transaction paradigm.

4.3 Federated DBMS

Characterization Another very active area of database research is the integration of existing "information islands" into coherent systems. Since legacy applications have to be preserved (in most cases, it is economically infeasible to re-code them for using a new system), very particular requirements have to be met. Database federations are considered as being a promising approach for solving the arising problems. A *federated DBMS* (FDBMS) provides for the interoperation of (probably heterogeneous) component DBMS (CDBMS) under one "common roof", by preserving local autonomy. The architecture of an FDBMS is shown in Figure 2. Note that the FDBMS has its own data model (possibly different from any data model used by the involved CDBMS). It may also have its own database. *Autonomy* means that CDBMS retain a separate and independent control over their data, even if they join a federation. In a sense, the FDBMS is an "ordinary" application from the local systems' point of view.

 DOI : 10.35291/2454-9150.2021.0349

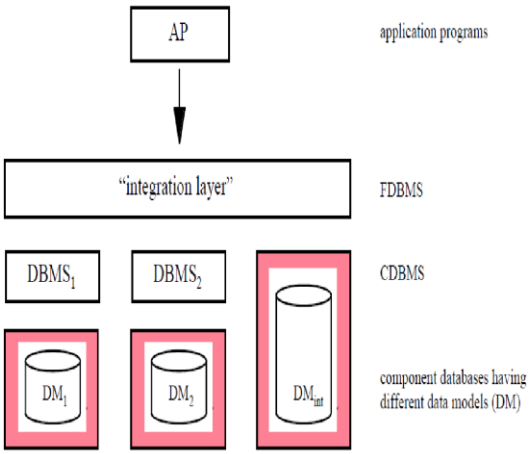

Figure 4: Architecture of a database federation

The FDBMS provides a uniform interface to global users, ensures location transparency, hides heterogeneity, and offers the usual DBMS functionality (like transaction processing, query languages, etc.). In particular, it has to care for the mapping between the global and the local data models. There are several ways to achieve this. One possibility is to apply the 5-level schema architecture /ShLa 90/, where export schemas of component systems are translated into generic schemas according to the global data model. Afterwards, the generic schemas are integrated into the global schema of the FDBMS. A simpler, and currently more practical approach, if heterogeneous CDBMS have to be integrated, is based on object-oriented technologies. Global types are presented to global users, and the integration of component systems is buried ("hard-wired") within the code of global methods /HäDi 92/ (operational integration). In either case, integration is a tedious job. FDBMS may also help to allow the introduction of, e.g., an object-oriented DBMS into an environment where a more traditional DBMS already is in use, and where both kinds of systems have to interoperate.

Security Issues

Since database federations are special cases of interoperable systems connected by a network, *network security issues have to be taken into consideration* (for tightly coupled as well as for loosely coupled federations /ShLa 90/). Database federations need strong and reliable mechanisms to identify and authenticate remote users (including mutual authentication and continuous authentication), as well as to encrypt sensitive data which are transmitted between the FDBMS and CDBMS. This does not mean that the required mechanisms have to be implemented by the FDBMS itself. It is also possible to apply security services offered by the network or by specialized systems like Kerberos. A typical aspect, however, is that access rights may depend on the kind of connection of the user requesting the data (local, remote, dial-up, etc.). Most of the additional security problems only arise for tightly coupled systems, providing for a global integration layer and a global authority to enforce a global security policy. Global access control is enforced by an independent reference monitor that is integrated into the FDBMS and has to cooperate with local reference monitors (two-level access control, see Fig.4). An access control system at the global layer is essential due to several reasons:

– *Integrated systems aggravate security problems*. This is quite obvious in case of personal data. The more data are available about humans, the better they have to be protected against misuse.

– Several *new protection objects "emerge" at the global layer*, e.g. global relationships between data of different component systems (in particular aggregated data). Their proper protection can only be ensured by the FDBMS, since no component system alone is aware of these protection objects.

– The FDBMS usually has a storage area of its own. At least data which are stored there have to be protected by global mechanisms.

The problems we are faced with mainly stem from the heterogeneity and autonomy of CDBMS. The *FDBMS has to cope with heterogeneous local security policies and mechanisms* (besides heterogeneity with respect to data models, semantic heterogeneity, different query languages, different transaction mechanisms, etc.). Consequently, the FDBMS needs at least to understand the local concepts in order to cooperate with the local systems in a meaningful way. In case of MAC systems, the following problems can arise:

– Local systems use *different security classes* (semantic differences).

– Local systems may have defined *different partial orderings* between their security classes. Some systems may even only support total orderings.

– *Different classification granules* may be supported (single-level vs. multi-level protection objects; e.g. tuple classification vs. attribute classification).

– Different security policies may be applied (support for polyinstantiation with different operational semantics (/LuHs 90, JaSa 90, SmWi 92b/), support for trusted subjects or dynamic changes of a process' clearance level, etc.).

DAC systems are mainly faced with the following aspects of heterogeneity:

– CDBMS can support *different kinds of access rights* (permissions, prohibitions or even both, with *different conflict resolution policies* in the latter case), and can apply *different closure assumptions* (open vs. closed world assumption).

 DOI : 10.35291/2454-9150.2021.0349

– *Different authorisation units* may be supported (users, groups, nested groups, roles with different role paradigms, role hierarchies, subject domains, etc.).

– The *protection object granules are usually different* (databases, protection object domains, types/classes/relations or objects/tuples, etc.), and *different actions* are supported to access these objects (relational operators, methods, etc.).

– Some systems may support *value-dependent access rights* (either via a view concept /GrWa76/ or by applying another technique like query modification /Ston 75/).

– The local systems can enforce *different authorization paradigms* (centralized vs. decentralized authorization, ownership vs. administration paradigm), and in case that decentralized authorization is supported, it may be based on different mechanisms (grant options vs. explicit grant permissions).

The situation becomes even more complicated if multiple CDBMS have to be integrated where some only support DAC, whereas others only support MLS. Hence, access control for database federations needs a formal foundation, i.e. a *canonical security model subsuming DAC and MLS*. *Information flow control* is required if the FDBMS is allowed to store data of component systems in its own data storage areas or in other component database systems. Probably, this problem can only be solved using classification-based techniques. The opportunities database federations offer for solving security problems are not that obvious. If access control is more important than performance, it is possible to integrate a CDBMS offering either no or only very restricted security mechanisms into a secure database federation. This way, the FDBMS acts as a secure interface to the data kept by an insecure DBMS. Furthermore, tightly coupled federations can be used to enforce system-wide security policies. Note that both examples require low authorization autonomy for CDBMS to prevent users from circumventing the FDBMS.

Existing Approaches

MLS for database federations has so far not been considered in much detail. The results of a panel discussion on security issues for database federations were published in /MLTS 92/ (also including some notes on DAC). An approach for tightly coupled federations has been presented in /IdQG 94/ and /IdGC94/. During the integration process, data from component systems are reclassified - if needed to get a consistent global schema. Since this reclassification tends to over classification, the authors do not only consider "static security" (the classification of subjects and protection objects is static), but also describe some ideas to support "dynamic security". The latter means that a group of subjects is able to get a higher clearance level (resulting in the ability to access more sensitive data) under predefined circumstances. This approach is based on Shamir's scheme of sharing a secret /Sham 79/ (or so-called "shadow keys" according to /Denn 82/). A predefined number of users has to combine their shares in order to get temporarily a higher clearance. A scheme for loosely coupled database federations has been presented in /Oliv 94/. This approach is very interesting, because it introduces the notion of trust between sites being involved in a database federation. CDBMS can selectively decide which other sites are allowed to access their data. The first attempt to develop a canonical security model for database federations comprising MLS as well as DAC has been presented in /Pern 92/. Local systems can apply a discretionary policy, a mandatory policy or a combination of both. The global canonical model comprises the functionality of local systems and allows for defining additional access restrictions at the global level. Concerning DAC for federations, most approaches that have so far been developed focus on traditional database technology (mostly relational technology) and are based on view concepts /ShLa 90, Pern 92/. The first DAC concept for database federations has been implemented in Mermaid /TeLW 87/, which is a front-end system to integrate multiple homogeneous relational DBMSs by ensuring distribution transparency. Authorisation autonomy has been preserved. Access rights are individually granted at the global level, but do not imply any right for involved CDBMSs (i.e. the corresponding local rights have to be granted explicitly). Access control is based on access control lists which are associated with certain schemas. A user having an entry within such an access control list is allowed to carry out the corresponding relational operator for any relation that belongs to this schema. Local access validations are carried out independently. Mermaid provides for a two-level access control and supports full authorization autonomy. In /WaSp 87/ an access control mechanism for heterogeneous federations (supporting relational and network CDBMS) has been described, which is based on views and an ownership paradigm. Authorization autonomy is achieved by providing only snapshots of local data to global users. Thus, global write accesses are not supported. Recently, a discretionary scheme for tightly coupled federations has been presented /JoDi 94/, supporting the integration of heterogeneous CDBMS, different degrees of local autonomy (within the same federation), and decentralized authorization based on a pessimistic protocol. Since federations are special cases of interoperable systems, the efforts of the OMG (and related committees like the ODMG) have to be taken into consideration. For the OMG CORBA /OMG 91/, some security issues have been considered /OMG 93/, which have so far concentrated on network security issues like authentication and encryption services. However, it is also planned to define a security

 DOI : 10.35291/2454-9150.2021.0349

service for CORBA environments, probably including access control.

V. ABOUT THE VULNERABILITIES IN DATABASE MANAGEMENT SYSTEM (VDBMS)

Based on our survey conducted the vulnerabilities in the database are defined as: poor architecture, misconfigurations, and vendor bugs incorrect usage [2].

5.1 Vendor bugs refer to buffer overflows and other programming errors that result in users executing the commands they are allowed to execute. Furthermore Downloading and applying patches usually fix vendor bugs and viruses.

5.2 Poor architecture refers to the result of inadequate factoring security into the design of how an application works there. These vulnerabilities are typically the hardest to fix because they require a major rework by the vendor. We can give an example of poor architecture; it would be when a vendor utilizes a weak form of inscription.

5.3 Misconfigurations are caused by not accurately locking down databases. Mostly the configuration options of databases can be set in a way that compromises security and safety for that database. Some of these parameters are concluded insecurely by default. But mostly it is not a problem unless you unsuspectingly change the configuration and setting. An example of this in Oracle is the REMOTE_OS_AUTHENT parameter. When you set REMOTE_OS_AUTHENT to true you are allowing unauthenticated users to connect to your database, so that he can do his task correctly.

5.4 Incorrect usage means building applications utilizing developer tools in ways that can be used to break into a database. SQL injection is an example of incorrect usage for developers.

The authors Marco Vieira and Henrique Madeira [3] have defined that the vulnerabilities in DBMS are an internal factor related to the set of security mechanisms available or not available in the database, the correct configuration of those mechanisms (it is a responsibility of the DBA), and the hidden flaws on the system configuration. He has described that security in database can be violated due to points as given below:

5.5 Irresponsible DBA: Refers to deactivation of the necessary security mechanisms such as user privileges, authentication, auditing, data encryption which allows intruders to find a way to get access to the data into the database.

5.6 Incorrect configuration: Permits unauthorized users or hackers to access the data in our system.

5.7 Hidden flaws in the database: May allow hackers to connect to the database server by exploring those faults.

5.8 Unauthorized users: Means these users "still" the credentials of authorized users in order to access the database server for searching the data.

5.9 Misused Privileges: Refers to authorized users taking advantage of their privileges to maliciously access or destroy our data in a database. Vulnerabilities are also defined by Hassan A.

Afyouni [4] in the following manners:

5.10 Configuration and Installation: Using a default installation and configuration that is known publicly. For example failure to change default password or default privileges or permissions.

5.11 User Mistakes: Sometimes Carelessness in implementing procedures failures to follow directly, or accidental errors with some faults. For example users lack a bad authentication process, technical information or implementation, untested disaster recovery plan in a database.

5.12 Software: Refers to vulnerabilities found in commercial software for all types of programs such as all applications, operating systems, database management systems and network systems with other different programs.

5.13 Design and Implementation: Inaccurate software analysis and design as well as coding problems and faults may lead to vulnerabilities in a database.

VI. ABOUT THE THREATS IN DATABASE MANAGEMENT SYSTEM (TDBMS)

Threat in databases is defined by Aziah Asmawi in [5] as a set of policies, measures and mechanisms to provide safety, availability and integrity of data and to combat possible attacks on the system from outsiders as well as insiders, both accidental and malicious. Aziah Asmawi has mentioned about SQL injection which can be executed by two ways by unauthorized user accessing the database via web page connected network:

Access through login page: This is the easiest technique in which it bypasses the login forms where users are authenticated by using password. This type of technique can be done by the attackers through: *'or' condition*, *'having' clause*, *multiple queries* and *extended stored procedure with package*.

***Access through URL*:** The attackers use this technique: manipulating the query string in URL and using the SELECT' and UNION statements. Further Ravi Sandhu [1] has described in his paper that threat to the database can be internal or external. By this technique he has characterized the security breach as incorrect data modification, unauthorized data observation and data unavailability.

As mentioned in [4] types of threats are following:

People: At this point the different people involved in the database management system can be a government authority

 DOI : 10.35291/2454-9150.2021.0349

, an employee or a person-in-charge, consultants, contractors, visitors, hackers, organized criminals, spies, terrorists and social engineers may deliberately or unintentionally exact damage on any of the database environment factors.

***Malicious Code*:** Refers to Software code , in**6** which most cases is intentionally written to damage or violate one or more of the database environment components are boot sector worms, viruses, spoofing code Trojan horses, denial-of-service flood, bots, rootkits, bots, E-mail spamming, macro code.

***Natural disaster*:** Calamities caused by nature can destroy any or the entire database environment components.

***Technological disasters*:** Refers to Some sort of malfunction in hardware or equipment, technological disorders like media failure, hardware failure, power failure, or network failure can inflict damage to database management systems, data files or data or whole database.

VII. SECURITY METHODS IN DATABASE MANAGEMENT SYSTEM (SMDBMS)

Here we will discuss some security methods in DBMS. In early days security methods in database management systems focused only on role based access control or maintaining the confidentiality or authenticity of the database. But in the current scenario the unauthorized user working on a web page which is connected via internet connection has access to the database, since all the queries sent by the user are converted to SQL query in that database. The user may send malicious queries and confirm or modify the transactions of the database without affecting the performance of the database. This type of attack is called SQL injection. But in the current scenario the security method of the database should focus on role based access control and maintain the CIA and avoid attacks due to the network. This section emphasizes the same, based on various papers and books available on similar topics or the same issue.

SECURING DATABASE BASED ON ACCESS CONTROL:

In this section we will discuss database security based on access control. The role based access control method has been proposed by Guoliang Zou, Jing Wang, Dongmei Huang [6] where he has implemented security using the following points:

Preventing illegal users from logging the system

Identify validation

Access Control Interface

Verification codes

Database security: storage procedure

Database security: oracle parameter

The author Ravi Sandhu has created various security approaches [1] where he has considered that access control policies in early days were based on the development of two different classes of models, the discretionary access control policy and on the required access control policy and procedure. Based on these models of early days [7] have proposed two assumptions: The first assumption was that the access control models for databases should be defined in terms of the logical data model; hence authorizations for a relational database should be defined in terms of relational model such as relations, relation attributes and tuples etc. The second assumption is that for databases, in accession to name-based access control, where the secure and protected objects are categorized by giving their names, content-based access control has to be promoted. Discretionary access control policy has subsidized the creation and development of System R access control for relational database management systems which altered strongly on some key features such as distributed authorization administration, effective grant and revoke of authorizations and the use of views for supporting and developing content-based authorizations. Furthermore the access control policies of an object oriented database (OODBMS) are defined in [8]. Here in this point the author has discussed two proposed security models for OODBMS. They are given below as:

Sorion Security Model: This is a security model proposed by Thurainsingham to associate a secure access control into the ORION model system.

Jajodia-Dogan Security Model: Jajodia-Dogan (6, 12) has proposed a security model for OODBMS that controls access by using the encapsulation characteristic of object oriented databases. Henceforth using the access control policies and procedure the confidentiality of the database can be supported.

The second security issue of database management systems has various fields of database integrity as described in [5]: *Physical database integrity protection*: It manages data integrity through physical obstacles such as fires and power failures. *Logical data integrity protection*: It refers to the assertion that information can be changed only by users.

Data element integrity protection: It involves data efficiency and data regularity. And the third security issue availability as described above belongs to the data availability from the database management system. Henceforth, Due to the availability of the company's whole information on the web page which is connected via Internet to its database, the whole data of that company is available using the SQL injection. The below section describes the security methods to prevent SQL injection in that scenario.

 DOI : 10.35291/2454-9150.2021.0349

VIII. SOME SECURITY METHODS TO PREVENT SQL INJECTION

Hence as a protection from SQL injection many Intrusion Detection Systems (IDS) have been suggested. A brief description of these IDS is discussed below:

Misuse Detection System for DBMS

(DEMIDS): This method has been proposed by Chung *et al*. (1999). It is called a misuse-detection system, created for relational databases. It uses audit data log to retrieve profiles describing typical behavior of users in Database Management System. The method is presented by Lee *et al.* (2000). This method is based on intrusions. Hence this method has used time signatures to discover database intrusions. On the other way similar work was proposed by Low *et al*.(2002).This method is used for Detecting Intrusion in Databases through Fingerprinting Transactions (DIDAFIT).It is a system created using misuse detection approach to show database intrusion detection at the application level in a database. But another approach towards a database specific intrusion detection mechanism is by Hu and Panda (2003). They proposed and developed a mechanism that is more capable of finding data dependency relationships among transactions and use this information to find hidden anomalies in a database log. Ke Chen *et al.* (2005) developed an intrusion detection model for a database system based on digital amnesty. It gives an additional layer of security against DBMS misuse. On other hand a real-time intrusion detection mechanism based on the profile of user roles has been prescribed by Bertino *et al.* (2005). This total approach is based on mining SQL queries stored in audit log files in a database. Rietta (2006) described an application layer intrusion detection system, which should take the form of a proxy server and apply an anomaly detection model based on distinct characteristics of SQL and the transaction history of an appropriate user application and user. Aziah Asmawi has proposed SQL Injection and Insider Misuse Detection System (SIIMDS) in 2008 to define both types of intrusions from external and internal threats. Malicious users may access a series of safe information and then apply different techniques to retrieve sensitive data by using that information. To address these inference problems,Yu Chen in [9] has created a semantic inference model (SIM) that symbolizes all the possible inference channels from any attribute in the system to the set of elevated sensitive attributes. Hence based on the SIM, the violation detection system keeps track of a user's query history in a database. When a new query is stiffed, all the channels where sensitive information can be stored will be recognized. If the probability of inferring sensitive information increases to a more specified threshold, then the current query request will be revoked. Using the security methods mentioned in section A and B secure and safe databases can be created. It may be accessed from anywhere and the security would be managed. Even though there is no such thing as a 100 percent guarantee in network security, awful obstacles can be placed in the path of SQL injection attack. Anybody of these defenses extremely reduces the chances of a successful SQL injection attack to prevent our data. Implementing all four is a best practice that will supply a high degree of protection and safety. Despite its extensive application, your web site does not have to be SQL injection's next suspect. The next section briefs up all the vulnerabilities, threats and security methods of database management systems in tabular format which will be beneficial for the development of secure and safe databases. There actually are a lot of methods that web site owners can do to secure against SQL injection attacks[19].

Table 1: Comparison details of VDBMS, TDBMS and SMDBMS

Vulnerabilities (VDBMS)		THREATS (TDBMS)	SECURITY METHODS SDBMS)
Vendor Bug	Buffer Overflow, Programming errors	May damage or violate the database	Unauthorized access control policy
Poor Architecture	Weak form of encryption	May damage database environment components (networks, applications, operating systems, DBMS and data)	1.Sorion Security Model 2.Jajodia-Dogan Security Model
Misconfiguration	Not properly locking database	Loss of integrity of the database	1.Physical database integrity protection 2.Logical data integrity protection 3.Data element integrity protection
Incorrect usage	SQL injection	Misuse of availability of database	Intrusion Detection System like 1. A Misuse Detection System for Database System (DEMIDS) 2.SQL Injection and Insider Misuse Detection System (SIIMDS) 3. Detecting Intrusion in Databases through Fingerprinting Transactions (DIDAFIT) 4. Semantic inference model (SIM)
	Deactivation	Easy access	Two principles should

 DOI : 10.35291/2454-9150.2021.0349

Irresponsible DBA	of necessary security mechanism	of data	be followed: 1. The access control models for databases should be expressed in terms of the logical data model; thus authorizations for a relational database should be expressed in terms of relations, relation attributes, and tuples. 2. For databases, in addition to name-based access control, where the protected objects are specified by giving their names, content-based access control has to be supported.
Hidden Flaws in DB	Undetected defects	Allow hackers to connect to the database server by exploring those defects.	Intrusion Detection System
Unauthorized Users	Unauthorized users "still" the credentials of authorized users	Easy access to database servers.	Intrusion Detection System
Misused Privileges	Authorized users take advantage of their privileges.	Maliciously access or destroy data	Database Administrator should provide security on the basis of above mentioned principles.

IX. CHALLENGES - WHY PROTECTING DATABASES IS EVEN MORE DIFFICULT TODAY

Despite the increased focus by research and industry toward improving security of our cyber infrastructures, today the protection of data, entrusted to enterprise information systems, is more challenging than ever. There are several factors underlying this trend. Data security concerns are evolving. In addition to the traditional requirements of data Confidentiality, integrity and availability, new requirements are emerging such as data quality [69], completeness, timeliness, and provenance [35]. In particular, it is important that data be complete, correct, and up-to-date with respect to the external world. The increasing quality of data will make data more valuable. Highly valuable data increases the potential to be gained from unauthorized access and the potential damage that can be done if the data is corrupted. The amount of data is increasingly large: "It is estimated that the amount of information in the world is doubling every 20 months, and the size and number of databases are increasing even faster" [1]. Therefore, protection mechanisms must be able to scale well. The intermediate information processing steps typically carried out by corporate employees such as typing an order received over the phone are removed. Users who are outside the traditional corporate boundary can have direct and immediate online access to business information which pertain to them. In a traditional environment, any access to sensitive information is through employees. Although employees are not always reliable, at least they are known, their access to sensitive data is limited by their function, and employees violating access policies may be subject to disciplinary action. When activities are moved to the Internet, the environment drastically changes. Today, due also to the offshoring of data management functions and the globalization of business enabled by the Internet, companies may know little or nothing about the users (including, in many cases, employees) accessing their systems and it is more difficult for companies to deter users from accessing information contrary to company policies. Finally, as a result of trends toward ubiquitous computing, data must be available to users anywhere anytime. Because of these increased risks, the adequate protection of information systems, managing and making available large data volumes, is not an option any longer. Not only will damage to the data affect a company's businesses and operations, it could also have legal consequences on companies especially if, as discussed by Schneier [83], laws were to be promoted enforcing liability of software products and applications. As Schneier argues in his paper, in the very near future insurance companies will move into cyber-insurance and we can certainly expect that "they will start charging different premiums for different security levels." All the above motivations are thus strong drives for the systematic adoption of solutions that are more articulated and comprehensive than the ones available today. Not only must adequate solutions be developed and deployed,

but organizations also need to show that they comply with security and privacy requirements. In particular, research efforts need to be devoted to a large number of topics including: Data Quality and Completeness. Users increasingly rely on information they find on the Web. This is the case for example of medical information. However, users do not, in general, have guarantees that the data is complete and of acceptable quality. We need techniques and organizational solutions to assess and attest the quality of data. Techniques in this respect may include simple mechanisms such as quality stamps that are posted on Websites. Other techniques include providing more effective integrity semantics verification and the use of tools for the assessment of data quality, based on techniques such as record linkage. Application-level recovery techniques are also needed for automatically repairing incorrect data. Intellectual Property Rights (IPR). Data in many cases are the results of intellectual activities of individuals and

 DOI : 10.35291/2454-9150.2021.0349

organizations. Questions concerning IPR are thus becoming increasingly relevant. To address some of these concerns, watermarking techniques for relational data have been recently proposed [84], [85] which can be used to detect IPR violations. Research is however needed to assess the robustness of such techniques and to investigate different approaches aimed at preventing IPR violations. Access control and privacy for mobile users. Users will be increasingly mobile and will have a large variety of devices available to them. Moreover, the deployment of computing power and sensors in every-day environments will make it possible for users to be always connected, sometimes without even being aware of it. In such contexts, several issues are relevant. Users will execute many more activities online; information about user identities, profiles, credentials, and permissions will be more frequently required. Such information will need to be secure and reliable; reliable user identification will be increasingly crucial. It is thus important on one hand to develop techniques for efficient storage of security relevant information on small devices; a relevant example in this respect is represented by the notion of portable access rights recently proposed by Bykova and Atallah [29]. On the other side, it is important that access control mechanisms be integrated with standards being developed for identity management [57] as well as with trust negotiation techniques [23]. Because large-sized streams of data are generated in such environments, efficient techniques for access control must be devised and integrated with processing techniques for continuous queries. Finally, the privacy of user location data, acquired from sensors and communication networks, must be assured. Database survivability. This is an important topic which has been largely unexplored, despite its relevance. Survivability refers to the ability of the database system to continue its functions, may be with reduced capabilities, despite disruptive events, such as information warfare attacks. To date, issues related to database survivability have not been investigated much. Liu [58] has proposed four database architectures for intrusion-tolerant database systems that focus on the containment of malicious transactions. Even though this is an important initial step, much more research needs to be devoted to techniques and methodologies assuring database system survivability.

X. CONCLUSION

Frameworks for DBMS construction can be used to develop special purpose active or object-oriented DBMS, and may also help to implement database federations. Active mechanisms are the most promising technology for solving problems in many areas of software systems in general, in particular also in database security. Although the technology is still premature and not very well understood, it may turn out to trigger a revolution in the design, construction and maintenance of software systems. Depending on the event model and the expressiveness of the conditions, very complex situations within a computer system can be monitored (which can also be used for intrusion detection /Denn 86, LuJa 88/) and an appropriate reaction can be triggered. Moreover, since active rules subsume deductive rules /Wido 93/, they can also be applied to represent knowledge about any kind of application (e.g. how to map abstract concepts onto concrete mechanisms). Database systems support the controlled and integrated inspection and modification of such information. Standard mechanisms are provided that can be used by several applications, so that even the reuse of functionality is supported. As a side-effect, numerous calls to the DBMS often can be saved.11 The applications "simply" send very abstract requests to the DBMS, and powerful servers (not necessarily mainframes, but perhaps "clusters" of high-end workstations) where these DBMS are running on process the data locally. Only the results the applications are really interested in have to be transmitted over the network. In summary, current trends in database technology do indeed have considerable impact on security concepts, in terms of both, better solutions that can be supported, and new problems that need to be solved. Unfortunately, commercial products in this area are – once again! – very slow to incorporate security features that are as advanced as the rest of the system from the very beginning. At best, they are going to retrofit them to the system in later releases. The security community is thus challenged not only to devise and evaluate appropriate concepts, but also to push for and foster the necessary technology transfer to DBMS builders and users. Organizations like the OMG in case of object-oriented technology can support the transmission of know-how from the research community to the software industry.

REFERENCES

[1] Current trends and new challenges of databases and web applications for systems driven biological research, December 2010, frontiers in Physiology, DOI:10.3389/fphys.2010.00147

[2] Current Trends in Database Technology and Their Impact on Security Concepts Klaus R. Dittrich and Dirk Jonscher Institut für Informatik, Universität Zürich, Winterthurerstr. 190, CH-8057 Zürich {dittrich,jonscher}@ifi.unizh.ch

[3] Database Trends and Directions: Current Challenges and Opportunities

[4] A Relative Study on different Database Security Threats and their Security Techniques

[5] Security Of Database Management Systems

[6] A Relative Study on Different Database Security Threats and their Security Techniques January 2020 DOI:10.13140/RG.2.2.11657.60000

[7] Security Issues and Their Techniques in DBMS - A Novel Survey, December 2014, Inter- national Journal of Computer Applications 85(13), DOI:10.5120/14905-3402

 DOI : 10.35291/2454-9150.2021.0349

[8] A Relative Study on Different Database Security Threats and their Security Techniques

[9] Data Breach Investigation Reports

[10]https://www.ictsecuritymagazine.com/wp-content/uploads/2016-Data-Breach-Investigations-Report.pdf

[11]https://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf

[12]https://www.ictsecuritymagazine.com/wp-content/uploads/2018-Data-Breach-Investigations-Report.pdf

[13]https://www.ictsecuritymagazine.com/wp-content/uploads/2019-Data-Breach-Investigations-Report.pdf

[14] A study on SQL injection techniques, December 2016, International Journal of Pharmacy and Technology 8(4):22405-22415

[15]https://www.blackhat.com/presentations/bh-dc-07/Shulman/Paper/bh-dc-07-Shulman-WP.pdf

[16] Security in Database Systems, Global Journal of Computer Science and Technology Network, Web & Security Volume 12 Issue 17 Version 1.0 Year 2012 Type: Double Blind Peer Reviewed International Research Journal Publisher: Global Journals Inc. (USA) Online ISSN: 0975-4172 & Print ISSN: 0975-4350

[17] Study on Database Management System Security Issues, November 2017,DOI:10 .3063 0/joiv.1.4-2.76

[18] Active Database Systems, March 1999, ACM Computing Surveys 31(1):63-103, DOI: 10 .1145/311531.311623

[19] A Literature Review on Evolving Database, March 2017, International Journal of Computer Applications 162(9):35-41, DOI:10.5120/ijca2017913365

 DOI : 10.35291/2454-9150.2021.0349