International Journal of Computer Applications (0975 – 8887)
Volume 169 – No.10, July 2017

A Novel Approach to Database Intrusion Detection

Jay Kant Pratap Singh Yadav
Assistant Professor
Department of CSE,
NIET, Greater Noida

Devottam Gaurav
Assistant Professor
Department of CSE,
NIET, Greater Noida

ABSTRACT
In this paper, we propose data mining approach for database intrusion detection. In each database, there are a few attributes or columns or columns that are more important or sensitive to be tracked or sensed for malicious modifications as compared to the other attributes. Our approach concentrates on mining pre-write as well as post-write data dependencies among the important or sensitive data items in relational database. These dependencies are generated in the form of association rules. Any transaction that does not follow these dependency rules are identified as malicious. We also suggest removal of redundant rules in our proposed algorithm to minimize the number of comparisons during detection phase.

General Terms
Database System, Intrusion Detection System, Sequential Mining.

Keywords
Data Mining, Intrusion Detection System, Data Dependency, Sensitive Attributes.

1. INTRODUCTION
In any critical Information system defending information from illegal access is extremely important [1]. Since database that contain confidential information hence the threat of attacks, the database intrusion detection system are employed for detecting intrusions into Database Management Systems (DBMSs)[2][3]. In the recent past the size of databases has increased swiftly .This led to the concentrating of developing algorithms that are capable of extracting knowledge from huge databases. Data mining has become prevalent than ever as it can determine important but concealed patterns from databases. Data mining is used to find valid, novel, potentially useful and ultimately understandable patterns in data [2][4]. The inherent information within databases, mainly, the interesting association relationship among the items may reveal valuable patterns that may be used for implementation of intrusion detection system. In present scenarios all the planning based on historical database. A huge amount of data to be stored in database, these data is accessed by number of users using web application. So that the number of entry point increases, which make database unsecured. Database consists of many attributes and it is very difficult for administrator to keep track of whether attributes are being accessed only by genuine transaction or not. So by finding data dependencies among the attributes in the database we can easily find out malicious transactions. The techniques employed use of data mining approach to generate data dependency among attributes and these dependencies generated are the form of association rules i.e. before one data item is updated in the database what others data items probably needs to be read and after this data item is update what other data items are likely to updated by same transaction. Transactions that do not follow any of the mined data dependency rules are marked as malicious [5][6]. But it has some limitations as listed below:

Sensitivity or Importance of attribute: In real applications all the attributes are not of equal importance and the attribute of higher importance are usually access less frequently, so it may be possible that there is no rule generated for these attribute therefore these attributes cannot be checked.

Pre-Write Data Dependency rules: There are two types of data dependency rules, namely, read rules and write rules. A read rules take care of attributes that are read in order to update an attribute. Write rules take care of attributes that are updated after updating an attribute. But no one rules take care about attributes that are updated before updating an attribute.

Redundant Data Dependency Rules: Data dependency rules generated are redundant in nature so it takes too much time to detect an malicious transaction because a large number of matching is done with respect to these rules, this degrade the efficiency of the Database intrusion Detection System.

Therefore, if we take care of high sensitive attributes, we assign higher weight to high sensitive attributes so that rules can be generated for sensitive attributes by which administrator check only those alarms generated due to unusual modification of sensitive data attributes without focusing all data attributes [5]. If we take care of pre-write data dependency, we generate write rules for attributes that are updated before updating an attribute. These rules are called pre-write rules. These rules minimize the false positive rate and increase true positive rate. If we remove redundant rules generated during training phase then the overall efficiency of the system to detect malicious transactions will improve a lot. Therefore, our motivation for proposed algorithm is to generate more rules for sensitive attributes by classifying database attributes into different sensitivity groups and assigning suitable weight to these groups, finding the pre-write rules and post-write rules which show the pre-write data dependency as well as post-data dependency relationships among attributes that are updated before updating an attribute and updated after updating an attribute. After that we remove all redundant rules.

The remaining part of the paper is summarized as follows: Section 2 gives the overview of definitions and types of analysis approaches used for IDS. In section 3, description of different data mining algorithms are given which are proposed by different eminent researchers. In section 4, we presented the view of our proposed algorithm for database intrusion detection system. In section 5, a methodology has been provided to test and evaluate our proposed technique. In section 6, we evaluated the performance of our proposed algorithm with a number of cases. Finally, Section 7 gives the view of the conclusion.

2. INTRUSION DETECTION SYSTEM
Along with several benefits the internet has brought it also creates number of ways to compromise the stability of the machines connected to internet. Security management operations protect computers from illegal disclosure of information, and the modification or damage to valuable data.

Defensive operation can be classified into two categories: static and dynamic. Static defensive mechanisms are proposed to provide barriers to the attacks. Keeping advanced operating systems and other software and deploying firewalls at entry points are few examples of static defense solutions. These mechanism are basic line of defense from intrusions and easy to deploy and provide significant improvement to unguarded system. These also act as the foundation to sophisticated defense mechanism [2]. Dynamic defense mechanisms monitor the system to find the evidence of intrusions. These operations aim to catch the attacks and record the information about the incidents such as source and nature of attack. Intrusion detection systems are examples of dynamic defense mechanism [2]. Intrusion detection is the process of tracking of tracking important events occurring in a computer system and analyzing them for possible presence of intrusions. Intrusion detection systems are the software or appliances this automate this monitoring and analysis process. There are number of different ways to classify IDS. Most of the research concentrate either in the field of analysis approach or in placement of IDS.

The analysis approach can broadly classify into two categories [2][5]

- Anomaly Detection
- Misuse Detection

Placement of IDS can be broadly classify into two categories

- Network Based System
- Host Based System

- **Anomaly Detection: -** The anomaly detection model bases its decision on the profile of a user's normal behavior. It analyzes a user's current session and compares it with the profile representing his normal behavior. An alarm is raised if significant deviation is found during the comparison of session data and user's profile. This type of system is well suited for the detection of previously unknown attacks [5]. The main disadvantage is that, it may not be able to describe what the attack is and may sometimes have high false positive rate.
- **Misuse Detection: -** A misuse detection model takes decision based on comparison of user's session or commands with the rule or signature of attacks previously used by attackers. For example, a signature rule for the guessing password attack can be "there are more than 6 failed login attempts within 4 minutes". The main advantage of misuse detection is that it can accurately and efficiently detect occurrence of known attacks. However, these systems are not capable of detecting attacks whose signatures are not available.
- **Network-Based System: -** Its data is mainly collected network generic stream going through network segments. This is generally accomplished by placing the network interface card in promiscuous mode to capture all network traffic that crosses its network segment. Network-based intrusion detection systems (IDS) identify and prevent misuse of network resources by examining packets as they pass sensors on the network. It is an intrusion detection system that tries to detect malicious activity such as denial of service attacks; port scans or even attempts to crack into computers by monitoring network traffic. A NIDS reads all the incoming packets and tries to find suspicious patterns known as signatures or rules [5][6].
- **Host Based System: -** Host based system where the first type of intrusion detection system to be developed and implemented. An IDS which looks for attack signatures in log files is the Host Based Intrusion Detection System. These systems started in 1980 when network were neither complex nor widely used as they are today. In that simpler environment, the easiest and best way to detect malicious action is to audit the logs for anomalous activities. Host Based IDS are installed locally on host machines making it a versatile system compared to Network Based System. Today's Host Based IDSs remain powerful tool for understanding previous attacks and determining proper methods to defeat their future application. Host Based IDS typically monitor system, event and security logs on windows and syslog in Unix environment to detect intrusion. If any of these file changes, the IDS will compare the new log entry with attack signature that are previously stored to find out if there is any match, the system respond with administrator alerts. The main disadvantages of host based systems are performance overhead and maintenance hassle.
- **Database Intrusion Detection System: -** Since database store valuable information of an organization or an application, its security has started getting attention [5]. Database Intrusion Detection System (IDS) is a new database security technology targeted specifically at monitoring and protecting relational databases where most of the organization store crucial data [2]. Database IDS focuses on malicious transaction attacks which cannot be prevented by traditional database security mechanisms. Database intrusion refers to an unauthorized access and misuse of database systems. DIDSs identify suspicious, abnormal or downright malicious accesses to the database system from both of internal and external users. These systems aim to detect intrusions as early as possible, so that any damage caused by the intrusions is minimized. Unfortunately, malicious transactions can seriously corrupt a database through a vulnerability denoted as damage spreading [7].

3. DATA MINING ALGORITHMS

Over the last few years, data mining has attracted a lot of attention due to increased generation, transmission and storage of high volume data and an imminent need for extracting useful information and knowledge from them Han el at. [1]. Data Mining refers to a collection of methods by which large sets of stored data are filtered, transformed, and organized into meaningful information sets Fayyad el at. [4]. It also applies many existing computational techniques from statistics, machine learning and pattern recognition. In recent years, researchers have started looking into the possibility of using data mining techniques in the emerging field of computer security, especially in the challenging problem of intrusion detection. At the first concept of intrusion detection system was suggested by Anderson (1980). He applied statistic method to analyze user's behavior and to detect those attackers who accessed system in an illegal manner. After that he proposed a prototype of intrusion detection expert system in 1987, subsequently, the idea of intrusion detection system was known progressively, and his paper regarded as significant landmark in this area. The author proposed a data mining framework for constructing intrusion detection model. The key idea is to apply data mining programs namely, classification, meta-learning, association rules, and frequent episodes to audit data for computing misuse and anomaly detection models that accurately capture the actual behavior i.e. patterns of intrusion and normal activities.

3.1 Association Rules Mining Algorithm

The basic definition of association rule using support and confidence, defined as follows [2]. Let I= $\{i_1,\ldots,i_m\}$ be a set of literals, called items. Database D= $\{T_1,\ldots,T_n\}$ is a set of transactions, where $T_i \subseteq I(1\leq i\leq m)$, that is going to be released. Each transaction T is an item set such that $T \subseteq I$. A unique identifier TID is associated with each transaction. We say that a transaction T supports X, a set of items in I, if $X\subseteq I$. The association rule is an implication formula like X=>Y, where $X\subset I$, $Y\subset I$ and $X\cap Y=\emptyset$. The item set XUY called generating item set which lead to the generation of an association rule. It consist of two parts: left hand side (LHS) of the arrow (here X) called rule antecedent and right hand side (RHS) of the arrow (here Y) called rule consequent. We say that a rule X=>Y in the database D with confidence C and support S, if $|X \cup Y|/|X| \geq C$ (where |X| is the number of occurrences of item set X in the database D) and $|XUY|/N \geq S$ (N is the number of transaction in D) respectively. Note that support of a rule is a measure of frequency of a rule, where confidence of a rule is a measure of strength of the relation between item sets, C and S are called domain parameters of association rule mining. The number of item sets and association rules increase exponentially with the number of items in the database. But only some of them are interested. So, because of interestingness problem we consider support and confidence higher than two user specified thresholds, MST (minimum support threshold) and MCT (minimum confidence threshold). A detailed overview of association rule mining algorithms and computationally efficient algorithms are presented in Han et al. [1]. The author Agrawal et al. [8] presented an efficient Apriori algorithm for association rule mining.

3.2 Weighted Association Rule Mining Algorithm

Wang et al [9] have proposed a weighted association rule mining technique in which they assign numerical weights to each item to reflect the interest/intensity of the item within the transaction. They first ignore the weight and find the frequent item sets from unweighted data mining and then introduce weight during rule generation. Tao et. al., [10] use the weighted support for discovering the significant item sets. A detailed overview of weighted association rule mining algorithms are presented in F. Tao et al. [10].

4. PROPOSED ALGORITHM

The proposed Algorithm performs the analysis of data dependencies among attributes. It should be noted that, for tracking malicious modifications of higher importance or sensitive attributes, we need to obtain data dependency rules for these attributes. If there is no rule for these attribute, it cannot be checked. Since attributes with high sensitivity are accessed less frequently, so there may not is any rule generated for these attributes. The author Agrawal et al., [8] presented an efficient Apriori algorithm for association rule mining. But it cannot be directly apply in our method. To find out rules for less frequent attributes, we add the concept of weighted association rule mining given by F. Tao et al. [10].In weighted association rule mining some weight is assigned for less frequent but critically important data items and this weight is multiplied with total count of the data items when we calculate the support count for that items. In modified Algorithm we get dependency rules for less frequent attributes. Based on the modification we categorize database attributes into different sensitivity groups and assigning suitable weights to each group, to generate data dependency rules for important but possibly less frequent attributes. Here we also take care to find out pre-write data dependency and post-write data dependency rules which give the relationship between attributes that are required to be updated before and after updating an attribute. After generating read data dependency rules, pre-write data dependency rules and post-write data dependency rules, we prune out redundant rules so that less comparisons are required during detection phase which improve the efficiency (in term of time) of our algorithm.

We divide the work of our proposed algorithm for database intrusion detection into two phases first one is training phase and second one is detection phase.

4.1 Training Phase

Training phase has following three components (Fig.1.):

- Security Sensitive Sequential Pattern Discovery.
- Read, Pre-Write and Post-write Sequence Set Generation.
- Non-Redundant Weighted Data Dependency Rule Generation.

4.2 Security Sensitive Sequential Pattern Discovery

Now, we explain the weighted sequence mining algorithm in this section. We modify the association rule mining algorithm to find out sequential patterns for less frequent and higher sensitive attributes by assigning weight for each attribute.

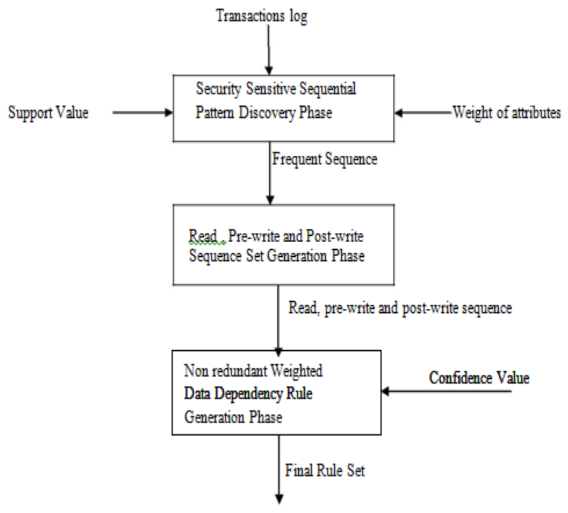

Transactions log
Support Value
Security Sensitive Sequential Pattern Discovery Phase
Weight of attributes
Frequent Sequence
Read, Pre-write and Post-write Sequence Set Generation Phase
Read, pre-write and post-write sequence
Non redundant Weighted Data Dependency Rule Generation Phase
Confidence Value
Final Rule Set

Fig. 1. Component of our proposed algorithm

We categorize the attributes into the following three sensitivity levels—High Sensitivity (HS), Medium Sensitivity (MS) and Low Sensitivity (LS). Also, modification (write) of an attribute of a particular sensitivity level is considered more important than accessing (read) the same attribute, from database integrity point of view. We consider an attribute say x, then $W(_{xw})>W(_{xr})$, where W is a weight function, xw denotes writing or modifying attribute x and xr denotes reading of attribute x. Here we are using the same example as given in [5] for discussion of weighted sequence mining algorithm.

For a given schema, we define six types of operations on the attributes based on the different sensitivity levels and mode of access. Numerical weights are assigned to each operation, which signify their relative order of importance. The six types of operations are: High Sensitive Write (HSW), High Sensitive Read (HSR), Medium Sensitive Write (MSW), Medium Sensitive Read (MSR), Low Sensitive Write (LSW) and Low Sensitive Read (LSR) such that, WHSW>WHSR>WMSW>WMSR>WLSW>WLSR. The weight of a given attrib_ID_seq of a certain transaction is same as the weight of the most sensitive operation applied on the attributes in that sequence. Let us say that there are attributes a_1, a_2, a_3, a_4, a_5, where $a_1, a_3 \in$ HS, $a_2 \in$ MS and $a_4, a_5 \in$ LS, and we have following five sequences:

<a_{1r}, a_{3r}, a_{2w}, a_{4w}>

<a_{1r}, a_{3w}, a_{4w}>

<a_{5r}, a_{2w}, a_{4r}>

<a_{4r}, a_{5w}>

Let 3, 2 and 1 be the weights of HS, MS and LS, respectively and 0.25 be the additional weight of write operation for all HS, MS and LS. In sequence (i) the most sensitive attributes are a_1, a_3, which are in HS. Hence, the weight of this sequence is 3. Sequence (ii) contains the same set of most sensitive attributes as (i) but since in this sequence a_3 is present with write operation, it is assigned a weight of 3 + 0.25. In the third sequence, the most sensitive attribute is a_2, which is in MS and it is with write operation. So, sequence (iii) gets a weight of 2 + 0.25. The last sequence contains sensitive attributes a_4, a_5, which are in LS and a_5 is with write operation. Hence, it gets a weight of 1 + 0.25.

The weights assigned to the sequences, used to calculate the support of each sequence in the transaction, are required in the second pruning step. If the support of any sequence is greater than the minimum support, then the sequence is considered to be a frequent sequence. Let there be a sequence s with weight Ws. Let N be the total number transactions. If s is present in n out of N transactions, then the support of sequence s is:

$$Support(s) = (n*w_s)/N$$

Assignment of weight to each attribute has a significant effect on the sequence mining algorithm. Sequences containing high sensitive attributes but otherwise accessed less frequently in the transactions can potentially become frequent sequences because count of each such sequence is enhanced by multiplying with its weight. The weighted support can now exceed the minimum support, making it a frequent sequence.

Consider the example transactions shown in Table 1. These transactions are generated from the bank database schema shown in Table 2 with attributes encoded into integers. In Table 3, the three sensitivity groups and the weight of each attribute are shown. The transactions and weights form the input for the weighted sequential pattern mining algorithm. The sequences generated from the algorithm are shown in Table 4. In this table, we also show the sequences that would have been generated if we execute existing algorithm given in [5].

Table 1. Example Transactions for the Sequential Pattern Mining Algorithm

Trans ID	Attribute access sequence
1	11r, 13w, 4r, 8r, 2r, 16r, 17r, 14r
2	7r, 2r, 7r, 2r, 14r, 15w
3	16r, 17r, 14r, 14r, 15w, 17w, 2r, 7w
4	11r, 12w, 2r, 4w, 16r, 17r, 14r
5	2r, 4w, 2r, 7w, 7r, 8r, 2r
6	11r, 13w, 4r, 8r, 2r, 2r, 4w
7	14r, 15w, 4r, 8r, 2r, 8r, 2r
8	7r, 8r, 2r, 2r, 2r, 8w, 5w, 2r, 4w
9	8r, 2r, 14r, 15w, 7r, 2r
10	14r, 15w, 16r, 17r, 14r, 14r,1 5w, 17w

Table2. Bank Database Schema

Table Name	Column Name (Integer Encoding of Attributes)
Customer	Name(14), Customer_id(15),Address(12), Phone_no(2)
Account	Account_id(4), Customer_id(17), Status(7), Open_dt(5), Close_dt(16), Balance(8)
Account_type	Account_type(11), Max_tran_per_month(13), Description(18)

Table3. Weight Table for the Attributes Used in the Bank Database

Sensitivity Group	Attribute	Weight	Write Weight
HS	7, 8, 13	3	.25
MS	5, 16	2	.25
LS	2, 4, 11, 12, 14, 15, 17, 18	1	.25

Table 4: Mined Sequence Using Minimum Support Value 25 %

Sequence Using Weighted Method	Sequence Using our Proposed Method
<16r, 17r, 14r, 15w, 17w, 7w>, <2r, 4w, 7w, 7r, 8r>, <7r, 8r, 8w, 2r, 4w>, <7r, 8r, 2r, 8w, 4w>, <8r, 2r, 14r, 15w, 7r>, <8r, 14r, 15w, 7r, 2r>, <11r, 13w, 8r, 2r, 4w>, <11r, 13w, 8r, 2r, 16r>, <13w, 4r, 8r, 2r>, <7w, 7r, 8r, 2r>, <2r, 7r, 14r, 15w>, <7r, 2r, 14r, 15w>, <14r, 15w, 2r, 7w>, <14r, 15w, 2r, 8r>, <14r, 15w, 8r, 2r>, <4r, 2r, 8r>, <13w, 8r, 16r, 17r 14r >, <13w, 8r, 2r, 16r, 14r>	<16r, 17r, 14r, 15w, 17w, 7w>, <2r, 4w, 7w, 7r, 8r>, <7r, 8r, 8w, 2r, 4w>, <7r, 8r, 2r, 8w, 4w>, <8r, 2r, 14r, 15w, 7r>, <8r, 14r, 15w, 7r, 2r>, <11r, 13w, 8r, 2r, 4w>, <11r, 13w, 8r, 2r, 16r>, <13w, 4r, 8r, 2r>, <7w, 7r, 8r, 2r>, <2r, 7r, 14r, 15w>, <7r, 2r, 14r, 15w>, <14r, 15w, 2r, 7w>, <14r, 15w, 2r, 8r>, <14r, 15w, 8r, 2r>, <4r, 2r, 8r>, <13w, 8r, 16r, 17r 14r >, <13w, 8r, 2r, 16r, 14r>

Algorithm for generating frequent mined sequence is as:

```
Input:
A set of Transactions T containing attribute sequences, weight of the attributes, support minSup
Output:
A set L that contains Frequent sequences.
Initialization:
L = {}
Algorithm:
L1← {large 1 - item sets whose support as calculated by eq.1>minSup}
k←2
while($L_{K-1} \neq \varphi$)
{
for each itemsets $l_1 \in L_{K-1}$
{
if(($l_1[2]=l_2[1]$)^($l_1[3]=l_2[2]$^...^($l_1[k-3]=l_2[k-2]$)^($l_1[1] \neq l_2[k-1]$))
{
c=$l_1$ join $l_2$; // join step: generate candidates
}
}
$C_k \leftarrow C_k \cup \{c\}$
}
for transactions t∈T//scan T for counts
{
$Ct \leftarrow \{c|c \in C_k \wedge c \subseteq t\}$for candidates c∈Ct
{
count[c]←count[c]+1
}
}
$L_k \leftarrow \{c|c \in C_k \wedge$Support count (as calculated in Eq.1)>minSup}
k←k+1
}
Return $L \leftarrow \cup_k L_k$
```

4.3 Read, Pre-Write and Post-Write Sequence Set Generation

To understand the next part we introduce some basic terms and also discussing the working of our algorithm.

Definition 1: A sequence is an ordered list of attributes along with read and/or writes operations performed on these attributes. We denote sequence s by $<a_{1o}, a_{2o}, a_{3o}, \ldots\ldots, a_{ko}>$ where a_1 to a_k are attributes and o is an operation that can take values either 'r' for read or 'w' for write operation.

Definition 2: The Read Sequence is denoted by ReadSeq of attribute a_j is a sequence of the form $<a_{1r}, a_{2r}, a_{3r}, \ldots\ldots, a_{kr}, a_{jw}>$, which is the sequence of attributes a_1 to a_k that are read before the attribute a_j written. All such sequences form a set named as read sequence set denoted by ReadSeqSet.

For example, consider the following SQL statement:

Update account set balance = balance+7000 where customerId=23;

In this update query, before updating the balance attribute, customerId and old account balance must be read. So read sequence for updating balance would be $<\text{customerID}_r, \text{balance}_r, \text{balance}_w>$.

The sequences generated in the previous step shown in Table 4 are next used to determine read, Pre-write and post-write sequences. According to the definitions, ReadSeq and PreWriteSeq, PostWriteSeq must contain at least one write operation each. Sequences that do not have any attribute with write operation, are not used for read and pre-write, post-write sequence generation. A sequence that contains a single attribute does not contribute to dependency rule generation. Hence, will be ignored.

For each write operation a_{jw} in a sequence, add $<a_{1r}, a_{2r}, \ldots, a_{kr}, a_{jw}>$ to ReadSeqSet where $a_{1r}, a_{2r}, \ldots, a_{kr}$ are the read operations on attributes a_1 to a_k before the write operation on attribute a_j. For example, sequence $<7_r, 8_r, 8_w, 2_r, 4_w>$ of Table 4 generates the following read sequences $<7_r, 8_r\ 9_w>$, $<7_r\ 8_r\ 2_r, 4_w>$. To generate pre-write sequences, for each write operation a_{jw} in a sequence, add $< a_{1w}, a_{2w}, \ldots, a_{kw}, a_{jw}>$ to PreWriteSeqSet where $a_{1w}, a_{2w}, \ldots$akw are write operations on attributes a_1 to a_k before the write operation on attribute aj. For example, sequence $<16_r, 17_r, 14_r, 15_w, 17_w, 7_w>$ of Table 4 generates the pre-write sequence $<15_w, 17_w>$ and $<15_w, 17_w, 7_w>$. To generate post-write sequences, for each write operation a_{jw} in a sequence, add $<a_{jw}, a_{1w}, a_{2w}, \ldots, a_{kw}>$ to PostWriteSeqSet where $a_{1w}, a_{2w}, \ldots a_{kw}$ are write operations on attributes a_1 to a_k after the write operation on attribute a_j .

Table 5. Read Sequences, Pre-write and Post-write Sequences

Weighted Method		**Our Proposed Method**		
Read Sequence Set	**Write Sequence Set**	**Read Sequence Set**	**Pre-write Sequence set**	**Post write Sequence Set**
< 16r, 17r, 14r, 15w >	< 15w, 17w, 7w>	< 16r, 17r, 14r, 15w>	< 15w, 17w>	< 15w, 17w, 7w>
< 16r, 17r, 14r, 17w >	<17w,7w>	< 16r, 17r, 14r, 17w >	<15w,17w,7w>	<17w,7w>
< 16r, 17r, 14r, 7w >	< 8w, 4w >	< 16r, 17r, 14r, 7w >	< 8w, 4w >	< 8w, 4w >
< 2r, 4w >,< 2r, 7w >	< 15w, 7w >	< 2r, 4w >,< 2r, 7w >	< 15w, 7w >	< 15w, 7w >
< 7r, 8r, 8w>	< 4w, 7w >	< 7r, 8r, 8w>	< 4w, 7w >	< 4w, 7w >
< 7r, 8r, 2r, 4w >	< 13w, 4w >	< 7r, 8r, 2r, 4w >	< 13w, 4w >	< 13w, 4w >
< 8r, 2r, 14r, 15w >		< 8r, 2r, 14r, 15w >		
< 8r, 14r, 15w >		< 8r, 14r, 15w >		
< 11r, 13w >		< 11r, 13w >		
< 11r, 8r, 2r, 4w >		< 11r, 8r, 2r, 4w >		
< 2r, 7r, 14r, 15w >		< 2r, 7r, 14r, 15w >		
< 7r, 2r, 14r, 15w,>		< 7r, 2r, 14r, 15w,>		
< 14r, 15w >		< 14r, 15w >		
< 14r, 2r, 7w >		< 14r, 2r, 7w >		
< 7r, 8r, 2r, 8r >		< 7r, 8r, 2r, 8r >		

For example, sequence $<16_r, 17_r, 14_r, 15_w, 17_w, 7_w>$ of Table 4 generates the post-write sequence $<15_w, 17_w, 7_w>$and $<17_w, 7_w>$.The read, pre-write and post-write sequences generated from the mined sequences of Table 4 are shown in Table 5.

4.4 Non-Redundant Weighted Data dependency Rule Generation

There are three types of data dependency rules, namely, read rules pre-write rules, post-write rules. A read rule of the form $a_{jw} \rightarrow a_{1r}, a_{2r}, \ldots, a_{kr}$ implies that attributes a_1 to a_k are read in order to write attribute a_j. Pre-Write rule of the form $a_{1w}, a_{2w}, \ldots, a_{kw}.\ a_{jw}$ implies that attributes $a_{1w}, a_{2w}, \ldots, a_{kw}$ are modified, before writing attribute a_{jw}. Post-Write rule of the form $a_{jw} \rightarrow a_{1w}, a_{2w}, \ldots, a_{kw}$ implies that After writing attribute a_{jw}, attributes $a_{1w}, a_{2w}, \ldots, a_{kw}$ are modified, These rules are generated from the read, pre-write and post-write sequences. Weighted data dependency rule generation uses weighted confidence. The confidence of the read, pre-write and post write rules are calculated by the following method.

Let R be a read rule of the form $a_{jw} \rightarrow a_{1r}, a_{2r}, \ldots a_{kr}$, generated from the read sequence rs ∈ ReadSeqSet . Let Count(a_{jw}) and Count($_{rs}$) be the total count of the attribute a_{jw} and that of rs among the total number of transactions.

The weighted confidence of the rule R is defined as:

Confidence (CR) = Count (rs)/count(a_{jw})

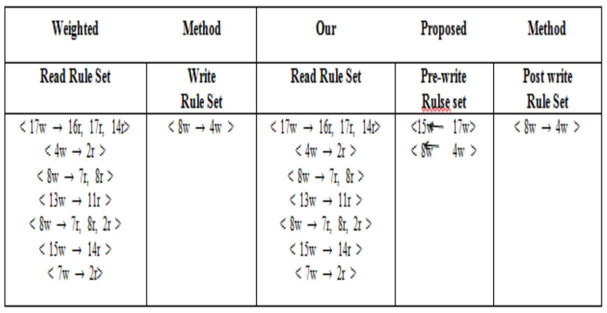
Count(a_{jw}) is defined as follows:

Count(a_{jw}) = $\sum (W_{aj}) + .25) + \sum \max(W(_{rs}))$

∀Transaction T, a_{jw}∈T, rs∈T ∀Transaction T, a_{jw}∈T, rs∈T
Count(rs) is defined as:
Count(rs) =$\sum$ max(W(rs))

The read rules, pre-write rules and post write rules generated from read, pre-write, post-write sequences given in Table 5 shown in Table 6.

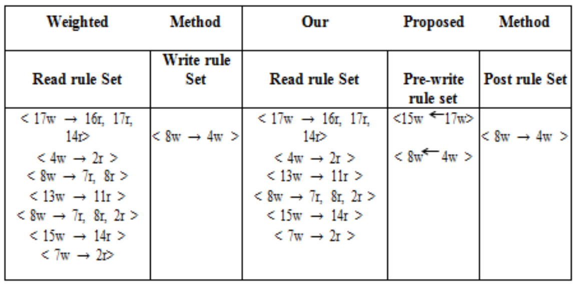
Table 6. Read, Pre-write and Post-write Dependency Rules (Confidence value 70%)



Weighted Method		Our Proposed Method		
Read Rule Set	Write Rule Set	Read Rule Set	Pre-write Rulse set	Post write Rule Set
< 17w → 16r, 17r, 14r> < 4w → 2r > < 8w → 7r, 8r > < 13w → 11r > < 8w → 7r, 8r, 2r > < 15w → 14r > < 7w → 2r>	< 8w → 4w >	< 17w → 16r, 17r, 14r> < 4w → 2r > < 8w → 7r, 8r > < 13w → 11r > < 8w → 7r, 8r, 2r > < 15w → 14r > < 7w → 2r >	<15w← 17w> < 8w← 4w >	< 8w → 4w >

Some of the read rules, pre-write and post-write rules generated in previous step are redundant rules. The redundant rules are pruned out from read rule set, pre-write rule set as well as post- write rule set and shown in Table 7. We consider a rule R to be redundant if it has the same antecedent as similar to another rule R* and R's consequent is a subset of R*'s consequent. Redundant rules are removed to make the procedure for identifying malicious transactions more efficient.

Table 7. Non redundant Read, Pre-write and Post Write Dependency Rules



Weighted Method		Our Proposed Method		
Read rule Set	Write rule Set	Read rule Set	Pre-write rule set	Post rule Set
< 17w → 16r, 17r, 14r> < 4w → 2r > < 8w → 7r, 8r > < 13w → 11r > < 8w → 7r, 8r, 2r > < 15w → 14r > < 7w → 2r>	< 8w → 4w >	< 17w → 16r, 17r, 14r> < 4w → 2r > < 13w → 11r > < 8w → 7r, 8r, 2r > < 15w → 14r > < 7w → 2r >	<15w ←17w> < 8w← 4w >	< 8w → 4w >

At last we find the non-redundant weighted data dependency rule set as shown in Table 8.

Table 8. Non redundant Weighted Data Dependency Rules

Weighted Method	Our Proposed Method
<17w → 16r, 17r, 14r>	<17w → 16r, 17r, 14r>
<4w → 2r>,	<4w → 2r>
<8w → 7r, 8r>	<13w → 11r>
<13w → 11r>	<8w → 7r, 8r, 2r>
<8w → 7r, 8r, 2r>	<15w → 14r>
<15w → 14r>	<7w → 2r>
<7w → 2r>	<15w←17w>
<8w → 4w>	<8w←4w> <8w→4w>

Input:

A set of transactions T containing set of attribute sequences, weight of the attributes, support MinSup and Confidence MinConf.

Output:

A set of rules AnsSet that can be used to detect malicious modifications of the data.

Algorithm:

Initialization:

Initialize two sets ReadRuleSet = {}, PreWriteRuleSet={}, PostWriteSeqSet = {}, for storing read and write rules respectively.

Initialize three sets ReadSeqSet = {}, PreWriteSeqSet = {}, PostWriteSeqSet = {} for storing read, Pre-Write and Post-Write sequences respectively

Create a set data dependency rules AnsSet=

Create a set data dependency rules AnsSet= {ReadRuleSet, PreWriteRuleSet, PostWriteRuleSet}.

Execute sequential mining algorithm as give in fig with minimum support minSup. At each step, calculate support of the sequences using equation (1).

For each sequential pattern Pi, where Pi contains at least one write operation

If (a1r, a2r, ...,akr, ajw∈ Pi and a1r, a2r, ..., akr ≠ Ø) where a1r to akr are all the

read operation on attributes a1 to ak before ajw, the write operation on attribute aj

For each write operation a_{jw}

ReadSeqSet=ReadSeqSet∪{a1r, a2r,....,akr, ajw }

If (a1w, a2w , ...,akw, ajw∈ Pi and a1w, a2w,....,akw ≠ Ø) where a1w to akw are all the

write operation on attributes a1 to ak before ajw, the write operation on attribute aj

For each write operation ajw

PreWriteSeqSet=PreWriteSeqSet∪{a1w, a2w,....,akw, ajw}

If (ajw, a1w, a2w, ...akw∈ Pi and a1w, a2w, ...akw ≠ Ø) where a1w to akw are all the write operation on attributes a1 to ak after ajw, the write operation on attribute a

For each write operation ajw

PostWriteSeqSet=PostWriteSeqSet∪ {ajw, a1w, a2w,...., akw}

For each read sequence rs of the form a1r, a2r,....,akr, ajw∈ ReadSeqSet

Construct read rule rr of the form

ajw →a1r, a2r,....,akr

Calculate the confidence C of rr using equation (2)

If(C>=minConf)

ReadRuleSet=ReadRuleSet∪ {rr}

For each rule ∈ReadRuleSet

If rr has same antencedent as another rule and rr's consequence is subset of that

```
rulei.err is redundant rule
ReadRuleSet=ReadRuleSet-{rr}
For each Pre-Write sequence pwr of the form $a1w$, $a2w$,....,$akw$, $ajw \in$ PreWriteSeqSet
Construct pre-write rule pwr of the form $a_{1w}$, $a_{2w}$, ...., $akw \leftarrow ajw$
Calculate the confidence C of pwr using equation (2)
        IF (C>=minConf)
PreWriteRuleSet=PreWriteRuleSet∪ {pwr}
For each rule∈ PreWriteRuleSet
If pwr has same antencedent as another rule and pwr's consequence is subset of that
rule i.e pwr is redundant rule.
PreWriteRuleSet=PreWriteRuleSet-{pwr}
For each Post-Write sequence powr of the form $ajw$, $a1w$, a2w,....,$a_{kw}$∈ PostWriteSeqSet
Construct post- write rule powr of the form $a_{jw} \rightarrow a_{1w}$, $a_{2w}$ , ...., $akw$
Calculate the confidence C of powr using equation (2)
IF (C>=minConf)
PostWriteRuleSet=PostWriteRuleSet∪ {powr}
For each rule ∈ PostWriteRuleSet
If powr has same antencedent as another rule and powr's consequence is subset of that rule i.e., powr is redundant rule
PostWriteRuleSet=PostWriteRuleSet-{powr}
Return AnsSet= {ReadRuleSet, PreWriteRuleSet, PostWriteSet}
```

4.5 Detection Phase

After the rules are generated, these are used to verify whether the incoming transactions are malicious or not. If an incoming transaction has a write operation, it is checked whether there are any corresponding read rules, pre-write rules or post-write rules. If the write operation violates these rules, it is marked as malicious and an alarm is generated. Otherwise, normal operation proceeds. For example, Let us take an incoming Transaction T: $16r$, $17r$, $14r$, $14r$, $17w$, $15w$, $2r$, $7w$. In this transaction the attributes that have been updated are 17, 15, 7. For attribute 17 there is one read rule $17w \rightarrow 16r$, 17r, 14r and one pre-write rule$15w \leftarrow 17w$ in Table 8. First one rule is satisfied because 16, 17, 14 are read before updating 17 but second one is not satisfied Hence transaction is marked as malicious. If we consider rules generated by weighted algorithm5, it can be seen that there is no pre-write rule for attribute 17 and one read rule17w→16r,17r,14r which is satisfied ,for attribute 15 there is only one read rule15w→ 14r which is satisfied because before updating attribute 15 attributes 14 are read in transaction. For 7 there is one read rule 7w→2r which is also satisfied because before updating attribute 7 attribute 2 is read in transaction. Hence transaction T is marked as normal transaction.

5. IMPLEMENTATION METHODOLOGY

5.1 Experimental Setup

The proposed technique has been tested and evaluated with the help of number of transaction log files. We developed System prototype using Java as front end and MS SQL 2000 Server. We have used Standard TPC-C benchmark Schema. Here we used the bank database of Table 1 for our experiments. All the experiments were conducted on a Compaq-PC, Intel(R) core (TM) 2 Duo processor, 2.10 GHz, with 2 GB of RAM running on Window 7 Professional N 64 bit operating system.

5.2 Test Applications

The overall working of our proposed algorithm depends on various parameters as, Minimum Support, Sensitivity ratio, Sensitivity group, Number of mean write operations in each transaction etc. We varied all these parameters to see the effect on overall working of proposed algorithm. So in the training phase, we have generated a number of sets of training data with each set of size 10 to 100 transactions having different distributions and performed various experiments for each parameter. In experiments for mean write operations, we have used the 90% transactions which contain at least one write (update) operation and 10% transaction with all read operation. and we check the performance of our proposed system by varying the mean number of write operations within the transactions. The performance of our system for this variation is represented by line chart shown in Section 6. In experiments for sensitivity group, we have also chosen the number of transactions containing most sensitive attributes in the training data. We used 20% out of the 100% transactions with highly sensitive attributes in the training data. And vary this parameter also, and effect on working of algorithm shown by line chart in Section 6. Similarly experiments with other parameters also performed and their effect on the working of proposed algorithm is shown in Section 6. In general, we considered support and confidence values are 25% and 70%, respectively. Once the transactions are generated, we execute the Existing Algorithm5 for generation of data dependency rule and after that, we execute our proposed algorithm on the training data with weight ratios 1:2:3 for LS, MS and HS groups. We have taken additional weight of write operation as 0.25 for all the three categories. We vary the different parameter as discussed above to see the relative performance of our proposed algorithm and existing algorithm. The overall comparative results are discussed in next section.

6. PERFORMANCE RESULTS AND ANALYSIS

6.1 Performance Metrics

We have evaluated performance of our proposed algorithm by identifying percentage of malicious transaction detected on different sensitivity ratio, for each sensitivity group, and mean write operation and compare the results with existing algorithm with the help of line and bar charts. At last we evaluate number of comparison required during detection phase with respect to redundant rules. In given subsections we discuss performance of our approach on different parameters one by one.

6.2 Minimum Support

As minimum support increases, percentage of malicious transactions detection decreases since the number of rules crossing the minimum support is less. Fig 2. shows the relative performance of over algorithm with existing algorithm[5].Our algorithm performs better than existing algorithm. This is because our proposed algorithm generates extra pre-write rules that contribute in identification of malicious transactions.

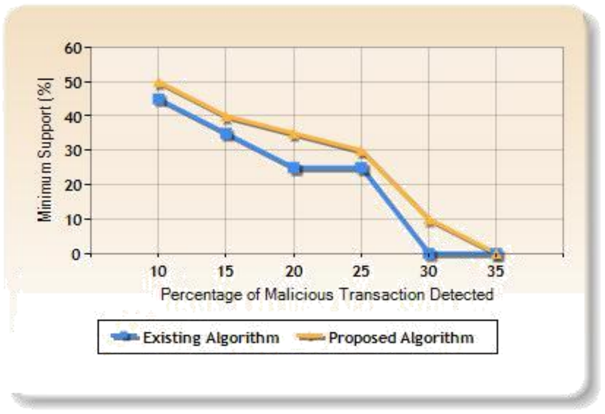


Fig. 2. Performance comparison for minimum support.

6.3 Sensitivity Ratio

When sensitivity ratio is increased, More sensitive attributes which were earlier below the support, can now cross the minimum support value, as higher the number of sequences generated by sequence mining algorithm, more read, pre-write and post-write rules are extracted. In Fig. 3, comparative performance is shown for each sensitivity ratio. It is seen that our proposed algorithm outperforms than existing algorithm for more sensitive attributes.

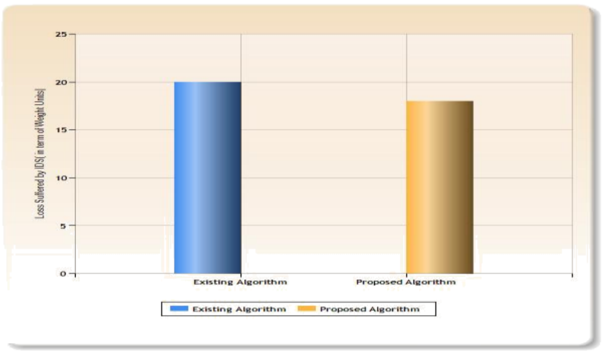


Fig.3. Performance comparison for different sensitivity ratios

6.4 Sensitivity Group

Fig. 4. shows comparative performance for each sensitivity group. It is seen that our proposed algorithm outperforms existing algorithm for more sensitive attributes. Because there are more rules for high sensitive attributes because high sensitive attribute cross minimum support easily, so more transaction are identified malicious on behalf of high sensitive attributes.

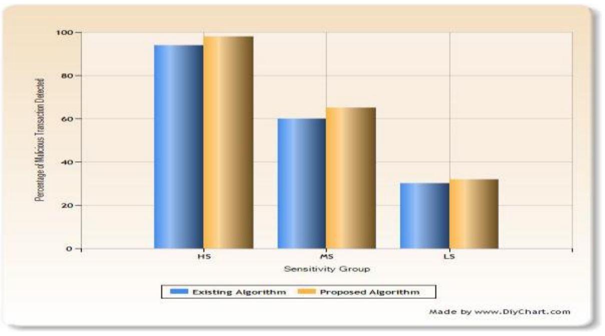


Fig.4. Performance comparison for different sensitivity groups

6.5 Mean Number of Write Operations

Fig.5. shows that as the mean number of write operations increases, the detection rate also increases .the reason is that, with more write operations the number of read pre-write and post-write rules also increases if there are more rules, a higher numbers of attributes checked against malicious modification. Fig.5. shows that our proposed algorithm outperforms existing algorithm because our algorithm generates more rules as compare to existing algorithm[5].

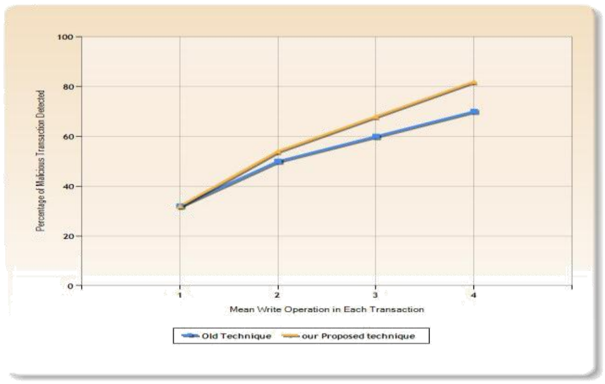


Fig. 5. Performance comparison for Mean write operations

6.6 Loss suffered by Ids (in terms of weight units)

Loss is computed by adding the weights of all the attributes whose malicious modifications are not detected by the IDS. It is observed from Fig. 6. that our proposed algorithm outperforms existing algorithm. This is because our proposed algorithm track the sensitive attributes with write operation much better way than existing algorithm and therefore overall loss is minimized.

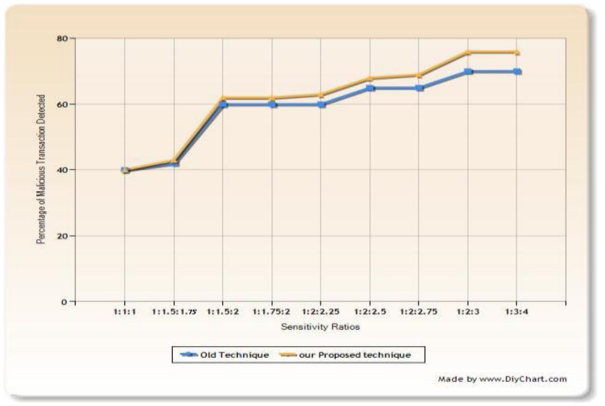


Fig.6. Performance comparison in term of Loss Suffered by IDS

6.7 Number of Comparison required during detection phase against redundant rules

Number of comparison required for identification of a malicious transaction, are equal to number of redundant rules in rule set .our proposed algorithm removes all redundant rules in training phase so number of comparison required to check maliciousness of a transaction by our algorithm is always one. Fig.7. shows that our algorithm outperforms existing algorithm because existing algorithm generates large number of redundant rules but our proposed algorithm generates no redundant rule.

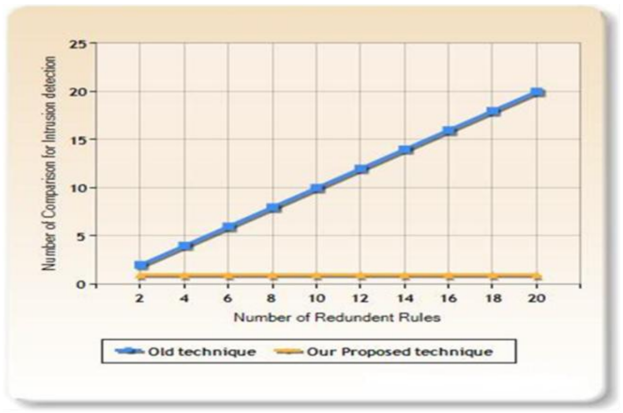


Fig.7. Number of comparison for intrusion detection versus number of redundant rules

6.8 Analysis Results

In this section, we analyze the performance of our algorithm in terms of their characteristics. As discussed the performance results for our proposed system in section 5.1, the proposed system achieves following main goals of database intrusion detection system.

- Our system detects more malicious transactions as compared to other existing system [5] therefore true positive rate of our system increases.
- Our system generates more rules as compared to other existing system so false positive rate decreases.
- Our system detects normal transaction as normal with high rate as existing system. So true negative rate increases.
- False negative rate of our system is very low because it checks three types of data dependency read and pre-write and post-write.

So finally we can say that our system is improved version of existing IDS [5].

7. CONCLUSIONS AND FUTURE WORK

Database intrusion detection is a new field of research focusing on the security in DBMS from the applications of data mining algorithms. Here, we proposed an effective technique for database intrusion detection in which we take care of three parameter, first one the sensitivity or importance of attributes and second one is pre - write data dependency and post-write data dependency between attributes(in form of rules) and last one is redundancy of rules. Our algorithm takes care of all these parameters. An example demonstrating our proposed algorithm is discussed. We also discussed the comparative performance metrics for database intrusion detection system, in which it shows that performance of the proposed system is better than other existing approach [5]. Our proposed system considers pre-write data dependency and post data dependency between attributes so generate pre-write data dependency rules and post-write data dependency rules. Existing system does not consider pre-write data dependency therefore generates only write data dependency rules which are similar to our post-write data dependency rules. By additionally, generating pre-write data dependency rules, our proposed algorithm decreases false positive rate and increases true positive rate .Our system also eliminate the redundant data dependency rules in training phase which reduces the number of comparison with rules in detection phase therefore improves the response time of the system.

Currently sensitive attributes are identified at the design level of DBMS. In future, we plan to use Role Based Access control (RBAC) administered databases for finding out sensitive attributes dynamically. Under an RBAC system, permissions are associated with roles usually grouping several user, rather than with single user. Generally, important roles like administrator access sensitive attributes and if their audit logs are mined, then some useful information regarding the attributes can be extracted. This will help in deciding the sensitivity of attributes.

8. REFERENCES

[1] J. Han, M. Kamber. “Data Mining: Concept and Techniques”, 2001, Morgan Kaufmann Publishers.

[2] D. Neelapala. “Use of FP growth Algorithm for Database Intrusion Detection”, UMI dissertation Publishing, 2008, pp. 230-260.

[3] W. L. Low, S. Y. Lee, P. Teoh. “DIDAFIT: Detecting Intrusions in Databases Through Fingerprinting Transactions”, In: proceedings of the 4th International Conference on Enterprise Information Systems (ICEIS), 2002, pp. 264-269.

[4] U. Fayyad, G.P. Shapiro, P. Smyth. “The KDD Process for Extracting Useful Knowledge from Volumes of Data”, In proceedings of the Communications of the ACM, 1996, pp. 27-34.

[5] A.Srivastava, S. Sural, A.K. Majumdar. “Weighted Intra-transactional Rule Mining for Database Intrusion Detection,” Lecture Notes in Artificial Intelligence, Springer Verlag, In proceedings of Pacific-Asia Conference in Knowledge Discovery and Data Mining, 2006, pp. 611-620.

[6] Y. Hu, B. Panda. “A Data Mining Approach for Database Intrusion Detection”, In proceedings of the ACM Symposium on Applied Computing, 2004, pp. 711-716.

[7] A. Rezk, H. Ali, M. El-Mikkawy, S. Barakat. “Minimize the False Positive Rate in a Database Intrusion Detection System”, International Journal of Computer Science & Information Technology (IJCSIT), 2011, Vol. 3, No 5, pp. 29-38.

[8] R. Agrawal, R. Srikant. “Fast algorithms for mining association rules”, In: Proceedings of the 20th International Conference on Very Large Databases, 1994, ACM SIGMOD, pp. 487–499.

[9] W. Wang, J. Yang, P. S. Yu. “Efficient Mining of Weighted Association Rules”, In proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2000, pp. 270-274.

[10] F. Tao, F. Murtagh, M. Farid. “Weighted Association Rule Mining using Weighted Support and Significance Framework”, 2003, In proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 661-666.

[11] E. Lundin, E. Jonsson. “Survey of Intrusion Detection Research”, Technical Report, 2002, Chalmers University of Technology.

[12] W. Lee, S.J. Stolfo. “Data Mining Approaches for Intrusion Detection”, In proceedings of the USENIX Security Symposium, 1998, pp. 79-94.

[13] D. Barbara, J. Couto, S. Jajodia, N. Wu. “ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection”, ACM SIGMOD, 2001, pp. 15-24.

[14] C.Y. Chung, M. Gertz, K. Levitt. “DEMIDS: A Misuse Detection System for Database Systems”, In proceedings of the IFIP TC-11 WG 11.5 Working Conference on Integrity and Internal Control in Information System,1999, pp. 159-178.

[15] V.C.S. Lee, J.A. Stankovic, S.H. Son. “Intrusion Detection in Real-time Database Systems via Time Signatures”, 2000, In proceedings of the Real Time Technology and Application Symposium, pp. 124.

[16] S.Y. Lee, W.L. Low, P.Y. Wong. “Learning Fingerprints for a Database Intrusion Detection System”, 2002, In proceedings of the European Symposium on Research in Computer Security, pp. 264-280.

[17] D. Barbara, R.Goel, S. Jajodia. “Mining Malicious Data Corruption with Hidden Markov Models”, 2002, In proceedings of the IFIP WG 11.3 Working Conference on Data and Application Security, pp. 175-189.

[18] Y. Zhong, X. Qin. “Research on Algorithm of User Query Frequent Item sets Mining”, 2004, Machine Learning Cybernetics, pp. 1671-1676.

[19] A. Srivastava, A. Bhosale, S. Sural. “Speeding up Web Access Using Weighted Association Rules”, Lecture Notes in Computer Science, Springer Verlag, 2005, In proceedings of International Conference on Pattern Recognition and Machine Intelligence (PReMI'05), 660-665.

[20] A. Kundu, S. Sural, A.K. Majumdar. “Database Intrusion Detection using sequence alignment”, 2010, In proceedings of international journal of information security, pp. 179-191.

[21] E. Bertino, E. Terzi, A. Kamra, A. Vakali.“Intrusion Detection in RBAC-Administered Databases”, 2005, In: Proceedings of the 21st annual computer security applications conference (ACSAC), pp. 170–182.

[22] R. Agrawal, R. Srikant. “Mining Sequential pattern”, In proceedings of the 1995 International Conference on Data Engineering, 1995, Taipei, Taiwan , pp. 203-205.

[23] S.Wenhui, T. Tan. “A Novel Intrusion Detection System Model for Securing Web-Based Database Systems”, 2001, In proceedings of the 25th annual international computer software and applications conference (COMPSAC), pp. 249–254.

[24] K. Takeda. “The Application of Bioinformatics to Network Intrusion Detection”, 2005, In proceedings of the international carnahan conference on security technology (CCST), pp. 130–132.

[25] S. Coull, J. Branch, B. Szymanski, E. Breimer. “Intrusion Detection: A Bioinformatics Approach”, 2003, In: proceedings of the annual computer security applications conference (ACSAC), pp. 24–33.

[26] E. Mohammadreza, S. Merar, S. Fatimah, A. Lilly Suraini. “Intrusion detection using Data Mining Techniques”, 2010, In: proceedings of Information retrieval and knowledge management (CAMP), IEEE, pp. 200-203.

[27] Yi Hu, C. Campan, J. Walden, I. Vorobyeva, J. Shelton. “An effective log mining approach for database intrusion detection”, 2010, In proceeding of International Conference on Systems Man and Cybernetics (SMC), IEEE , pp. 2299 – 2306.

[28] M. Ektefa, S. Memar, F. Sidi, Affendey L. S.. “Intrusion Detection Using Data Mining Techniques”, 2010, In Proceedings of the International Conference on Information Retrieval & Knowledge Management, (CAMP), IEEE, pp.200-203.

[29] U.P. Rao, G.J. Sahani, D.R. Patel. “Detection of Malicious Activity in Role Based Access Control (RBAC) Enabled Databases”, 2010, In Proceeding of Journal of Information Assurance and Security 5, pp. 611-617.

[30] R. Brace, P. Mell. “Intrusion detection system”, 2001, NIST special publication on Intrusion Detection System.

[31] R. Sandhu, D. Ferraiolo, R. Kuhan. “The NIST Model for Role Based Access Control: Towards Unified Standard”, 2000, In Proceedings of the 5th ACM workshop of Role Based Access Control.

[32] S.Hashmi, Y. Yang, D. Zabihzadeh, M. Rangavari. “Detecting Intrusion Transaction in Databases Using Data Dependencies and Analysis Expert Systems”, 2008, Vol. 25, pp.460-473S.