Weighted Role Based Data Dependency Approach for Intrusion Detection in Database

Udai Pratap Rao[1], and Nikhil Kumar Singh[2]
(Corresponding author: Udai Pratap Rao)

Department of Computer Engineering, S. V. National Institute of Technology[1]
Surat (Gujarat) 395007, India
Department of Information Technology, UVPCE, Ganpat University[2]
Mehsana, (Gujarat), India
(Email: upr@coed.svnit.ac.in)

(Received Mar. 23, 2016; revised and accepted May 28 & June 10, 2016)

Abstract

In this paper, an approach is ascertained to detect malicious activities in RBAC (Role Based Access Control) enabled database.The proposed approach introduces weighted role based data dependency rule mining algorithm (WRBDDRM), that mines weighted role-wise data dependency rules from database log. The weights are intended to relate the sensitivity of attributes. The proposed algorithm also uses separate support and confidence for each role to generate the dependency rules and these data dependency rules are used to detect the malicious activities in the database. Transactions which disobey any of the data dependency rules are detected as malicious transactions.

Keywords: Database security, insider threats, role based access control, weighted support and confidence, sensitivity of attributes

1 Introduction

Many security steps have already been taken to prevent databases from intrusions [25]. Access control systems, intrusion detection system, authentication systems, antivirus software and firewalls are few examples of such security measures. In order to safeguard the databases from malicious actions, intrusion detection security measures have been widely considered to detect the malicious actions in the database. Such kind of security measures are exclusively targeted to database protection and are variant of basic IDS [3]. Intrusion detection is extensively used in different areas for detection of malicious or intrusive activity. The major areas for detection of intrusive activity are computer network [1, 14, 18, 22, 31], database [9, 34, 35], wireless sensor network [11, 23], software code [19], and electric power system [32] etc.. This paper is mainly focused on the detection of malicious activities in databases.

Currently, databases are the central component of many information systems, and therefore represents critical assets to organizations[25]. However,protecting databases from attacks presents unique bottle-neck [8, 9, 21, 39]. Well recognized and accepted security measures, such as anti-virus software, firewalls, access control methods, and file permissions, protect information systems at the network or operating system level but refrains protection against database specific attacks [6, 9, 20, 21, 25]. This happen due to the fact that actions which are malicious for DBMS may not be malicious for operating system and network. Network and operating system defense mechanisms are mainly designed to defend against external attackers [18, 20], but insider attacks are the primary threat in case of transaction-level database [25]. Recent database security research focuses on techniques to mitigate against insider attacks [6, 13, 21, 25]. The most basic level of insider attack is from authorized users, who do not posses database administrator rights [13, 21, 25]. The second category of insider threat is a group of authorized users who do not have database administrator rights, referred to as collaborators [7, 9]. A counter measure framed to protect the database from an aggregation of malicious queries may not detect those same queries if they are originated from collaborating users [9]. The last insider category is database administrators [20].

This paper synthesizes current research on database intrusion detection and proposes a database intrusion detection model that incorporates the key capabilities developed in earlier research [34, 40].

The rest of the paper is framed as follows: Section 2 reflects the related work. Section 3, contains motivation and contribution. Section 4 exhibits the related terms needed to understand the approach, transaction representation and system architecture of our proposed WRBDDRM approach. Section 5 contains learning and detection algorithms of WRBDDRM. Performance results and analysis are emphasized in Section 6. Finally, we conclude our work in Section 7.

2 Related Work

Database IDS mainly uses one of the two approaches: misuse detection and anomaly detection. The anomaly detection is more effective than misuse detection because it detects known and unknown attacks.

Chung et al. [10] proposed the system called DEMIDS (Detection of Misuse in Database Systems) in 2000. DEMIDS defines notion of distance measure. Pair wise shortest distance and schema distance of set of attributes is measured with the help of integrity constraints. Then frequent item sets are extracted from the database log with the help of distance measure. The frequent item sets are considered as the profiles of users, which can later be used by security officer to verify existing security policies. This approach gives the basic idea of database IDS and does not provide the explored view.

In 2000, Lee et al. [24] contemplated database intrusion detection system for real-time database systems. For detecting intrusions, they exploited real-time properties of data. They keep track of update rates of data objects, which are unknown to the intruder. This approach is exclusively applicable to real time database where time of access is utmost important.

Low et al. [26] suggested a fingerprint (signature) based approach in 2002. In this, the signature of legitimate transactions are stored where new transactions executed by the user are checked against previously stored fingerprints. Signature based approaches are only possible if there are fixed number of applications which can access database. In this approach, the signature in the form of fingerprint introduces the matching overhead and parallelly effects the accuracy.

Hu et al. [13] proposed data mining approach in 2004 to identify intrusion. They used rule based classification for the system. During learning phase frequent sequential patterns are extracted and used for generating classification rules viz. read rules and write rules. These rules are used during detection phase for detecting the intrusion. They focused particularly on malicious modification of data, while malicious read operation are not caught by their proposed approach.

Vieira et al. [43] proposed DBMTD (Data Base Malicious Transaction Detector) mechanism in 2005. The approach is signature based. They have done the manual profiling of transactions, which is not possible if number of valid transactions by applications is too high. Usefulness only for small and fixed number of applications is the major drawback of the approach.

Bertino et al. [8] proposed intrusion detection scheme for RBAC-enabled database in 2005. Naive bayes classifier is used for detecting intrusion in RBAC enabled database. They have proposed three levels of triplets that can be used to transform database log after processing. If role obtained by classification is same as original one which has executed the query then transaction is said to be a legitimate transaction.

Later in 2008, Kamra et al.[21] improved their approach by using quiplets instead of triplets. Drawback of their both approaches is that if database is not RBAC-administered then there is a need to maintain separate profiles for each user, which will add large memory overhead to store the profiles and execution time for classification. The approach is restricted to work at query level and not at transaction level is another major limitation of the work presented.

Srivastava et al. [40] proposed Weighed Data Dependency Rule Miner (WDDRM) algorithm in 2006 and improved the approach [16] by considering the sensitivity of attributes. This helps in extracting sequences with attributes which are sensitive but less accessed.

Mathew et al. [30] proposed data-centric approach to insider attack detection in database systems in 2010. Their experimental result shows that their technique is very effective, accurate, and is promising in complementing existing database security solutions. This approach was the first data-centric approach for detecting database intrusion.

In 2010, Rao et al. [33] improved the approach presented in [8] by extending it at transaction level. They have shown that their approach outperforms compared to query level approach [8]. In 2014, an approach for enhancing the detection rate in database Intrusion Detection System proposed [35]. This novel approach provides the flexibility in profile matching constraints. They are able to enhance the detection rate by reducing the false positive and false negative rate.

In 2015, Rao et al. [34] proposed an RBDDRM (Role based Data Dependency Rule Miner) approach for detection of database privilege abuse in RBAC (Role Based Access Control) administered database. This approach mines role wise data dependencies from database log which are considered as role profiles and are used to detect the privilege abuse by database users. The main focus is on the read, write, and conditional rules to strengthen the approach. The approach proposed in [34] is further improved by considering the sensitivity of attributes.

3 Motivation and Contribution

Database breaches have always been a threat to the privacy of individuals and organizations [17]. According to Verizon data breach investigation report of 2012, out of all attacks involving insider, 90% were malicious and was performed intentionally. It indicates that detection of malicious insider attack in the database is of great concern [42].

Insider attacks have not only grown frequently, but also found significantly more damaging to businesses than external attacks. In 2008, the Identity Theft Resource Center (ITRC) in the United States said that one in six breaches (7.7%) was attributed to insiders, more than twice of that found in 2007 (16%) [15]. The ITRC 2008 report reached 656 breaches in which 35,691,255 records were exposed by the end of 2008, reflecting an increase of 47% over last year's total of 446. The ITRC 2013 breach report reached 614 breaches, in which 91,982,92 records were exposed by the end of 2014 [16]. In its 2008 Data Breach Investigations Report, based on more than 500 forensic investigations of security breaches, Verizon Business found that half of all internal breaches were conducted by IT administrators [5]. The 2008 CSI Computer Crime and Security Survey [37] reported continuing trends in the frequency and severity of insider abuse and financial fraud. From 2004 to 2011, respondents consistently reported insider abuse as the second most frequent type of security incident [27, 28, 29, 36, 37, 38]. The specific rate of insider abuse incidents ranged from as low as 42% of all reported incidents [29] to as high as 59% [27, 36]. Financial fraud is another form of insider attack and accounted for 8% to 12% of reported incidents between 2004 and 2011 [27, 28, 29, 36, 37, 38]. Insider attacks are not just frequent but expensive too. The two insider related categories of computer security incidents named (i) insider abuse, and (ii) financial fraud account for a major portion of computer security losses [36, 37]. As per the statistics shown by Verizon 2010 data breach investigation report [4], it is clear that some effective method is required to detect the malicious activities in the database.

Rao et al. [34] inspired from the concept of Hu and Panda [13] and proposed role based data mining approach in which rules are generated for each role separately. Conditional rules are generated along with the read and write rules. In this approach no rules will be generated if some attributes are less frequent and more sensitive to be attacked. Using sensitivity of attributes we can improve the performance of the database IDS [34]. We propose the following measures to improve the performance of [34].

1) Proposed approach extracts data access dependencies based on the weighted scheme [40] that exists between the attributes of the database using [12]. In our approach, access dependencies are extracted separately for each role based on the sensitivity of the attribute.

2) We modify the definition of read, write and conditional rules given in [34]. These modified read, write and conditional rules present strict checking of user input transactions for malicious behavior. The definition of read, write, conditional rules in [13, 34] contain only one attribute on LHS. For example:

$$\begin{aligned} w(a) &\rightarrow r(b,c)r(b,d) \\ w(f) &\rightarrow c(k,d). \end{aligned}$$

In our approach, we define the read, write, and conditional rules in which one or more attributes are allowed on LHS. For Example:

$$\begin{aligned} w(a,e,f) &\rightarrow r(b,c)r(b,d) \\ w(e,f) &\rightarrow c(b,c). \end{aligned}$$

3) We use different support and confidence for each role based on the fact that how much dependency rules are required to detect the malicious pattern precisely. Each role in the DBMS is having different access pattern of database. If some role in the system accesses the database more frequently then higher support is required and if accesses of the database are less frequent then lower value of support is needed to get the desirable frequent sequences. Now, with the help of these frequent sequences generated for each role, we use separate confidence for that role, so that generated rules can effectively detect the actual behavior of the user input transaction. If the confidence value is high, then fewer rules will be generated, hence the chances of higher false negative rate. If the confidence value is less then more rules will be generated and the chances of higher false positive rate. So, we have chosen the confidence value separately for each role in such a way that lowers the FP and FN values. To extract such data dependencies, we use database access history which is assumed to be free from attacks. These dependencies are then used to extract dependency rules. These data dependencies which are in the form of access rules (different for each role) are used to detect the database insider abuse.

4 Weighted Role Based Data Dependency Rule Miner

We have used sequential pattern mining algorithm [2] for extracting data dependencies in the database system. Sequential patterns extracted are then converted to data dependency rules. We use the rule-based classification technique for detection of malicious activity. The rules generated at the end of our approach reflect data dependencies among attributes of the database.

Our approach is concerned more about insider attack and detects external attacks as well if an intruder disobeys any data dependency rules. Our work is based on relational database; we call our proposed algorithm as *WRBDDRM (Weighted Role Based Data Dependency Rule Miner)*.

4.1 Terminologies

In this section, we explain some of the formal definitions needed to understand the approach.

Read Operation: Read operation on a set of attributes is represented as $r(a_1, a_2, ..., a_n,)$. Read operation r_1, is said to be contained in read operation r_2, if r_2 have all attributes present in r_1 (means r_2 is the subset of

r_1). Read operation r_2 may have extra attributes that are not present in r_1.

Write Operation: Write operation on a set of attributes is represented as $w(a_1, a_2, \cdots, a_n)$. Write operation w_1 is said to be contained in write operation w_2, if w_2 has all attributes present in w_1. Write operation w_2 may have extra attributes that are not present in w_1.

Conditional Operation: Conditional operation on set of attributes means there is condition on those attributes in a query of transaction. For example in a query *"select name from student id=1"*, there is a condition operation on attribute *id*. There may be several attributes in one operation. Conditional operation is represented as $c(a_1, a_2, \ldots a_n)$.

Remark 1. *Attributes within one operation is an unordered set of attributes i.e. their sequence does not matter.*

Sequence: Sequence is an ordered list of one or more read, write or conditional operations. Each read, write or conditional operation in a sequence can have one or more attributes on which operation is believed to be performed simultaneously (sequence of attributes in an operation is irrelevant). Sequence is denoted as

$$< o_1(a_{11},\ a_{12}, \ldots\ a_{1w}),\ o_2(a_{21},\ a_{21}, \ldots\ a_{2x}),$$

$$o_3(a_{31},\ a_{32}, \ldots\ a_{3y}), \ldots\ o_n(a_{n1},\ a_{n2}, \ldots\ a_{nz})\ >$$

Here, o_i denotes read (r), write (w) or conditional (c) operation. a_{kl} denotes the attribute of the relation. A sequence $< o_{11},\ o_{12},\ o_{13}, \ldots\ o_{1n}\ >$ is said to be contained in another sequence $< o_{21},\ o_{22},\ o_{23}, \ldots\ o_{2m}\ >$, if there exist integers $i_1 < i_2 < i_3 < \ldots$ such that $o_{11} \subseteq o_{2i1}, o_{12} \subseteq o_{2i2},\ o_{13} \subseteq o_{2i3}, \ldots\ o_{1n} \subseteq o_{2in}$, and also o_{1k} and o_{2ik} must be same operation (read, write or conditional) where $1 <= k <= n$.

Support: The support for a sequence is defined as the fraction of total transactions that contain the sequence. Transaction is also the sequence of operations.

Read Sequence: Read sequence is a sequence with all operations as read operations except last operation which must be write operation; all read, write operations in a read sequence can have several attributes. Read sequence of set of attributes is denoted as

$$< r(a_{11},\ a_{12}, \ldots a_{1w}),\ r(a_{21},\ a_{22}, \ldots a_{2x}),$$

$$r(a_{31},\ a_{32}, \ldots a_{3y}), \ldots w(a_{n1},\ a_{n2} \ldots\ldots\ldots a_{nm})\ >$$

which represents that transaction may need to perform all read operations in order before the transaction updates attribute $(a_{n1},\ a_{n2} \ldots\ldots\ldots a_{nm})$.

Read Sequence Set (ReadSeqSet): Read sequence set is the collection of all read sequences.

Read Rule: Read rule is a rule with exactly one write operation on LHS and ordered sequence of read operation(s) on RHS. Read rule is denoted as,

$$w(a_{n1},\ a_{n2} \ldots a_{nm})\ \rightarrow r(a_{11},\ a_{12}, \ldots\ a_{1w}),\ r(a_{21},\ a_{22},$$

$$\ldots a_{2x}), \ldots\ r(a_{(n-1)1},\ a_{(n-1)2}, \ldots\ a_{(n-1)z})\ >$$

Remark 2. *A rule r_1 is said to be contained in rule r_2, if LHS of both the rules is same and operations on RHS of r_1 is subset of operations on RHS of r_2. Also, attributes of all operations on RHS of r1 must be the subset of attributes of corresponding operations on RHS of r_2 [34].*

Write Sequence: Write sequence is a sequence with all operations as write operations; all write operations in a write sequence can have several attributes except first write operation which must have exactly one attribute. Write sequence is denoted as,

$$<w(a_{11},\ a_{12} \ldots\ldots a_{1n}),\ w(a_{21},\ a_{22}, \ldots a_{2x}),$$

$$w(a_{31},\ a_{32}, \ldots a_{3x}) \ldots\ w(a_{n1},\ a_{n2}, \ldots a_{nz})\ >$$

which shows that transaction may need to perform all write operations in order after the transaction updates attribute $(a_{11},\ a_{12} \ldots\ldots a_{1n})$.

Write Sequence Set (WriteSeqSet): Write sequence set is collection of all write sequences.

Write Rule: Write rule is a rule with exactly one write operation on LHS and ordered sequence of write operation(s) on RHS. Write rule is denoted as,

$$w(a_{11},\ a_{12} \ldots\ldots a_{1n}) \rightarrow\ < w(a_{21},\ a_{22}, \ldots a_{2x}),$$
$$w(a_{31},\ a_{32}, \ldots a_{3x}) \ldots w(a_{n1},\ a_{n2}, \ldots a_{nz})\ >$$

Write rule can be easily generated from write sequence.

Conditional Sequence: Conditional sequence is a sequence of exactly two operations; in which first operation must be conditional operation on one or more attributes and second operation must be read or write operation on one or more attributes. Conditional sequence is denoted as $<$ c $(a_{11},\ a_{12}, \ldots\ a_{1x})$, r/w $(a_{21},\ a_{22} \ldots\ldots a_{2n})$ $>$ which represents that transaction may need to perform conditional operation on set of attributes immediately before the transaction read/write attributes ($a_{21},\ a_{22} \ldots\ldots a_{2n}$).

Remark 3. *A conditional sequence $<c_{11},\ o_{12}>$ (o_{12} here must be read or write operation) is said to be contained in another sequence $<o_{21},\ o_{22},\ o_{23}, \ldots o_{2m}>$, if there exist integers i, j=i+1 such that $c_{11} \subseteq o_{2i}$ and $o_{12} \subseteq o_{2j}$, and also o_{2i} is conditional operation and o_{2j} is same operation (read, write or conditional) as o_{12}.*

Conditional Sequence Set: Conditional sequence set is collection of all conditional sequences.

Conditional Rule: Conditional rule is a rule with exactly one read/write operation on LHS and exactly one conditional operation on RHS. Conditional rule is denoted as,

$$r/w(a_{21},\ a_{22}\ldots\ldots a_{2n}) -> c(a_{11},\ a_{12},\ldots a_{2x}).$$

Conditional rule can be easily generated from conditional sequence.

Confidence: Confidence of a rule can be defined as the fraction of support of the sequence from which rule is generated to the support of the operation on LHS.

Weighted Support and Confidence [40]: Few attributes in every database that are much important to be sensed or tracked for malicious modifications as compared to the other attributes. More the sensitivity of an attribute is, more is its weight. Srivastava et al.[40] have categorized the attributes in three sets: High Sensitivity (HS), Medium Sensitivity (MS), and Low Sensitivity (LS). The sensitivity of an attribute depends on the database application. From an integrity perspective, in addition, the sensitivity for modifications of attributes are more vital than reading them. Let x be the same attribute and if x ∈ HS then W ($\mathrm{x_w}$) > W ($\mathrm{x_r}$), in which W represents function of weight, $\mathrm{x_w}$ indicates writing or altering attribute x and $\mathrm{x_r}$ signifies attribute x reading.

We arrange all the attributes into the aforementioned three sets on the basis of their sensitivities and allocate numerical weights to each set; once schema is given for instance, say $w_1,\ w_2,\ w_3 \in \mathbf{R}$, is the real number set and $w_3 = w_2 = w_1$ are the weights of HS, MS and LS, respectively for each category. Lets assume d_1,d_2, d_3∈**R** are the additional weights associated with write operations, such that $d_3 =\ d_2\ = d_1$. Let read operation accesses the attribute x and w_1 denotes the weight associated with x. If write operation also accesses the attribute x then the weight associated to x will be $w_1 +\ d_1$ and can be represented as,

$$\begin{aligned} W(x_r) &= w_1 \\ W(x_w) &= w_1 +\ d_1. \end{aligned}$$

Let us assume a sequence s with weight w_s and let N be the total number transactions. If s is present in n transactions out of N transactions, then the support of sequence s can be given as [40],

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

Suppose R be a rule of read operation having the form $a_{jw} \rightarrow a_{1r}, a_{2r}, \cdots, a_{kr}$ produced from the read sequence $rs \in ReadSeqSet$. Let $Count(a_{jw})$ and $Count(rs)$ be the aggregate count of the attribute a_{jw} and that of rs among the total transactions. The weighted confidence of the rule R is defined as [40],

$$Confidence\ (C_R) = Count\ (rs)\ /\ Count\ (a_{jw}).$$

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

$$\begin{aligned} Count(a_{jw}) = & \sum_{\forall Transaction \neq T, a_{jw} \in T and rs T} (w3 + d3) \\ & + \sum_{\forall Transaction T,\ rs \in T} max(W(rs)). \end{aligned}$$

$Count(rs)$ is defined as,

$$Count(rs) = \sum_{\forall Transaction T, rs \in T} max(W(rs)).$$

Maximum disobeyed confidence: Maximum disobeyed confidence finds the severity of the malicious activity by using the confidence of rules [34]. It uses maximum function to get the highest confidence from confidences of disobeyed rules. The maximum confidence from disobeyed rules shows the severity of the malicious activity. Let there is a rule $\mathbf{r(a)} \rightarrow \mathbf{c(b)}$ with 70% confidence. This means that out of all transactions reading attribute 'a', 70% of transactions have conditional operation on attribute 'b' immediately before reading attribute a. While 30% of transaction that does not support this rule; do not have conditional operation on attribute b immediately before reading attribute a. So, transaction detected as malicious might be from these 30% of transactions which are not malicious. But, if transaction disobeys rule with 100% confidence then there are more chances of that transaction being malicious.

4.2 Transaction Representation

Transactions from the database log (during learning phase) and transactions from users (during the detection phase) need to be preprocessed. After the pre-processing it is represented in the format needed by our approach which is similar to the representation used by [34]. It can be better understood by following example.

```
Select  a, b, c, d
from    table_name
where   e = "xyz" and f = "abc"

Update  table_name
set  a = "pqr", e = "xyz"
where  c = "abc"
```

Above transaction of two queries, after pre-processing is represented as,

$$< c(e,\ f),\ r(a,\ b,\ c, d),\ c(c),\ w(a,\ e)\ >$$

.

Attributes used by WHERE clause are considered as conditional operation on those attributes.

4.3 Proposed System Architecture

The proposed system architecture as given in Figure 1 has two phases: learning phase and detection phase. In the learning phase, the database log is used to mine role profiles where the transactions are converted to a representation as discussed in Section 4.2. After completion of pre-processing, preprocessed transactions are fed to proposed algorithm to extract the role-profiles. These profiles are then stored for later use and represents the normal behavior of the role.

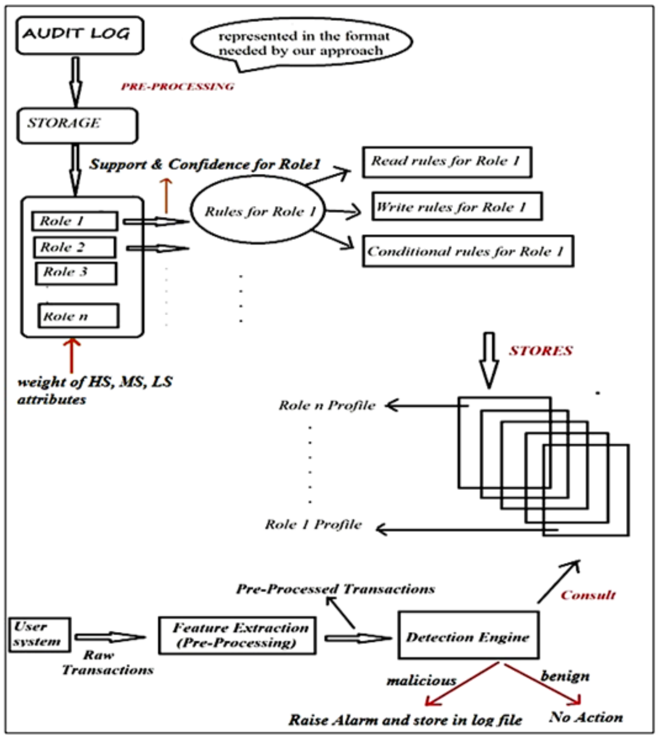

AUDIT LOG
represented in the format needed by our approach
PRE-PROCESSING
STORAGE
Support & Confidence for Role1
Rules for Role 1
Read rules for Role 1
Write rules for Role 1
Conditional rules for Role 1
Role 1
Role 2
Role 3
Role n
weight of HS, MS, LS attributes
STORES
Role n Profile
Role 1 Profile
Pre-Processed Transactions
Consult
User system
Raw Transactions
Feature Extraction (Pre-Processing)
Detection Engine
malicious
benign
Raise Alarm and store in log file
No Action

Figure 1: Proposed system architecture

In the detection phase, new user transaction is checked when it is executed. The transaction from the user is preprocessed first and converted to the same presentation used during the learning phase. The preprocessed transaction is given as input to the detection engine which consults the role profiles and checks the transaction against all the rules to the corresponding role of the users. If the transaction disobeys any rule, an alarm is raised (entry in the log of possible abuses). Otherwise, it is considered as normal transaction and no action is taken in such case.

5 The Algorithm

Our proposed approach works in two phases viz. learning phase and detection phase.

5.1 Learning Phase

Preprocessed transactions are fed to the proposed WRBDDRM learning algorithm after feature extraction from the database log. The steps of this algorithm are described in Algorithm 1. At the end of WRBDDRM learning algorithm, role wise profiles are generated and stored on the permanent storage. Terminologies for WRBDDRM learning algorithm are listed in Table 1.

Algorithm 1 WRBDDRM Learning Algorithm

```
1: Input: Set of preprocessed transactions from the database log; $T$
2: Output: Role wise rules i.e. read, write and conditional rules for each role
3: Initialize sensitivity group of attributes and for each role $k$, $1 \le k \le n$
4: Initialize $T_k = \{\ \}$
5: for each transaction t in T do
6:     Insert t to$T_k$, where k is role who executed t
7: end for
8: for all role k, where $|T_k| > 0$ do
9:     Initialize $RS_k = \{\}$, $WS_k = \{\}$, $CS_k = \{\}$, $RR_k = \{\ \}$, $WR_k = \{\ \}$, $CR_k = \{\ \}$
10:    Generate sequential pattern $x_i = Weighted\_AprioriAllalgorithm(T_k, min\_support_k)$
11: end for
12: for all sequential pattern $x_i$, where $|x_i|>1$ do
13:    if $|x_i|=2$, If the last operation in $x_i$ is write or read operation and first operation in $x_i$ is conditional then
14:        add $x_i$ to $CS_k$
15:    end if
16:    if last operation in $x_i$ is write and all other operations in $x_i$ are read then
17:        add $x_i$ to $RS_k$
18:    end if
19:    if first operation in $x_i$ is write and all other operations in $x_i$ are also write then
20:        add $x_i$ to $WS_k$
21:    end if
22: end for
23: for every sequence s in $CS_k$ do
24:    if $weighted\_confidence$ of conditional rule r generated from sequence s >$min\_confidence_k$ then
25:        if If no rule in $CR_k$ (with $weighted\_confidence$ equal to$weighted\_confidence$of r) contains r then
26:            Add rule r to $CR_k$
27:        end if
28:        Delete the rules from $CR_k$ which has same $weighted\_confidence$ as r and are contained in r
29:    end if
30: end for
31: for every sequence s in $RS_k$ do
32:    if $weighted\_confidence$of read rule r generated from sequence s >$min\_confidence_k$ then
33:        if no rule in $RR_k$ (with $weighted\_confidence$ equal to $weighted\_confidence$of r) contains r then
34:            Add rule r to $RR_k$
35:        end if
36:    end if
37:    Delete the rules from $RR_k$ which has same $weighted\_confidence$ as r and are contained in r
38: end for
39: for every sequence s in $WS_k$ do
40:    if $weighted\_confidence$ of write rule r generated from sequence s >$min\_confidence_k$ then
41:        if no rule in $WR_k$ (with $weighted\_confidence$ equal to $weighted\_confidence$ of r) contains r then
42:            Add rule r to $WR_k$
43:        end if
44:        Delete the rules from $WR_k$ which has same $weighted\_confidence$ as r and are contained in r
45:    end if
46:    Store $RR_k$, $WR_k$, and $CR_k$ to the permanent storage
47: end for
```

WRBDDRM learning algorithm generates role wise rules based on the sensitivity of attributes while considering separate support and confidence for each role.

Weighted AproriAll algorithm is used to gener-

Table 1: Terminologies

T	Set of all preprocessed transactions from the database log;
n	Different types of roles in the application;
T_k	Set of preprocessed transactions executed by role k;
$\lvert T_k \rvert$	Number of transactions executed by role k;
RS_k	Set of read sequences mined for role k;
WS_k	Set of write sequences mined for role k;
CS_k	Set of conditional sequences mined for role k;
RR_k	Set of read rules generated for role k;
WR_k	Set of write rules generated for role k;
CR_k	Set of conditional rules generated for role k;
X	Sequential patterns;
$min\_support_k$	Minimum support defined for role k;
$min\_confidance_k$	Minimum confidence defined for role k;
x_i	One out of many sequential patterns;
$\lvert x_i \rvert$	Length of sequential pattern x_i.

ate sequential patterns for role k with minimum supportmin$_support_k$. The steps of this algorithm are described in Algorithm 2.

Algorithm 2 Weighted AprioriAll algorithm

```
1: Input: Transactions of role k ($T_k$) and minimum support of role k ($min\_support_k$)
2: Output: Sequential pattern of role k having minimum support as $min\_support_k$
3: $\mathbf{L_1}$ = {large 1-sequences}
4: k=2
5: for ($\mathbf{L_{k-1}} \neq \emptyset$) do
6:     $\mathbf{C_k}$ = Candidate_Gen ($\mathbf{L_{k-1}}$) // candidates generation algorithm given below
7: end for
8: for each sequence s in the dataset do
9:     for each candidate in $\mathbf{C_k}$ do
10:        $\mathbf{n_{c_k}}$= Increment the count that is contained in s
11:        $\mathbf{Wc_k}$ = weight of each candidate by using the concept in equation 1 and 2
12:        Candidate_Support = weighted support of candidate with weight $\mathbf{Wc_k}$ and $\mathbf{n_{c_k}}$
13:        $\mathbf{L_k}$ = Candidates in $\mathbf{C_k}$ with Candidate_Support>min_support
14:    end for
15: end for
```

Candidate generation algorithm is used in step 6 of Weighted AproriAll algorithm for generating new candidates. The steps of candidate generation algorithm are described in Algorithm 3.

5.2 Detection Phase

In the detection phase, new user transaction is preprocessed and fed to detection engine. The detection engine reads the stored rules (outcome of learning phase) and

Algorithm 3 Candidate Generation algorithm

```
1: Input: Set of all large (k-1)-Sequences i.e. $L_{k-1}$
2: Output: Set of all candidate k-Sequences i.e. $C_k$
3: Insert into $C_k$
4: Select $p.litemset_1, \ldots, p.litemset_{k-1}, q.litemset_{k-1}$
5: From $L_{k-1}p$, $L_{k-1}q4$. Where $p.litemset_1 = q.litemset_1 \ldots\ldots$
       $p.litemset_{k-2} = p.litemset_{k-2};$
6: Delete all sequences $C_k$ such that some (k-1) subsequences of c is not in $L_{k-1}$
```

checks new user transaction against dependency rules of related role (role of the user who has executed the transaction). If the transaction is compliant with the rules then it is normal otherwise malicious and an alarm is raised. Raising of alarm means an entry is made to log of probable attacks. The steps of detection phase algorithm are described in Algorithm 4.

5.3 Methodology of WRBDDRM Algorithm

Learning phase and detection phase can be best understood by an example. Table 2 and Table 3 show the preprocessed transactions from the database log which are executed by Role 1 and Role 2. We have considered *(a, e, s, l, r, m)* as high sensitive attributes, *(d, j, h, i)* as medium sensitive attribute, and *(b, c, f, g, h, i, m, o, p, q)* as low sensitive attributes. Weight for high sensitive, medium sensitive, and low sensitive attributes are considered as 3, 2, and 1 respectively. Rules for Role 1 is generated by considering support 40% and confidence as 75 % and rules for Role 2 are generated by considering support 45% and confidence as 70%.

Table 4 shows the strong association read, write and

Algorithm 4 WRBDDRM Detection Algorithm

```
1: Input: Preprocessed user input transactions
2: Output: Malicious or normal transaction
3: for For every role k do
4:   Initialize $RR_k$={}, $WR_k$={}, $CR_k$={}
5:   Retrieve $RR_k$, $WR_k$, $CR_k$ from permanent storage to memory
6: end for
7: for each read operation in t do
8:   for every attribute a of the read operation do
9:     for every rule r for attribute a in $CR_k$ do
10:      if r is disobeyed and $max\_disobeyed\_confidence < weight\_confidence(r)$ then
11:        $max\_disobeyed\_confidence = weight\_confidence(r)$
12:      end if
13:    end for
14:  end for
15: end for
16: for each write operation in t do
17:   for For every attribute 'a' of write operation do
18:     if If r is disobeyed && $max\_disobeyed\_confidence< weigh\_cofidence(r)$ then
19:       $max\_disobeyed\_confidence = weight\_confidence(r)$
20:     end if
21:     if $max\_disobeyed\_confidence \neq 0$ then
22:       Add the entry to log of possible attacks along with $max\_disobeyed\_confidene$
23:     end if
24:   end for
25: end for
```

Table 2: Example transactions for role 1

T1	<[e,a] r[a,b] c[a] w[d,c] r[e,c] w[d,a]>
T2	<[a,d] w[a,b,c]>
T3	<[d,a] w[e,c] w[a]>
T4	<[e,a] r[b] c[d] w[b,c] r[d,e,c] w[d,a]>
T5	<[d,e,a] r[e,b] c[b,c] w[e,c]>

Table 3: Example transactions for role 2

T1	<[g,h] r[p] w[r] c[q] r[g,i,m] r[f,h,k]>
T2	<[k] r[g,l] c[i,m] r[g,h,l] w[n]>
T3	<[h] r[f,h,k] w[f,n] r[g] c[i] w[l]>
T4	<[p] r[p] w[q] r[g] c[h,k] w[i] r[k] r[f,h]>
T5	<[g,i,m] r[g] c[i,q] w[i] r[g] c[i,k] w[l] r[g,l,>

conditional rules generated for Role 1 and Role 2. Number of strong association conditional rules are 21, number of strong association read rules are 12, and the number of strong association write rules are 4.

Now with the help of rules as shown in Table 4, new input transactions can be classified as normal or malicious. Let, new user transaction is $<c\,[d,a]\,w\,[e,c]\,w\,[a]>$. As

Table 4: Generated rule set

Rule ID	Confidence	RuleSet
	Conditional RuleSet (CR_k)	
R1	100%	r[(e b)] => c[(d e a)]
R2	75%	r[(b)] => c[(d e a)]
R3	100%	w[(e)] => c[(d a)]
R4	100%	w[(e c)] => c[(d a)]
R5	100%	r[(a b)] => c[(e a)]
R6	100%	r[(e)] => c[(e a)]
R7	88%	w[(c)] => c[(a)]
R8	100%	r[(f)] => c[(h)]
R9	100%	r[(f h)] => c[(h)]
R10	100%	r[(f h k)] => c[(h)]
R11	100%	r[(f k)] => c[(h)]
R12	75%	r[(h)] => c[(h)]
R13	100%	r[(h k)] => c[(h)]
R14	100%	r[(k)] => c[(h)]
R15	100%	w[(l)] => c[(i)]
R16	100%	r[(g h l)] => c[(i m)]
R17	100%	r[(g l)] => c[(i m)]
R18	100%	r[(h l)] => c[(i m)]
R19	100%	r[(l)] => c[(i m)]
R20	100%	r[(g l)] => c[(k)]
R21	100%	r[(l)] => c[(k)]
	Read RuleSet (RR_k)	
R22	100%	w[(a b)] => r[(a d)]
R23	100%	w[(a b c)] => r[(a d)]
R24	100%	w[(a c)] => r[(a d)]
R25	75%	w[(b)] => r[(a d)]
R26	100%	w[(d c)] => r[(a b)]
R27	100%	w[(d)] => r[(b)] r[(e c)]
R28	100%	w[(d a)] => r[(b)] r[(e c)]
R29	100%	w[(f n)] => r[(f h k)]
R30	74%	w[(n)] => r[(f h k)]
R31	100%	w[(i)] => r[(g)]
R32	100%	w[(l)] => r[(g)]
R33	77%	w[(n)] => r[(g l)] r[(g h l)]
R34	100%	w[(a b)] => r[(a d)]
	Write RuleSet (WR_k)	
R35	100%	w[(d c)] => w[(d a)]
R36	76%	w[(b)] => w[(d a)]
R37	75%	w[(c)] => w[(a)]
R38	100%	w[(f n)] => w[(l)]

the first operation is conditional operation in the transaction so there are no rules for it. Second operation is write on attributes e and c. One conditional rule R4 is present for write operation on e and c together and one conditional rule R3 is present for the write operation on e. For the given transaction, both the rules R4 and R3 are obeyed. Moving further, third operation is the write on attribute 'a' for which rule set contains no rule. As the transaction $<c\,[d,a]\;w\,[e,c]\;w\,[a]>$ obeys all the rules, therefore, it is considered as normal transaction.

Now, consider a new transaction <c(a), w(c), w(b)>. In this first operation is conditional operation on attribute 'a', therefore no rule on conditional operation. Second operation is the write operation on attribute 'c'. R7 and R37 are rules of write operation on the attribute 'c'. New transaction follows rule R7. But rule R37 is not followed, so new transaction will be detected as malicious and max-disobeyed-confidence is equal to confidence of R37 which is equal to 75%. Similarly, on attributes 'b' last operation is the write operation . R2, R25, and R36 are write rules on attribute 'b'. New transaction does not follow the rule R2, which has confidence equal to 75% that is equal to max-disobeyed-confidence. So, max-disobeyed-confidence will remain same i.e. 75%. Now rule R25 is also not followed by new transaction which has confidence equal to 75% and is equal to max-disobeyed-confidence. The max-disobeyed-confidence will remain same i.e. 75%. Now rule R36 is also not followed by new transaction which has confidence equal to 76% and greater than max-disobeyed-confidence. The max-disobeyed-confidence will be updated from 75% to 76%.

6 Performance Results and Analysis

In this section, the comparison between our approach (WRBDDRM) and existing approach (RBDDRM) [34] is presented.

6.1 Experimental Setup

We use Java programming language for implementation and testing of our approach using Net Beans IDE 7.4. We use TPC-C (online transaction processing benchmark) [41] database schema. We have considered only two roles in the system viz. customer and administrator which are represented by Role1 and Role2 in our implementation.

The synthetic dataset is used to evaluate the performance. We manually generated 40 genuine transactions executed by different roles. Synthetic dataset is generated from 40 genuine transactions which are populated to 600 genuine transactions randomly. While randomly populating transactions, we consider the frequency with which both the roles interact to the database. In our system, Role 1 is the customer who interacts with database more frequently than the Role 2; an administrator. So while populating, frequency of execution of any transaction by Role 1 is more than the frequency of execution of any transaction by Role 2. Attacks are generated by randomly changing the some attributes in the operations of benign transaction by another operation of the same relation (same relation of schema). In this way, 100 malicious transactions are generated. Results are taken on 100 benign and 100 malicious transactions to evaluate false negatives, true negative, true positives, false positive and recall value.

6.2 Performance Results

The statistical results of the metrics: true negative, false positive, false negative, true positive and recall for RBDDRM [34] and WRBDDRM are shown in Table 5 and Table 6 respectively. We have shown here five instances of both the approaches to make the analysis clear. In RBDDRM [34] and WRBDDRM, we have varied minimum support value from 20% to 45% and minimum confidence value from 60 to 75. In proposed WRBDDRM, we vary the sensitivity of attributes i.e. HS, MS, and LS in the range of 1 to 4 to get the results. Instances for RBDDRM [34] are represented as Id# and instances for WRBDDRM is represented as WId# in Table 5 and Table 6 respectively. Role 1 and Role 2 are represented as R1 and R2 in Tables 5 and 6 respectively.

The values of True Negative, False Positive, False Negatives and True Positives are in terms of percentage. Recall Value is shown on the scale of 0 to 1.

6.3 Analysis

As there is no common fix parameters in both the approaches; RBDDRM [34] and WRBDDRM, therefore we can not compare them directly. In RBDDRM [34] support and confidence for all the roles are same while in our proposed approach, for each role support and confidence are different. WRBDDRM also uses different sensitivity parameters for the attributes. We have taken different instances of both the approaches on the basis of their respective parameter values for comparison.

Both RBDDRM [34] and WRBDDRM are compared by taking instances of respective approaches, i.e. Id# and WId# on X-axis and performance evaluation metrics on the Y-axis. The same is presented in Figures 2, 3, 4, 5, and 6 respectively.

From the graphs, we can see that Id1 is an instance of RBDDRM [34] in which support and confidence are 30 and 60 respectively. WId1 is an instance of the WRBDDRM in which support, confidence for Role 1 and Role 2 are 30, 60 and 35, 65 respectively. Weights for HS, MS, and LS attributes are 2, 1.5, and 1 respectively. Here, we see that support and confidence of Role 1 is same in both the approaches and support and confidence for Role 2 in our approach is more than RBDDRM [34]. Due to this, less number of rules are generated compared to RBDDRM [34]. Weights are associated with the attributes

Table 5: Results of RBDDRM algorithm [34]

Instances of RBDDRM	Support (%)	Confidence(%)	TN(%)	FP(%)	FN(%)	TP(%)	Recall
	R1&R2	R1&R2					
Id1	30	60	45	55	47	53	.53
Id2	35	60	55	45	47	53	.53
Id10	25	75	73	27	53	47	.47
Id11	25	60	55	45	38	62	.62
Id13	45	70	82	18	67	33	.33

Table 6: Results of proposed WRBDDRM algorithm

Instances of WRBDDRM	Minimum Support (%)		Minimum Confidence(%)		Weight of Attributes			TN (%)	FP (%)	FN (%)	TP (%)	Recall
	R1	R2	R1	R2	HS	MS	LS					
WId1	30	35	60	65	2	1.5	1	55	45	41	59	.59
WId2	30	35	60	65	2.5	2	1	43	57	38	62	.62
WId10	25	20	60	55	2	1.5	1	34	66	7	93	.93
WId11	25	20	55	55	2	1.5	1	38	62	10	90	.90
WId13	45	45	70	70	2	1.5	1	55	45	44	56	.56

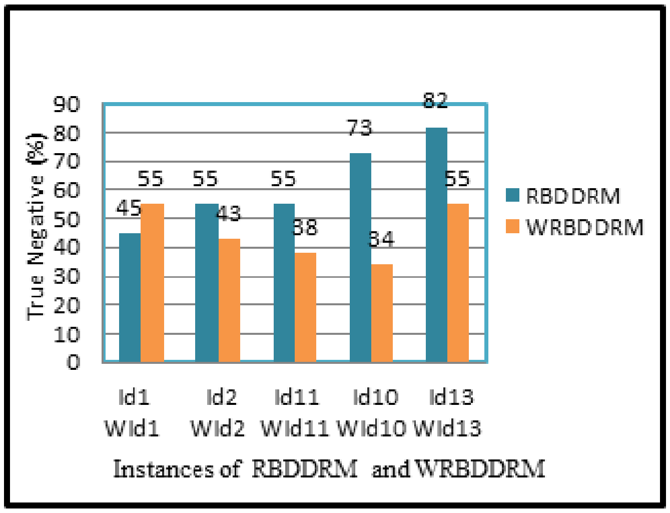


Figure 2: TN Vs various instances of both the approaches

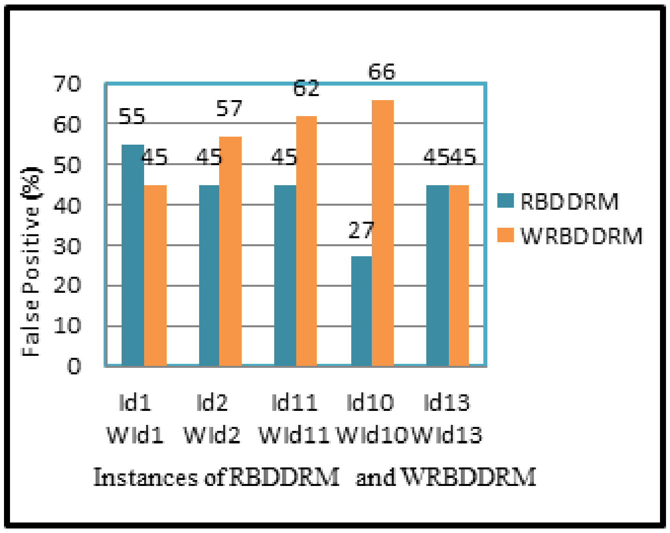


Figure 3: FP Vs various instances of both the approaches

on the basic of their sensitivity and relatively more rules will be generated that too results in increased recall value. Similarly, we can compare for other instances of both the approaches for any performance evaluation metrics used.

Figure 2 and Figure 3, shows the comparison of RBDDRM approach [34] and WRBDDRM approach for true negative and false positive respectively. Figure 4 and Figure 5 show the comparison of false negatives and true positives of both the approaches.

It is also observed from the graph as shown in Figure 3 that false positives of our approach are more than the existing RBDDRM [34] approach; but there is a significant improvement in case of false negatives as shown in Figure 4. False negatives in WRBDDRM are lesser and true positive rate is higher than RBDDRM [34]. Due to the lower false negative rate and higher false positive rate, attack detection capability of WRBDDRM is reasonably high. Comparisons for attack detection capabilities i.e. recall value of our approach and RBDDRM [34] is shown in Figure 6. Improvement in case of false positives for our approach is not adequate because of generating more conditional rules due to the consideration of the attributes' sensitivity. This can be reduced by considering the separate support and confidence for conditional rules.

From the graph as shown in Figure 6, we can say that, for every similar type of instance in both the approaches, recall value of WRBDDRM is higher than the RBDDRM

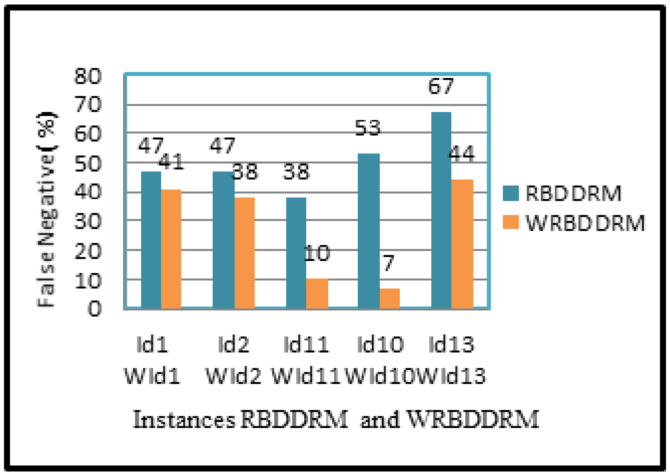


Figure 4: FN Vs various instances of both the approaches

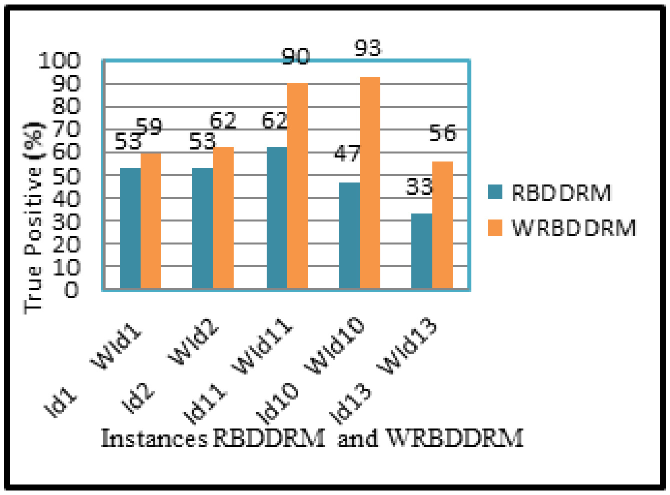
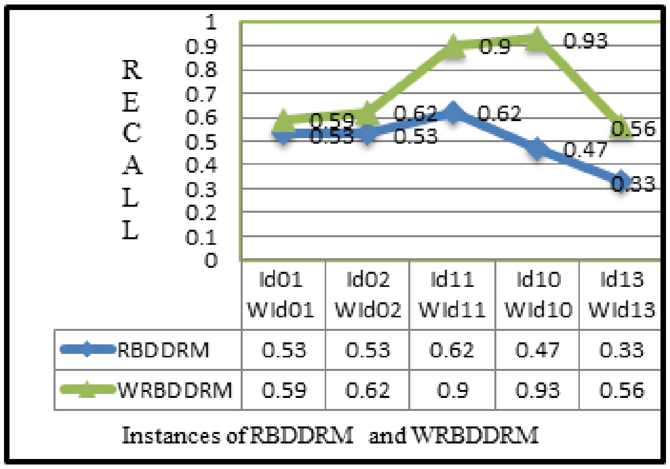


Figure 5: TP Vs various instances of both the approaches



	Id01 WId01	Id02 WId02	Id11 WId11	Id10 WId10	Id13 WId13
RBDDRM	0.53	0.53	0.62	0.47	0.33
WRBDDRM	0.59	0.62	0.9	0.93	0.56

Figure 6: Recall Vs various instances of both the approaches

approach [34].

7 Conclusion

The proposed approach incorporated weighted data dependency rules to strengthen the security of database IDS. By analyzing the experimental results, it is clearly observed that our approach WRBDDRM outperforms in terms of attack detection capability i.e. recall value compared to RBDDRM algorithm.

Acknowledgments

This research work is supported by Institute Research Grant (Ref. No.: Dean (R&C)/1503/2013-14, dated: 17-02-2014) of S. V. National Institute of Technology Surat (Gujarat) 395007-India.

References

[1] M. H. Aghdam and P. Kabiri, "Feature selection for intrusion detection system using ant colony optimization," *International Journal of Network Security*, vol. 18, no. 3, pp. 420–432, 2016.

[2] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the Eleventh IEEE International Conference on Data Engineering*, pp. 3–14, 1995.

[3] R. Bace and P. Mell, "NIST special publication on intrusion detection systems," *Technical Report, DTIC Document*, 2001.

[4] W. Baker, M. Goudie, A. Hutton, C. D. Hylender, C. Novak, D. Ostertag, C. Porter, and M Rosen, "2010 data breach investigations report," *Verizon RISK Team*, 2010.

[5] T. Barbaro and M. Andzeller, *2008 Data Breach Investigations Report*, 2008. (`http://www.verizonenterprise.com/resources/security/databreachreport.pdf`)

[6] E. Bertino, S. Jajodia, and P. Samarati, "Database security: research and practice," *Information Systems*, vol. 20, no. 7, pp. 537–556, 1995.

[7] E. Bertino, T. Leggieri, and E. Terzi, "Securing dbms: characterizing and detecting query floods," in *Proceedings of the 2004 Information Security Conference*, pp. 195–206, Springer, 2004.

[8] E. Bertino, E. Terzi, A. Kamra, and A. Vakali, "Intrusion detection in RBAC-administered databases," in *Proceedings of 21st IEEE Annual Computer Security Applications Conference*, pp. 1–10, 2005.

[9] A. S. Chikhale, and S. S. Dhande, "Protection of data base security via collaborative inference detection," *International Journal of Advanced Research*, vol. 3, no. 2, pp. 665–670, 2015.

[10] C. Y. Chung, M. Gertz, and K. Levitt, "Demids: A misuse detection system for database systems," in *Integrity and Internal Control in Information Systems*, pp. 159–178. Springer, 2000.

[11] L. Coppolino, S. D. Antonio, A. Garofalo, and L. Romano, "Applying data mining techniques to intrusion detection in wireless sensor networks," in *Proceedings of IEEE 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 247–254, 2013.

[12] S. Hashemi, Y. Yang, D. Zabihzadeh, and M. Kangavari, "Detecting intrusion transactions in databases using data item dependencies and anomaly analysis," *Expert Systems*, vol. 25, no. 5, pp. 460–473, 2008.

[13] Y. Hu and B. Panda, "A data mining approach for database intrusion detection," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, pp. 711–716, 2004.

[14] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Computer Communications*, vol. 49, pp. 1–17, 2014.

[15] Identity Theft Resourse Center, *2008 Data Breach Insider Theft Category Summary,* 2008. (http://www.idtheftcenter.org)

[16] Identity Theft Resourse Center, *2013 Data Breach Insider Theft Category Summary,* 2013. (http://www.idtheftcenter.org/images/breach/Insider_Theft_Summary_2013.pdf)

[17] Info Security, *22 Million User IDS May Have Been Stolen From Yahoo Japan,* 2013. (http://www.infosecurity-magazine.com/view/32498/22-million-user-ids-may-have-been-stolen-//from-yahoo-japanutm_medium=twitterutm_source=twitterfeed)

[18] J. Jabez and B. Muthukumar, "Intrusion detection system (ids): Anomaly detection using outlier detection approach," *Procedia Computer Science*, vol. 48, pp. 338–346, 2015.

[19] H. Jelodar, J. Aramideh, "Common techniques and tools for the analysis of open source software in order to detect code clones: A study," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 64–69, 2014.

[20] A. Kamra and E. Bertino, "Design and implementation of an intrusion response system for relational databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 875–888, 2011.

[21] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," *The VLDB Journal*, vol. 17, no. 5, pp. 1063–1077, 2008.

[22] D. Kumar and N. Kumar, "An approach for collaborative decision in distributed intrusion detection system," *International Journal of Computer Applications*, vol. 133, no. 13, pp. 8–14, 2016.

[23] M. Kumar, K. Dutta, I. Chopra, "Impact of wormhole attack on data aggregation in hieracrchical WSN," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 70–77, 2014.

[24] V. Lee, J. Stankovic, and S. H. Son, "Intrusion detection in real-time database systems via time signatures," in *Proceedings of 6th IEEE Real-Time Technology and Applications Symposium*, pp. 124–133, 2000.

[25] P. Liu, "Architectures for intrusion tolerant database systems," in *Proceedings of IEEE 18th Annual Computer Security Applications Conference*, pp. 311–320, 2002.

[26] W. L. Low, J. Lee, and P. Teoh, "Didafit: Detecting intrusions in databases through fingerprinting transactions," in *Proceedings of 4th International Conference on Enterprise Information Systems* , pp. 121–128, 2002.

[27] W. Lucyshyn, L. A. Gordon, M. P. Loeb, and R. Richardson, "2004 CSI/FBI computer crime and security survey," *Computer Security Institute*, 2004.

[28] W. Lucyshyn, L. A. Gordon, M. P. Loeb, and R. Richardson, "2005 CSI/FBI computer crime and security survey," *Computer Security Institute*, 2005.

[29] W. Lucyshyn, L. A. Gordon, M. P. Loeb, and R. Richardson, "2006 CSI/FBI computer crime and security survey," *Computer Security Institute*, 2006.

[30] S. Mathew, M. Petropoulos, H. Q. Ngo, and S. Upadhyaya, "A data-centric approach to insider attack detection in database systems," in *Proceedings 13th International Symposium on Recent Advances in Intrusion Detection*, pp. 382–401, Springer, 2010.

[31] G. V. Nadiammai and M. Hemalatha, "Effective approach toward intrusion detection system using data mining techniques," *Egyptian Informatics Journal*, vol. 15, no. 1, pp. 37–50, 2014.

[32] S. Pan, T. H. Morris, and U. Adhikari, "A specification-based intrusion detection framework for cyber-physical environment in electric power system.," *International Journal of Network Security*, vol. 17, no. 2, pp. 174–188, 2015.

[33] U. P. Rao, G. J. Sahani, and D. R. Patel, "Machine learning proposed approach for detecting database intrusions in rbac enabled databases," in *Proceedings of IEEE International Conference on Computing Communication and Networking Technologies (ICCCNT'10)*, pp. 1–4, 2010.

[34] U. P. Rao and N. K. Singh, "Detection of privilege abuse in RBAC administered database," in *Intelligent Systems in Science and Information*, pp. 57–76, Springer, 2015.

[35] U. P. Rao, N. K. Singh, A. R. Amin, and K. Sahu, "Enhancing detection rate in database intrusion detection system," in *Proceedings of IEEE Science and Information Conference (SAI'14)*, pp. 556–563, 2014.

[36] R. Richardson, "The 12th annual computer crime and security survey," *Computer Security Institute*, pp. 1–30, 2007.

[37] R. Richardson, "The 13th Csi computer crime and security survey," *Computer Security Institute*, pp. 1–30, 2008.

[38] R. Richardson, "15th annual 2010/2011 computer crime and security survey," *Computer Security Institute*, pp. 1–44, 2011.

[39] A. C. Squicciarini, I. Paloscia, and E. Bertino, "Protecting databases from query flood attacks," in *Proceedings of IEEE 24th International Conference on Data Engineering*, pp. 1358–1360, 2008.

[40] A. Srivastava, S. Sural, and A. K. Majumdar, "Weighted intra-transactional rule mining for database intrusion detection," in *Proceedings of the 10th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 611–620, Berlin, Heidelberg, 2006.

[41] TPC, *TPC Benchmark C, Standard Specification, Ver. 5.1*, July 11, 2016. (`http://www.tpc.org/tpcc`)

[42] Verizon, *Data Breach Investigations Report*, 2012. (`http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2012-ebk_en_xg.pdf`)

[43] M. Vieira and H. Madeira, "Detection of malicious transactions in dbms," in *Proceedings of IEEE 11th Pacific Rim International Symposium on Dependable Computing*, pp. 1–8, 2005.

Udai Pratap Rao is currently an Assistant Professor at the Department of Computer Engineering, S.V. National Institute of Technology Surat, Gujarat, India. He obtained his Ph.D. degree in Computer Engineering in 2014. His research interests include data mining, database security, information security & privacy, and big data analytics.

Nikhil Kumar Singh obtained his master's degree in Computer Engineering from S. V. National Institute of Technology Surat, Gujarat, India. He has completed his B.Tech in Computer Science and Engineering from Institute of Engineering and Rural Technology, Allahabd, India. Nikhil Kumar Singh is currently an assistant professor at U. V. Patel College of Engineering, Ganpat University, India. He has research interests in computer networks, data security, web mining and network security.