Amended Anticipation Model for Fast Exposure of Mischievous Communications in Database Systems

Sushant Yadav[#1], Mrs. Mamta Yadav[#2]
M.Tech Scholar, Assitant Professor
YCET, Narnaul (India)

Abstract *— Database Security is an concept that includes the following properties: authenticity (guarantees that a service or piece of information is authentic), confidentiality (absence of unauthorized disclosure of a service or piece of information), integrity (protection of a service or piece of information against illicit and/or undetected modification), and availability (protection of a service or piece of information against possible denials of service caused by malicious actions). Current intrusion detection systems use logs to detect malicious transactions. Logs are the histories of the transactions committed in the database. The disadvantage of using logs is that they need lot of memory. In addition to this sometimes even after a transaction is detected as malicious it cannot be rolled back. In this paper we present a method by which we can overcome the uses of logs and can detect malicious transactions before they are committed. We use specific user-profiles to store the sequence of commands in a transaction and use a prevention model for instant detection of malicious transactions.*

Keywords — *Database Security, intrusion detection systems.*

I. INTRODUCTION

Database security is also a specialty within the broader discipline of computer security. Information is the most serious resource for many organizations. In many cases, the success of an organization depends on the availability of key information and, therefore, on the systems used to store and manage the data supporting that information. The security of data against unauthorized access or corruption due to malicious actions is one of the main problems faced by system administrators. Due to the growth of networked data, security attacks have become a dominant problem in practically all information infrastructures. Imitation was carried out for a single as well as multiple users providing sequence of queries varying the size of the Database. A new approach for detecting malicious access to a database system is proposed and tested in this work. The proposed method relies upon manipulating usage information from database logs into three dimensional null-related matrix clusters that reveals new information about which sets of data items should never be related during defined temporal time frames across several applications. If access is detected in these three dimensional null-related clusters, this is an indication of illicit behaviour, and further security procedures should occur. In this thesis, we describe the null affinity algorithm and illustrate by several examples its use for problem decomposition and access control to data items which should not be accessed together, resulting in a new and novel way to detect malicious access that has never been proposed before.

A typical database application is a client-server system where a number of users are connected to a DBMS via a terminal or a desktop computer (the trend today is to access database servers through the internet using a browser). The user actions are translated into SQL commands by the client application and sent to the database server. The results are sent back to the client to be demonstrated in the adequate format by the client application. In many cases, the system includes an application server that runs business rules code (i.e., code directly related to data application handling) and performs load balancing of the multiple client sessions. The main goal of security in DBMS is to protect the system and the data from intrusion, even when the potential intruder gets access to the machine where the DBMS is running. To protect the database from intrusion the DBA must prevent and remove potential attacks and vulnerabilities.

The system susceptibilities are an internal factor related to the set of security mechanisms available (or not available at all) in the system, the correct configuration of those mechanisms (which is a responsibility of the DBA), and the hidden flaws on the system implementation. Vulnerability prevention consists on guarantying that the software used has the minimum vulnerabilities possible. This can be achieved by using adequate DBMS software. On the other hand, as the usefulness of the security mechanisms depend on their correct configuration and use, the DBA must correctly configure the security mechanisms by following administration best practices. Susceptibility removal consists on reducing the vulnerabilities found in the system. The DBA must pay attention to the new security patches

release by software vendors and install those patches as soon as possible. Furthermore, any configuration problems detected on the security mechanisms must be immediately corrected.

Our proposed method also has several benefits over other traditional methods of insider threat detection in that it requires little storage space, can be easily adjusted to reflect new applications, is very quick in operation once the historical data has been properly clustered, and requires only a moderate amount of computational time to calculate.

Intrusion exactly means interrupting or interfering in others work. In a better way it can be defined as any set of actions that attempts to compromise integrity, confidentiality and availability of resource. Intrusion detection is a security technology that attempts to identify either individual who is trying to break into system and misuse information without authorization and\or those who have legitimate access to the resource but are taking undue advantage of their rights. The job of Intrusion Detection System (IDS) is to dynamically monitor the events occurring in a system and alert when any suspicious activity occurs so that defensive action can be taken to prevent or minimize damage. In general, the main goal of IDS is to detect malicious transactions before they are being committed and then dropping and rolling them back. If the malicious transactions have been committed and have caused damages, then locating the damaged parts and repairing them on time will be much more problematic. Intrusion detection systems serve three essential security functions: they monitor, detect and respond to unauthorized activity.

II. BACKGROUND HISTORY

A. Existing Intrusion Detection Systems: Marco and Henrique proposed a Database Malicious Transaction Detector (DBMTD) in 2005. DBMTD mechanism is a log based mechanism for the detection of malicious transactions in DBMS. In practice malicious database transactions are related to security attacks carried out either externally or internally to the organization. External security attacks are intentional unauthorized attempts to access or destroy the organization's private data. These attacks are perpetrated by unauthorized users (*hackers*) that try to gain access to the database by exploring the system vulnerabilities (e.g., incorrect configuration, hidden flaws on the implementation, etc). On the other hand, internal security attacks are intentional malicious actions executed by authorized users. These users use their normal rights to execute illicit actions for their personal benefit or to harm the organization by causing loss or corruption of critical data.

Given the growing threat represented by security attacks to databases, and the fact that databases are where most of the relevant data of organizations is actually stored, a practical mechanism to detect the execution of malicious transactions in databases is of utmost importance. However, to the best of our knowledge, none of the existing commercial DBMS provides such a mechanism. Manual supervision and audit procedures are normally the only tools the DBA can use to detect potential database intrusion.

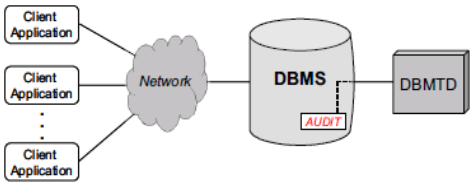

Figure 1. DB system using the DBMTD mechanism.

In a typical database environment the profile of the transactions that each user is allowed to execute is usually known by the DBA, as the database transactions are programmed in the database application code. In other words, the transactions are not ad hoc sequences of SQL commands. On the contrary, database transactions are well defined sequences of commands performing a finite set of predefined actions. For example, in a banking application users can only perform the operations available at the application interface (e.g., withdraw money, check account balance, etc). No other operation is available for the end users. Particularly, end users cannot execute ad hoc SQL commands.

The DBMTD mechanism uses the profile of the transactions defined in the database applications (authorized transactions) to identify user attempts to execute malicious transactions. DBMTD is built on top of the auditing mechanism implemented by most commercial DBMS. The audit lo is used by DBMTD to obtain the sequence of commands executed by each user, which is then compared with the profile of the authorized transactions to identify potential malicious transactions. IDS are based basically on two models Anomaly Model and Misuse Model. Anomaly model establishes a normal activity profile for the system and if any activity fails to match the profile of the normal profile then the IDS considers it as an intrusion attempt. The misuse model is based on the assumption that there are ways to represent attacks in the form of a pattern or a signature so that even variation of the same attack can be detected. Here the IDS maintain a database of all the known attack signatures. It raises an alarm whenever the attack signature matches the one that the IDS have in its database. There are possibilities that the IDS might be unable to detect an intrusion attempt (false negative) or might catch a normal behavior as intrusion (false positive). IDS are of three types namely Network based, Host based, combined IDS. A network based IDS deployed outside the firewall monitors the data packets traveling over the network and any possible attack on the data packets to

modify or read them are recorded by the IDS. A host based IDS is deployed on the host machine.

B. Types of Attacks: Attacks can be used to disclose information, to sidestep authentication mechanisms, to alter the database, and to execute arbitrary code, in certain instances, on the database server itself. On the basis of relationship between intruder and victim, attacks to the database can be classified as:

• ***Insider:*** An authorized user, who can be from own enterprise's employees or their business partners or customers, misuses his privilege or performs unauthorized access. They have privileges to access the application or system but misuse it and are usually harder to defend.

• ***Outsider:*** An unauthorized user coming from outside, frequently via the Internet who tries to gain access to system. They do not have proper rights to access the system and can be defended using strong security mechanisms.

• ***Attempted Break Ins:*** When an unauthorized user tries to gain access to a computer system is most often detected by typical behavior profiles or violations of security policies.

• ***Masquerader (Internal) Attacks:*** When an authorized user pretends to be as another user. These attacks are also called internal because they are caused by already authorized users. It is also detected by a typical behavior profiles or violations of any security policies.

• ***Penetration Attack:*** Usually detected by monitoring for specific patterns of activity like when a user attempts to directly violate the system's security policy.

• ***Leakage:*** Moving potentially sensitive data from the system is mostly detected by a typical usage of I/O resources.

• ***Denial of Service:*** Denying, by making the resources unavailable to other users. It is often detected by a typical usage of system resources like denying other users, the use of system resources by making them unavailable.

• ***Malicious Use:*** It involves various attacks such as file deletion, viruses etc. It is often detected by typical behavior profiles, violations of security policies, or use of special privileges.

III. Methodology and Planning of Work

The early research mainly focused on network-based and host-based intrusion detection. However, in spite of the significant role of databases in information systems, very limited research has been carried out in the field of intrusion detection in databases. We need intrusion detection systems that work at the application layer and potentially offer accurate detection for the targeted application. The approaches used in detecting database intrusions mainly include data mining and Hidden Markov Model (HMM). Chung presents a misuse detection system called DEMIDS which is tailored to relational database systems. DEMIDS uses audit logs to derive profiles that describe typical behavior of users working with the DBS. The profiles computed can be used to detect misuse behavior, in particular insides abuse. DEMIDS sue "working scope" to find numerous item sets, which are sets of feature with certain values. They define a notation of distance measure that captures the closeness of set of attribute with respect to the working scopes. These distance measures are then used to guide the search for frequent item-sets in the audit logs. Misuse of data, such as tampering with the data integrity, is detected by comparing the derived profiles against organizations security police or new audit information gathered about users. The main drawback of the approach presented is a lack of implementation and experimentation. The approach has only been described theoretically, and no empirical evidence has been presented of its performance as a detection mechanism.

A. First Approach: Database Intrusion Detection System for Role Based Enabled Database: The proposed approach in this section is, as from query based approach to transaction based approach. The main advantage of this approach is to extract the information among queries in the transaction. For example consider the following transaction:

Begin transaction
Select: a1, a2, a3, a4, a5 from t1, t2;
Update: t2 set a4= a2+1.2(a3);
End transaction

Where t1 and t2 are tables of the database and a1, a2, a3 are the attributes of table t1 and a4, a5 are the attributes of table t2 respectively. This example shows the correlation between the two queries of the transaction. It states that after issuing select query, the update query should also be issued by same user and in the same transaction. The approach based on the RBAC database uses the Naïve Bayes classifier as a learning algorithm to generate the role profiles on training data, and the training data which one is extracted from the log file and Users (Local/Remote) Database server Audit Server Application Layer stored into the form of particular representation to represent the user transaction behaviour.

B. Second Approach: Database Intrusion Detection System Using Legal Transaction Profiles: Basically this proposed approach is divided into three steps: Auto-generation legal profile phase, Detection phase, Action phase. It takes the advantage over the manual transaction profiles mechanism. As in this case the time to generate the legal transaction profile is reduced, also it overcomes the disadvantage of the existing system based on manual profile generation. The log file is used from which the history of the transactions are removed and stored into the offline audit trail and this can be done using the inclusion of existing auditing mechanism. Later the generated legal transaction profiles from offline audit trail are used at the detection phase to match with the

executable transactions; if any deviation is there then particular executable transaction is marked as malicious otherwise committed into the database. The last phase is the action phase and it may take the action based on the alarm generated by the database IDS.

C. Third Approach: Database Intrusion Detection System Using Counting Bloom Filter (CBF):A Bloom filter is used to define the bit array of *m* elements of n bits size and initially all set to 0. The filter uses a group H of k independent hash functions 1., *k h* wit range {1, . . . , n} that independently map each element in the universe to a random number uniformly over the range. For each element *x*ÎS , the bits B [hi(x)] are set to 1 for 1 _ i _ k. (A bit can be set to 1 multiple times.) To answer a query of the form "Is *y*ÎS ?", we check whether all ( ) *i h y* are set to 1. If not, y is not a member of S, by the construction. If all ( ) *i h y* hi(y) are set to 1, it is assumed that y is in S, and hence a Bloom filter may yield a false positive. The main problem with the bloom filter is the false positive i.e. it gives the wrong answer with correct query, and it is resolved using the counting bloom filter (CBF) where insertion and deletion of the set of the elements are possible. It also uses as similar to the bloom filter, *k* (random hash) functions, each of which maps or hashes some set element to one of the *n bits* array positions. To insert an element into a set, the element is passed into k hashing functions and k index values are obtained. All counters in counting bloom filter at corresponding index values are incremented. The overall approach based on the CBF is divided into the three phases. The initial phase is as similar to the automatic transaction profile generation algorithm to generate the authorized transactions. This process insures the correctness of the genuine profiles as declared as the legal profiles, its do automatically instead of manually thus it reduces the time to require for manual transaction profile generation. This next phase is all about the construction of the counting bloom filter (CBF) where random weights are assigned automatically corresponding to commands of legal transactional profile. After that the construction of the CBF is done by incorporating the hash functions. At the final stage of detection phase the constructed CBF along with the weights are loaded and the counter values in CBF are decremented using weight of identified command based on the executable transaction, if all the bits in the CBF are zero then the transaction is declared as valid.

An Intrusion Detection System should involve some preventive measures at the beginning in order to put a stop to the entrance of intruders into system. Then detection method should be used, to identify if any intruder had successfully bypassed the preventive step. If any intrusion has been detected then an appropriate response should be taken to defense from the attacker. In brief it can be said to have: Intrusion Prevention + Intrusion Detection + Appropriate Response in a database system to get rid of intruders. It will be better to include role based access and use data mining technique for intrusion detection in the IDS for producing more fine results. Speed, accuracy and adaptability are the common problems in IDS. The extensive amount of data that intrusion detection systems need to monitor in order to observe the entire situation causes speed problem. To handle this situation, the most important portion of information should be removed in order to provide efficient detection of attacks. The adaptation and accuracy issues of the intrusion detection can be solved by incorporating learning algorithms. In case of intrusion detection, learning means discovering patterns of normal behavior or pattern of attacks. In this way intrusion detection can combine the advantages of both signature based and anomaly-based IDS. The block diagram of a hybrid Database IDS is shown in figure.

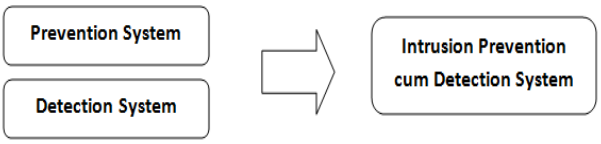

Fig.2 Combination of IPS and IDS system

From the figure, it is clear that initially the user sends service request via web based application, to the application server. The application server issues request to the database. Then, user can log to database. At this level, preventive measures can be taken by using exact password entry logins in role based authentication and analyzing the requested https. As the user logs into the database, the database session gets started and the SQL statements that are received from the application are passed to the detection module.

D. Assigning Null Affinity Temporal Values: In a typical database environment, transactions are programmed into various database application interfaces, so as long as the database applications remain stable, the set of transactions that are executed will not change. For example, in an educational database application, users can only perform and interact with data items that are available at their user-dependent application interface (e.g., viewing grades, paying for classes, entering grades, dropping a student from a class, etc). Other operations are not available for that particular class of end-user. Normally, end-users cannot execute ad hoc queries against the database. It is a very realistic assumption to use transactional profiling to detect malicious data access, resulting in a reduced risk of false alarms with other intrusion detection mechanisms.

2D Usage Array

For each application *Ay* that is run on the database, choose a time frame *Tz*. The time frame needs to be

chosen with a granularity that returns the usage of the database, and the temporal frame can be adjusted as necessary. For each data item *Ix* available to an application, an attribute usage value, denoted as *use(Ix, Ay, Tz*), is defined as:

use(Ix, Ay, Tz) ={1 if application Ay uses data item Ix during time frame Tz

0 if otherwise}

The matrix is stored in an *M x N* two dimensional data usage array at position *Ai,j* where *M* is the number of data items (rows) and *N* is the number of temporal frames (columns) and *i* is the particular data item and *j* is the particular time frame under examination.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
I_a	1	0	0	0	0	0	0
I_b	1	1	1	1	1	1	1
I_c	1	0	1	0	1	0	1
I_d	0	0	0	0	0	1	0
I_e	0	0	0	0	0	1	0

Fig.3 Example 2D usage matrix

In the above figure, an application used data item *Ia* only during time frame *T1*. In this example, we can define the time frame to be the days of week, so time frame *T1* would correspond to Monday during a given week and year. If it is so preferred, the temporal granularity can be changed so that the time frame under examination has a larger or smaller amount of granularity. For example, the usage matrix could be defined for only each hour during the Monday *T1* time frame from the above example. The time frame could also have a much larger granularity; for example, the data usage matrix could be defined as the previous twelve months of usage.

3D Usage Matrices:

The 2D usage arrays are calculated for each application that is run against the database, resulting in a three dimensional relationship. If an application is not suspected for potential misuse, the security engineer can extricate that particular application from the process and focus on other, more susceptible, applications. The result is then an *M x N x P* relationship where *M* is the number of data items, *N* is the number of temporally related time frames that were chosen in the first step, and *P* is the number of applications that are run against the database. This 3D usage data is used next to find and cluster elements that should not be used together across three dimensions, time, application level, and data item level, resulting in a novel way to detect misuse of the database.

Computing Null Temporal Affinity Energy Levels

Given the 3D usage matrix of a particular system, we have defined a *Null Temporal Affinity Energy (NE)* methodology that processes the 3D matrix so that dense clumps of zeros are clustered together across all thee dimensions. The *NE* algorithm was devised so that a three dimensional matrix that possesses dense clumps of zeros in all three dimensions will have a large *NE* (null energy) level when compared to the same three dimensional matrix whose elements across the y and z axes have been permuted so that its numerically small elements are more uniformly distributed throughout the relation. The x axis (time) is not permuted because each of the time frames are related by some fixed metric; this relationship will be used to expand or contract the three dimensional relationship to home in on particular areas of potential misuse in an area of future work described.

The proposed null energy level is the sum of the bond strengths in the 3D array of each nearest neighbor in three dimensions, where the bond strength is defined as the inverse of their product. The NE value, then, is given by:

$$NE(U)_{Tx} = \tfrac{1}{2} * \sum_{i=1}^{i=M}\left(\sum_{j=1}^{j=N}\left(\sum_{k=1}^{k=P}\left(\overline{A_{i,j,k}} * \left[\overline{A_{i+1,j,k}} + \overline{A_{i-1,j,k}} + \overline{A_{i,j+1,k}} + \overline{A_{i,j-1,k}} + \overline{A_{i,j,k+1}} + \overline{A_{i,j,k-1}}\right]\right)\right)\right)$$

A represents the binary inverse of *Ax*. *U* is a nonnegative *M x N x P* three dimensional array consisting of results of the *use* function over time period *Tx*; *U* was derived earlier. Since all the values of *U* are binary, the *NE* value is very efficiently calculated, even when given large three dimensional matrices. The value of ½ is present so that each bond is only counted once in the total *NE* sum. To better understand what the *NE* value represents, one can visually circle adjacent zeroes going up, down, left, or right in the same two dimensional plane, and also potentially circling zeros in the plane above and below the current element.

III.RESULT

Comparison of All Three Proposed Approaches:

For the comparison we consider the set of parameters to evaluate each approach with other one. The complete details of comparison are given in below table 1.

Table 1. Comparison of proposed approaches for database IDS

Approaches	Learning Time	False Positive	False Negative	Load on Server
First Approach	less	no	no	Yes
Second Approach	less	no	no	Yes
Third Approach	more	no	no	Yes
Only Based on Auditing Mechanism	-	no/yes	no/yes	Less

Based on the information in the above table as we can see the proposed approaches are very much useful to handle the malicious transaction once it is executed by the unauthorized user. The proposed approached also applicable to handle the internal misuse over the database. If we see the load on the database server for proposed mechanisms then it is quite high because of the inclusion of one additional layer of security into the database but it is less in auditing mechanism.

The security in the DBMS is one of the main concerns of the researchers now-a-days and there is an interest to develop the possible database intrusion detection systems.

The proposed method relies upon manipulating usage information from database logs into three dimensional null-related matrix clusters that reveals new information about which sets of data items should never be related during defined temporal time frames across several applications.

IV. CONCLUSION

This paper has proposed a new mechanism to detect malicious data access. As database systems play a vital role in organizational information architectures, procedures must be in place to ensure that these resources are not being used maliciously. We have presented the concepts and underlying architecture and shown how they can be applied. Our proposal relies on using historical data stored by the database logs on what data items were used at a particular time by various applications. We have used specific user-profiles to store the sequence of commands in a transaction and use a prevention model for instant detection of malicious transactions.

This information is then processed to reveal clumps of data items that should not be used together during certain time frames, resulting in a three dimensional usage matrix. This matrix allows a better prediction of potential misuse by allowing quicker and more precise prediction of items that should not be used together across the time, data item, and application dimensions

Suspicious queries are then compared to the maximized usage array and a distance value is calculated for each non conforming action. These distances are summed to reveal how far from what was expected this access is. If the access is above a certain threshold, further security procedures are performed. It is concluded that by choosing optimal value of size of Threshold Energy Value and number of combination of IPS and IDS functions the detector can be made to prevent a malicious transaction with a probability of almost 99% plus.

REFERENCES

[1] Marco Vieira and Henrique Madeira, “Detection of Malicious Transactions in DBMS”, IEEE Proceedings- 11th Pacific Rim International Symposium on Dependable Computing, Dec 12-14,2005, PP: 8.

[2] Korra Sathya Babu, “Prevention of Unwanted Transactions in DBMS”, Department of computer Science and Engineering, NIT Rourkela, 2008.

[3] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks ", Comm. of the ACM(1970).

[4] Ravi Sandhu and Pierangela Samarati, “Access Control: Principles and Practice”, IEEE Communications Magazine, September 1994.

[5] Yi Hu and Brajentra Panda, “Identification of malicious transactions in Database Systems”, Proceedings of 7th International database engineering & Applications symposium, 16-18 July, 2013, PP 329-335.

[6] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol”, IEEE Transactions on Networking, 2000, PP 281-293.

[7] TPC Council, “TPC BenchmarkTM C, Standard Specification, Version 5.10.1”, February 2009.

[8] Gordon, L. Loeb, M., Lucyshyn, W. and Richardson, R. Computer Security Institute. Computer crime and security survey, 2006.

[9] Fonseca, J., Vieira, M., and Madeira, H. Online detection of malicious data access using DBMS auditing. In *Proceedings of the 2008 ACM Symposium on Applied Computing*. SAC'08. ACM, New York, NY, 1013-1020, 2008.

[10] Chung, C. Y., Gertz, M., Levitt, K. DEMIDS: a misuse detection system for database systems.In *integrity and internal Control information Systems: Strategic Views on the Need For Control*, Norwell, MA, 159-178,2000