Database Intrusion Detection using Weighted Sequence Mining

Abhinav Srivastava[1], Shamik Sural[1] and A.K. Majumdar[2]
[1] School of Information Technology
[2] Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur, 721302, India
Email: {abhinavs@sit, shamik@sit, akmj@cse}.iitkgp.ernet.in

Abstract— **Data mining is widely used to identify interesting, potentially useful and understandable patterns from a large data repository. With many organizations focusing on web-based on-line transactions, the threat of security violations has also increased. Since a database stores valuable information of an application, its security has started getting attention. An intrusion detection system (IDS) is used to detect potential violations in database security. In every database, some of the attributes are considered more sensitive to malicious modifications compared to others. We propose an algorithm for finding dependencies among important data items in a relational database management system. Any transaction that does not follow these dependency rules are identified as malicious. We show that this algorithm can detect modification of sensitive attributes quite accurately. We also suggest an extension to the Entity-Relationship (E-R) model to syntactically capture the sensitivity levels of the attributes.**

Index Terms— **Data dependency, Weighted Sequence mining, Intrusion detection, E-R Model**

I. INTRODUCTION

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 [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 [2]. 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.

Intrusion is commonly defined as a set of actions that attempt to violate the integrity, confidentiality or availability of a system. Intrusion Detection is the process of tracking important events occurring in a computer system and analyzing them for possible presence of intrusions [3]. Intrusion Detection Systems (IDSs) are the software or hardware products that automate this monitoring and analysis process. In intrusion detection, it is assumed that all the prevention techniques are compromised and an intruder has potentially entered into the system. Hence, intrusion detection system is considered to be the second line of defense. In general, there are two types of attacks (i) inside and (ii) outside. Inside attacks are the ones in which an intruder has all the privileges to access the application or system but he performs malicious actions. Outside attacks are the ones in which the intruder does not have proper rights to access the system. He attempts to first break in and then perform malicious actions. Detecting inside attacks is usually more difficult compared to outside attacks.

Intrusion detection systems determine if a set of actions constitute intrusions on the basis of one or more models of intrusion. A model classifies a sequence of states or actions as "good" (no intrusion) or "bad" (possible intrusions). There are mainly two models, namely, anomaly detection and misuse 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. The main disadvantage is that, it may not be able to describe what the attack is and may sometimes have high false positive rate. In contrast, 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.

In this paper, we propose a new approach for database intrusion detection using a data mining technique which takes the sensitivity of the attributes into consideration in the form of weights. Sensitivity of an attribute signifies how important the attribute is, for tracking against

malicious modifications. This approach mines dependency among attributes in a database. The transactions that do not follow these dependencies are marked as malicious transactions. Early result on this work has been reported in [4]. We also propose an extension to the existing E-R model notations to represent sensitivity of the attributes.

The rest of the paper is organized as follows. In Section II, we mention some of the related work done in the area of database intrusion detection. We describe the principle behind weighted data dependency rule mining (WDDRM) with an example and explain the complete algorithm in Section III. An extension to E-R model is discussed in section IV. We present experimental results in Section V. Finally, we conclude the paper with some discussions.

II. Related Work

Intrusion detection started emerging as an interesting research topic since the beginning of the 1980s. In the 1990s, intrusion detection became an active area of research and even some commercial IDSs were built [5]. Over the last few years, several research works have been carried out that attempt to apply data mining for intrusion detection. Lee et al [6] suggest a method for network intrusion detection using data mining techniques. They consider classification, link analysis and sequential analysis as potential data mining algorithms along with their applicability in the field of intrusion detection. Barbara et al [7] have built a testbed for the detection of network intrusions using data mining.

Though intrusion detection is comparatively a well researched field, only a few researches have considered the problem of database intrusion detection. Chung et al [8] use "working scope" to find frequent itemsets, which are sets of features with certain values. They define a notion of distance measure that captures the closeness of a set of attributes with respect to the working scopes. Distance measures are used to guide the search for frequent itemsets in the audit logs. Lee et al [9] have proposed real-time database intrusion detection using time signatures. It monitors the database behavior at the level of sensor transactions. Sensor transactions are usually small in size and have predefined semantics such as write only operations and well defined data access patterns. In real-time database systems, temporal data objects are used. This temporal data has to be updated periodically. If a transaction attempts to update a temporal data which has already been updated in that period, an alarm is raised.

Lee et al [10] suggest fingerprinting of the access patterns of genuine database transactions and using them to identify potential intrusions. They summarize SQL queries into compact and effective regular expression fingerprints. If a given query does not match any of the existing fingerprints, it is reported as malicious. Barbara et al [11] use hidden markov model (HMM) and time series to determine malicious data corruption. They build a database behavioral model using HMM that captures the changing behavior over time. Malicious patterns are the ones that are not recognized by the HMM with high probability. Zhong et al [12] use an algorithm that mines user profiles based on the pattern of submitted queries. Hu et al [13] determine dependency among data items where data dependency refers to the access correlations among data items. These data dependencies are generated in the form of classification rules, i.e., before one data item is updated in the database, which other data items probably need to be read and after this data item is updated, which other data items are most likely to be updated by the same transactions. Transactions that do not follow any of the mined data dependency rules are marked as malicious transactions.

None of the above mentioned approaches consider sensitivity of the attributes in a database. They treat all the attributes at the same level and of equal importance, which is not always the case in real applications. Hence, there is a need for assigning higher weights to high sensitivity data items. Weighted data mining problems are being studied only very recently. Wang et al [14] have proposed a weighted association rule mining technique in which they assign numerical weights to each item to reflect interest/intensity of the item within the transaction. They first ignore the weight and find the frequent itemsets from the unweighted data and then introduce weight during rule generation. Tao et al [15] use weighted support for discovering the significant itemsets during the frequent itemset finding phase. Srivastava et al [16] have recently proposed the use of weighted association rule mining for speeding up web access by prefetching the URLs.

We discuss our work on weighted data dependency rule mining for intrusion detection in detail in the next section.

III. Weighted Data Dependency Rule Mining

A. MOTIVATION

In recent years, database size has grown in two ways: the number of records, or objects in the database and the number of fields or attributes per object. It is now quite common to have databases containing of the order of 10^9 objects, each having 10^2 or 10^3 attributes [2]. As a result, it is difficult for administrators to keep track of whether the attributes are being accessed only by genuine transactions or not. By categorizing the attributes into different types based on their relative importance or sensitivity, it becomes comparatively easier to track only those few attributes whose unintended modification can potentially have larger impact on the database application security.

It has been observed that IDSs often raise a large number of alarms, many of which are triggered

incorrectly by benign events [17]. Categorization of attributes enables the administrator to check only the alarms generated due to unusual modification of sensitive data instead of checking all the data attributes. Since the primary aim of a database intrusion detection system is to minimize the loss suffered by the database owner, it is important to track high sensitive attributes more accurately.

It should be noted that, for tracking malicious modifications of sensitive attributes, we need to obtain data dependency rules for these attributes. If there is no rule for an attribute, it cannot be checked. Since high sensitivity attributes are usually accessed less frequently, there may not be any rule generated for these attributes. The motivating factor for dividing database attributes into different sensitivity groups and assigning suitable weights to each group, is to bring out the data dependency rules for important but possibly less frequent attributes. Once rules are generated for sensitive attributes, it is possible to check against these rules for each transaction. If any transaction does not follow the mined rules, it is marked as malicious.

We divide the work of mining weighted data dependency rules for database intrusion detection into the following three components:

- Security Sensitive Sequence Mining
- Read-Write Sequence Generation
- Weighted Data Dependency Rule Generation

Fig. 1 shows the components of WDDRM schematically. First, audit logs are given as input for security sensitive sequence mining. The mining step generates the frequent sequences which are above a user specified support value. These sequences are then passed to the Read-Write sequence generation module, which computes the read and write sequences. Finally, read-write rules are extracted from the read-write sequences. The rules, which are above a minimum user specified confidence value, are included in the final set of rules. We discuss these components in detail in the following subsections.

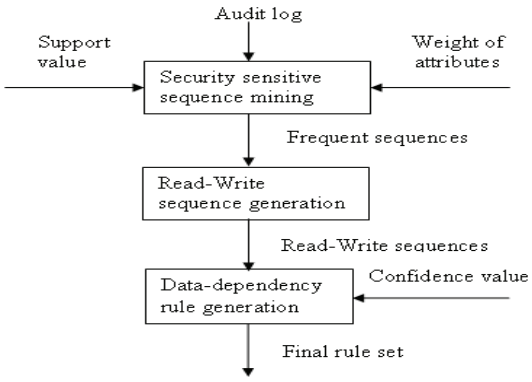


Figure 1. Components of WDDRM Architecture

B. DEFINITIONS

In this subsection, we define the terminologies used in the rest of the paper.

Definition 1 *SEQUENCE*: A sequence is an ordered list of attributes along with the operations performed on these attributes. A sequence denoted as *Seq* is of the form $<a_{1o}, a_{2o}, a_{3o} \ldots .a_{ko}>$, 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 *READ SEQUENCE*: A read sequence denoted as *ReadSeq* of attribute a_j is a sequence of the form $<a_{1r}, a_{2r}, \ldots a_{kr}, a_{jw}>$, which is the sequence of attributes a_1 to a_k that are read before attribute a_j is 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 + 100
 where customerId=5;
```

In this update query, before updating the *balance* attribute, *customerId* and old value of *balance* must be read. So read sequence for updating *balance* would be $<customerId_r, balance_r, balance_w>$.

Definition 3 *WRITE SEQUENCE*: A write sequence denoted as *WriteSeq* of attribute a_j is a sequence of the form $<a_{jw}, a_{1w}, a_{2w}, \ldots .a_{kw}>$, which is the sequence of attributes a_1 to a_k that are written after attribute a_j is written. All such sequences form a set named as write sequence set denoted by *WriteSeqSet*.

For example, consider the following SQL statement:

```
Update Account set balance = 1000, date =
'09/10/2005' where customerId = 5;
```

In the above transaction, it is seen that after updating *balance* attribute, *date* attribute is updated. So write sequence would be $<balance_w, date_w>$.

It must be noted that both read and write sequences can occur simultaneously in the same transaction. If we consider the last transaction, the complete sequence considering both read and write sequences would be: $<customerId_r, balance_w, date_w>$.

C. SECURITY SENSITIVE SEQUENCE MINING

Finding sequences among attributes along with the corresponding {read, write} access operations is similar to the problem of mining sequential patterns. Agrawal et al [18] have proposed an algorithm for detecting frequent sequential patterns in data. At first, it generates probable frequent sequence sets called candidate sets and then uses

two pruning steps to find the actual maximal frequent sequences. The first pruning step uses the downward closure property, which states that a sequence of length K cannot be frequent if all its subsequences of length K-1 are not frequent. If any of its subsequence is not frequent, then this sequence of length K is not considered as a candidate. The second pruning strategy is to drop the sequence from the set of frequent sequences if its count is below a user-specified minimum support value. Support of the sequence s is calculated as the percentage of transactions that contain the sequence s.

In this method, all the data items are considered at the same level without any weightage. We modify this traditional sequential mining algorithm and make it security sensitive by introducing weights for each attribute based on the sensitivity group. Higher the sensitivity of an attribute, more is its weight. We categorize the attributes into three sets: High Sensitivity (*HS*), Medium Sensitivity (*MS*) and Low Sensitivity (*LS*). The sensitivity of an attribute depends on the particular database application where it is used. Also, modification of the sensitive attributes is considered more important than reading them from the point of view of integrity. For the same attribute say x, if $x \in HS$ then $W(x_w) > W(x_r)$, where W is a weight function, x_w denotes writing or modifying attribute x and x_r denotes reading of attribute x.

For a given schema, we categorize the attributes into the above-mentioned three sets and assign numerical weights to each set. Let $w_1, w_2, w_3 \in R$, where R is the set of real numbers and $w_3 \le w_2 \le w_1$ are the weights of *HS*, *MS* and *LS*, respectively. Let $d_1, d_2, d_3 \in R$ be the additional weights of the write operations for each category such that $d_3 \le d_2 \le d_1$. Let $x \in HS$ be an attribute that is accessed in a read operation. Then the weight given to x is w_1. If it is accessed in write operation, then the weight given to x is $w_1 + d_1$.

In order to perform security sensitive sequence mining, we assign weights to each sequence based on the sensitivity groups of the attributes accessed in that sequence. The weight of a given sequence is the same as the weight of the most sensitive attribute present in that sequence. The weight assigned to a sequence also depends on the operation applied on the attributes. 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

(i) $<a_{1r}, a_{3r}, a_{2w}, a_{4w}>$
(ii) $<a_{1r}, a_{3w}, a_{4w}>$
(iii) $<a_{5r}, a_{2w}, a_{4r}>$
(iv) $<a_{4r}, a_{5w}>$

Let 3, 2 and 1 be the weights of *HS*, *MS* and *LS*, respectively and 0.75, 0.50, 0.25 be the additional weights of write operation for *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 + 0.75. 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 + 0.50. 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 are normalized so that they add up to unity.

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 W_s. Let N be the total number transactions. If s is present in n out of N transactions, then the support of sequence s is:

$$\text{Support}(s) = (n * W_s) / N \qquad (1)$$

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 I. These transactions are generated from the bank database schema shown in Table II with attributes encoded into integers. In Table III, 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 IV. In this table, we also show the sequences that would have been generated if categorization was not used at all.

TABLE I. EXAMPLE TRANSACTIONS FOR THE SEQUENCE MINING ALGORITHM

Txn ID	Attribute Access Sequence
1	$11_r, 13_w, 4_r, 8_r, 2_r, 6_r, 1_r, 3_r$
2	$7_r, 2_r, 7_r, 2_r, 3_r, 9_w$
3	$6_r, 1_r, 3_r, 3_r, 9_w, 1_w, 2_r, 7_w$
4	$11_r, 12_w, 2_r, 4_w, 6_r, 1_r, 3_r$
5	$2_r, 4_w, 2_r, 7_w, 7_r, 8_r, 2_r$
6	$11_r, 13_w, 4_r, 8_r, 2_r, 2_r, 4_w$
7	$3_r, 9_w, 4_r, 8_r, 2_r, 8_r, 2_r$
8	$7_r, 8_r, 2_r, 2_r, 2_r, 8_w, 5_w, 2_r, 4_w$
9	$8_r, 2_r, 3_r, 9_w, 7_r, 2_r$
10	$3_r, 9_w, 6_r, 1_r, 3_r, 3_r, 9_w, 1_w$

TABLE II. BANK DATABASE SCHEMA

Table Name	Column Name (Integer Encoding of Attributes)
Customer	Name (9), Customer_id (10), Address (4), Phone_no (1)
Account	Account_id (2), Customer_id (3), Status (7), Open_dt (5), Close_dt (6), Balance (8)
Account_type	Account_type (11), Max_tran_per_month, (13) Description (12)

TABLE III. WEIGHT TABLE FOR THE ATTRIBUTES USED IN THE BANK DATABASE

Sensitivity Group	Attributes	weights
HS	7, 8, 13	3
MS	5, 6	2
LS	1, 2, 3, 4, 9, 10, 11, 12	1

TABLE IV. MINED SEQUENCE USING MINIMUM SUPPORT VALUE 25%

Sequence using Non-weighted Method	Sequence using Weighted Method
$<3_r\ 9_w\ 2_r>$ $<4_r\ 8_r\ 2_r>$ $<6_r\ 1_r\ 3_r>$ $<2_r\ 3_r>$ $<2_r\ 7_r>$ $<7_r\ 2_r>$ $<11_r\ 2_r>$	$<8_r\ 3_r\ 9_w\ 7_r\ 2_r>$ $<13_w\ 4_r\ 8_r\ 2_r>$ $<7_w\ 7_r\ 8_r\ 2_r>$ $<4_w\ 7_w\ 7_r\ 8_r>$ $<2_r\ 7_w\ 7_r\ 8_r>$ $<3_r\ 9_w\ 2_r\ 7_w>$ $<3_r\ 9_w\ 2_r\ 8_r>$ $<3_r\ 9_w\ 8_r\ 2_r>$ $<6_r\ 1_r\ 3_r\ 7_w>$ $<6_r\ 3_r\ 9_w\ 7_r>$ $<7_r\ 8_r\ 2_r\ 8_w>$ $<8_r\ 2_r\ 3_r\ 7_r>$ $<8_r\ 2_r\ 6_r\ 3_r>$ $<8_r\ 6_r\ 1_r\ 3_r>$ $<11_r\ 13_w\ 8_r\ 2_r>$ $<11_r\ 8_r\ 2_r\ 6_r>$ $<13_w\ 8_r\ 4_w>$ $<2_r\ 7_r\ 3_r>$ $<4_r\ 2_r\ 8_r>$ $<7_r\ 2_r\ 3_r>$ $<7_r\ 3_r\ 9_w>$

D. READ-WRITE SEQUENCE GENERATION

The sequences generated in the previous step are next used to determine read and write sequences. According to the definitions, *ReadSeq* and *WriteSeq* must contain at least one write operation each. Sequences that do not have any attribute with write operation, are not used for read and write sequence generation. A sequence that contains a single attribute does not contribute to dependency rule generation and hence, is also filtered out. The read-write sequences are generated as follows.

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 $<3_r\ 9_w\ 2_r\ 7_{w>}$ of Table IV generates the following read sequences $<3_r\ 9_{w>}$, $<3_r\ 2_r\ 7_w>$. To generate write sequences, for each write operation a_{jw} in a sequence, add $<a_{jw}, a_{1w}, a_{2w}, \ldots a_{kw}>$ to *WriteSeqSet* 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 . For example, sequence $<3_r\ 9_w\ 2_r\ 7_{w>}$ of Table IV generates the write sequence $<9_w\ 7_w>$. The read-write sequences generated from the mined sequences of Table IV are shown in Table V.

TABLE V. READ SEQUENCES AND WRITE SEQUENCES

Non-Weighted Method	Weighted Method
$<3_r\ 9_w>$	$<4_w\ 7_w>$ $<9_w\ 7_w>$ $<13_w\ 4_w>$ $<8_r\ 3_r\ 9_w>$ $<2_r\ 7_w>$ $<3_r\ 9_w>$ $<3_r\ 2_r\ 7_w>$ $<6_r\ 1_r\ 3_r\ 7_w>$ $<6_r\ 3_r\ 9_w>$ $<6_r\ 3_r\ 7_w>$ $<7_r\ 8_r\ 2_r\ 8_w>$ $<11_r\ 13_w>$ $<8_r\ 4_w>$ $<7_r\ 3_r\ 9_w>$

E. WEIGHTED DATA DEPENDENCY RULE GENERATION

We extract two types of data dependency rules, namely, read rules and write rules. A read rule is of the form $a_{jw} \rightarrow a_{1r}, a_{2r}\ldots a_{kr}$. It implies that attributes a_1 to a_k must be read in order to write attribute a_j. Write rule is of the form $a_{jw} \rightarrow a_{1w}, a_{2w}, \ldots a_{kw}$. It implies that after writing attribute a_{jw}, attributes $a_{1w}, a_{2w}, \ldots, a_{kw}$ are modified. Both types of rules are obtained from the read and write sequences generated in the previous step. The rules, which are above a user defined minimum confidence, are considered for the final set of rules. The rules generated from the read-write sequences of Table V are shown in Table VI. The generated rules are used to verify whether an incoming transaction is malicious or not. If an incoming transaction has a write operation, it is checked if there are any corresponding read or write rules. If the new write operation does not satisfy any of these rules, it is marked as malicious and an alarm is generated. Otherwise, the transaction is computed and normal operation proceeds. The complete algorithm for the weighted data dependency rule mining is shown in Fig. 2.

TABLE VI. READ AND WRITE DEPENDENCY RULES

Non-Weighted Method	Weighted Method
$<9_w \rightarrow 3_r>$	$<9_w \rightarrow 3_r>$ $<8_w \rightarrow 7_r\ 8_r\ 2_r>$ $<4_w \rightarrow 8_r>$ $<4_w \rightarrow 7_w>$ $<9_w \rightarrow 7_w>$ $<13_w \rightarrow 4_w>$ $<13_w \rightarrow 11_r>$ $<7_w \rightarrow 2_r>$

Let us take an example incoming transaction T: $3_r, 9_w, 7_w, 2_r, 13_w$. In this transaction, the attributes that have been updated are 9, 7 and 13. For attribute 9, there is one read rule $9_w \rightarrow 3r$ and one write rule $9_w \rightarrow 7_w$ in Table VI. Both of these rules are satisfied because 3 is read before updating 9 and 7 is updated after the updation of 9. For attribute 7, there is one read rule $7_w \rightarrow 2_r$ which is also satisfied because before updating attribute 7, 2 is read in the transaction. For 13 there is one write rule $13_w \rightarrow 4_w$ and one read rule $13_w \rightarrow 11_r$ and both of which are not satisfied in the transaction. Hence, transaction T is marked as malicious. If we consider the rules generated by Non-weighted method, it can be seen that there is no rule for attribute 13. Hence, it cannot check the malicious updation of attribute 13 and will mark it as a non-malicious transaction.

INPUT:

A set of transactions *T* containing attribute sequences, weights of the attributes, support *minSup* and confidence *minConf*.

OUTPUT:

A set of rules *WDDR* that can be used to detect malicious modification of the data.

ALGORITHM:

Initialization: *ReadSeqSet* = ø, *WriteSeqSet* = ø, *ReadRuleSet* = ø, *WriteRuleSet* = ø and *WDDR* = {*ReadRuleSet*, *WriteRuleSet*}

Execute sequential mining algorithm with definition of support as in Eq. (1) and input value of *minSup*.

For each sequential pattern P_i, with at least one write operation
  If (a_{1r}, a_{2r}, …a_{kr}, a_{jw} ∈ P_i and a_{1r}, a_{2r}, …a_{kr} ≠ ø)
    For each write operation a_{jw}
      ReadSeqSet = *ReadSeqSet* ∪ { a_{1r}, a_{2r}, …a_{kr}, a_{jw} }

  If (a_{jw}, a_{1w}, a_{2w}, ….a_{kw} ∈ P_i and a_{1w}, a_{2w}, ….a_{kw} ≠ ø)
    For each write operation a_{jw}
      WriteSeqSet = *WriteSeqSet* ∪ { a_{jw}, a_{1w}, a_{2w}, ….a_{kw} }

For each read sequence a_{1r}, a_{2r}, …a_{kr}, a_{jw} ∈ *ReadSeqSet*
  Construct read rule rr of the form a_{jw} → a_{1r}, a_{2r}…….a_{kr}
  If (Confidence of rule rr ≥ *minConf*)
    ReadRuleSet = *ReadRuleSet* ∪ {*rr*}

For each write sequence a_{jw}, a_{1w}, a_{2w}, .…a_{kw} ∈ *WriteSeqSet*
  Construct write rule wr of the form a_{jw}→a_{1w}, a_{2w},...a_{kw}
  If (Confidence of rule wr ≥ *minConf*)
    WriteRuleSet = *WriteRuleSet* ∪ {*wr*}

Return *WDDR* = {*ReadRuleSet*, *WriteRuleSet*}

Figure 2. WDDRM Algorithm

IV. Extension To E-R Model For Representing Attribute Sensitivities

The entity-relationship (E-R) model is a widely used data model, which is based on a perception of the real world. The model consists of a collection of basic objects called entities, and relationships among these entities. We propose an extension to the exiting E-R diagram notations to syntactically capture the sensitivity of the attributes. As explained before, sensitivity is a measure of the importance of the attribute. Since an E-R diagram depicts all the information related to the attribute, we extend it to show the sensitivity information also. We add a new symbol '*' to the existing set of symbols to represent the sensitivity of an attribute.

Star: denoted as '*' represents whether attribute is sensitive or not.

Repeated use of this symbol makes an attribute more sensitive as compared to others. We use a part of the bank database schema of Table II to show how sensitivity can be captured in the existing E-R diagram.

In Fig. 3, we show an extended E-R diagram, which consists of two entity sets, *Customer* and *Account*, related through a binary relationship set *has*. The attributes associated with *Customer* are *Customer id*, *Name*, *Address* and *Phone no*. The attributes associated with *Account* are *Account id*, *Balance* and *Status*. The sensitivity of the attributes are depicted using '*'. In the *Account* relation, *Balance* attribute is given the highest number of '*'s, which makes it the most sensitive attribute compared to the other attributes in the set. *Status* is given one less number of '*' compared to *Balance*.

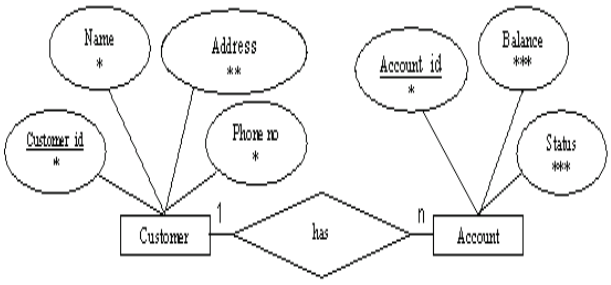


Figure 3. Extended E-R model with Attribute's Sensitivity

A database generated from this E-R model can store the attribute's sensitivity as part of the column

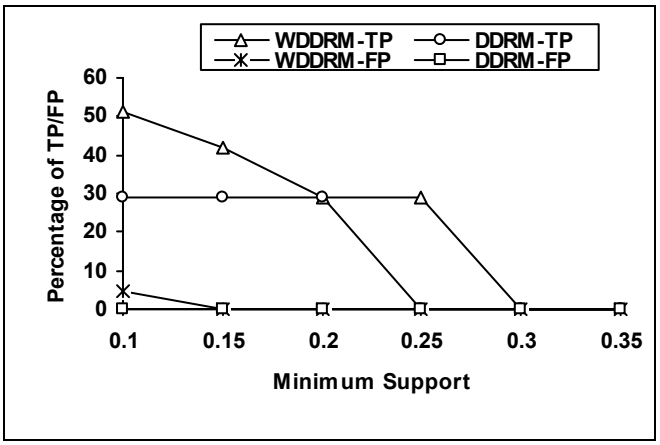
definitions. This information can then be directly used by the security sensitive mining algorithm as proposed here.

V. Experimental Results

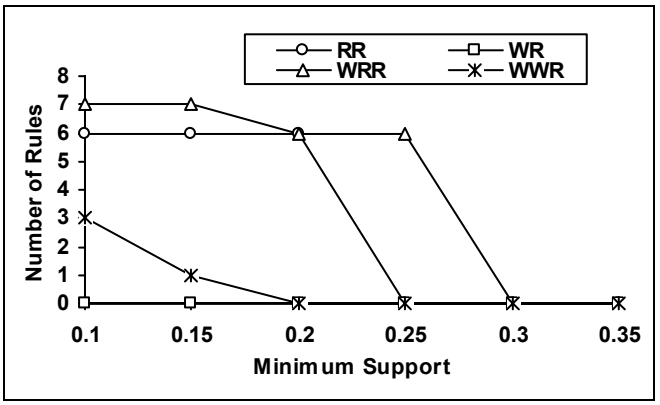
Our work is primarily focused on the development of a database intrusion detection system keeping sensitivity of the attributes into consideration. We have carried out several experiments to show the effectiveness of the proposed method. The system has been developed using Java as front end and MS SQL 2000 Server as the backend database. We have used the standard TPC-C benchmark schema for large scale simulation as suggested by the Transaction Processing Council (TPC). In this schema, we have categorized all the attributes into three sensitivity groups, namely, Low, Medium and High. After categorizing the attributes into different sensitivity groups, we used these attributes in SQL queries for accessing the database.

We vary the parameters that affect overall working of the proposed system. We use standard metrics, namely, True Positive (TP) and False Positive (FP) to analyze the effectiveness of our approach. The number of read and write queries are decided using normal distribution with user specified mean (μ) and standard deviation (σ). We start with write query mean 3 and vary it in different experiments. We also analyze our results for different weight ratios of the three sensitivity groups. We denote our approach by WDDRM and compare with non-weighted approach denoted by DDRM.

In Fig. 4(a), we show the percentage of True Positives (TP) and False Positives (FP) against different minimum support with write query mean 3, weight ratios 1:2:3 and minimum confidence 50%. It can be seen from the figure that WDDRM performs better than DDRM. As minimum support increases, TP decreases since the number of rules crossing the minimum support is less. In Fig. 4(b), we plot the number of rules generated against minimum support for mean 3 and weight ratios 1:2:3. We denote non-weighted read/write rules as *RR* and *WR*, respectively while weighted read/write rules are denoted as *WRR* and *WWR*. It is evident from the figure that at low values of minimum support, there are more rules and weighted rules are more than non-weighted rules. As the minimum support increases, many of the sensitive attributes, which were previously above the support, go now below the support, hence do not contribute in rule generation. Since the number of rules decreases, detection rate also decreases.



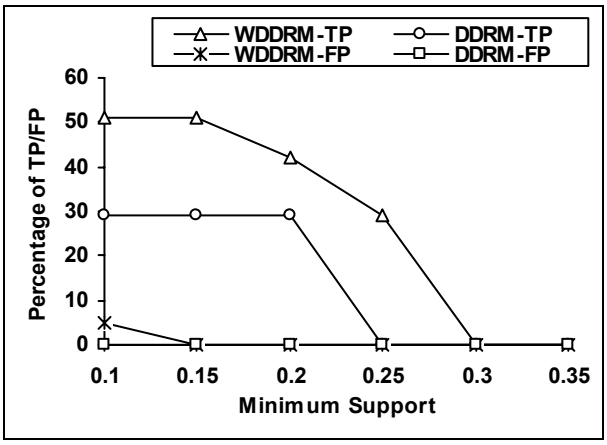
4(a)



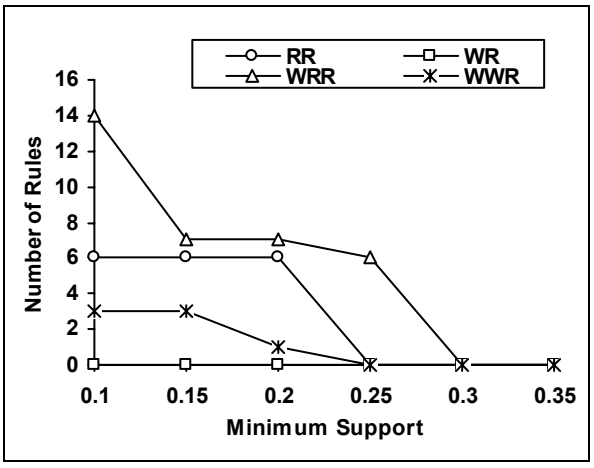
4(b)

Figure 4. (a) TP/FP Vs. Minimum Support for Weight ratio 1:2:3 (b) Number of Rules Vs. Minimum Support for Weight ratio 1:2:3

To see the effect of increase of weight ratios among different sensitivity groups, we plot the variation of TP/FP against minimum support at weight ratios 1:3:4 and 1:5:6 in Figs. 5 and 6, respectively. It can be seen that detection rate is better at weight ratio 1:5:6 compared to 1:3:4 which in turn is better than 1:2:3. Also, the number of rules at weight ratio 1:5:6 is higher than 1:3:4 which is more than at 1:2:3 of Fig. 4. When weight ratio is increased, more sensitive attributes, which were earlier below the support, can now cross the minimum support value. As higher number of sequences is being generated by the sequence mining phase, more read-write rules are extracted, which leads to a higher detection rate.

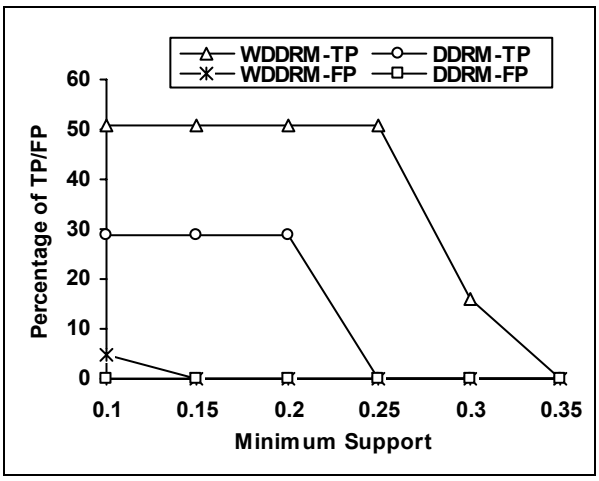


5(a)

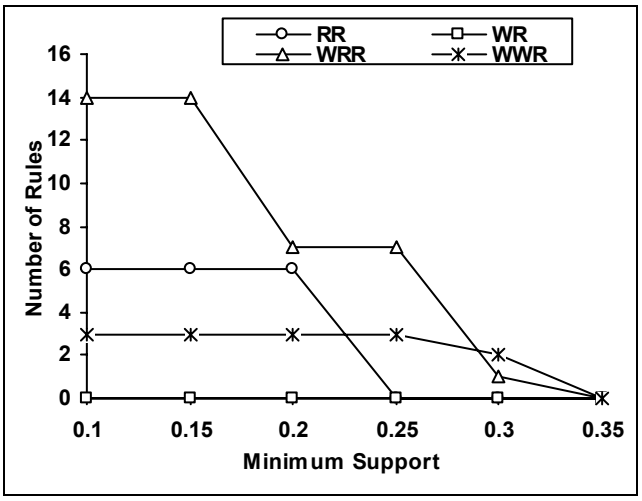


5(b)

Figure 5. (a) TP/FP Vs. Minimum Support for Weight ratio 1:3:4 (b) Number of Rules Vs. Minimum Support for Weight ratio 1:3:4



6(a)



6(b)

Figure 6. (a) TP/FP Vs. Minimum Support for Weight ratio 1:5:6 (b) Number of Rules Vs. Minimum Support for Weight ratio 1:5:6

We next show the variation of Percentage Detection against the mean number of write operations in Fig. 7. The minimum support is fixed at 10% and the weight ratio among the sensitive groups is kept at 1:2:3. It can be seen from the figure 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 and write rules also increases. If there are more rules, a higher number of attributes can be checked against malicious modification, and hence, percentage detection rate also increases.

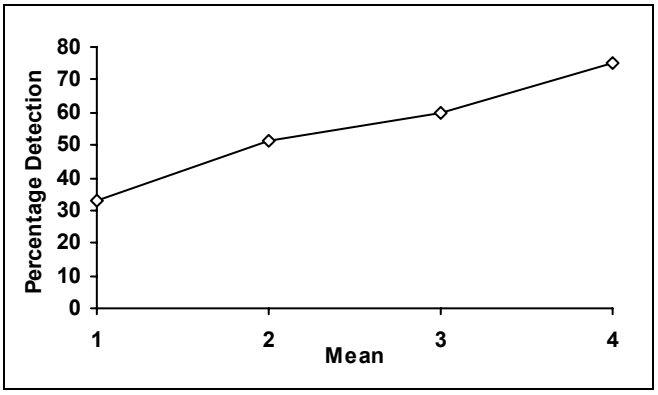


Figure 7. Percentage Detection Vs. Mean number of write operations

Finally, in Fig. 8, we plot the variation of Percentage Detection against the weight of write operations. As mentioned earlier, write operation is more important than read from the information warfare point of view. Also, write operation on a high sensitivity attribute is more important than on low sensitivity attributes. In the previous experiments, the weight of write operations are kept the same for all the three sensitivity groups. In Fig. 8, we show the effect of the variation of the weight of write operations while keeping the weight ratio 1:2:3 constant among sensitive groups. The user specified

minimum support value is 10% and mean is 3. It can be seen from the plot that as the weight of write operation increases, the rate of detection of malicious transactions also increases. The reason is that, as weight increases, the number of frequent sequences crossing the minimum support increases, which helps in generating more rules leading to a higher detection rate.

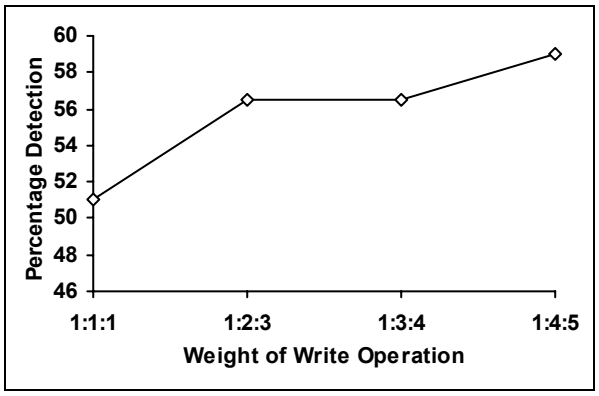


Figure 8. Percentage Detection Vs. Weight of write operations

VI. Conclusions and Discussions

We have proposed a database intrusion detection system using data mining techniques. In this work, we have identified some of the limitations of the existing intrusion detection systems in general, and their incapability in treating database attributes at different levels of sensitivity in particular. We have suggested a novel weighted data dependency rule mining algorithm that considers the sensitivity of the attributes while mining the dependency rules. Experimental results show that our algorithm outperforms some of the previous work done in this area. We also suggest that the sensitivity levels of attributes can be syntactically captured during data modeling through a simple extension of the E-R diagram notations.

Currently, sensitive attributes are identified at the design level. 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 users, 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 the attributes.

The proposed database intrusion detection system generates more rules as compared to non-weighted approach. There is a need for a mechanism to find out which of these new rules are useful for detecting malicious transactions. Such a mechanism helps in discarding redundant rules. We plan to use some learning mechanism to filter out extra rules generated by our approach.

Acknowledgment

This work is partially supported by a research grant from the Department of Information Technology, Ministry of Communication and Information Technology, Government of India, under Grant No. 12(34)/04-IRSD dated 07/12/2004.

References

[1] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers (2001).
[2] U. Fayyad, G. P. Shapiro, P. Smyth, The KDD Process for Extracting Useful Knowledge from Volumes of Data, Communications of the ACM, pp. 27-34 (1996).
[3] R. Bace, P. Mell, Intrusion Detection System, NIST Special Publication on Intrusion Detection System (2001).
[4] A. Srivastava, S. Sural, A.K. Majumdar, Weighted Intra-transactional Rule Mining for Database Intrusion Detection, Lecture Notes in Artificial Intelligence, Springer Verlag, Proceedings of Pacific-Asia Conference in Knowledge Discovery and Data Mining, pp. 611-620 (2006).
[5] E. Lundin, E. Jonsson, Survey of Intrusion Detection Research, Technical Report Chalmers University of Technology, (2002).
[6] W. Lee, S.J. Stolfo, Data Mining Approaches for Intrusion Detection, Proceedings of the USENIX Security Symposium, pp. 79-94 (1998).
[7] D. Barbara, J. Couto, S. Jajodia, N. Wu, ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection, ACM SIGMOD, pp. 15-24 (2001).
[8] C. Y. Chung, M. Gertz, K. Levitt, DEMIDS: A Misuse Detection System for Database Systems, IFIP TC-11 WG 11.5 Working Conference on Integrity and Internal Control in Information System, pp. 159-178 (1999).
[9] V.C.S. Lee, J.A. Stankovic, S.H. Son, Intrusion Detection in Real-time Database Systems Via Time Signatures, Real Time Technology and Application Symposium, pp. 124 (2000).
[10] S.Y. Lee, W.L. Low, P.Y. Wong, Learning Fingerprints for a Database Intrusion Detection System, Proceedings of the European Symposium on Research in Computer Security, pp. 264-280 (2002).
[11] D. Barbara, R. Goel, S. Jajodia, Mining Malicious Data Corruption with Hidden Markov Models, IFIP WG 11.3 Working Conference on Data and Application Security, pp. 175-189 (2002).
[12] Y. Zhong, X. Qin, Research on Algorithm of User Query Frequent Itemsets Mining, Machine Learning Cybernetics, pp. 1671-1676 (2004).
[13] Y. Hu, B. Panda, A Data Mining Approach for Database Intrusion Detection, Proceedings of the ACM Symposium on Applied Computing, pp. 711-716 (2004).
[14] W. Wang, J. Yang, P. S. Yu, Efficient Mining of Weighted Association Rules, Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 270-274 (2000).
[15] F. Tao, F. Murtagh, M. Farid, Weighted Association Rule Mining using Weighted Support and Significance Framework, Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 661-666 (2003).

[16] A. Srivastava, A. Bhosale, S. Sural, Speeding up Web Access Using Weighted Association Rules, Lecture Notes in Computer Science, Springer Verlag, Proceedings of International Conference on Pattern Recognition and Machine Intelligence (PReMI'05), pp. 660-665 (2005).
[17] K. Julisch, M. Dacier, Mining Intrusion Detection Alarms for Actionable Knowledge, Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 366-375 (2002).
[18] R. Agrawal, R. Srikant, Mining Sequential Patterns, International Conference Data Engineering, pp. 3-14 (1995).

Abhinav Srivastava is currently working towards his M.Tech (Information Technology) degree at the School of Information Technology, Indian Institute of Technology, Kharagpur, India. He received his B.Tech (Computer Science & Engineering) degree from the Harcourt Butler Technological Institute (HBTI), Kanpur, India in 2001.

He has worked as Member of Technical Staff in Persistent Systems Private Limited, Pune, India from 2001 to 2004. His research interests include Computer Security, Data Mining and Networks.

Shamik Sural received the B.E. degree in Electronics & Tele-communication Engineering from Jadavpur University, Calcutta, India in 1990, M.E. in Electrical Communication Engineering from Indian Institute of Science, Bangalore in 1992 and the Ph.D. degree from Jadavpur University in 2000.

He has worked in a number of companies belonging to the IT industry both in India as well as the USA in various capacities. Since 2002, he is an Assistant professor at the School of information Technology, Indian Institute of Technology, Kharagpur, India. Dr. Sural has served on the Program Committee and Executive Committee of a number of international conferences including International Database Engineering and Applications Symposium, IEEE Conference on Fuzzy Systems, Annual Computer Security Applications Conference, Asian Mobile Computing Conference, International Workshop on Distributed Computing and others. His research work has been funded by the Ministry of Communication and Information Technology, Department of Science and Technology, Govt. of India and the National Semiconductors Corporations, USA.

Dr. Sural is a member of the IEEE and the IEEE Computer Society. He has been elected as the Chairman of the IEEE Kharagpur Section for the year 2006. He has published more than fifty papers in reputed journals and conferences. His research interests include Database Security and Data Mining and Multimedia Database Systems.

A.K. Majumdar is a professor of the Computer Science and Engineering Department of the Indian Institute of Technology, Kharagpur, West Bengal. He served as the Head of the Computer Science and Engineering Department of IIT. Kharagpur from 1992 to 1995 and again from 1998 to 2001.

He received M. Tech and Ph. D. degrees from the University of Calcutta, in Applied Physics in 1968 and 1973, respectively. He also earned a Ph. D degree in Electrical Engineering from the University of Florida, Gainesville, Florida, USA, in 1976. Before joining the IIT Kharagpur in 1980, he served as a faculty member at the Indian Statistical Institute, Calcutta, and Jawaharlal Nehru University, New Delhi. He was a Visiting Professor in the Computing and Information Sciences Department of the University of Guelph, Canada in 1986 – 87.

Dr. Majumdar has more than 140 research publications in international journals and conferences. He has supervised several Ph. D and Masters' level theses. He is Fellow of the Institute of Engineers (India), Fellow of the Indian National Academy of Engineering and a Senior Member of the IEEE. His research interests include data and knowledge based systems, medical information systems, design automation and image processing.