2017 International Conference on Innovations in information Embedded and Communication Systems (ICIIECS)

Detection of Malicious Transactions in Databases Using Dynamic Sensitivity and Weighted Rule Mining

Indu Singh
Department of Computer Science and Engineering
Delhi Technological University
New Delhi, India
indu.singh.dtu14@gmail.com

Shubhi Sareen, Himanshu Ahuja
Computer Science and Engineering
Delhi Technological University
New Delhi, India
shubhisareen@gmail.com, himanshuahujadtu@gmail.com

***Abstract*—The development of an Intrusion Detection System for Database Security has grown into an undeniable necessity in the past few years. In order to maintain scalability and dynamicity of database systems, detecting privilege abuse is of foremost importance. Most researchers have presented either a static or database dependent solution to the existing problem of Insider Attacks. Our Approach, Dynamic Sensitivity Driven Rule Generation Algorithm (DSDRGA) detects intrusive transactions and therefore safeguards crucial data from modification. Sensitivity of data attributes and the data dependency rules are dynamically determined by mining frequent sequences from normal user data access patterns retrieved from the audit log. The extent of weighted similarity between the mined rules and the incoming transactions is used to classify the transaction as malicious or normal user behavior. DSDRGA gives promising results in the detection of malicious transactions when evaluated on a characteristic banking dataset.**

***Index Terms*—Database Intrusion Detection; Dynamic Sensitivity; Weighted Association Rule Mining; Role Based Access Control**

I. Introduction

In recent times, the exponential expansion in the usage of networked systems has exposed the databases to a large number of threats, and therefore it is of utmost importance to safeguard the databases against malicious attacks. In addition to the threat of external attacks, the databases are also subjected to internal attacks, which are a result of internal users abusing their access privileges. Intrusion Detection Systems (IDS) [1] are employed to detect such security breaches in a database and interpret inconsistencies in user-access patterns to identify unauthorized attempts to access databases by individuals working in the organization.

IDS are classified into signature-based and non-signature based intrusion detection systems. The signature based IDS detects attacks by comparing the input with known patterns of attacks [2], it therefore fails to detect previously unknown threats. Moreover, it is inefficient to record the information about signatures of all types of attacks. Such systems require regular updates, and therefore are unadaptable to unprecedented attacks on the system. On the other hand, the non-signature based intrusion detection systems [3] compare the input transaction with normal user transaction patterns, and evaluate the corresponding deviation to classify the input as a normal or malicious activity. Such systems prove to be effective in tackling with insider threats.

Through this paper, we aim to detect and prevent malicious transactions in the database. The NIST model [4] standardized the Role Base Access Control in RDBMS by assigning each user a specified role, and therefore adding a layer of security in the database. The Role Based Access limits the user to a domain of attributes hence restricting the user to access the operations out of its scope. A large number of IDSs are designed to tackle external attacks, but in this paper, we present a novel approach Dynamic Sensitivity Driven Rule Generation Algorithm (DSDRGA) to detect and prevent insider attacks as well.

In our approach, we dynamically determine the sensitivity of attributes and the data dependency rules by mining frequent sequences [5] from normal user data access patterns from the audit log. The incoming transactions, i.e. the new transactions are checked against the normal user access patterns with the help of our proposed detection algorithm, and an alarm is generated if the transaction is classified as malicious. This prevents the modification of sensitive database information. The main advantage of our approach lies in its ability to dynamically allocate sensitivity to operations, making it flexible towards any kind of database.

The rest of the paper is organized as follows. Section II describes the related work in the field of database security and IDS. In Section III, the algorithm and architecture of the proposed approach is illustrated. Section IV presents the experimental results of our approach. Section V draws the conclusion of the paper.

II. RELATED WORK

Numerous researchers are currently working in the field of Network Intrusion Detection Systems, but only a few have proposed research work in Database IDSs. Several systems for Intrusion Detection in operating systems and networks have been developed, however they are not adequate in protecting the database from intruders. ID system in databases work at query level, transaction level and user (role) level. Bertino et. al in [1] described the challenges to ensure data confidentiality, integrity and availability and the need of database security wherein the need of database IDSs to tackle insider threats was discussed.

Database IDSs include Temporal Analysis of queries and Data dependencies among attributes, queries and transaction. Lee et al. [6] proposed a Temporal Analysis based intrusion detection method which incorporated time signatures and recorded update gap of temporal attributes. Any anomaly in update pattern of the attribute was reported as an intrusion in the proposed approach. The breakthrough introduction to association rule mining by Aggarwal et al. [7] helped in finding data dependencies among data attributes, which was incorporated in the field of intrusion detection in Databases.

During the initial development of data dependency association rule mining, DEMIDS [8], a misuse detection system for relational database systems was proposed by Chung et al. Profiles which specified user access pattern were derived from the audit log and Distance Metrics were further applied for recognizing data items These were used together in order to represent the expanse of users. But once the number of users for a single system becomes substantial, maintaining profiles becomes a redundant procedure. Another flaw being the system assuming domain information about a given schema.

Hu et al. [9] presented a data mining based intrusion detection system, which used the static analysis of database audit log to mine dependencies among attributes at transaction level and represented those dependencies as sets of reading and writing operations on each data item. In another approach proposed by Hu et al. [10], techniques of sequential pattern mining have been applied on the training log, in order to identify frequent sequences at the transaction level. This approach helped in identifying a group of malicious transactions, which individually complied with the user behavior. The approach in [10] was improved in [11] by Hu et. al by clustering legitimate user transaction into user tasks for discovery of inter-transaction data dependencies.

The method proposed in [13] extends the approach in [9] by assigning weights to all the operations on data attributes. The transactions which didn't follow the data dependencies were marked as malicious. The major disadvantage of user assigned weights is the fact that they are static and unrelated to other data attributes. Kamra et. al [14] employed a clustering technique on an RBAC model to form profiles based on attribute access which represented normal user behavior. An alarm is raised when anomalous behavior of that role profile is observed.

Y. Yu et. al.[15] illustrated a fuzzy logic based Anomaly Intrusion Detection System. A Naïve Bayes Classifier is used to classify an input event as normal or anomalous. The basis of classifier is formed by the independent frequency of each system call from a process in normal conditions. The ratio of the probability of a sequence from a process and the probability not from the process serves as the input of a fuzzy system for the classification.

A hybrid approach was described by Doroudian et. al [16] to identify intrusion at both transaction and inter-transaction level. At the transaction level, a set of predefined expected transactions were specified to the system and a sequential rule mining algorithm was applied at the inter transaction level to find dependencies between the identified transactions. The drawback of such a system is that sequences with frequencies lower than the threshold value are neglected. Therefore, the infrequent sequences were completely overlooked by the system, irrespective of their importance. As a result, the True Positive Rate falls down for the system.

The above drawback was overcome by Sohrabi et. al [17] who proposed a novel approach ODARDM, in which rules were formulated for lower frequency item sets, as well. These rules were extracted using leverage as the rule value measure, which minimized the interesting data dependencies. As a result, True Positive Rate increased while the False Positive Rate decreased. In recent developments, Ranao et. al[18] presented a Query Access detection approach through Principal Component Analysis and Random Forest to reduce data dimensionality and produce only relevant and uncorrelated data. As the dimensionality is reduced, both, the system performance and True Positive rate increases.

III. PROPOSED APPROACH

We propose an Intrusion Detection mechanism, Dynamic Sensitivity Driven Rule Generation Algorithm (DSDRGA) that uses rule generation in the learning phase and then detects malicious queries in the detection phase. Our approach is novel in comparison to previously proposed systems, as the proposed algorithm DSDRGA is database flexible. It dynamically allocates sensitivity to the operations, therefore rendering the learning phase self-reliant and self-sufficient.

Our IDS is composed of 2 parts, the learning phase and the detection phase. In the learning phase, extracted transactions are pre-processed into read and write sequences. The initial part of the learning phase dynamically determines the sensitivity of all the read and write operations. After calculating the sensitivity, frequent sequences are generated using an improvised version of WSPAN algorithm [5]. Further, read and write rules are generated from these frequent sequences. It is assumed that the provided transaction and database logs don't contain any malicious transactions.

In the detection phase, incoming SQL Queries are parsed and compared with the previously determined rules. The extent of similarity between these two sets is determined by calculating the harmonic mean between Similarity Index and Overlap Coefficient. We have defined the Similarity Index and Overlap Coefficient, in order to incorporate the sensitivity of attributes into account. Therefore, these two measures collectively identify whether a transaction is normal or malicious.

A. *System Architecture*

The proposed IDS work independently without obstructing the functioning of the database management system (DBMS). Malicious Transaction submitted by an attacker, who successfully bypasses the initial security system is recognized and terminated by the proposed system.

The architecture of the proposed system consists of two segments, the Learning Phase and the Detection Phase. In the following section, we define the basic terms, required to understand the architecture and working of the system.

B. *Definitions*

This section includes the definitions of all the terms which are implemented in our approach.

Transaction: A transaction is a set of queries executed by a user. Each transaction is represented by a unique transaction ID. eg: A transaction containing n queries, $T_{\text{ID}} = \langle Q_1, Q_2, Q_3, \dots Q_{\text{n}} \rangle$ where Q_{k} is the k^{th} query of the transaction.

User Role: Each transaction is committed by a user. In our DIDS system each user is assigned a role which is recorded when the user commits a transaction. Each user belongs to a set of pre-defined set of User Roles $R = \{r_1, r_2, \dots r_{\text{m}}\}$.

Operation: An operation is defined as read or write action on an attribute of the relation. $O(A)$ represents an operation on the attribute of the relation where $O(x)$ belongs to the set $\{r(read), w(write)\}$

Sensitive Operations: The operations which have calculated sensitivity strictly greater than 0.5 are considered to be sensitive elements. Rules are generated with respect to sensitive elements.

Highly Sensitive Operations (μ) : The operations which have a calculated sensitivity strictly greater than 0.75 are considered to be highly sensitive elements. The key elements in Detection Phase is a Highly sensitive element.

Read Sequences: The read sequences are of the form $\{R(x_1), R(x_2), R(x_3), \dots O(x)\}$, where $O(x)$ is either a read or write operation. It implies that element x_1, x_2, x_3 etc. are read prior to $O(x)$.

Write Sequences: The write sequences are of the form $\{O(x), W(x_1), W(x_2), R(x_3), \dots\}$ where $O(x)$ is either a read or write operation. It implies that element x_1, x_2, x_3 etc. are updated after the execution of $O(x)$.

Read Rules: Read rules are generated from the Read Sequences. A read rule is defined as $\{R(x_1), R(x_2), R(x_3), \dots\} \to O(x)$, where $x \in \mu$.

Write Rules: Write Rules are generated from the Write Sequences. A write rule is defined as $O(x) \to \{W(x_1), W(x_2), R(x_3), \dots\}$, where $x \in \mu$.

Key Operation (K) : The antecedent in the write rule and the consequent in the read rule is called the Key Operation.

C. *Learning Phase*

In the first step of the learning phase, we extract information from the audit log files of the database which contains data of transactions executed by trusted users. Our system requires the audit log file to record the user role that committed the transaction.

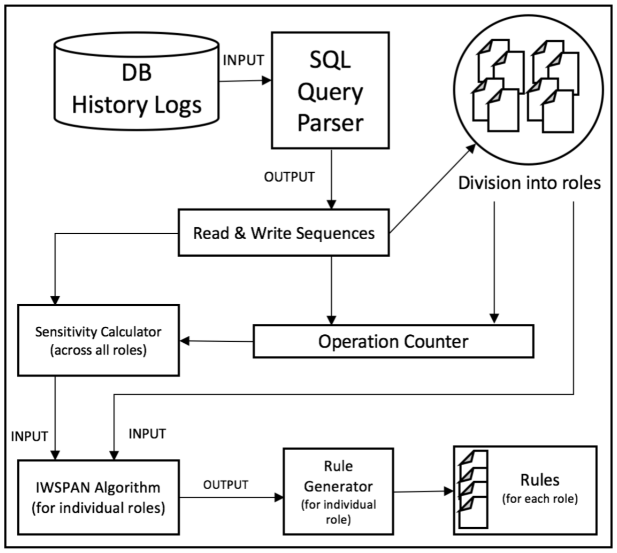

Fig. 1. Architecture of the Learning Phase

*1) **Transaction Pre-processing**:* After the audit log files are extracted they are pre-processed such that the information can be manipulated to find frequent sequences. Using a parsing algorithm each query in a transaction is parsed and converted into a read or write sequence.

The parsing algorithm also assigns each transaction a unique Transaction ID (T_{ID}) and records the user role which committed the transaction. The final output, after parsing the transactions includes three columns: (T_{ID}), the user role which committed the transaction and the read and write sequence generated by the parsing algorithm. The sequences are grouped by user roles and stored into separate files for the next step of the learning phase.

*2) **Sensitivity Calculation**:* After extracting read and write sequences for various user roles , we build an Operation Counter table, which keeps a record of the number of occurrences of each operation, along with the number of user roles which are accessing the operation. Using this table, we further calculate the sensitivity of each operation.

The sensitivity of any operation can at most be 0.9 and at least be 0.1 since no attribute is completely accessible ($\alpha_{O(x)} = 1$) or completely inaccessible ($\alpha_{O(x)} = 0$). Both, the number of occurrences of an operation and the number of rules having access are both individually normalized using Feature Scaling between restricted boundaries of $a = 0.1$ and $b = 0.9$.

$$X' = a + \frac{X - X_{min}}{X_{max} - X_{min}} \times (b - a) \tag{1}$$

We chose the feature scaling normalization method since it incorporates the relative use of different operations and the relative accessibility of the operations by the user roles.In a similar manner, we define $n_{\text{r}}(x)$, the Normalized Value of user roles accessing the operation x and $n_{\text{t}}(x)$, the Normalized Value of the number of occurrences of the operation x. Here N_{rmin} & N_{rmax} is the minimum and maximum number of operations accessed by any user role respectively.

$$n_r(x) = a + \frac{N_r(x) - N_{rmin}}{N_{rmax} - N_{rmin}} \times (b - a) \tag{2}$$

$N_{\text{t-min}}$ & $N_{\text{t-max}}$ is the minimum and maximum number of transactions of any operation.

$$n_t(x) = a + \frac{N_t(x) - N_{tmin}}{N_{tmax} - N_{tmin}} \times (b - a) \tag{3}$$

The advantage of this normalization is that now, both the number of occurrences of an operation and the number of user roles having access will have the same priority in the formula mentioned below.

We calculate the total accessibility ($\alpha_{O(x)}$) of an operation x as the harmonic mean of $n_{\text{t}}(x)$ and $n_{\text{r}}(x)$. Harmonic mean is used against weighted average in calculating ($\alpha_{O(x)}$) as it subtly highlights whether the values of $n_{\text{t}}(x)$ and $n_{\text{r}}(x)$ are at the edges or in the median of the scale. If either of the values is too low, the harmonic mean will reduce considerably.

$$\alpha_{O(x)} = \frac{2 \times n_r(x) \times n_t(x)}{n_r(x) + n_t(x)} \tag{4}$$

Sensitivity (δ) is defined from the accessibility (α) as,

$$\delta_{O(x)} = 1 - \alpha_{O(x)} \tag{5}$$

Each of the operations is assigned with this dynamically calculated sensitivity. The sensitivity of the operations is then fed into the Sequential Pattern Mining algorithm.

*3) **Sequential Pattern**:* Sequential patterns contain those operations which are performed multiple times simultaneously in a particular order. In our approach, we generate frequent sequences for all user roles using the improvised version of Weighted Sequential pattern mining algorithm. This algorithm is executed for each user role with two arguments - (a) The set of pre-processed transactions for that user role and (b) Sensitivity of all operations.

We have set the minimum threshold on support as (φ_{min}). Running the WSPAN [5] algorithm separately for each user role is not computationally expensive as the total number of transactions processed, whether collectively or separately remains the same.

Consider the following transactions as an example:
Suppose, a particular user role has executed the transactions below on the Sample Banking Schema having 5 user roles.

Table	Column Name (Character Encoding)
Customer	Customer_ID(A), Cust_Name(B), Phone (C)
Account	Account_ID(D), Balance(E), Acc_Password(F)
Employee	Employee_ID(G), Employee_Salary(H)

Table 1. Bank Database Schema

update Customer add values (10898292, 'John Kane', 9999999, 'AC10929');
update Employee SET H = H + 3000 where G = 'EP-1978';

Operation	R(A)	R(B)	R(C)	R(D)	R(E)	R(F)	R(G)	R(H)
$n_r(x)$	0.9	0.9	0.9	0.9	0.9	0.58	0.42	0.42
Operation	**W(A)**	**W(B)**	**W(C)**	**W(D)**	**W(E)**	**W(F)**	**W(G)**	**W(H)**
$n_r(x)$	0.42	0.58	0.74	0.74	0.58	0.42	0.42	0.26

Table 2. $n_r(x)$ for various operations

Operation	R(A)	R(B)	R(C)	R(D)	R(E)	R(F)	R(G)	R(H)
$n_t(x)$	0.9	0.449	0.43	0.55	0.23	0.1	0.18	0.105
Operation	**W(A)**	**W(B)**	**W(C)**	**W(D)**	**W(E)**	**W(F)**	**W(G)**	**W(H)**
$n_t(x)$	0.156	0.322	0.32	0.24	0.26	0.11	0.1	0.12

Table 3. $n_t(x)$ for various operations

Each query is parsed and formed into Read and Write Sequence. These read and write sequences are used to calculate $n_r(x)$ and $n_t(x)$. Two tables are formed, one for the normalized value of the number of user roles having access ($n_r(x)$) on an operation (Table 2), and the other containing normalized number of occurrences ($n_t(x)$) in transactions (Table 3).

The operation $W(D)$ can be accessed by four user roles. Using (2), and $b = 0.9$ & $a = 0.1, n_r(W(D)) = 0.74, asinTable2.$ Using equations (4) and (5) and values of $n_r(x)$ and $n_t(x)$ from Table 2 and table 3 respectively, the sensitivity of all the operations was calculated for a sample Banking dataset as follows (Table 4):

Operation	R(A)	R(B)	R(C)	R(D)	R(E)	R(F)	R(G)	R(H)
Sensitivity	0.1	0.426	0.446	0.334	0.684	0.886	0.807	0.893
Operation	W(A)	W(B)	W(C)	W(D)	W(E)	W(F)	W(G)	W(H)
Sensitivity	0.809	0.589	0.589	0.688	0.69	0.89	0.9	0.898

Table 4. Sensitivity of Operations

This sensitivity is used to find out Sensitive elements (δ) and Highly Sensitive elements (μ) and to generate rules.

*4) **Rule Generation**:* After the formation of frequent sequences, data dependency rules are generated from these frequent sequences for each user role. Read and Write rules so generated are added to the set of rules for that user role, if and only if the key operation has sensitivity greater than the minimum sensitivity threshold (φ). Such rules are added to the set of rules for that user role. The following algorithm for generating rules arises from the observation that a set of less sensitive operations should be performed in order to access a high sensitivity operation.

Our rule generation algorithm **Dynamic Sensitivity Driven Rule Generation Algorithm (DSDRGA)** inputs the sequential patterns generated using WSPAN [5] for each user role and outputs the frequent rules of each user role. A minimum threshold for the sensitivity of key operation in a rule is set according to the database. This threshold limits the number of rules generated by the algorithm. The algorithm runs for all extracted frequent sequence for each user role. One sequence may generate more than one rule. A Flagged-Elements Array F is maintained which stores the indices of the elements that have already been used as key operations. On each run the highest sensitivity operation among the un-flagged elements in a sequential pattern is found [line 2]. If the sensitivity of this operation is less than the threshold, the function jumps to the next sequence, else, the operation is set as the Key Operation (K) of the sequence.

From lines 9 - 19, the algorithm checks if the operations preceding K are *read* operations, and add the generated read rule to the Read Rule Set, RR {}. From lines 20 - 30, the algorithm checks if the operations succeeding K are write operations, and add the generated write rule to the Write Rule Set, WR{}. Then the key operation is Flagged and added to

Algorithm 1: DSDRGA Rule Generation

```
Data: $\varphi_{min} = \delta_{threshold}$,  $RR = \phi$,  $WR = \phi$,
$K = Key\ Operation$,  $F = Flagged\ Elements$,
$Stack\ St = \phi$,  $Queue\ Q = \phi$
Result: Set of Read and Write Rules
1  for user role $\Omega \leftarrow 1$ to $n$ do
2      for frequent sequence $\rho$ of $\Omega \leftarrow 1$ to $m$ do
3          F = $\phi$;
4          Function formRule ($\rho_{k,m}$, F)
5              K = $\mu \notin$ F;
6              $\gamma$ = current_element if ($\delta_k \leq \varphi_{min}$) then
7                  break;
8              end
9              while (element before $K \in$ Read_Operation
                 & Sequence Bound) do
10                 if ($\gamma \in$ Write_Operation) then
11                     while (top(St) != -1) do
12                         $RR\{\} \leftarrow$ pop(St);
13                     end
14                 end
15                 if ($\gamma \in F$) then
16                     continue;
17                 end
18                 push (St, $\gamma$);
19             end
20             while (element after $K \in$ Write_Operation &
                 Sequence Bound) do
21                 if ($\gamma \in$ Read_Operation) then
22                     while (front(Q) != rear(Q)) do
23                         $WR\{\} \leftarrow$ deQueue(Q);
24                     end
25                 end
26                 if ($\gamma \in F$) then
27                     continue;
28                 end
29                 enqueue (Q, $\gamma$);
30             end
31             F[ ] $\leftarrow$ K;
32             formRule ( $\rho_{k,m}$, F);
33         end
34     end
35 end
```

the list of Flagged Elements F[] for that sequence [line 31]. In line 32, $formRule()$ executes with the new list of flagged elements on the same frequent sequence.

In the example, W(A) $\in$ μ $(S > 0.75)$ in the first transaction and $W(G)$, $R(H)$ and $W(H)$ $\in$ μ in the second transaction. Hence, these will act as the *key elements (K)*. $W(B), W(C), W(D)$ will act as *Sensitive* elements (δ) $(S > 0.5)$. The sequences will generate the following rules:

W(A)* $\rightarrow$ *W(B), W(C), W(D)

For the first transaction, since, W(A) is the highly sensitive element, it therefore results in a write rule with W(A) as the key element.

$$\textit{R(G)} \rightarrow \textit{R(H), W(H);} \quad \textit{R(H)} \rightarrow \textit{W(H)}$$

For the second transaction, all three elements are highly sensitive elements. However, if we consider W(H) as a key element, it violates the definition of Write Rules. Therefore, the outputs for the query are two read rules with R(G) and R(H) as key elements.

D. Detection Phase

Once the user rules are formed for each role (Ω) using the rule generation algorithm, the system can now be used to detect anomalies in new transactions made by a user. Each query from a new transaction is first pre-processed and converted into rules where the Key Operation is a Highly Sensitive Operation.

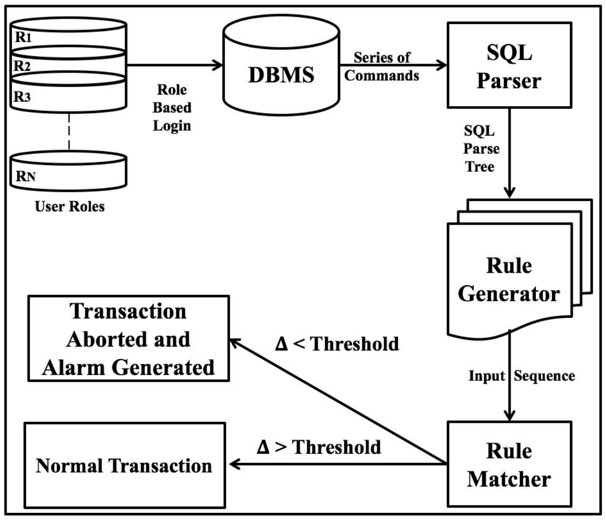

Fig. 2. Architecture for Detection Phase

Then the Detection engine runs for each of the rules for the new transaction and finds the similarity of the rules of the new input query with the rules present in the user roles.

If the key operation of the rules for the new transaction does not match the key element of any rule of that user role, an alarm is raised directly because the user role tries to access a highly sensitive inaccessible element. Otherwise, similarity index and overlap coefficient between the rules of the new transaction and the mined rules of the corresponding user rule is calculated.

The advantage of our definition of Similarity Index(σ) and Overlap Coefficient(θ) over the previously defined Szymkiewicz-Simpson coefficient and SrensenDice index is the fact that it also considers sensitivity of the operations rather than only considering the number of matches.

We use average sensitivity of matched elements and average difference in sensitivity of unmatched elements to calculate Similarity Index(σ) and Overlap Coefficient(θ).

$$x_1 = \sum_{i=1}^{N} \frac{S(X_i)}{N} \tag{6}$$

$$x_2 = \left| \sum_{i=1}^{M_1} \frac{S(X_i)}{M_1} - \sum_{i=1}^{M_2} \frac{S(X_i)}{M_2} \right| \tag{7}$$

$$x_3 = x_1 + x_2 \tag{8}$$

Where, N is the number of operations in antecedent of consequent. $S(X_i)$ is the sensitivity of the i^{th} operation in the rule, M_1 and M_2 are the number of unmatched operation in actual user role and input user role respectively. X_1 is the average sensitivity of matched operations whereas X_2 is the average difference in sensitivity of unmatched elements. σ and θ are defined as follows; here M is the $max(M_1, M_2)$.

$$\sigma = \frac{x_1}{x_3} \qquad \theta = 1 - \left\{ \frac{M}{M+N} \times x_2 \right\} \tag{9}$$

If the value of the average difference in sensitivity of unmatched elements is equal to zero ($X_2 = 0$), the definition of Similarity Index and Overlap Coefficient changes as follows:

$$\sigma = x_1 \qquad \theta = \frac{N}{M+N} \tag{10}$$

After calculating the values of the Similarity Index and Overlap Coefficient, their harmonic mean (Δ) is calculated. The harmonic mean provides a better estimate of total weighted similarity, rather than average value because the value of harmonic mean precisely determines whether the values of σ and θ are at the edges or in the median of the scale.

$$\Delta = \frac{2 \times \sigma \times \theta}{\sigma + \theta} \tag{11}$$

The Value of Weighted similarity is calculated by multiplying Δ with the sensitivity of the key element and is stored in TS (Total Similarity).

$$TS = \delta_K \times \Delta_{max} \tag{12}$$

If the value of TS (Total Similarity) is less than the threshold value, then alarm is raised. Otherwise the transaction is further proceeded.

In Algorithm 2, The value of TS is initialized to 0. In the lines 5 to 11, Detection Phase finds the total similarity between the rules generated for the input query and all the rules of the user's role. The lines 12 to 15 calculates the similarity index, overlap coefficient and harmonic mean to calculate the weighted similarity between the generated rules from the input query and the predefined rules of the user role. From lines 13 - 17, The maximum value of Δ obtained by the algorithm, for various rules of the user role having the same key element is computed. This maximum value of Δ is compared with the value of $R_{threshold}$ and an alarm is raised accordingly, in the lines 18 to 20. If the key element of the generated rule doesn't match any of the predefined rules, an alarm is raised by the lines 22-24.

Algorithm 2: Detection Phase

```
Data: Rules from the new Query using $\varphi_{min}$
K = Key Element, WR = Write Rule, RR = Read Rule
$\sigma$ = Similarity Index, $\theta$ = Overlap Coefficient, TS = Total Similarity
Result: Checks whether the new query is malicious or normal
1  for (user role $\Omega \leftarrow$ 1 to $m$) do
2      if ($K \in \mu$) then
3          for (($\Omega \in$ User Role) $C_i \leftarrow C_1$ to $C_n$) do
4              maxScore = $-\infty$;
5              for ($\Omega$ where ( $K[C_i]$ == $K[Q]$)) do
6                  if (WR) then
7                      match Consequent;
8                  end
9                  if (RR) then
10                     match Antecedent;
11                 end
12                 Calculate $x_1,\ x_2,\ \sigma,\ \theta$ & $\Delta$;
13                 if ($\Delta$ > maxScore) then
14                     maxScore = $\Delta$;
15                 end
16             end
17             TS = (maxScore * $\delta$(K));
18             if ( $TS < R_{Threshold}$) then
19                 Raise Alarm;
20             else
21                 continue;
22             end
23         end
24     else
25         Raise Alarm;
26     end
27 end
```

IV. RESULTS

The performance of the proposed method was analyzed by conducting several experiments on a typical banking database schema adhering to the TPC - C benchmark [12]. The database consisted of two types of logs - one with the normal user transactions of various user roles and other which comprised a mixture of normal and malicious user transactions. These two types of logs in the database were used to evaluate the performance of our system. In total, 12,000 transactions were generated. The database logs comprising malicious transactions were utilized for evaluating the true positives (TP), false negatives (TN), false positives (FP) and true negatives (TN). Using these measures, the recall, precision and accuracy were calculated to evaluate the system.

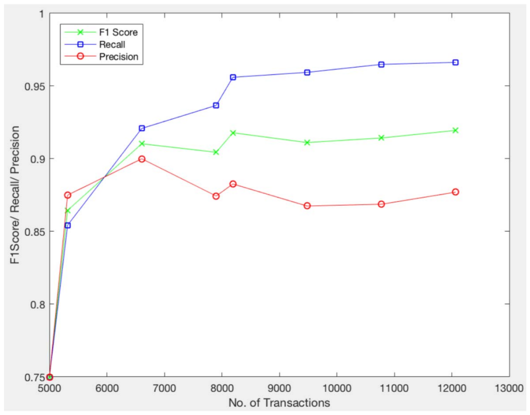

Fig. 3. Variation of F1Score, Precision and Recall with No. of Transactions

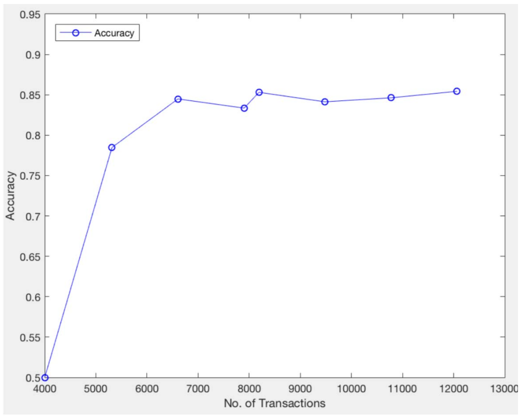

Fig. 4. Variation of Accuracy with Number of Transactions

Figure 3 depicts the variation in precision, recall and F1 Score of the system, with change in the number of transactions in the database. It can be inferred from the graph that the value of Precision first increases from 0.87500 to 090000 and then decreases to 0.87704, on further increase in the number of transactions. The value of Recall increases from 0.85423 to 0.96627, with increase in the number of transactions. With increase in number of transactions, the number of rules generated for each transaction increases. Therefore, this ensures that rules revolving around highly sensitive elements are not overlooked. This results in a decrease in number of false negatives and increase in the number of true positives, therefore leading to an increase in the value of recall. The value of precision initially increases with increase in number of transactions, but further decreases, as the number of false positives increases due to an increase in the number of rules generated. However, the minimum value of precision obtained by DSDRG surpasses the performance of previously proposed systems.

Figure 4 represents the variation in the value of Accuracy with change in number of transactions in the database. It is observed from the graph that the value of accuracy increases from 0.78481 to 0.85431 as the number of transactions increases from 5000 to 12000.

Technique	Reference	Elements Analyzed				Performance
		Command Syntax	Scalability	RBAC	Intrusion Prevention	
Integrated Dependency with Sequence Alignment Analysis	Hu et al. [10], 2004	✓	✓	✗	Partial	Recall = 0.9
	Srivastava et al. [11], 2006	✓	✗	✗	Partial	Precision = 0.8
	Sohrabi et al.[15], 2014	✓	✗	✗	Partial	Recall = 0.65, Precision = 0.77
Inter-transaction Dependency Analysis	Doroudian et al. [14], 2014	✓	✓	✗	Partial	Recall = 0.77, Precision = 0.78
DSDRGA		✓	✓	✓	Yes	Recall = 0.9369, Precision = 0.8779

Table 5. Performance Comparison for the Proposed System

Table 5 represents the difference in the performance of various techniques based on the algorithms used and performance parameters (precision and recall). The maximum value of precision and recall has been recorded for the various systems. It can be inferred from the table, that our algorithm surpasses the alternative methods in term of performance. This can be attributed to the dynamic nature in determining the sensitivity of data attributes and taking it into account in the detection phase as well.

V. Conclusion and Future Work

Through this paper, we presented a novel approach DSDRGA tailored for RBAC-administered Databases, based on data mining techniques in order to detect malicious transactions. The proposed system dynamically determines sensitivity of data attributes and extracts data dependency rules by accessing transactions from the audit log. The experimental results on the synthetic database of 12000 transactions exemplifies the fact that our approach is efficacious in detecting intrusions in a database. Our future work will emphasize on a further detailed formulation of sensitivity of operations, which can enhance the performance of the system.

References

[1] Bertino and R. Sandhu, ”Database Security Concepts, Approaches, and Challenges”, IEEE Transactions on Dependable and Secure Computing, Vol. 2, No. 1, pp. 2-19, 2005.

[2] D.E. Denning, ”An Intrusion-Detection model”, IEEE Transactions on Software Engineering, Vol. SE-13, pp. 222-232, Feb. 1987.

[3] H S Vaccaro and G. E. Liepins, ”Detection of Anomalous Computer Session Activity”, In Proceedings of the 1989 IEEE Symposium on Security and Privacy, pages 280-289, Oakland, California, 1-3 May 1989.

[4] D. Ferraiolo, R. Sandhu, S. Gavrilla, D. Kuhn, R. Chandramouli, ”NIST standard for role-based access control”, ACM Transactions on Information and System Security (TISSEC), Vol 4, Issue 3, 2001.

[5] G. Lan, T. Hong, H. Lee, ”An efficient approach for finding weighted sequential patterns from sequence databases”, Applied Intelligence, vol. 41, no. 2,pp. 439-452, 2014.

[6] V. C. S. Lee, J. A. Stankovic, and S.H. Son, ”Intrusion Detection in Real-time Database Systems Via Time Signatures”, in Proceedings of the 6th IEEE Real Time Technology and Application Symposium (RTAS),pp. 124-133, 2000.

[7] R. Agrawal, T. lmieliiski, and A. Swami, ”Mining Association Rules between Sets of Items in Large Databases”, in Proceedings of the 1993 ACM SIGMOD International Conference on Management of data, 1993.

[8] C. Y. Chung, M. Gertz, and K. Levi , ”DEMIDS: A Misuse Detection System for Database Systems”, in Third Annual Working Conference on Integrity and Internal Control in Information Systems, pp.159-178, 1999.

[9] Y. Hu and B. Panda, ”Identification of Malicious Transactions in Database Systems”, in Proceedings of the International Database Engineering and Applications Symposium (IDEAS ’03), 2003.

[10] Y. Hu and B. Panda, ”A Data Mining Approach for Database Intrusion Detection”, in Proceedings of the ACM Symposium on Applied Computing, pp. 711-716, 2004.

[11] Y. Hu and B. Panda, ”Mining inter-transaction data dependencies for database intrusion detection”, Innovations and Advances in Computer Science and Engineering, Springer, 2010.

[12] TPC-C benchmark: http://www.tpc.org/tpcc/default.asp

[13] A. Srivastava, S. Sural, and A. K. Majumdar, ”Weighted intra-transactional rule mining for database intrusion detection”, in Proceedings of the Pacific-Asia Knowledge Discovery and Data (PAKDD), 2006.

[14] A. Kamra, E. Bertino, and E. Terzi, ”Detecting anomalous access patterns in relational databases”, The International Journal on Very Large Data Bases, Springer, 2008, ISSN:1066-8888.

[15] Y. Yu and H. Wu, ”Anomaly Intrusion Detection Based upon Data Mining Techniques and Fuzzy Logic.” IEEE Conference on Systems, Man and Cybernetics, 2012.

[16] M. Doroudian and H.R. Shahriari, ”A Hybrid Approach for Database Intrusion Detection at Transaction and Inter-Transaction Levels”, 6th Conference on Information and Knowledge Technology (IKT), pp. 1-6, 2014.

[17] Mina Sohrabi, M. M. Javidi, S. Hashemi, ”Detecting intrusion transactions in database systems: a novel approach”, Journal of Intelligent Info Systems 42:619-644 DOI 10.1007 Springer 2014.

[18] C. A. Ranao and S. Chao, ”Anomalous query access detection in RBAC-administered databases with random forest and PCA”, Journal Information Sciences, Volume 369, Issue C, Pages 238-250, 2016.