Advanced Techniques for Enhanced Database Security

Er. Bhavna Sharma

Abstract— Database security is the system, processes, and procedures that protect a database from unintended activity. Unintended activity can be categorized as authenticated misuse, malicious attacks or inadvertent mistakes made by authorized individuals or processes. Database security is also a specialty within the broader discipline of computer security. Information is the most critical resource for many organizations. The protection 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. 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. There is a need to prevent such systems from unauthorized access to the database structure, data values and privileges. In this paper we are implementing intrusion detection cum prevention model for database. We will consider some detection and prevention techniques and added response module to present such a model.

***Index Terms*—Access,Database,Security,User**

I. INTRODUCTION

Database Security breaches can be classified as unauthorized data observation, incorrect data modification and data unavailability. Unauthorized data observation results in the disclosure of information to users not intended to gain access to such information. Incorrect modification of data either intentional or unintentional, results in an incorrect database state. Data unavailability results in improper and untimely functioning of the organization.

Most DBMS are based on the relational data model. The relational data model defines a database as a collection of relations, where each relation is a table with rows and columns. The consistency of the data stored in different tables is assured by a set of relational integrity mechanisms, including referential integrity mechanisms that assure data in one table is consistent with data in other tables. 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 displayed 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 vulnerabilities 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 effectiveness of the security mechanisms depend on their correct configuration and use, the DBA must correctly configure the security mechanisms by following administration best practices. Vulnerability 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.

Basic working of Security in DBMS

In database management system the above solutions are met with different components or methods. Initially to gain access to a database a user has to provide identification and authentication. Identification is the means by which a user provides a claimed identity to the system (i.e. using a username). Authentication is the means of establishing the validity of this claim (password). Issuing identification and initial authentication key to the users is the job of DBA. The user can change his identification and authentication to enter into the database late on.

Access Control Methods ensures confidentiality in DBMS. If anybody tries to access the data object the Access Control Methods checks for the rights of the user against the data object with the set of authorizations stated by the Administrators. Access is the ability to do something with a computer resource. This usually refers to a technical ability (e.g. read, modify, delete, execute or use an external connection. Authorization is the permission to the computer resource. To ensure Data Integrity the access control mechanism and semantic integrity constraints are used together. 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, which must be verified against the database state.

The Recovery subsystem and the Concurrency control mechanism ensures that data is available and correct despite hardware and software failures and accesses from concurrent application programs.

The main goal of database security mechanisms is to protect the data stored in the database from unauthorized accesses or malicious actions in general. In spite of all these steps taken to secure the database still there are possibilities for hackers to take the chance of breaking into the system. A hacker may get a password in many ways and try to operate the system. Unfortunately, the risk of malicious actions and attacks in DBMS becomes a real threat in such a situation and the proposal of instant intrusion detection mechanisms for DBMS is a logical and relevant step. That is just the goal of the present work. Typical database security attacks can be classified as: 1) intentional unauthorized attempts to access or destroy private data; 2) malicious actions executed by authorized users to cause loss or corruption of

ISSN: 2278 – 1323

critical data (e.g., system manager that have legitimate access to database servers but should not access database data); and 3) external interferences aimed to cause undue delays in accessing or using data, or even denial of service. In the present work we are particularly interested in the first two types of attacks, which in practice correspond to malicious transactions executed by authorized users or by unauthorized users that gain access to the database by exploring system vulnerabilities. As database security mechanisms are not design to primarily detect intrusions (are designed to avoid the intruder's access to database data), there are many cases where the execution of malicious sequences of SQL commands (transactions) cannot be detected (or avoided).
The following points present some examples:

– The DBA does not activate the necessary security mechanisms (e.g., authentication, user privileges, data encryption, auditing, etc), which allows intruders to get access to database data.
– The security mechanisms available are incorrectly configured permitting potential intruders (*hackers*) to access the database.
– Hidden flaws in the database implementation may allow hackers to connect to the database server by exploring those defects.
– Unauthorized users "still" the credentials of authorized users in order to access the database server.
– Authorized users take advantage of their privileges to maliciously access or destroy data.

Malicious transactions may damage the database integrity and availability. However, in spite of the pertinence of the detection of malicious database transactions, the reality is that very few practical mechanisms which are able to identify users executing malicious transactions has been proposed so far.

This paper proposes a new mechanism for the detection of malicious transactions in DBMS. As in a typical database environment it is possible to define the profile (sequence of SQL commands) of all the transactions that each user is allowed to execute, the proposed mechanism uses that profile of valid transactions to identify user's attempts to execute invalid sequences of commands. The sequences of commands that constitute each valid transaction are obtained by analysis of the execution profile of the database client applications.

Unauthorized access to data resources is a major threat faced by all organizations. While organizations typically have very complex firewalls and intrusion detection systems in place in order to detect external threats, these systems do little to protect organizations against another significant threat, the malicious insider. These insiders will often not be after the physical systems, rather they will attempt to corrupt or capture the data these systems contain. This threat needs to be protected against, and more work needs to be performed in the prevention and detection of these attacks. Reality, however, shows that such mechanisms for enforcing security policies are often ignored or not adequately used by security professionals. While the reasons for this vary, they typically are related to how complex it is to define implementable security policies and how difficult enforcement against "privileged users" can become. Much more emphasis needs to be placed on simple, well-defined internal access control mechanisms.
We propose a detection system that involves knowledge about the set of data items that are used during the normal execution of an application during certain defined time frames. This is done for some set (or all) applications that are run against the database, resulting in a three dimensional relation. This information is then used to cluster groups of data items that have low temporal affinity, meaning that they are rarely, if ever, used together during standard operating conditions. This information can be collected by investigating historical data kept in the logs by the database management system. This raw historical usage data is then transformed using three dimensional matrix operations that allow security engineers to better analyze potential misuse from current dynamic database operations. This method has not been proposed before, and it shows promise to reveal previously unknown usage patterns that can indicate illicit behavior. 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 literally 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. DETECTION TECHNIQUES

There are two common approaches in database IDSs to detect intrusions when focusing primarily on audit data and access pattern: misuse-based intrusion detection and anomaly-based intrusion detection.

Misuse-Based Intrusion Detection System/Knowledge Based detection:

Such systems use well-known attack patterns or signatures as the basis for detection. It tries to identify activities matching a signature stored in a database. Whenever a match is found an alarm is triggered.
Advantage:
• False alarm rate is very low and any action if found unclear is not allowed. Hence, their accuracy is very high.
• Easy creation of attack signature databases.
Disadvantage:
• Created attack signatures may not cover all attacks as new attacks are hard to forecast.
• Updation of signature database is difficult.
The creation of the well-known attack patterns or signatures is a tedious, manual process that requires detailed knowledge of each software exploit that is supposed to be captured. Simplistic signatures tend to generate large numbers of false positives and too specific ones cause false negatives. The user activity is compared against a repository of signatures that define characteristics of an intrusion. The purpose of attack signatures is to describe the essential features of attacks. For a signature to be good, signature must be narrow enough to capture precisely the characteristic aspects of exploit, it attempts to address and at the same time, it should be flexible enough to capture variations of the attack. Failing to generate a good signature can cause either large amounts of false positives or false negatives.

Anomaly-Based Intrusion Detection System/Behavior Based Detection:

These systems use user profile as the basis for detection. When it detects any sufficient deviation of the recent activity from the normal profile of activities or expected behavior of user then it considers such an activity as intrusion. Then, immediately it generates an alarm to take appropriate action.

Advantages:

• It can detect attempts that try to exploit new and unexpected vulnerabilities.

• Anomalies are recognized without getting inside the causes and characteristics.

• Ability to detect abuse of user privileges.

Disadvantages:

• It may have very high false alarm rate.

• The broad knowledge of expected user behavior is required which makes it difficult to implement.

• User behaviors can vary with time, thereby requiring a constant update of the normal behavior profile database.

But intrusive activity does not always coincide with anomalous activity. There are four possibilities, each with a non-zero probability:

1. *False Negatives:* These are intrusive but not anomalous. That is, the activity is intrusive but because it is not anomalous we fail to detect it. These are called false negatives because the intrusion detection system falsely reports absence of intrusions.
2. *False Positives:* These are not intrusive but anomalous. That is, the activity is not intrusive, but because it is anomalous, we report it as intrusive. These are called false positives because the intrusion detection system falsely reports intrusions.
3. *True Negatives:* These are not intrusive and not anomalous. The activity is not intrusive and is not reported as intrusive.
4. *True Positives:* These are intrusive and anomalous. The activity is intrusive and is reported as such because it is also anomalous.

III. PROPOSED WORK

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 privileges to execute illicit actions for their personal benefit or to harm the organization by causing loss or corruption of critical data.

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.

One more Intrusion detection system was proposed in which explained how to identify malicious transaction by checking for data dependency. Before any updating of data, it has to be read and after updating it has to be written back. There exists the pre-write set and post write set for a updating. Then identification of a malicious transaction is done by comparing the consistency in the pre-write set and post-write set of the user transactions. The detection model proposed in used to detect the malicious transaction after the transaction has been committed.

Another intrusion detection model was proposed in which detect the malicious transactions before they are committed. It considers a situation when an intruder performs a malicious transaction by transferring some amount from account A to account B without the intervention of the account A's owner. Before the intrusion is detected and repaired the money may be drawn away. This transaction cannot be repaired. It is better to prevent a malicious transaction than detecting a curing it later which is an additional burden. Logs are used in recovery purposes to maintain the ACID properties. Logs are difficult to manage especially when systems of different systems are using. Logs cannot be employed in embedded systems. Since logs are usually stored in buffer if buffer is turned off then the logs are lost. If at all the logs are stored in the secondary storage the time is taken more to fetch and compare with the transaction profile. If a masquerader gets a log, he can know all the sequence of work done by the user.

Considering above drawbacks an Intrusion Prevention System without using logs was proposed. It's a space efficient method of implementation for detecting a malicious transaction since it do not use logs for the detector. We follow the model proposed in our present work for intrusion detection. In sequence in counting bloom filters were used for intrusion detection. In our present work we prove by our experimental evaluation that the same prevention model can work just by using simple counting bloom filters and still it can be made to detect malicious transactions 99.85% times if optimal size of CBF and optimal no. of hashing functions are used.

Bloom Filters

A Bloom filter is a method for representing a set of elements (also called keys) to support membership queries. It was invented by Burton H. Bloom in 1970. An empty Bloom filter is a bit array of *m* bits, all set to 0. There must also be *k* different hash Functions defined, each of which maps or hashes some set element to one of the *m* array positions. In order to add an element, the element is supplied to each of the *k* hash functions to get *k* array positions and the bits at all these positions are set to 1. A simple Bloom Filter with 4 hash functions is shown in Fig. .

ISSN: 2278 – 1323

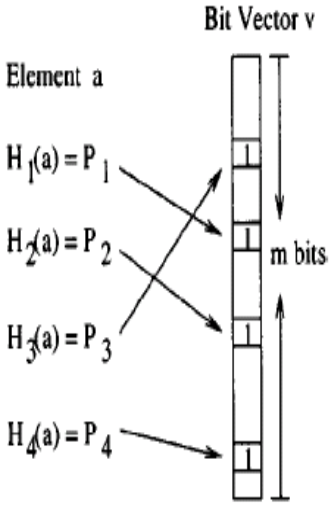

Fig.2 Bloom Filter with four hash functions.

Example: Choose m = 6 (number of bits) and k = 2 (no of hash functions).

Let the hash function taken be
h1(x) = x mod 6
h2(x) = (2x + 3) mod 6

We first initialize the Bloom filter B [1::6], and then insert 9 and 11 in a set A.

h1 (x)	h2(x)	Bloom Filter (B)
Initialize:		0 0 0 0 0 0
Insert 8:		
2	1	0 1 1 0 0 0
Insert 12:		
0	3	1 1 1 1 0 0

Now let us attempt some membership queries:

h1(x)	h2(x)	Answer
Query 14:		
2	1	Yes (Wrong Answer 14 not inserted)
Query 10:		
4	5	No (Correct Answer 10 not inserted)

In this example 8 and 12 are inserted into set A and a corresponding bloom filter B is set for set A. Due to these insertions first four bits of the bloom filter are set. When a person queries for number 14 in set A the answer he gets is yes i.e. the number is present. The reply to the query is yes because hashing functions generates index values 1 and 2 which were set in the bloom filter. So this results in a wrong answer to query. When query is made for number 10 the answer comes correct because index values 4 and 5 which were generated by counting bloom filter for number 10 were not set in the bloom filter.

This example shows that bloom filter has limitation and it does not always give correct answer to queries. This limitation can be reduced to a certain extent by using a variation of bloom filter which is known as counting bloom filter.

Counting Bloom Filters

A Counting Bloom Filter (denoted CBF) is a vector *B* of *m* **counters**. Along with it available to us are *k* (random hash) functions, each of which maps or hashes some set element to one of the *m* 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. To delete an element from the set reverse process is followed and corresponding counters are decremented.
Thus a counting Bloom filter (CBF) generalizes a Bloom filter data structure so as to allow membership queries on a set that can be changing dynamically via insertions and deletions.

CBF	0	1	1	0	1	0	2	0	2	1	0	3

Fig.3.1 shows a Counting Bloom Filter.

Prevention Model

The prevention model is an instant detection model and it prevents the malicious transactions from getting committed. Malicious transactions are major threat to organizations nowadays. Majority of malicious transactions are performed by the people internal to organization. This model shown acts before the database and prevents the transactions from getting committed in database. To build the instant detector we use a counting bloom filter.

This prevention model consists of Client Applications, Network and a database management system equipped with Intrusion Prevention System (IPS). The intrusion detection system in DBMTD mechanism proposed in was acting after the DBMS as shown. Thus it allowed all the transactions to be committed before detecting whether it is malicious or not. Our intrusion prevention system comes in between the network and DBMS as shown. Thus it checks all the transactions before they are committed and prevents the malicious ones from getting committed. So this model will even help us to avoid damage from those malicious transactions which cannot be rolled back. Unlike the DBMTD mechanism which fails to avoid damage in such a situation.

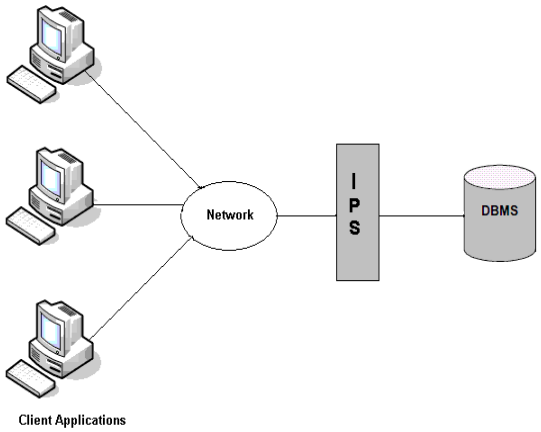

Fig.3.2 Database Management System with Intrusion Prevention System

Intrusion Prevention System
Setting up an intrusion prevention system along with a database management system comprises of three phases namely (1) Profiling the transactions and assigning to users (2) Giving weight to commands and setting the CBF's for transactions and (3) Instant detection and prevention.

- **Profiling the transactions and assigning to users**

Profiling consists of representing the authorized transactions as sequences of valid commands. In this phase the administrator and higher authorities sits together and identifies different transactions that are possible in the system. These transactions are then manually organized into a strict sequence of valid commands. Some typical transaction profiles are shown in fig. 5.Profiling of transactions also prevents attack of intruders to a certain extent because unless the intruder knows the strict sequence of commands he cannot perform a transaction.

- **Giving weight to commands and setting the CBF's for transactions**

In this phase the DBA assigns weights to all commands identified during profiling phase. Weights can be assigned randomly. After this phase each transaction can be viewed as a strict sequence of weights. The list of commands and their corresponding weights is stored. After setting weights of all commands the next job of DBA is to set a counting bloom filter for each transaction. After setting all counting bloom filters they are also stored so that they can be loaded during the instant detection and prevention phase.

- **Instant Detection and Prevention**

In this the user is allowed to enter commands but at any time the user can only execute one transaction. The same user performing transactions from one client machine cannot execute two transactions in parallel. When a user selects a particular transaction like making a payment or submitting a new order, etc IPS loads corresponding CBF and a weight_command list. The weight_command list contains weights and corresponding commands allowed in the system. If user's command is valid then counter values in CBF are decremented using weight of the identified command. The user is allowed to enter all the commands one by one. When user stops entering commands the counting bloom filter is checked. If all the counter values in CBF are zero then the transaction is declared as valid and committed into database otherwise the transaction is treated as a malicious transaction. This procedure is summarized in Instant_Detector() algorithm.

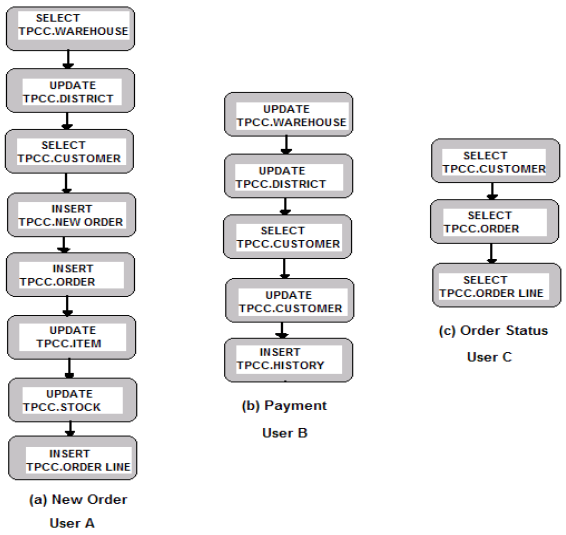

Fig. 3.3 Typical profiles of database transactions

III. COCLUSION

The great demand of web facing application providing faster access to information has made databases more vulnerable. There is a need to prevent such systems from unauthorized access to the database structure, data values and privileges. We would be implementing intrusion detection cum prevention model for database. We have considered some detection and prevention techniques and added response module to present such a model.

This paper has proposed a new mechanism to detect malicious data access. As database systems play a vital role in organization 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. 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.

This work has revealed several areas of improvements and further work. We are working on adding a spatial dimension to our model as the physical location of a user is often an important security metric. The resulting four dimensional matrix requires a reworking of our clustering algorithm and modifications to the distance calculations. As mentioned previously, we plan to develop an automatic method to allow the temporal time frame to be dynamically set so as to show several characterizations of the system. As the usage array is already clustered, we will be able to focus in on certain areas that we know are hotspots for potential attacks, and tailor the system to these dimensions. The clustering is what allows us to have this view of the system.

REFERENCES

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

[2] 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.

[3] 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.

[4] McCormick W., Schweitzer P., White, T. Problem Decomposition and Data Reorganization by a Clustering Technique, Operations Research. 993–1009, 1972.

[5] Navathe, S., Ceri, S., Wiederhold, G., and Dou, J. Vertical partitioning algorithms for database design. *ACM Trans. Database Syst.* 9, pp. 680-710, 1984.

[6] Vieira, M. and Madeira, H. Detection of Malicious Transactions in DBMS. In *Proceedings of the 11th Pacific Rim international Symposium on Dependable Computing* (December 12 - 14, 2005). PRDC. IEEE Computer Society, Washington, DC, 350-357, 2005.

[7] Bertino, E., Kamra, A., Terzi, E., and Vakali, A. 2005. Intrusion Detection in RBACadministered Databases. In *Proceedings of the 21st Annual Computer Security Applications Conference*. ACSAC. IEEE Computer Society, Washington, DC, 170-182, 2005.

[8] Schonlau, M. and Theus, M. Detecting masquerades in intrusion detection based on unpopular commands. In *Information Processing Letters*, vol. 76, 33–38, 2000.

[9] Ramakrishnan, R. and Gehrke, J. *Database Management Systems*. 3rd. McGraw-Hill. 2002.

AUTHOR
Er. Bhavna Sharma
Educational Detail – B.Tech , M.Tech
Department – CSE
University – Kurukshetra University

ISSN: 2278 – 1323