Design and Implementation of Database Intrusion Detection System for Security in Database

Udai Pratap Rao
Dept. of Computer Engineering, NIT Surat,
Gujarat, INDIA-395007

Dhiren R. Patel
Dept. of Computer Engineering, NIT Surat,
Gujarat, INDIA-395007

ABSTRACT
In this paper, we propose database intrusion detection mechanism to enhance the security of DBMS. In a typical database environment, it is possible to define the profile of transactions that each user is allowed to execute. In our approach, we use the transactions profile and overall system architecture is divided into two parts, learning phase and intrusion detection phase. The learning phase generates authorized transactions profile automatically and is used at detection phase to check the behaviour of executable transactions. We also implement the detection phase with the help of Counting Bloom Filter (CBF) and comparing both the approaches.

Keywords
Database Security, Database Auditing, Transaction Profile, Counting Bloom Filter (CBF).

1. INTRODUCTION
Database security research [1] is mainly concerned about the protection of database from unauthorized access. The unauthorized access may be in form of execution of malicious transactions that may breach the security of database and lead to break the integrity over the database. These malicious transactions (database intrusions) are to be taken care of. Researchers have taken interest to develop the database IDS (intrusion detection system) [2] to protect the database from malicious transactions. Security in database [3] is integrated to design and implementation of mechanism for protection of database. The objective of database security approaches is to provide the protection of data stored in the database from malicious action. Usually database security attacks can be classified as external attack and insider attack. In external attack unauthorized attempts are taken place to access or destroy private data and in insider attack the malicious actions are executed by authorized users. The protection of database by the use of encryption techniques, where the database may be encrypted but this kind of system may lead to degradation of query performance. However, in connection with detection of database intrusions few appropriate mechanisms have been proposed in [4, 5]. This paper proposes an innovative mechanism for the detection of database intrusions in DBMS. The proposed approach is also extended to incorporate the CBF. This approach is considered as transaction level approach and is used to detect the malicious transactions in the database.

Rest of this paper is organized as follows. In section 2, we discuss related background. In section 3, system architecture of our proposed database IDS is defined. In section 4, the implementation and results are given. In section 5, the performance evaluation and analysis of the proposed system is given. In section 6, the approach in an addition to the database intrusion detection as Counting Bloom Filter is given with conclusion and references at the end.

2. RELATED WORK
Research work in the field of database intrusion detection has been going on for more than two decades. The approaches used in detecting database intrusions mainly include data mining and traditional security measures. Chung et al. [6] presents a misuse detection system called DEMIDS which is personalized to relational database systems. This approach uses audit logs to derive profiles that describe typical behaviour of the users working with the DBS. The profiles computed can be used to detect misuse behaviour in particular insides abuse. The main drawback of this approach is that it has only been described theoretically, and no empirical evidence has been presented. Lee et al. [7] have proposed a real-time database intrusion detection using time signatures. Real-time database systems have a deal with data that changes its value with time. This intrusion detection model observes the database behavior at the level of sensor transaction. If a transaction attempts to update a temporal data which has already been updated in that period, an alarm is raised. Wenhui et al. [8] proposed a two-layer mechanism to detect intrusions against a web-based database services. However, they have not used different level of granularity or intra-transactional and inter-transactional features in their model. Hu et al. [9] determine the dependency among data items where data dependency refers to the access correlations among data items. These data dependency are generated in the form of association rules. Transactions that do not follow any of the mined data dependency rules are marked as malicious transactions. In this paper equal important are given to the each attributes and there is no concept of attribute sensitivity. The attributes sensitivity problem addresses by Srivastava et al. [10], where some of the attributes are considered more sensitive to malicious modification compared to others. They suggest a weighted data mining algorithm for finding dependencies among sensitive attributes. Any transaction that does not follow these dependency rules is identified as malicious. So in this approach more numbers of rules are generated as compare to the approach presented in [9]. The main problem with this concept is the

identification of proper support and confidence values. Zhong et al. [11] uses query templates to mine user profiles. They developed an elementary transaction level user profile. A constrained query template is a four tuple < op, f, t, c > where op is type of the SQL query, f is the set of attributes, t is the set of tables, and c is the constrained condition set. It uses an algorithm that mines user profile based on the pattern of submitted queries. An algorithm of mining database user query profiles of transaction level is presented. This algorithm changes the computing method of the support and confidence in association rules mining by adding query structure and attribute relations to the computation. Kamra et al. [12] have proposed a role based approach for detecting malicious behaviour in RBAC (role based access control) administered databases. Classification technique is used to work out role profiles of normal user behaviour. An alarm is raised if roles estimated by classifier for given user is different than the actual role of a user. The approach is well suited for databases which employ role based access control mechanism. It also addresses insider threats scenario directly. It does not detect transaction level dependency; hence some of the database attacks may be undetected. The approach presented by Rao et al. [13] is at transaction level and eliminates the problem of [12]. A technique by Lee et al. [14] detects illegitimate database accesses by matching SQL statements against a known set of legitimate database transaction fingerprints. This technique fails again unknown database attacks. In [15], the authors addressed the detection of malicious DBMS transactions with the assumption with transactions profile, and these transactions profile were generated manually. Hence this approach is not effective because manual generating transaction profiles mechanism is more time consuming process.

Here we propose the database IDS which incorporates to generate authorised transactions profile automatically instated of manually and detection phase is also automated to ensure the performance of the system.

3. SYSTEM ARCHITECTURE OF OUR PROPOSED IDS

The proposed system architecture as given in figure 1, wherein database application users pass raw queries through the database IDS Database IDS checks that the incoming online transaction is authorized for that user or not. If the transaction is authorized then database IDS allows the transaction to commit into the DBMS. We have implemented three algorithms: automatic profile generating algorithm works as profile creator, SQL query parsing algorithm works as feature selector and automatic malicious transaction detection algorithm works as detection engine in our proposed system.

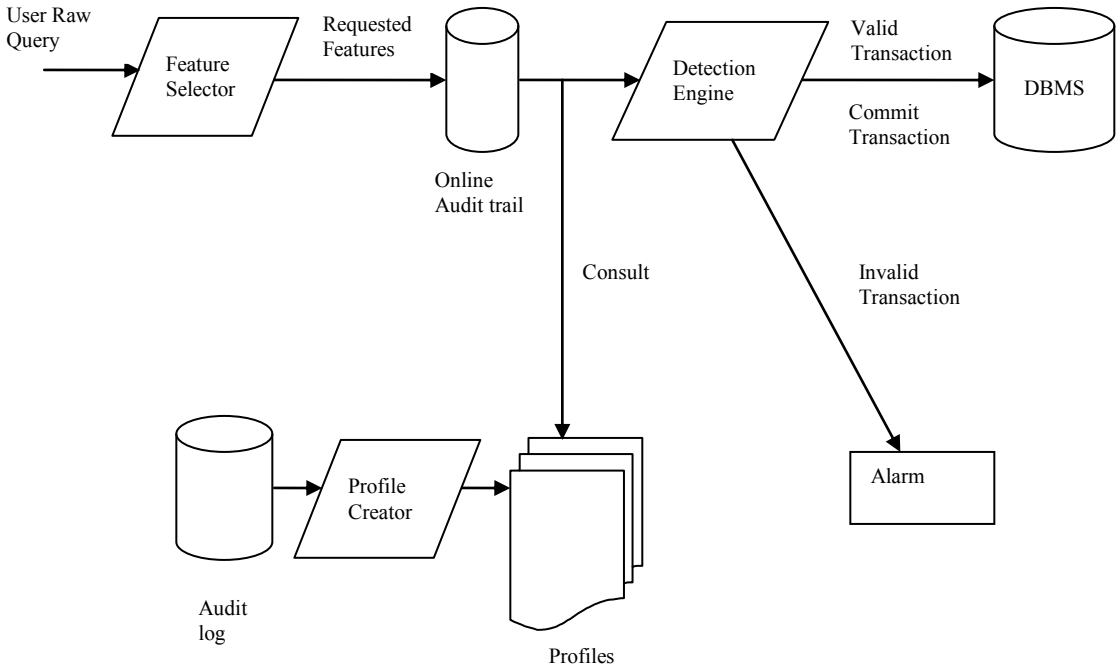

Figure.1: Architecture of Proposed Database IDS

In our proposed system offline audit log contains the data of authorised transactions and these information are extracted from the log file by the help of the DBMS auditing mechanism. The basic history about the log file is that it consists the information about the committed transactions those are executed in the secure environment by the authorised users. Profile creator takes these offline audit log data as input and generates the transactions profile and these transactions profile are considered as authorised profiles and stored at the system, after that these authorised transactions profile are used at the detection phase. Therefore any database application user wants to execute any transaction in DBMS then he/she will submit the raw queries to feature selector. Feature selector extracts the required features like command type, target object from online raw queries submitted by the users and store it in the online audit trail table. Now detection engine first generate the transaction profile for the data stored in online audit trail and this profile is compared with authorised transactions profile. If online transaction profile matches with authorised transaction profile then the detection engine allows the particular executable transaction to commit into the DBMS. If online transaction profile does not match with authorised transactions profile then the detection engine will never allow the particular transaction to commit into the DBMS and marked as a malicious and system raised the alarm.

4. IMPLEMENTATION AND RESULTS

We implemented our proposed approach from learning to the detection phase. The extracted information from the log file are stored in the offline audit table as shown in the table 1 which consists the information like username, transaction number and sequence of commands executed on particular object .The input to the learning phase is given in table 1. The algorithm as given below is used to generate the transactions profile and these profiles are considered as authorised transactions profile. The produced output as authorised transactions profile of learning phase is shown in figure 2.

Transaction Profile Generating Algorithm

Read records from audit table;

Sort audit table by session ID and sequence no;

For each session ID of audit table do
{
while (current session ID contain operations) do
{
print sequence no, command type and target object of operation;
increment the value of sequence no by one;
}
increment the value of session ID by one;
}

Table 1. Offline Audit-log Table (Input to Learning Phase)

User name	Session id	Transaction id	Sequence no	Command type	Target object
Sales	9	1	1	Select	Order
Sales	9	1	2	Select	Product
Sales	10	2	13	Select	Order
Sales	12	12	33	Select	Order
Sales	10	2	14	Delete	Order-line
Sales	9	1	4	Insert	Order-line
Ware	11	3	25	Select	Ware
Sales	12	12	34	Insert	Order
Sales	10	2	15	Delete	Order
Sales	9	1	5	Update	Stock
Sales	12	12	35	Insert	Order-line
Sales	10	2	16	Update	Customer
Ware	11	3	26	Select	Product
Sales	9	1	3	Insert	Order
Ware	11	3	27	Select	Stock
Ware	13	5	21	Select	Ware
Ware	13	5	22	Select	Stock
Sales	14	7	17	select	Order
Sales	14	7	18	select	product
Sales	14	7	19	insert	order
Ware	13	5	23	Update	Stock
Sales	14	7	20	insert	order-line
Sales	12	12	36	Update	Stock
Sales	14	7	21	update	stock
Sales	14	7	22	insert	order-line
Sales	14	7	23	update	stock

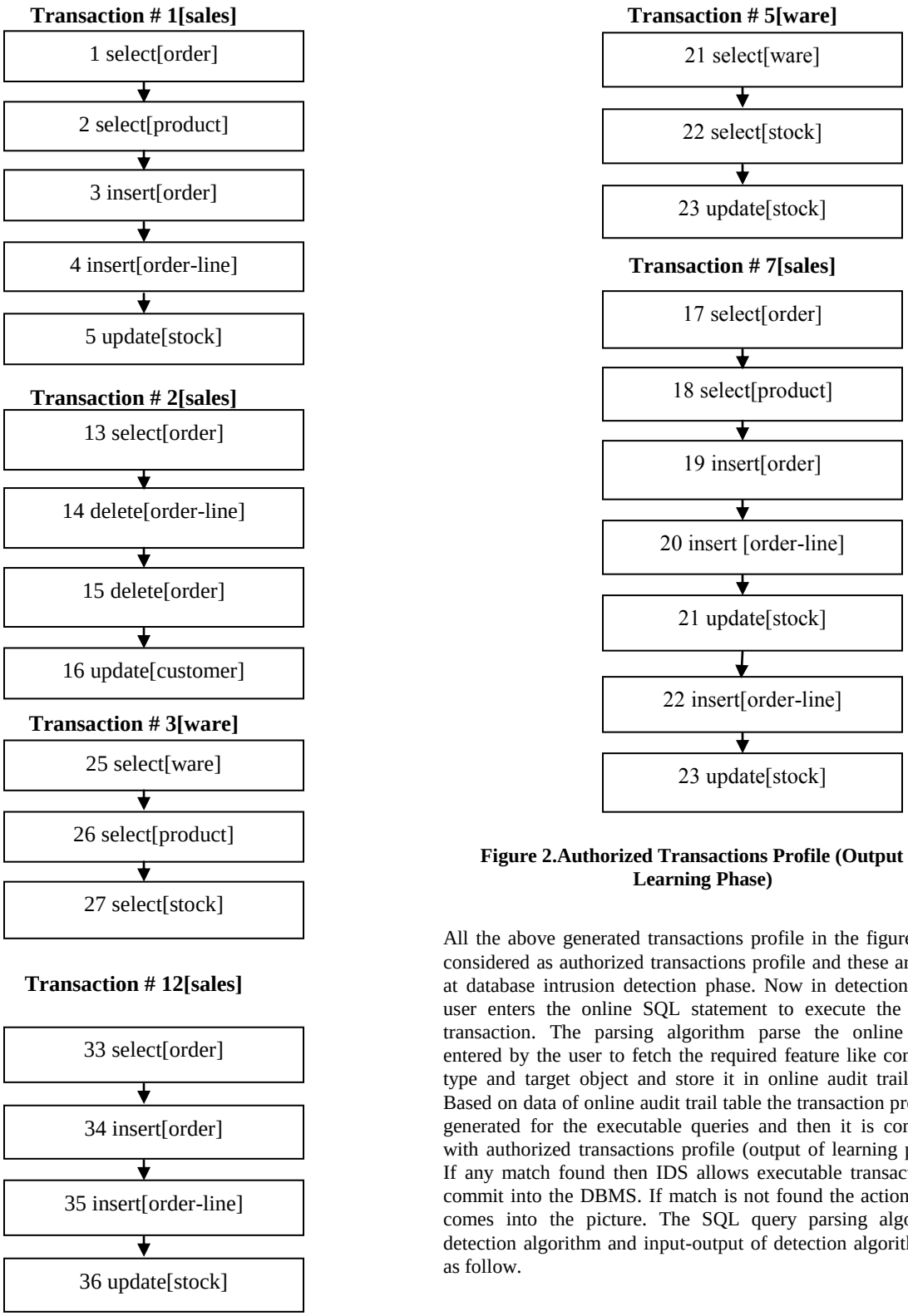

Figure 2.Authorized Transactions Profile (Output of Learning Phase)

All the above generated transactions profile in the figure 2 are considered as authorized transactions profile and these are used at database intrusion detection phase. Now in detection phase user enters the online SQL statement to execute the online transaction. The parsing algorithm parse the online query entered by the user to fetch the required feature like command type and target object and store it in online audit trail table. Based on data of online audit trail table the transaction profile is generated for the executable queries and then it is compared with authorized transactions profile (output of learning phase). If any match found then IDS allows executable transaction to commit into the DBMS. If match is not found the action phase comes into the picture. The SQL query parsing algorithm, detection algorithm and input-output of detection algorithm are as follow.

SQL query Parsing Algorithm

```
Fetch the SQL query as string;
/* declare two variable as string*/

cmd_type = NULL;
tar_obj = NULL;

scan the SQL query word by word separated by space;

if first word is SELECT then
cmd_type = SELECT;
go for to find a word FROM;
/*store the word that is after the word FROM in variable
tar_obj*/
tar_obj = word after FROM word;

else if first word is INSERT then
cmd_type = INSERT;
go for to find a word INTO;
/* store the word that is after the word INTO in variable tar_obj
*/
tar_obj = word after INTO word;

else if first word is DELETE then
cmd_type = DELETE;
go for to find a word FROM;
/*store the word that is after the word FROM in variable
tar_obj*/
tar_obj = word after FROM word;

else if first word is UPDATE then
cmd_type = UPDATE;
/*store the word that is after the word UPDATE in variable
tar_obj*/
tar_obj = word after UPDATE word;
end if

store the value of cmd_type and tar_obj variable in ONLINE
AUDIT Table;
```

Malicious Transaction Detection Algorithm

```
Read the records from ONLINE_AUDIT table;
online_cnt = 0;
while(ONLINE_AUDIT table contain records)
        {
        onlile_cnt = online_cnt + 1;
        }

read the records from OFFLINE_AUDIT table order by
session_ID;
offline_cnt = 0;
for each session_ID of ONLINE_AUDIT table do
        {
        /* calculate no of operation in current session_ID*/
        while(current session_ID contain contain operations)
                {
                offline_cnt = offline_cnt + 1;
                }
        if(online_cnt == offline_cnt)
                {
                while(ONLINE_AUDIT table contain
records)
                        {
compare command type and target object of
ONLINE_AUDIT table with OFFLINE_AUDIT table for
current session_ID;
                        }
                if command type and target object of both
the table are match for current session_ID then
                        {
                        flag = 1;
                        goto  endtran;
                        }
                else
                        flag = 0;
                }

/* increment value of session id*/
        session_ID = session_ID + 1;
        }
endtran:
if (flag = 1) then
        Transaction is valid;
Else
        Malicious transaction;
```

Transactions 1, 2,7and 12 as shown in figure 2 are valid for user sales and transaction 3 and 5 are valid for user ware. Now suppose user sales want to execute any transaction then he will establish a connection with database and he will perform the operations. Suppose he is entering the data as given in the table 2.

Table 2. Online Audit Log Table

User name	Session id	Transaction id	Sequence no	Command type	Target object
Sales	14	7	17	select	Order
Sales	14	7	18	select	product
Sales	14	7	19	insert	order
Sales	14	7	20	insert	order-line
Sales	14	7	21	update	product
Sales	14	7	22	insert	order-line
Sales	14	7	23	update	product

Transaction profile for the above data as in the table 2 is generated and generated profile for the executable SQL statement is given in the Figure 3.

Detection algorithm compares the transaction profile as shown in figure 3 with authorised profiles as shown in figure 2. There are two update operations for transaction no. 7 in figure 2, and

both are on *stock object*, but in case of executable transaction update operations on *product object* and there is deviation between the authorised transaction profile and the executable transaction profile then this current executable transaction is considered as a malicious transaction by detection algorithm.

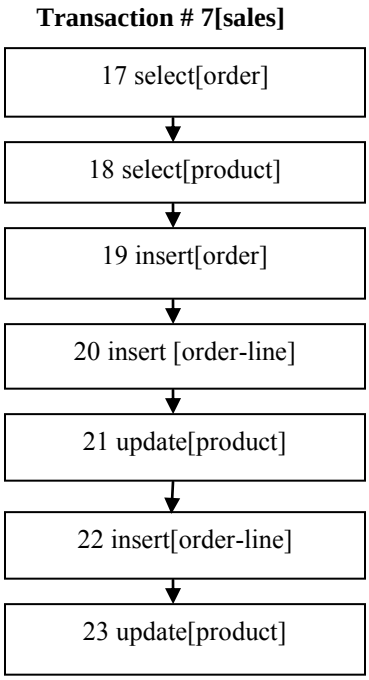

Figure 3. Transaction Profile for Executable Transaction

5. PERFORMANCE EVALUATION AND ANALYSIS

We have evaluated our approach based on the several factors and it gives a better result than earlier proposal [15] of database security mechanism. Experimental results for different criteria are discussed here.

5.1 Performance of Automatic Transaction Profile Generation Algorithm

In [14, 15] authors fetch the required data in audit trail using the auditing mechanism of DBMS. Then they manually analyze this audit trail and manually generate the transactions profile from the data stored in audit trail, the process itself takes more time as compare to automatic transaction profile generating mechanism. We have implemented the *Automatic* transaction profile generating algorithm that takes the offline audit trail data as input and generates the transactions profile for it within milliseconds automatically.

We implemented the automatic transactions profile generating algorithm at different number of transactions in audit trail and proposed algorithm generates the transactions profile for those data within millisecond and we observed that the time taken to generate the transactions profile by our approach is slightly varied with respect to the number of transactions once these are increased. We also observed that the same can be achieved by manual transactions profile mechanism and these may be also useful for the database IDS system, but we believe that this procedure takes the time more as compare to our proposed algorithm. The result graph for no. of transactions profile vs. time taken in millisecond to generate those transactions profile is shown in the figure 4. The time to generate the number of profiles is varied from application to application of database as well as size of the transactions where the size of the transaction is equal to the number of operations in the particular transaction.

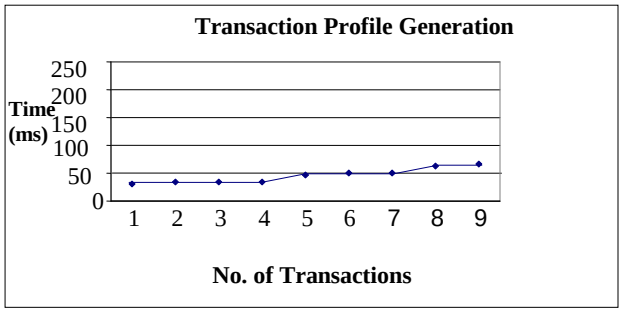

Figure 4. Time for Transactions Profile Generation

We have also measured the performance of the database system/ server with and without the database intrusion detection mechanism. The overall evaluated result is shown in the figure 5. We supplied the large number of transactions to the system /server to monitor the performance of the system. The result itself shows that without inclusion of database intrusion detection mechanism the system/ server performance is quite improved this is expected only because there is no database IDS is installed over the system/server. But for proper monitoring as well as protection of database, the database IDS is required and it is useful for the system when the application lever security mechanism is being compromised, so any transaction must routed with the database IDS and therefore it takes some more time as compare to the other where database IDS is not considered .

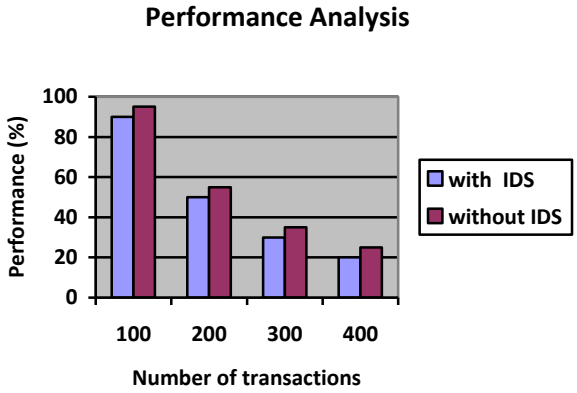

Figure 5. Performance of the System

In case of intrusion detection algorithm the transaction profile of online operations are usually compared with authorized transactions profile manually [14], so it takes more time. Here we have implemented the malicious transaction detection algorithm that compares online transaction profile with stored

authorized transactions profile automatically and it identifies the user behaviours quickly.

5.2 Learning Time

The learning time depends upon the time taken to extract the information from the log file using the DBMS auditing mechanism and storing it into the offline audit table plus time required to generate the authorised transaction profile. Overall learning time depends on two points.

1. We assume that valid users have been allowed to commit their transactions into the database and it was done in the secure environment and this before the commencement of the learning phase, and then only it can be ensured that the offline audit table is having the genuine information and supported by the particular application. If n number of transactions are designed to support by the particular application then in the secure environment the learning phase may take the time to generate the profiles as t time but in case where if we do not consider the execution of the transaction into the secure environment the learning algorithm may take time as t + t1 (where t1 is some additional time) because learning phase may be revisited.
2. If design of the application is changed the learning phase is revisited as we can say repetition of learning to learn new transactions from the database. So in this case learning time is fully depends upon the time required to learn the new more transaction as authorised.

5.3 Accuracy

In [15] author defines the transactions profile as sequence of commands executed for particular action. For example, the transaction withdraw should have command sequence like select-delete-update. Here object dependency is not checked so anyone can update any important object. So it is not accurate. In our proposal we define the transaction profile as sequence of commands executed on particular object. For example, in our approach transaction withdraw should have command sequence like select(usr_balance)-delete(master)-update(usr_balance). Now if any user executes the transaction withdraw then there sequence of commands on particular object must be same as authorized sequence. If users update the data only in those objects to which they are authorized then only the online entered transaction is considered as a valid transaction. In our proposed approach we are also checking the object dependency.

5.4 Coverage

The coverage is calculated by considering the percentage of malicious transactions are detected by the proposed database IDS. To measure the coverage of our proposed mechanism we have submitted random transactions to the system. One hundred and ten random (malicious) transactions have been submitted, corresponding to the execution of 730 SQL commands. From the submitted malicious transactions our proposed approach has detected all 110 transactions, resulting in coverage of 100%.

5.5 False Positive and False Negative

False positive and false negatives are important to evaluate the system performance. False negatives are riskier than false positives because say for example, if there wasn't an attack and the IDS picked it (false positive), its not much of a harm for the database but if there was an attack and the IDS doesn't detect it (false negative), then it can be harmful for the database and leads to break the consistency over the database.

In our proposed mechanism the existence of false positives depends on how complete the definition of authorized transactions are defined by the database developer and how these transactions are generated by automatic transaction profile mechanism. It also depends upon the dependency of the objects for that actions are performed. If the DBA define the set of valid transactions in a fully complete way by considering all possible constraints then the number of false positives will be zero. We have also evaluated the case of false negative and observed that there is no chance of this. Our proposed approach checks the transactions properly and if the transaction is not tally with the stored authorised transaction then always it is blocked.

6. COUNTING BLOOM FILTER (CBF) BASED APPROACH

6.1 Background

A Bloom filter [16] 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 $h_1, ..., h_k$ with range {1, . . . , n} that independently map each element in the universe to a random number uniformly over the range. The main problem with the bloom filter is the false positives. It gives the wrong answer with correct query, and this problem is resolved by the use of counting bloom filter (CBF) where insertion and deletion of set of elements are possible. Similar to the bloom filter, it uses *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. 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 by allowing the membership queries and CBF can be changed dynamically by insertions and deletions operations. It resolves the problem of a standard bloom filter with false positives.

6.2 Database Intrusions Detection with Counting Bloom Filter (CBF)

The overall approach based on the CBF is divided into the three phases.

6.2.1 Initial Phase

It is similar to the automatic transaction profile generation algorithm to generate the authorised transactions as supported by the database application. This process insures the correctness of the genuine profiles as declared authorized profiles. This algorithm works automatically instead of manually thus it reduces the time required for manual transaction profile generation.

6.2.2 Construction of CBF

In counting bloom filter (CBF), random weights are assigned automatically corresponding to commands of authorized transactions profile those are generated by the automatic transaction profile mechanism. Transaction is viewed into a strict sequence of weights with respect to the sequence of commands of particular transaction and considered as authorized profile. The list of commands of the particular transaction and their corresponding weights are stored over the system. After assigning the weights to the commands of the transaction the construction of the CBF is done by incorporating the hash functions and the constructed CBF's are stored so that they can be loaded during the detection phase and detection is done automatically once any executable transaction comes.

6.2.3 Detection Phase

At the detection phase the executable transactions are considered and validated by the database IDS system to ensure that the particular transaction is valid or not .It is done by using the constructed counting bloom filter (CBF) as discussed in the above phase. For a particular selected transaction by the user the detection phase automatically load the corresponding CBF and weights of commands of the transaction. The list contains weights and corresponding commands allowed in the system. If user's commands over the transaction are valid the counter values in CBF are decremented using weight of identified command, if all the bits in the CBF are zero then the transaction is declared as valid.

6.2.4 Comparison of both approaches

Both approaches presented in this paper are compared to know the effectiveness of the system performance. Based on the implementation of the both approaches with respect to the number of transactions we got important results shown in the Figure 6. This figure shows the result for system performance considering three cases: 1) system performance without both approachs 2) system performance with automatic transactions profile approach and 3) system performance with CBF based approach.

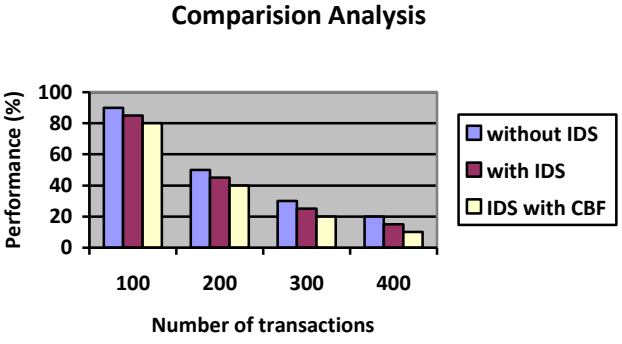

Figure 6. Comparison of Both Approaches

Results are based on the supplied large number of the transactions and observed that the performance of the system in case of without both approach is high but it does not guarantee to secure the database completely. We observed that the performance of the system is quit high in case of first approach as compare to the CBF based approach. However CBF based approach is better even the system performance is low because it guarantees about the complete protection of the false positive cases. The performance of the system in the CBF is low because of the construction of CBF. At the time of detection constructed CBF and weights of the operations of the particular transaction is loaded and deletion of the bits are done over the constructed CBF. So the insertion and deletion of the bits over the CBF may take some time. Another important thing is to incorporate the number of hash function, inclusion of hash function in the numbers may ensure the security over the database but it may lead to degrading of system performance.

7. CONCLUSION

We have proposed two approaches to detect the intrusions in database. They provide an additional layer of security in DBMS. It can be considered as generic approach for any database and overcomes the limitation of the exiting database security mechanisms. We are extending our work with the help of CBF to ensure the security in database. Our comparisons show that it performs better.

8. REFERENCES

[1] Fonseca, Vieira, M., Madeira, H, "Integrated intrusion detection in database," In Bondavalli, A., Brasileiro, F, Rajsbaum S.(eds), LADC 2007. LNCS, vol 4746, pp 198-211. Springer, Heidelberg , 2007.

[2] Jinfu Chen, Yasheng Lu, and Xiaodong Xie, "An Auto-generating Approach of Transactions Profile Graph in Detection of Malicious Transaction" , in Proceedings of Third International Conference on International Information Hiding and Multimedia Signal Processing, pp. 562-565, IEEE, 2007.

[3] E. Bertino, S. Jajodia, and P. Samarati, "Database security:Research and practice", Information Systems Journal, Volume 20, Number 7, 1995.

[4] Jose Fonseca, Marco Vieira, "Monitoring Database Application Behaviour for Intrusion Detection", PRDC '06 , pp. 383- 386, IEEE 2006.

[5] Jose Fonseca, Marco Vieira, and Henrique Madeira, "Detecting Malicious SQL" , C.lambrinoudakis, G.Pernul, A M. Tjoa(Eds.): Trusbus 2007, LNCS 4657, pp. 259-268, Springer Heidelberg, 2007.

[6] C. Y. chung, M. Gertz, K. Levitt, "DEMIDS: A Misuse Detection System for Database systems", IFIP TC-11 WG 11.5 Conference on integrity and internal control in information system, PP. 159-178, 1999.

[7] V. C. S. Lee, J.A. Stankovic, S. H. Son, "intrusion detection in real-time database system Via time signatures", real time technology and application symposium, PP. 124, 2000.

[8] Wenhui S., Tan T., "A novel intrusion detection system model for securing web based database systems", In Proceedings of the 25th annual international computer software and application conference (COMPSAC), pp. 249-254, 2001.

[9] Y. Hu, B. Panda, "A data mining approach for database intrusion detection", In Proceedings of the ACM Symposium on applied computing, pp. 711-716, 2004.

[10] Srivastava, A., Sural, S., Majumdar, A. K., "Weighted intra-transactions rule mining for database intrusion detection", In Proceedings of the Pacific-Asia knowledge discovery and data mining (PAKDD), lecture notes in artificial intelligence, Springer. Pp. 611-620, 2006.

[11] Zhong Y., Qin X., "Database intrusion detection based on user query frequent itemsets mining with constraints", In Proceeding of the 3rd international conference on information security, pp. 224-225, 2004.

[12] Bertino E., Terzi E., Kamra A., Vakali A., "Intrusion Detection in RBAC-Administered Database", In Proceeding of the 21st annual computer security application conference (ACSAC), pp. 170-182, 2005.

[13] Udai Pratap Rao, G. J. Sahani, Dhiren R. Patel, "Detection of Malicious Activity in Role Based Access Control (RBAC) Enabled Databases", International Journal of Information Assurance and Security, pp. 611-617, Volume 5, Issue 6, USA, ISSN 1554-1010,2010.

[14] Lee S.Y., Low W.L, Teoh P., "DIDAFIT: Detecting Intrusions in Database Through Fingerprinting Transactions", in proceedings of the 4th International Conference on Enterprise Information system(ICEIS) 2002, pp. 121-128.

[15] Marco Vieira, Henrique Madeira, "Detection of Malicious Transactions in DBMS", Dependable Computing, 2005. Proceedings. 11th Pacific Rim International Symposium on 12-14 Dec. 2005.

[16] Flavio Bonomi, Michael Mitzenmacher, Rina Panigrahy, Sushil Singh and George Varghese1, "An Improved Construction for Counting Bloom Filters" , ESA 2006, LNCS 4168, pp. 684–695, 2006.

9. AUTHORS PROFILE

Udai Pratap Rao received the B.E. degree in Computer Science and Engineering in 2002 & M.Tech degree in Computer Science and Engineering in 2006, and currently working as Assistant Professor in the Department of Computer Engineering at S. V. National Institute of Technology Surat (Gujarat)-INDIA. His research interests include Data Mining, Database security, Information Security, and distributed systems.

Dr. Dhiren R. Patel is currently a Professor of Computer Engineering at NIT Surat, India. He carries 20 years of experience in Academics, Research & Development and Secure ICT Infrastructure Design. His research interests cover Security and Encryption Systems, Web Services & Programming, SOA and Cloud Computing, Digital Identity Management, e-Voting, Advanced Computer Architecture etc. Besides numerous journal and conference articles, Prof. Dhiren has authored a book "Information Security: Theory & Practice" published by Prentice Hall of India (PHI) in 2008. He is actively involved in Indo-US security research collaborations.