A dubiety-determining based model for database cumulated anomaly intrusion

Junkai YI [1], Gang LU [1], Kevin LÜ[2]

[1]School of Information Science and Technology, Beijing University of Chemical Technology
15 BeiSanhuan East Road, ChaoYang District, Beijing, 100029, China
[2]Brunel University, Uxbridge UB8 3PH, UK

Abstract— The concept of Cumulated Anomaly (CA), which describes a new type of database anomalies, is addressed. A typical CA intrusion is that when a user who is authorized to modify data records under certain constraints deliberately hides his/her intentions to change data beyond constraints in different operations and different transactions. It happens when some appearing to be authorized and normal transactions lead to certain accumulated results out of given thresholds. The existing intrusion techniques are unable to deal with CAs. This paper proposes a detection model, Dubiety-Determining Model (DDM), for Cumulated Anomaly. This model is mainly based on statistical theories and fuzzy set theories. It measures the dubiety degree, which is presented by a real number between 0 and 1, for each database transaction, to show the likelihood of a transaction to be intrusive. The algorithms used in the DDM are introduced. A DDM-based software architecture has been designed and implemented for monitoring database transactions. The experimental results show that the DDM method is feasible and effective.

Keywords: Database security; Intrusion detection; Anomaly intrusion

1. Introduction

The number of anomalous usage originated inside an organization is increasing steadily [7][11][12]. They are usually made by "authorized" users of the system. Typically, there is a specific type of intrusions, a user who is authorized to modify data records under certain constraints deliberately hides his/her intentions to change data beyond constraints in different operations and different transactions. It happens when certain authorized and normal transactions submitted result in the accumulated amount of data out of some thresholds. Often, in this type of attack, each individual transaction is legitimate; however, the accumulated results of the attacker's operations are malicious. We refer this type of intrusions as *Cumulated Anomaly (CA)* intrusion.

The existing *Intrusion Detection Systems* (IDS) can be grouped into two classes: (1) *misuse detection*, which maintains a database of known intrusion techniques or behaviors and detects intrusions by comparing users' behaviors against the database [14, 9]; (2) *anomaly detection*, which analyzes user behaviors and the statistics of a process in a normal situation, and checks whether the system is being used in a different manner [5, 21].

In general, *misuse detection* model cannot detect new, unknown intrusions [14]. *Anomaly detection* needs to maintain the records of users' behaviors and the statistics for normal usages, which is referred to as "profiles". The profiles tend to be large. To detect intrusions, it needs a large amount of system resources, and often delays detection decision makings. If attackers hide their intensions, *anomaly detection* will not be able to detect them. So it is fair to say that neither *anomaly detection* nor *misuse detection* would be able to effectively detect *CAs.*

In this study, we investigate *Cumulated Anomaly* intrusions and propose a model for detection. In this model, each transaction is treated as an audit record, which includes the database account, the SQL statement, the time when it is submitted, and the specific data it refers to. By applying a cluster process on the audit records, transaction patterns can be derived. According to these patterns, monitoring rules are set up to specify what transaction patterns to monitor, including the database operations (such as *select*, *update* and *delete*) and the database objects referred to (such as tables and columns). Besides, monitoring rules also define different monitoring modes, including counting the occurrences of transactions belonging to the same user, and data change frequency to make sure that the monitoring rules can inspect closely some complex transactions in which some malicious intentions may hide. In addition, membership functions [11] of fuzzy set theories, which are used to specify the monitoring rules, are applied in the model to monitor and present the possibility of anomalies of transactions in real time. The values of the parameters in the monitoring rules can be extracted from training processes and be modified whenever it is needed. The membership functions are used to assist the rules to indicate the likelihood of a transaction being intrusive or not. If a transaction matches the pattern of a monitoring rule, an indicator (degree) within the interval $[0,1]$ will be calculated. This indicator is used to represent the dubiety degree of a transaction. By this method, the dubiety of database transactions can be denoted quantitatively. Therefore, this model is named as *Dubiety-Determining Model* (DDM).

The main contributions of this study are: (1) address a specific type of anomalies - *Cumulated Anomaly*; (2) propose a new method, the DDM, to monitor *Cumulated Anomaly*; (3) design software system architecture for database transaction monitoring based on the DDM; (4) implement the DDM in a database system; (5) evaluate the system to verify the effectiveness of the DDM.

The rest of the paper is as follows. Section 2 reviews some related work briefly. Section 3 describes the DDM. The design and implementation issues are discussed in Section 4. In Section 5, the experimental results are introduced. Section 6 presents the final remarks.

2. Related Work

Besides access policies, roles, administration procedures, physical security, security models, and data inference, *misuse detection* and *anomaly detection* at databases have been used to detect anomaly intrusions or intrusion attempts made at the databases.

Chung *et al* developed DEMIDS [4], which was a misuse detection system for relational database systems. DEMIDS uses anomaly detection methods for the detection of misuse of prifileges. The main idea is based on frequent *itemsets*. They comprise relations, attributes and values which a user most often uses in his/her SQL statements. The frequent *itemsets* are derived in the training phase for each user. DEMIDS developed a distance measure between such a set and a SQL statement. In the monitoring phase DEMIDS uses this measure to compare a user's frequent *itemset* and his actual queries. An alarm is raised when the measure exceeds a threshold. [1] provides two approaches to anomaly detection in relational databases. The first one is based on the comparison of reference values. These values are obtained with a combination of statistical functions on the elements of single attributes. The second approach uses Δ-relations. Δ-relations record the changes of the values of the monitored attributes between two runs of the system. DIDAFIT [10] is a database misuse detection system that identifies anomalous database accesses by matching SQL statements with a known set of legitimate database transaction fingerprints. It modifies the semantics of an SQL statement with random data, derives a general form of a user's statements and compares the form and the current SQL statement. A similar approach is also presented in [9]. [13] presents a framework for a statistical anomaly prediction system using a neuro-genetic forecasting model, which predicts unauthorized invasions, based on previous observations and takes further action before intrusion occurs. In this paper, the authors propose an evolutionary time-series model for short-term database intrusion forecasting using genetic algorithm owing to its global search capability. D_DIPS [7] monitors transactions issued by users and malicious transactions are viewed as intrusion behaviors. If a malicious transaction is identified, the system cancels the transaction before it succeeds. This method assumes that there are no direct interactive ways for database users to by pass the system. However, if this assumption does not hold, the system will be invalid.

In the existing database intrusion detection researches, fuzzy set theory is mainly used with other theories such as neural network in building profiles for anomaly detection [8, 14, 21, 19]. For example, in [3], Chen R.C. and Hsieh C.C use a fuzzy Adaptive Resonance Theory (ART) and neural network to detect anomaly intrusion of database operations, by monitoring the connection activities to a database.

3. The Dubiety-Determining Model (DDM)

The existing researches have pointed out that the users' profiles can be used for *misuse detection* or *anomaly detection* [5, 9, 14, 21]. However, they approaches do not focus on the data changed by database transactions. Besides, they discover anomaly after it has occurred. In DDM, users' profiles include not only the patterns of their SQL statements, but also the change of data. DDM monitors the dubiety degrees of database transactions quantitatively in real time for *Cumulated Anomaly*.

In DDM, monitoring rules are set up according to the patterns of transactions (like *fingerprint* defined in [9] and [10]). However, transactions matching rules are not definitely anomalous, because in *Cumulated Anomaly* detection, we care not only about the patterns of transactions, but also the data referred, as Fig. 1 shows.

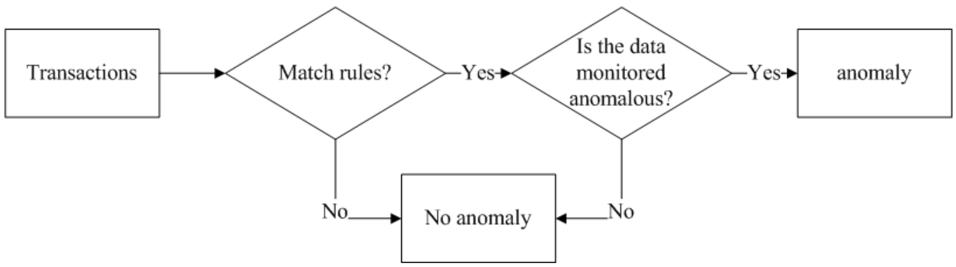


Fig.1 The Detection Flow of DDM

In this section, after giving some basic definitions, we introduce the two sub-models employed in the DDM, which are *statistical sub-model* and *membership function sub-model*. They are the key components of the DDM. The algorithm of training monitoring rules is then presented.

3.1. Definitions in DDM

In this paper we adopt the relational database model as the underlying data model. We define relational database as

$$DB = \langle RS, IC \rangle$$

where $RS = \{R_1, \ldots, R_n\} = \bigcup_{i=1}^{n} \{R_i\}$ is the set of all the relations in the database, and IC is a set of integrity constraints in the database. Furthermore, for a relation $R \in RS$, it is defined that $R = \langle A_1, \ldots, A_n \rangle$ where each A_i is an attribute of R. The set of the attributes of R is denoted by $attr(R) = \{A_1, \ldots, A_n\}$. The function $attr$ also operates on multi-relations, and the result is the set of the attributes of all the relations operated, i.e. $attr(R_1, \ldots, R_n) = \bigcup_{i=1}^{n} attr(R_i)$. The value of attribute A_i of tuple t in a relation R is denoted by $R.t.A_i$. When it is discussed for a definite or the same relation, the prefix R can be omitted, i.e. $t.A_i$.

Definition 1. *Query Statement*

An SQL query statement submitted to the database can be initially considered as 5-tuple:

$$S = < opr, Rs, AT, cond, V >$$

where

- $opr \in \{select, delete, insert, update, execute\}$,
- Rs is the set of relations referred by opr;
- AT is the set of attributes referred by opr;
- $cond$ is the specified condition in S, i.e. the *where* sub-clause. If there is no *where* sub-clause in the statement, we just have $cond = NULL$;
- V is the set of values referred by opr for each item in AT, denoted by $R_i.t.A_j.v$ where $R_i \in Rs$, t is the tuple affected by opr, and $A_j \in attr(R_i)$.

Example 1.

For example, for the statement

update table1 set col1=3, col2='hello' where col3=0;

we suppose that tuple t satisfies the condition $t.col3=0$, then we have

$$S=<update,\{table1\},\{col1,col2\},col3=0,\{3,'hello'\}>$$

for the statement.

Lemma 1. *Equivalent Statements*

For SQL query statements S_1 and S_2, it is also defined:

if

$$\begin{aligned} S_1.opr &= S_2.opr \\ S_1.Rs &= S_2.Rs \\ S_1.AT &= S_2.AT \\ S_1.cond &= S_2.cond \end{aligned}$$

then S_1 and S_2 are called *Equivalent Statements*, denoted by $S_1=S_2$.

Definition 2. *Pattern of Statemenst*

Given an SQL query statement S, by replacing $S.V$ and the specific data value in $S.cond$ with a uniform token $\$TOKEN\$$, the pattern of S can be obtained, which is denoted by p. This operation is defined as $P_{TOKEN}(S)$. p is a 5-tuple:

$$p=P_{TOKEN}(S)=<opr_p,Rs_p,AT_p,cond_p,\$TOKEN\$>$$

where

- $opr_p=S.opr$,
- $Rs_p=S.Rs$;
- $AT_p=S.AT$;
- $cond_p$ is $S.cond$ with the specific data value replaced by $\$TOKEN\$$. If there is no *where* sub-clause in the statement, we just have $cond_p=NULL$;
- $\$TOKEN\$$ is the token by which the specific data values in S are replaced.

Example 2.

For example, for the statement

$$S=<update,\{table1\},\{col1,col2\},col3=0,\{3,'hello'\}>,$$

we have

$$p=P_{TOKEN}(S)=<update,\{table1\},\{col1,col2\},col3=\$TOKEN\$,\$TOKEN\$>$$

Lemma 2. *Equivalent Pattern*

For patterns p_1 and p_2, it is also defined:

if

$$\begin{aligned} p_1.opr_p &= p_2.opr_p \\ p_1.Rs_p &= p_2.Rs_p \\ p_1.AT_p &= p_2.AT_p \\ p_1.cond_p &= p_2.cond_p \end{aligned}$$

then p_1 and p_2 are called *Equivalent Patterns*, denoted by $p_1=p_2$.

Definition 3. *Transaction*

A transaction T executed on a database can be regarded as 4-tuple:

$$T = \langle acct, S, m, ts \rangle$$

where

- $acct$ is the database account who executes T ;
- S is the statement submitted to the database in T ;
- m is the set of measures on T to be monitored, which will be defined in detail in Definition 4;
- ts is the time stamp recording the time when T is executed.

Lemma 3. *Equivalent Transactions*

For transactions T_1 and T_2, it is defined:

if

$$T_1.acct = T_2.acct$$
$$T_1.S = T_2.S$$

then T_1 and T_2 are called *Equivalent Transactions*, denoted by $T_1 = T_2$.

Definition 4. *Measure of Transactions*

For a transaction T, there are two types of values of m, i.e. $m \in \{m_c, m_s\}$. m_c is the frequent number of Equivalent Transactions of T executed in a time window tw, denoted by

$$T.m_c = Count(T_i)$$

where $T_i.ts \geq T.ts - tw$ and for $i \neq j$, $T_i = T_j$.

Given a relation $R_i \in T.S.Rs$ and its attribute $R_i.A_j \in T.S.AT$, supposing t is the tuple affected by $T.S.opr$, m_s is defined as the sum of the margins between corresponding item $v = R_i.t.A_j.v \in T.S.V$ (Definition 1) and the data of $R_i.t.A_j$ before S is submitted in each T in a time window tw, i.e.

$$T.m_s = Sum(R_i.t.A_j.v - R_i.t.A_j),$$

where the data type of $R_i.A_j$ is numeric and

$$T_i.ts \geq T.ts - tw;$$
$$\text{for } i \neq j, T_i = T_j;$$
$$R_i \in T_i.S.Rs;$$
$$A_j \in attr(R_i).$$

If the data type of $R_i.A_j$ is not numeric, it is defined that $T.m_s = NULL$.

Definition 5. *Dubiety Degree of Transactions*

Given a transaction T, and a membership function f, $f(T.m) \in [0,1]$ where $T.m \in \{m_c, m_s\}$ is called the *Dubiety Degree* of T. $f = 0$ means T is normal or *completely acceptable*, and $f = 1$ implies T is malicious or *completely unacceptable*.

Definition 6. *Audit Record*

An audit record ar for a transaction T can be regarded as 6-tuple:

$$ar = \langle aid, T.acct, T.S, T.ts, data1, data2 \rangle$$

where

- aid is the unique ID for each audit record;
- $T.acct$ is the database account who executes T ;
- $T.S$ is the statement submitted to the database in T ;
- $T.ts$ is the time stamp recording the time when T is executed.;
- $data1$ and $data2$ are the data values to which $T.S$ referred, respectively before and after $T.S$ is executed.

Definition 7. *Cumulated Anomaly*

Given a membership function f and a transaction T, the occurrences of T with different time stamps are denoted by T_i. They consist of a set of transactions $TS=\{T_1,\dots,T_n\}=\bigcup_{i=1}^{n}\{T_i\}$, where $T_1=T_2=\cdots=T_n$. If there is $STS=\bigcup_{j=1}^{m}\{T_{i_j}\}\subseteq TS$, where $i_j\in[1,n]$ and $T_{i_1}.ts\le T_{i_2}.ts\le\cdots\le T_{i_m}.ts$, and for STS,

$$f(T_{i_1}.m)\le f(T_{i_2}.m)\le\cdots\le f(T_{i_m}.m)=1$$

stands, it is said that T causes *Cumulated Anomaly*.

3.2. Statistical Sub-model of DDM

Given a metric for a random variable X and n observations $X_1,\dots,X_n$, the purpose of the statistical sub-model of X is to determine whether a new observation X_{n+1} is abnormal with respect to the previous observations. The mean avg and the standard deviation $stdev$ of $X_1,\dots,X_n$ are defined as:

$$avg=\frac{X_1+X_2+\cdots+X_n}{n}. \tag{1}$$

$$stdev=\sqrt{\frac{\sum_{i=1}^{n}(X_i-avg)^2}{n}}. \tag{2}$$

A new observation X_{n+1} is defined to be abnormal if it falls outside a *confidence interval* that is standard deviations from the mean, which is denoted by *CI*:

$$CI=avg\pm dev. \tag{3}$$

where $dev=d\times stdev$ with d as a parameter. Given a time window tw, by letting $X_n=T_n.m$, this sub-model can be applied to $T.m\in\{m_c,m_s\}$. Therefore, it would apply for the case of *Cumulated Anomaly* to determine $T.m$ for a transaction T.

3.3. Membership Function Sub-model of DDM

According to Definition 5, a membership function f is used to "measure" the Dubiety Degree for each transaction T.

An appropriate membership function is the basis of quantitative analysis on fuzzy attributes and plays a key role in fuzzy mathematics. The most widely used functions include *S*-shaped functions (F_S), *Z*-shaped functions (F_Z) and π-shaped functions (F_π). With *U*-shaped functions (F_U) defined as complementarities of π-shaped functions, these four types of membership functions are defined in Fig. 2. Their curves are illustrated in Fig. 3.

$$F_S(x,a,c)=\begin{cases}0 & x\le a\\ 2\left(\dfrac{x-a}{c-a}\right)^2 & a<x\le\dfrac{a+c}{2}\\ 1-2\left(\dfrac{c-x}{c-a}\right)^2 & \dfrac{a+c}{2}<x\le c\\ 1 & x>c\end{cases}$$

$$F_Z(x,a,c)=1-F_S(x,a,c)$$

$$F_\pi(x,a,b,c)=\begin{cases}F_S(x,a,b) & x\le b\\ F_Z(x,b,c) & x>b\end{cases}$$

$$F_U(x,a,b,c)=1-F_\pi(x,a,b,c)$$

Fig.2 The Definition of the Membership Functions

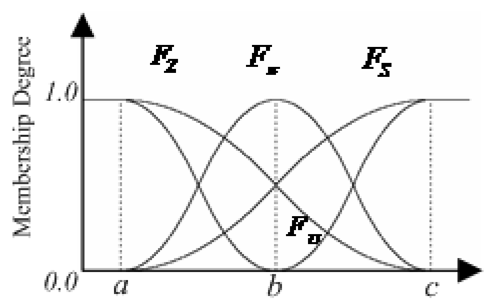


Fig.3 The Curves of the Membership Functions

In Fig. 2 and Fig. 3, we assume that $a\le b\le c$. It is straightforward to prove that when $a=b=c$, F_S and F_Z both have only two values which are 0 and 1, while F_π only has 0 and F_U only has 1 as their values. By adjusting the values of *a*, *b* and *c*, the shapes of F_π and F_U can be changed. For example, the smaller the difference between a and c is, the narrower F_π and F_U are. However, F_S and F_Z are not related to the parameter *b* according to the definition. Their shapes can only be adjusted by *a* and *c*. To make these two functions more flexible, we define *b* as the *median point* for them, i.e.

$$F_S(b)=\frac{1}{2} \tag{4}$$

and

$$F_Z(b)=\frac{1}{2} \tag{5}$$

As a result, the shapes of F_S and F_Z can be adjusted by adjusting *b* as well as by *a* and *c*. According to (4) and (5), F_S and F_Z are made Modification 1, as Fig. 4 shows.

$$F_S(x,a,b,c)=\begin{cases}0 & x\le a\\ \frac{1}{2}\left(\frac{x-a}{b-a}\right)^2 & a<x\le b\\ 1-\frac{1}{2}\left(\frac{c-x}{c-b}\right)^2 & b<x\le c\\ 1 & x>c\end{cases}$$

$$F_Z(x,a,b,c)=1-F_S(x,a,b,c)$$

$$F_\pi(x,a,b,c)=\begin{cases}F_S(x,a,\frac{a+b}{2},b) & x\le b\\ F_Z(x,b,\frac{b+c}{2},c) & x>b\end{cases}$$

$$F_U(x,a,b,c)=1-F_\pi(x,a,b,c)$$

Fig.4 Modification 1 of the Membership Functions

3.4. Integration of the Two Sub-models of DDM

Given a set P containing n observations $X_1,\ldots,X_n$ of a metric for a random variable X, i.e. $P=\{X_n \mid n=1,2,\ldots\}$, there must be a minimum X_{min} and a maximum X_{max} in it. The mean of all the elements in P is avg as (1) defines. By (3), we have

$$CI=[avg - d\times stdev, avg+d\times stdev] \tag{6}$$

with $d\,(d\ge 0)$ as a parameter. By defining $l_{min}=avg-X_{min}$ and $l_{max}=X_{max}-avg$, we additionally define $d\times stdev=\max(l_{min},l_{max})$ (so that the smaller one of l_{min} and l_{max} is included in *CI* as well). By taking $d\times stdev=l_{max}$ as example, we can prove it is reasonable to do so.

Proof.

$$\because d\times stdev=\max(l_{min},l_{max}) \text{ and } d\times stdev=l_{max}$$
$$\therefore l_{min}=avg-X_{min}\le X_{max}-avg=l_{max}$$
$$\therefore |X_i-avg|\le|X_{max}-avg|$$

Additionally, by (6) and (2),

$$avg+d\times\sqrt{\frac{\sum_{i=1}^{n}(X_i-avg)^2}{n}}=X_{max}$$
$$\Rightarrow \frac{1}{d^2}=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i-avg}{X_{max}-avg}\right)^2$$

$$\because \left|X_i - avg\right| \le \left|X_{max} - avg\right|$$
$$\therefore \left(\frac{X_i - avg}{X_{max} - avg}\right)^2 \le 1$$
$$\therefore \sum_{i=1}^{n}\left(\frac{X_i - avg}{X_{max} - avg}\right)^2 \le n$$
$$\therefore \frac{1}{d^2} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - avg}{X_{max} - avg}\right)^2 \le 1$$

Only in the case that $X_1 = X_2 = \cdots = X_n$, $\frac{1}{d^2} = 1$ stands. By Chebyshev'(???)s inequality, the probability of a value falling outside the interval $[avg\text{ - }d \times stdev, avg + d \times stdev]$ is at most $\frac{1}{d^2}$. The statement above is valid because $\frac{1}{d^2} \le 1$. The proof for $d \times stdev = l_{min}$ is similar. □

To be precise, we define $CI = [avg\text{ - }l_{min}, avg + l_{max}]$ which is $CI = [X_{min}, X_{max}]$. Initially, by assigning X_{min}, avg and X_{max} to the parameters of membership functions *a*, *b* and *c*, respectively, *CI* is mapped to the interval $[0,1]$. As a result, X_{min} and X_{max} are mapped to 0 or 1 (depends on different membership functions), and avg is mapped to a value within $[0,1]$. However, as defined previously, 0 means completely acceptable and 1 means completely unacceptable (implying anomaly). Because X_{min} and X_{max} are both in *CI*, we should make sure that $F(X_{min}) < 1$ and $F(X_{max}) < 1$ (meaning X_{min} and X_{max} do not cause anomaly), where $F \in \{F_Z, F_S, F_\pi, F_U\}$. From the definitions of F_S, F_Z, F_π and F_U (Fig. 2 and Fig. 4), it can be seen that F_S is the basis of the rest three ones. Thus, we only need to adjust F_S to make $F(X_{max}) < 1$ for it. To remain *b* as the median point, the values in $[0, 0.5]$ are unchanged. On the contrary, to make $F(X_{max}) < 1$ for F_S, the values in $(0.5,1]$ are reduced by $\alpha (0 < \alpha < 1)$, i.e.

$$\begin{aligned} F_S &= \alpha\left[1 - \frac{1}{2}\left(\frac{c-x}{c-b}\right)^2 - \frac{1}{2}\right] + \frac{1}{2} \\ &= \frac{\alpha+1}{2} - \frac{\alpha}{2}\left(\frac{c-x}{c-b}\right)^2 \end{aligned} \qquad (b < x \le c).$$

Thus, we have Modification 2 of the membership functions, as Fig. 5 shows. As a result, we have $F_S(X_{max}) = F_S(c) = \frac{\alpha+1}{2} < 1$ but not $F_S(X_{max}) = 1$. The rest three functions have the homologous consequence. The parameter α can be assigned to a proper value by users according to the applications. Nevertheless, it is recommended that α is not less than 1 significantly. That ensure $F_S(X_{max})$ less than 1 not too much, and the result values in $(b, c]$ differentiable.

$$F_S(x,a,b,c)=\begin{cases}0 & x\le a\\ \frac{1}{2}\left(\frac{x-a}{b-a}\right)^2 & a<x\le b\\ \frac{\alpha+1}{2}-\frac{\alpha}{2}\left(\frac{c-x}{c-b}\right)^2 & b<x\le c, 0<\alpha<1\\ 1 & x>c\end{cases}$$

$$F_Z(x,a,b,c)=1-F_S(x,a,b,c)$$

$$F_\pi(x,a,b,c)=\begin{cases}F_S(x,a,\frac{a+b}{2},b) & x\le b\\ F_Z(x,b,\frac{b+c}{2},c) & x>b\end{cases}$$

$$F_U(x,a,b,c)=1-F_\pi(x,a,b,c)$$

Fig.5. Modification 2 of the Membership Functions

By Definition 5, any $X_n = T_n.m$ can be mapped to a real number in $[0,1]$. This real number denotes the Dubiety Degree of T_n.

3.5. Training the Parameters of Membership Functions

An automated solution to train the rules in DDM to "learn" the parameters both in advance and during monitor is developed. So that the values of parameters *a*, *b* and *c* of membership functions in the rules can be assigned automatically according to normal audit sets. In this way, they can be adjusted during the monitoring process as well.

Given

$$TS=\{T_1,\dots,T_n\}=\bigcup_{i=1}^{n}\{T_i\}, \text{ where } T_1=T_2=\cdots=T_n,$$

and

$$P_n=\bigcup_{i=1}^{n}\{X_i\}=\bigcup_{i=1}^{n}\{T_i.m\},$$

as it is stated above, we have

$$\begin{cases}a=X_{min}\\ b=avg\\ c=X_{max}\end{cases}.$$

Thus, for $P_n=\{X_1,\dots,X_n\}$, we have

$$\begin{cases}a_n=X_{min(n)}=\min(X_1,\dots,X_n)\\ b_n=avg_n=\frac{X_1+\cdots+X_n}{n}\\ c_n=X_{max(n)}=\max(X_1,\dots,X_n)\end{cases}.$$

When a new value $X_{n+1}=T_{n+1}.m$ comes, for $P_{n+1}=P_n\bigcup\{X_{n+1}\}=\{X_1,\dots,X_n,X_{n+1}\}$, we will have

$$
\begin{cases}
a_{n+1} = \min(X_1, \ldots, X_n, X_{n+1}) = \min(\min(X_1, \ldots, X_n), X_{n+1}) \\
\quad = \min(a_n, X_{n+1}) \\
b_{n+1} = avg_{n+1} = \dfrac{X_1 + \cdots + X_n + X_{n+1}}{n+1} = \dfrac{n \times avg_n + X_{n+1}}{n+1} \\
\quad = \dfrac{nb_n + X_{n+1}}{n+1} \\
c_{n+1} = \max(X_1, \ldots, X_n, X_{n+1}) = \max(\max(X_1, \ldots, X_n), X_{n+1}) \\
\quad = \max(c_n, X_{n+1})
\end{cases} \tag{7}
$$

Consequently, a parameter *n* is employed to "remember" the number of times of previous calculations. As a result, the parameters *a*, *b* and *c* of membership functions can be calculated basing on their historical values, as (7) shows. We call *n* as *Maturation Degree* of monitoring rules. It indicates how the parameters of membership functions can be affected by a new value X_{N+1}. According to statistic theory, the bigger *n* is, the smaller probability of $a_{N+1} \neq a_N$ and $c_{N+1} \neq c_N$ is, and the less b_N can be affected by X_{N+1}.

3.6. Deriving Patterns from Audit Records

Given a set of audit records $ARS = \{ar_1, ar_2, \ldots, ar_n\} = \bigcup_{i=1}^{n} \{ar_i\}$, with a token $\$SOMEVALUE\$$, we have pattern for each $ar_i.S$ in it:

$$p_i = P_{SOMEVALUE}(ar_i.S) \, .$$

By selecting $ar_i.T.acct$ and p_i as cluster-features, a cluster is processed on ARS to derive patterns from audit records:

$$\text{if } ar_i.T.acct = ar_j.T.acct \text{ and } p_i = p_j ,$$
$$\text{then } ar_i \cong ar_j$$

where $1 \le i \neq j \le n$, and $ar_i \cong ar_j$ means ar_i and ar_j belong to the same cluster. All of the audit records in the same cluster form a subset of ARS, which is denoted by ARC. This cluster process generates $a(1 \le a \le n)$ cluster subsets ARC_k:

$$\bigcup_{k=1}^{a} ARC_k = ARS$$

If ARC_k has m_k items in it, and ARS has m_{total} items in total, we will have

$$percentage_k = \frac{m_k}{m_{total}} \times 100\% \, .$$

If we order ARC_k by m_k or $percentage_k$ in descending order, it can be known that what patterns by which accounts have occurred, and their frequencies of occurrences. By taking this result for reference, a database administrator can easily decide to set up monitor rules aiming at what patterns. The administrator also can set up any arbitrary rules whose patterns have not appeared.

4. The DDM-Based Software Architecture

4.1. The Software Architecture

The architecture for database transaction monitoring based on DDM is designed as shown in Fig. 6.

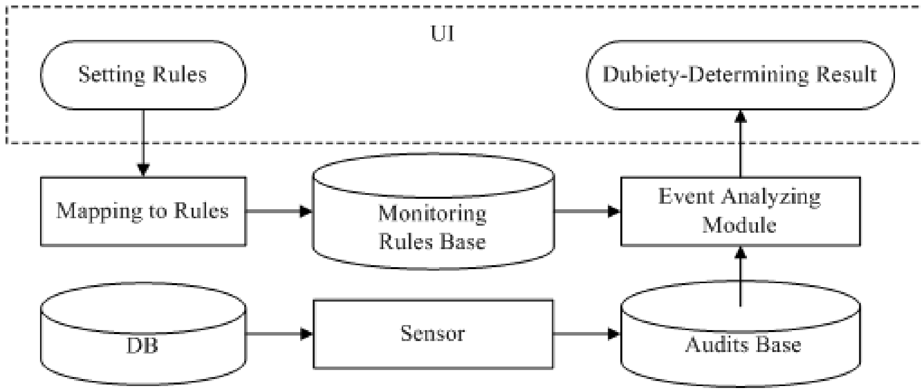


Fig.6 The Architecture for Database Transaction Monitoring Based on DDM

The user interface (UI) provides tools for interactions, which includes *Setting Rules* and display *Dubiety-Determining Results*. *Setting Rules* allows users to set up monitoring policies. These monitoring policies are then formatted and transferred into *Monitoring Rules Base* by *Mapping to Rules*. The information about each database transaction is organized into *Audits Base* by *Sensor*. *Event Analyzing Module* selects every new audit record from *Audits Base*, and then checks against the monitoring rules in *Monitoring Rules Base*. Finally, *Event Analyzing Module* calculates dubiety degree for the audit record, and forwards the results to *Dubiety-Determining Result*.

Other main modules/components of the architecture are:

Audits Base is built to store the audit records generated by *Sensor*, while *Monitoring Rules Base* is used to store monitoring rules defined manually.

Setting Rules, used to define monitoring rules, specifies which attributes of transactions to monitor, what types of membership functions to use, and what the values of the parameters in membership functions are, etc.

Mapping to Rules. When the information about the monitoring policy and membership function is decided, *Mapping to Rules* converts it into the format of monitoring rules. The rules are stored in *Monitoring Rules Base*.

Sensor. This module monitors the transactions of application databases in real time. By analyzing each transaction processed, it collects information about the transaction, and then stores it in *Audits Base*.

Event Analyzing Module. This is the centre of the whole architecture. The monitoring algorithm is implemented in this module. For each record in *Audits Base*, *Event Analyzing Module* is processed and matched against the rules in *Rules Base*. The value of the monitored attribute is then obtained. By substituting this value in the membership function defined in the rule, the result of the function is calculated as the degree of dubiety. The detail of this algorithm is presented in the next section.

4.2. Basic Data Structures Required in DDM

There are two basic data structures required in DDM: *Audit Record* and *Monitoring Rule*. *Audit Record* is for recording the information about each database transaction. *Monitoring Rule* is the structure for specifying the format of the monitoring rules. The details of the two structures are defined as follows.

Audit Record. This data structure is 6-tuple recording information of each database transaction, which matches Definition 6 completely:

<AID, UID, SQLText, Time_stampe, Data1, Data2>

where

- *AID* is the identifier for each audit record.
- *UID* records the user name of the transaction.
- *SQLText* records the content of the SQL statement of the transaction.
- *Time_stamp* records the time when the transaction is executed.

- *Data1* is the first data field that the transaction relates to. For example, the data value before update.
- *Data2* is the second data field that the transaction relates to. For example, the data value after an update.

To make it clearer, from now on in this paper, we will use the term *audit record* instead of *transaction*.

Monitoring Rule. This data structure is 6-tuple defining the format of the monitoring rules:

<RID, UID, Action, Obj1, Obj2, Condition, Time_window, Mon_type, Function, Enable>

where

- *RID* starting with the letter *R* is the identifier for each monitoring rule.
- *UID* indicates which user the rule is aimed at.
- *Action* indicates what type of operations the rule is related to, such as *select, update, delete* and so on.
- *Obj1* and *Obj2* records for which database object (table, view, procedure, and so on) the rule is valid. *Obj1* is the first object that *Action* refers to, such as a table, a view or a procedure. *Obj2* is the second one. If *Obj1* is a table or a view, *Obj2* will be a field name.
- *Condition* indicates the condition of *Action*. Usually it is the condition part (*where* clause) of the SQL statement.
- *Time_window* specifies a number of hours as a time range. The audit records occurred in that time range before the currently being checked one will be sought by the rule.
- *Mon_type* is the type of monitor. It has two values: *C* and *S*. *C* is used for counting numbers and *S* is for recording the sum value.
- *Function* is sub-tuple recording the information of the membership function used by the rule:

<FID, A, B, C>

where

- *FID* specifies which type of membership function to use. It has four values. 'Z' means F_Z. 'S' means F_S. 'P' means F_π, while 'U' means F_U.
- *A*, *B*, and *C* store the values of *a*, *b*, and *c* respectively (definition of membership function).

Enable is a switch. When it is 1, the rule is valid; otherwise, it is not.

4.3. The Algorithm of Dubiety Determining in DDM

The algorithm of calculating the dubiety degrees of audit records by membership functions is illustrated in Fig. 7. If an audit record is not matched by any monitoring rule, there will be no detection result for it. If it is matched by more than one rule, the one generating the highest result will be selected.

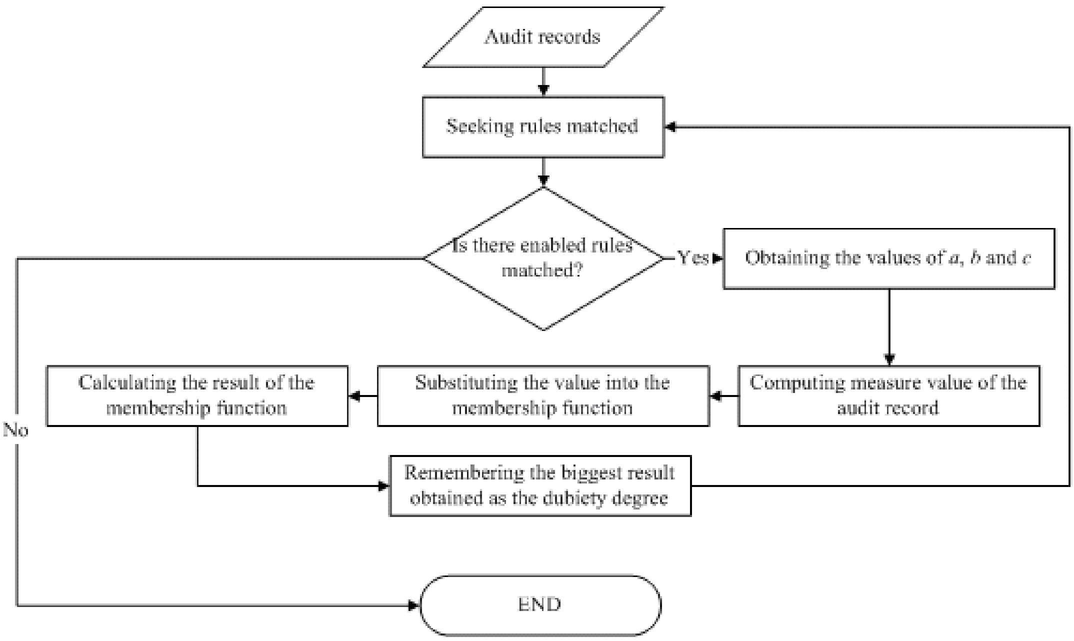


Fig.7 Flow Diagram of the Algorithm

Algorithm 1 *MembershipFunctions* (*x*, *a*, *b*, *c*, *fid*)

Select case *fid*
    Case ”s” or ”S”
        return $F_S(x, a, b, c)$
    Case ”z” or ”Z”
        return $F_Z(x, a, b, c)$
    Case ”p” or *fid* =”P”
        return $F_\pi(x, a, b, c)$
    Case ”u” or *fid* =”U”
        return $F_U(x, a, b, c)$
End select

Algorithm 2 Determining the dubiety value of a designated audit record

Input: *aid*, which is the AID of the audit record whose dubiety value will be determined

Operation: If any rule matches, record information of *aid*, *rid*, *fid*, *a*, *b*, *c*, *x*, and *result*

select the enabled rules which match *aid* on UID and roughly on SQLTEXT
initialize *buf* = 0 as a buffer variant
for each selected rule **do**
    if the rule matches the audit in detail **then**
        Obtain *fid* and the values of *a*, *b* and *c*

```
    Compute the measured value of the audit as x, such as the times of executed or accumulated value of updated data
        result = MembershipFunctions (x, a, b, c, fid)
        if result ≥ buf then
            buf=result
            update the record with aid, rid, fid, a, b, c, x and result
        end if
    end if
end for
```

Algorithm 1 implements the membership functions. Algorithm 2 determines the dubiety degree of a designated audit record.

5. Experimental Assessment

This section presents the measure performance of a number of experiments. We implemented a system based on the architecture introduced in Section 3, which was used to test and verify the DDM method. The experiments are performed on a computer with one CPU of 2.99GHz and 512MB RAM. The operating system is Microsoft Windows Server 2003 SP1. The DBMS is Microsoft SQL Server 2000. The example database *Northwind* of SQL Server is used in this study. It includes trade data records for a company called *Northwind Traders*, which engaged in the import and export trade business.

5.1. Experiment 1

In Experiment 1, *Audits Base* and *Monitoring Rules Base* are built according to the two basic structures defined. According to Section 3.1, 30,000 typical audit records are generated and 19 monitoring rules are set up.

*Data set***.** AIDs are generated in ascending order of Time_stamp, the values of which are randomly generated precise to second (system clock) in a period of three months (from 2006-07-22 to 2006-10-23). Seven user names appear in the field of UID: *Ann*, *Bob*, *Charles*, *Dennis*, *Eva*, *Fabre*, and *Gama*. The values of fields SQLText are randomly generated as common database operations in the form of SQL statements. The content of SQLText includes selecting data from a table, updating the data in a table, inserting data into or deleting data from a table, executing a procedure, or opening a database.

By applying the patterns deriving approach as Section 3.6 states, 183 patterns of all the 30000 audit records are derived. After sorting them by **Percentage** in descending order, Table 1 shows the top 10 ones. **UID** is the database account. **SQL Text** is the SQL statement with specific data values replaced by token $\$SOMEVALUE\$$. **UID** and **SQL Text** compose a pattern. **Count** is the number of occurrences of a pattern. **Percentage** is the occurrence percentage of a pattern in all of the 30000 audit records.

Table 1 Top 10 Ones of the Derived Patterns

UID	SQL Text	Count	Percentage
Administrator	select * from orders where customerid=$SOMEVALUE$	740	2.466667
Ann	select * from orders where customerid=$SOMEVALUE$	730	2.433333
Dennis	select * from orders where customerid=$SOMEVALUE$	729	2.43
Eva	select * from orders where customerid=$SOMEVALUE$	708	2.36
Fabre	select * from orders where customerid=$SOMEVALUE$	707	2.356667
Bob	select * from orders where customerid=$SOMEVALUE$	693	2.31
Clerk	select * from orders where customerid=$SOMEVALUE$	693	2.31
Administrator	select * from [order details] where orderid=$SOMEVALUE$	180	0.6
Eva	update orders set freight=$SOMEVALUE$ where orderid=$SOMEVALUE$	170	0.566667
Administrator	update customers set phone=$SOMEVALUE$ where costomerid=$SOMEVALUE$	168	0.56

Finally, 19 typical monitoring rules are set up into the *Monitoring Rules Base* according to the patterns. Each rule is specified with one of the four types of membership functions, and the parameters *a*, *b*, and *c* are assigned manually. For instance, as Table 2 shows (in which the column of Enable is not listed to make the table not too wide), we have

R09=< *R09*, *Fabre*, *update*, *order details*, *UnitPrice*, *ProductID=43*, *5000*, *S*, < *S*, *5.0*, *10.0*, *32.0* >, 1>.

Table 2 Monintoring Rule R09

RID	UID	Action	Obj1	Obj2	Condition	Time_window	Mon_type	FID	A	B	C
R09	Fabre	Update	order details	UnitPrice	ProductID =43	5000	S	S	5	10	32

That means *R09* is used to monitor the audit records where UID is *Fabre*, *update [order details] set UnitPrice=p where ProductID=43* as SQLText, and *p* is a number. The data items before and after update operation are recorded in the fields Data1 and Data2. When an audit record of that type occurs, *R09* seeks the audit records of that type which have occurred over the past *5000* hours, and sums up the difference between each pair of Data1 and Data2 in each audit record. Then, the sum is substituted into the $F_S(x, 5.0, 10.0, 32.0)$ defined in *R09*. Finally, a result value of the function is assigned as the dubiety degree of that audit record. As this is a real-time process; an audit record has been examined as soon as it has arrived. All of the 19 monitoring rules are listed in Table 3. To make the table not too wide, the column of Enable is not listed.

Table 3 19 Monitoring Rules

RID	UID	Action	Obj1	Obj2	Condition	Time_window	Mon_type	FID	A	B	C
R01	Fabre	select	orders	EmployeeID	<NULL>	3000	C	S	10	50	80
R02	Fabre	select	orders	CustomerID	<NULL>	3000	C	S	10	41	72
R03	Dennis	select	order details	UnitPrice	OrderID =11545	3000	C	S	5	35	85
R04	Ann	update	customers	Phone	<NULL>	3000	C	U	10	50	78
R05	Charles	update	order details	Discount	<NULL>	3000	S	P	-1	0	1
R06	Eva	delete	orders	<NULL>	EmployeeID =2	3000	C	S	0	0	0
R07	*	insert	customers	<NULL>	<NULL>	3000	C	Z	5	40	100
R08	Ann	update	customers	Address	<NULL>	3000	C	Z	0	60	350
R09	Fabre	update	order details	UnitPrice	ProductID =43	5000	S	S	5	10	32
R10	*	use	master	<NULL>	<NULL>	5000	C	S	0	0	0
R11	Gama	exec	SalesByCategory	<NULL>	<NULL>	3000	C	S	5	25	50
R12	Bob	use	monitor	<NULL>	<NULL>	2400	C	P	10	33	50
R13	Dennis	use	monitor	<NULL>	<NULL>	2400	C	P	15	52	80
R14	Charles	exec	Ten Most Expensive Products	<NULL>	<NULL>	2400	C	S	5	5	5
R15	Eva	logon	<NULL>	<NULL>	<NULL>	2400	C	S	1	50	200
R16	Bob	select	customers	<NULL>	<NULL>	5000	C	S	20	100	190
R17	*	select	customers	CustomerID	<NULL>	3500	C	S	50	200	300
R18	*	select	orders	<NULL>	CustomerID =	3500	C	S	50	200	400
R19	*	select	order details	<NULL>	OrderID =	3500	C	U	50	180	330

*Results***.** Experiment 1 consists of two tests. In Test 1, all of the 19 monitoring rules are enabled. 9971 of the 30000 audit records are detected as dubious or anomalous. The rest are regarded as normal, as these audit records do not match any of the 19 rules. Among the 9971 results, there are 1380 ones with results being neither *0* nor *1*. The rest ones are either *0* or *1*. Table 4 lists 5 examples. In Table 4, all of the dubiety degrees of these audit records are between *0* and *1*. That means they are "dubious": not completely acceptable or unacceptable. By the "degree", we know *how* dubious a record is.

Table 4 5 Example Results of Test 1

AID	RID	FID	A	B	C	X	Result
1118	R18	S	50	200	400	192	0.329208
1124	R02	S	10	50	72	41	0.5
1126	R18	S	50	200	400	194	0.338547
1127	R18	S	50	200	400	195	0.343265
1128	R07	Z	5	40	100	26	0.8120439

For the record of **AID** *1118*, **RID** is *R18*, while **X** is *192.0*. When *R18* is matched again in the record of **AID** *1126*, **X** is *194.0*. This can be explained because both *R02* and *R18* matched the **AID** *1124*. For *R18*, its **X** is *193.0*, while for *R02*, **X** is *41.0*. Therefore, **AID** *1124* has *0.5* as **Result** by *R02* and *0.333861* by *R18*. Because the **Result** of *R02* is greater than that of *R18*, the audit record is more dubious as measured by *R02* than by *R18*. As a result, *R02* is selected for AID *1124*.

In Test 2, 6 rules including *R02* are disabled. In the results, all the records picked up by *R02* in Test 1 are now picked up by *R01* (in Test 1, R02's dubious degree is higher than R01's). This is because these records are matched by both *R01* and *R02*. When *R02* is disabled in Test 2, *R01* is used where *R02* was selected before. The results of these two tests are summarized in Table 5.

Table 5 Summarized Data of the Two Tests

Test		**Total**	**UID**						
			Ann	**Bob**	**Charles**	**Dennis**	**Eva**	**Fabre**	**Gama**
1	All results	9971	1463	1431	1383	1357	1422	1549	1366
	Not 1	1380	219	247	130	186	275	152	171
2	All results	1811	152	307	136	64	170	982	2
	Not 1	558	66	207	0	63	153	69	0

5.2. Experiment 2

In Experiment 2, it is supposed that there is a product whose *ProductID* is 9 in *Products*. Assume a member of staff, *Ann,* is authorized to modify *UnitPrice* of *Product* 9. However, if the *UnitPrice* has been changed too much or too often, it could be suspicious. It is defined that *UnitPrice* should not be changed for more than 4 times in 30 days, and the sum of changed value should not be more than 3 dollars in 90 days. A practical case is simulated by previous assumptions.

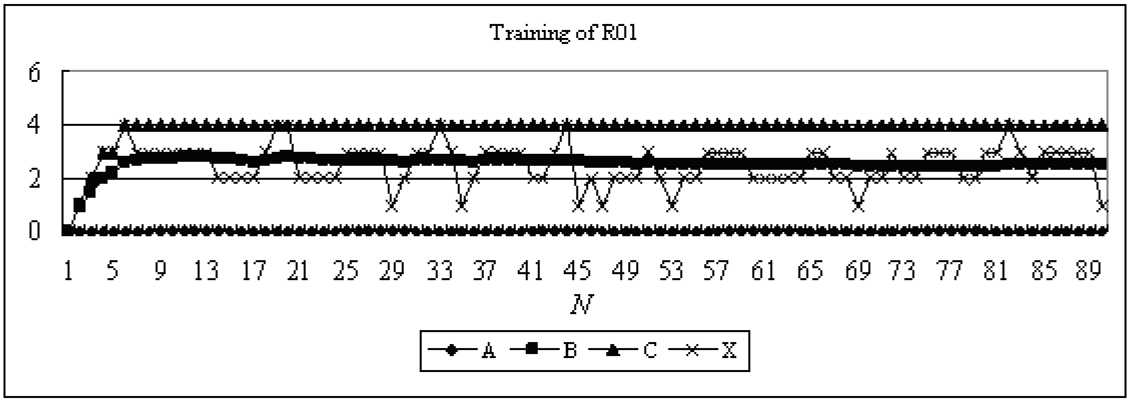
The existing 30000 audit records in Experiment 1 are considered as normal ones here. Besides them, more 12 additional ones for *Ann*'s updating *UnitPrice* of *Products 9*, which may cause *Cumulated Anomaly*, are imitated into *Audits Base*. All of the audit records distribute in three months.

In *Monitoring Rules Base*, two new monitoring rules *R01* and *R02* are set up for the assumptions in Experiment 2, instead of all of the 19 ones in Experiment 1. F_S is used for *R01* while F_U is used for *R02*, as Table 6 lists.

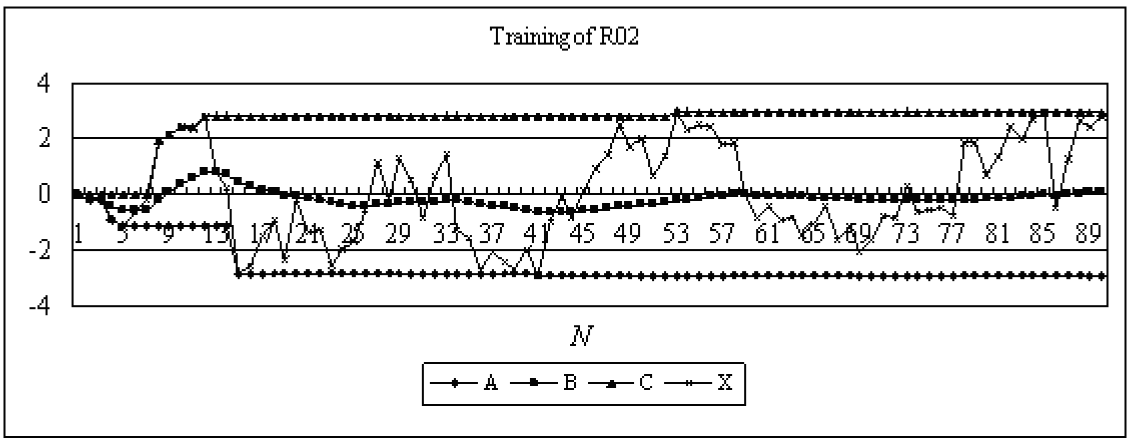
Table 6 Two Monitoring Rules in Experiment 2

RID	**UID**	**Action**	**Obj1**	**Obj2**	**Condition**	**Time_window**	**Mon_type**	**FID**	**A**	**B**	**C**
R01	Ann	update	Products	UnitPrice	ProductID=9	720	C	S	0	0	0
R02	Ann	update	Products	UnitPrice	ProductID=9	2160	S	U	0	0	0

Initially, the values of the parameters *a*, *b* and *c* of F_S and F_U are 0, and the *Maturation Degrees* of the two rules (*N* described in Section 3.5) are both 0. 89 imitated normal audit records for *Ann*'s updating *UnitPrice* of *Products 9* in 4 years are used to train the two rules, and then the two trained rules are used to detect *Cumulated Anomaly* in *Audits Base*. Fig. 8 shows the process of training the two monitoring rules. In the figure, *A*, *B* and *C* respectively stands for the parameters *a*, *b* and *c*. After training, for *R01*, we have $a = 0$, $b \approx 2.5$ and $c = 4$; for *R02*, we have $a = -2.95$, $b \approx 0.068$ and $c = 2.92$. In the experiment, we let $\alpha = 0.9$, which makes $F_S(X_{max}) = F_S(c) = 0.95$.



(a)



(b)

Fig.8 Training of the Two Monitoring Rules

Fig. 9 shows all results. Fig. 9 (a) shows the value of *UnitPrice* after *Ann* updates it for each time. Fig. 9 (b) shows the monitor result of using the rule of *R01*. We can see that the dubiety degree is increasing gradually. However, it does not reach 1 all the while. That means *Ann*'s operations become more and more malicious but no anomaly occurs by *R01*. Fig. 9 (c) shows the results of monitoring the modified *UnitPrice* of *Product 9* over 90 days by *R02*. It is shown that the dubiety degree is more and more close to 1. At the end the dubiety degree reaches 1. According to the definition of DDM, anomalies may occur. When *R01* and *R02* are both enabled, the results are shown in Fig. 9 (d). Fig. 9 (d) also can be regarded as the combinations of Fig. 9 (b) and Fig. 9 (c) by selecting the point with the higher dubiety degree between (b) and (c) for each *AID*. In general, when several monitoring rules are matched to the same audit record, the one with the highest dubiety degree will be selected. From the results, we can see *Ann*'s operations cause anomaly.

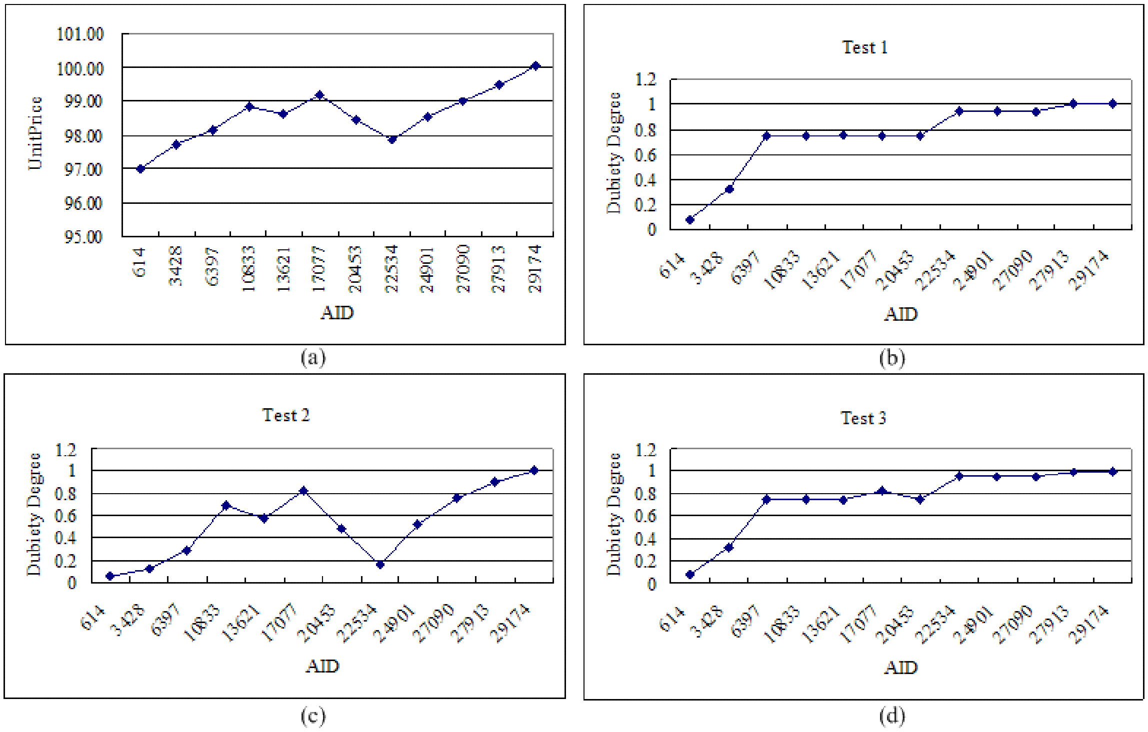


Fig.9 Result of Experiment 2

6. Conclusion

The cumulated information of data changed by database transactions may give the hints about malicious intrusions, which is an issue has not been well focused in previous studies. In this paper, we expose a novel concept: *Cumulated Anomaly*. The goal of this study was to give an insight into the database transactions when *Cumulated Anomaly* would occur. A novel detection method *Dubiety-Determining Model* (DDM) has been proposed aiming at the detecting *Cumulated Anomaly* intrusions.

By generalizing the audit records with a token and the cluster process on them, the DDM derives patterns of SQL statements in the database transactions. Monitoring rules are initialized according to the patterns. A learning solution ensures the DDM being trained with a normal audit records set. It also makes the DDM to "learn" during the process of monitoring. During the monitoring process, the DDM assigns a real number between 0 and 1 to each database transaction. The real number is called dubiety degree, which tells how a transaction is dangerous, and whether *Cumulated Anomaly* occurs.

Basing on the DDM, software system architecture is designed and implemented. Two experiments are performed to verify the effectiveness of the DDM with it. The experiment results mainly identify two things. Firstly, *Cumulated Anomaly* does exist in database transactions. Secondly, the DDM is rule-based model, and it measures transactions quantitatively in real time. Finally, the DDM can monitor database transaction in real time to predict before and discover after *Cumulated Anomaly* has occurred. In conclusion, a novel database anomaly *Cumulated Anomaly* is defined and represented in this paper, and the DDM proposed aiming at *Cumulated Anomaly* is capable of identifying it. By the DDM, database security is focused on from a new aspect.

We currently work on improving the performance of the algorithms and constructing the entire detection system. The research on considering anomalies caused by cumulative updates to sets of variables in DDM that, for example, jointly describe a state of an asset to be protected, would be an interesting area.

Acknowledgement

This work is supported by the Infrastructure Project of Science and Technology of the Chinese National Eleventh Five-Years Development Program. The number of the project is 2006BAK31B04, and by the UK EPSRC ICG grant.

Reference

[1] Adrian SPalka, Jan Lehnhardt (2005): A comprehensive approach to anomaly detection in relational databases. Data and Applications Security 2005, LNCS 3654: 207-221.

[2] Cai ZM, Guan XH, Shao P, Peng QK, Sun GJ (2003): A rough set theory based method for anomaly intrusion detection in computer network systems. Expert Systems 20 (5): 251-259.

[3] Chen R.C., Hsieh C.C. (2004): An anomaly intrusion detection on database operation by fuzzy ART neural network. Proceedings of International Computer Symposiums 2004: 839-844.

[4] Chung C.Y., Michael G. and Karl L. (2000): DEMIDS: a misuse detection system for database systems. Working Conference on Integrity and Internal Control in Information Systems: 159--178.

[5] Darren M., Fredrik V. and Giovanni V. (2006): Anomalous system call detection. ACM Transactions on Information and System Security, Vol. 9, No. 1: 61-93.

[6] Francesco M. M., Mauro M. and Marina M. (2006): Auditing sum-queries to make a statistical database secure. ACM Transactions on Information and System Security, Vol. 9, No. 1: 31-60.

[7] Jiazhu Dai, Huaikou Miao (2005): D_DIPS: An intrusion prevention system for database security. Proceedings of International Conference on Information, Communications and Signal Processing 2005, LNCS 3783: 481-490.

[8] Kuok C.M., Fu A., Wong M.H.(1998): Mining fuzzy association rules in databases. SIGMOD Record, 27(1): 41-46.

[9] Lee S.Y., Low W.L. and Wong P.Y. (2002): Learning fingerprints for a database intrusion detection system. Proceedings of European Symposium on Research in Computer Security 2002, LNCS 2502: 264-279.

[10] Low W.L., Lee J., Teoh P. (2002): DIDAFIT: Detecting intrusions in databases through fingerprinting transactions. Proceedings of International Conference on Enterprise Information Systems 2002: 121-128.

[11] Pedrycz W., Gomide F. (1998): An Introduction to Fuzzy Sets: Analysis and Design. Cambridge, Mass. MIT Press.

[12] P. Ramasubramanian, A. Kannan (2004): Quickprop neural network short-term forecasting framework for a database intrusion prediction system. Proceedings of International Conference on Artificial Intelligence and Soft Computing 2004, LNAI 3070: 847-852.

[13] Ramasubramanian P., KANNAN A. (2006): A genetic-algorithm based neural network short-term forecasting framework for database intrusion prediction system. Soft Computing 10 (8): 699-714.

[14] Sato I., Okazaki Y. and Goto S. (2002): An improved intrusion detecting method based on process profiling. Transactions of the Information Processing Society of Japan vol.43, no.11: 3316-26.

[15] Somjai B., Robert C. S. and David A. M. (2003): Rule-based COTS integration. Journal of Research and Practice in Information Technology, Vol. 35, No. 3: 197-204.

[16] Susan M. B., Rayford B. V. (2000): Fuzzy data mining and genetic algorithms applied to intrusion detection. Proceedings of National Information Systems Security Conference 2000.

[17] Valeur F., Mutz D. and Vigna D. (2005): A learning-based approach to the detection of SQL attacks. Proceedings of International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment 2005, LNCS 3548: 123-40.

[18] Xu JY, Sung AH and Liu QZ (2005): Online fraud detection system based on non-stationery anomaly detection. Proceedings of the 2005 International Conference on Security and Management: 364-370.

[19] Zhang Y.J., Leng H.W. (2005): An investigation of immune detection algorithm with vaccine operator and fuzzy match. Proceedings of the International Conference on Machine Learning and Cybernetics 2005, Vol. 7: 3943-3948.

[20] Zhu, D., G. Premkumar and Zhang X., Chu C. (2001): Data mining for network intrusion detection: a comparison of alternative methods. Decision Sciences, 32(4): 635-660.

[21] Zhu T.Q., Xiong P. (2005): Optimization of membership functions in anomaly detection based on fuzzy data mining. Proceedings of International Conference on Machine Learning and Cybernetics 2005, Vol. 4: 1987-1992.