# Analysis of Data Dependency Based Intrusion Detection System*

Yermek Nugmanov[1], Brajendra Panda[1], and Yi Hu[2]

[1] Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, AR 72701
{ynugmano,bpanda}@uark.edu
[2] Computer Science Department
Northern Kentucky University
Highland Heights, KY 41099
huy1@nku.edu

**Abstract.** This research focuses on analyzing the cost effectiveness of a database intrusion detection system that uses dependencies among data items to detect malicious transactions. The model suggested in this paper considers three main factors: the quality of intrusion detection, the probability of intrusion, and the cost structure of an organization whose data is protected by the intrusion detection system. We developed a step by step approach that helps in determining the optimal configuration expressed by the response strategy and the threshold value. The experimental results show that our model is capable of finding the optimal configuration while taking the cost structure of an organization into consideration.

**Keywords:** Database intrusion detection, semantic analyzer, cost analysis, response strategy.

## 1 Introduction

Although there exist many security tools for protecting computer systems from attacks, as per [1], none of them can provide absolute security. Therefore, all important events that occur in a computer system must be supervised and examined for a possible presence of malicious activity by intrusion detection systems. Of the two universally recognized models for intrusion detection [2, 14], misuse detection and anomaly detection, the latter typically uses a threshold to define which events are considered normal and which are considered intrusive [3, 4, 15]. By changing the threshold, an organization may find the optimal balance between successful detections and false alarms. However, finding of the optimal configuration is a difficult task. Some recent works have focused on the optimization techniques for anomaly detection systems

* Research of Yermek Nugmanov and Brajendra Panda was supported in part by AFOSR under grant FA 9550-04-1-0429. Research of Yi Hu was supported in part by the KY NSF EPSCoR program.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 111–125, 2009.

[4, 5]. At present, the majority of existing host-based anomaly detection systems are ineffective in detecting attacks on databases, since they are focused on tracking and analyzing events that occur in operating systems and applications, and not on the database itself. Although a few models have been developed for detecting malicious activities in databases, to the best of our knowledge, none of these methods have been analyzed for their cost–effectiveness. The objective of this work is to evaluate the data dependency based database intrusion detection system [12] for optimization based on response strategy and threshold value.

## 2 Background

A limited research has been done to address the problem of malicious transaction detection in database systems. In [6] an architecture for intrusion-tolerant database systems is proposed. An intrusion-tolerant database management system is able to operate and deliver essential services even in case of attacks. However, this approach is more focused on the localization of attacks and recovery of the damage, than on developing a specific intrusion detection system. A database intrusion detection scheme based on data dependency rule mining is presented in [11]. A similar method that uses weighted sequence mining techniques is offered in [7]. The major drawback of this detection method is that the weights of attributes must be assigned manually. The method presented in [8] detects intrusion in databases that employs role-based access control and it uses Naïve Bayes Classifier to predict the role which the observed SQL command most likely belongs to, and compares it with the actual role. If the roles are different, the SQL statement is considered illegal. An intrusion detection system for real-time database systems has been discussed in [9]. Researchers in [10] proposed a misuse detection system for databases based on the observation that there exist certain regularities in "access patterns" of users. Our research is based on the model presented in [12] that detects malicious activities in a database management system by using data dependency relationships.

Regarding the effectiveness of intrusion detection, a study presented in [3] showed that such methods are subject to the base-rate fallacy, coming to the conclusion that "in order to achieve substantial values of the Bayesian detection rate, we have to achieve a low false alarm rate". He found that in most cases such a rate is unattainable. Following that, researchers in [13] developed the techniques for building an intrusion detection system on the basis of cost-sensitive models. The first comprehensive study on the cost effectiveness of intrusion detection systems appeared in [5], which addresses the problem of finding the optimal configuration of a single intrusion detection system, and various combinations of multiple intrusion detection systems. An optimization scheme based on game theory has been offered in [4].

## 3 The Model

### 3.1 Data Dependency Based Intrusion Detection Model

In this section, we very briefly discuss the data dependency based database intrusion detection model, which was presented in [12]. This model has two components, namely, the static semantic analyzer and the dynamic semantic analyzer. The static semantic analyzer is employed to analyze the database application program statically to discover intra-transaction data dependencies represented by the read, pre-write, and post-write set. If a transaction does not conform to the read, pre-write, or post-write sets, it will be identified as a malicious transaction. This is treated as the first line of defense. In case, the malicious transactions are well-crafted and are compliant with the data dependencies discovered by the static semantic analyzer, the access patterns to these sets can be used to discover malicious transactions. The access probabilities for these sets depend on the execution path of the database application and the normal user access patterns of the database. The dynamic semantic analyzer is designed to calculate the access probability based on the database log.

To have a better understanding on the data dependency based database intrusion detection model, we illustrate an example here. Suppose during normal database operation phase, two transactions $T_1$ and $T_2$ are generated by the database application. Assume that we have the SQL statements in $T_1$ and $T_2$ as shown in Table 1.

**Table 1.**

| $T_1$ | $T_2$ |
|---|---|
| Update Table1 set m = i + j where …<br>Update Table1 set n = m + k where …<br>Update Table1 set t = m + p + q where … | Update Table1 set m = i + r where… |

The static semantic analyzer will generate the read, pre-write, and post-write sets as illustrated in Table 2. Let us use data item *m* to illustrate the purpose of these sets and how they can be used to identify malicious database transaction. Data item *m* has non-empty read set and post-write set. The read set states that before data item *m* is updated by the transaction, either data items in {*i*, *j*} or {*i*, *r*} have to be read by the transaction. The post-write set states that after *m* is updated, data item *n* and *t* have to be updated by the same transaction. Please note that the where clauses have been ignored to keep this example short and may not be so in reality. If a transaction updates data item *m* without complying with rules specified by the read set and post-write set, it will be identified as an anomalous transaction.

**Table 2.**

| | Read Set | Pre-Write Set | Post-Write Set |
|---|---|---|---|
| m | { {i, j}, {i, r} | { ∅ } | { {n,t}} |
| n | { {m, k} } | { {m} } | { ∅ } |
| t | { {m, p, q} } | { {m, n} } | { ∅ } |

If an attacker's transaction is well-crafted and conform to the data dependencies specified, the dynamic semantic analyzer then steps in. Say, in reality, the access probabilities of {*i*, *j*} and {*i*, *r*} in the read set of *m* are different for normal user transactions and set $\{i, r\}$ is only infrequently used for updating m in special occasions. If the attacker's transaction updating $m$ reads $\{i, r\}$ instead of $\{i, j\}$ before modifying $m$, the access probability of the read set generated by the dynamic semantic analyzer can be used to identify this anomalous transaction. For more information on the data dependency based database intrusion detection model, interested readers may refer to [12].

As per the requirement of the data dependency based database intrusion detection method, there can be only two types of users: normal users and intruders. Normal users are authorized users, who connect to the database only through the database application and, therefore, can generate only legal transactions. Intruders are unauthorized users, who connect to the database from a remote terminal masquerading as normal users. There are many ways to pretend to be a normal user. For example, an intruder can obtain a password of the legitimate account or take control of a normal user's database connection. In any case, in order to get an access to the database, an intruder must find some vulnerability and exploit it.

### 3.2 Probability of Intrusion

There are three main factors that contribute to the cost of anomaly detection. These factors should be taken into consideration in the analysis of the optimal configuration of a data dependency based intrusion detection system. The first factor is the detection rate of the system. The method described in [4] was adapted to estimate this parameter. The second factor is the ratio of intrusions to legal activities. We assess this value by assuming that the number of intrusions depends on the number of vulnerabilities in the computer system that may allow an attacker to establish a connection to the database protected by the system. Bayes' formula for posterior probability is used to find the actual probability of attack in presence or absence of an alarm signal. The last factor is the losses incurred by an organization in different outcomes of a single attack. These values define which response strategy the organization needs to apply. The optimization conditions for the response strategy are found by means of linear programming. The optimal value of threshold is derived computationally.

If both valid and illegal transactions are generated at the same rate, the expected rate of intrusions among all transactions can be expressed the as follows:

$$\lambda = \frac{\alpha\psi}{\alpha\psi + \varphi}$$

where $\alpha$ is the proportion of successfully exploited vulnerabilities to all discovered vulnerabilities, $\psi$ is the rate at which vulnerabilities are discovered and $\varphi$ is the rate at which normal users establish connections to the database. The value of $\varphi$ can be easily retrieved by analyzing the database log. In order to find $\psi$, it is necessary to find out what sort of vulnerabilities can be used by an intruder to establish a connection to the database. For example, if the operating system on which the database application is deployed has a vulnerability that allows getting unauthorized access to this system, the intruder finally will be able to steal the password of a legitimate database user and establish a connection to the database. Then, using the information supplied by the vendor or other organizations such as Computer Emergency Response

Team, we can estimate the rate at which such vulnerabilities are discovered. The most difficult is to evaluate the value of $\alpha$ which depends on a large number of various factors.

The probability that certain vulnerability will be successfully used for an attack is greatly influenced by the personality of the individual who first discovered this vulnerability. If the vulnerability was discovered by the vendor, it will very likely not be announced before the hot fix is released. If the vulnerability is discovered by a potential intruder, everything depends on the intruder's behavior: a hacker can either conduct an attack or publish the discovered vulnerability. We can assume that, in the first case, there is always a possibility that instead of attacking a single target, the hacker can randomly choose a target or perform a mass attack against all known users of the vulnerable software or hardware. If the vulnerability is published, the probability that this vulnerability will be used against a single organization considerably increases. All these factors significantly complicates the calculation of $\alpha$. Hence, to simplify the problem, we assume that $\alpha$ is the fraction of vulnerabilities successfully utilized by hackers before the patch has been applied by a vendor. Thus, relying on the statistics of successful attacks we can estimate the value of $\alpha$, which, in turn, allows us to find the intrusion rate.

### 3.3 Evaluation of the Data Dependency Based Database Intrusion Detection

As explained earlier, the data dependency based intrusion detection system uses the dependencies among the data items in the database. Before a data item is updated in the database, some other data items are read or written, and after the update, other data items can be written too. Malicious transactions are detected by comparing read, pre-write and post-write sets against data items actually read or written by user transactions. Also the access probability for these sets can be used to identify malicious transactions. Without loss of generality we can formulate data dependency based intrusion detection method as the following two rules:

1. *Static Detection Rule*: Each update operation of a transaction must conform to the data dependencies represented by the read, pre-write, and post-write sets;
2. *Dynamic Detection Rule*: Each update operation of a transaction must conform to the normal user access patterns of different sets in the read, pre-write, and post-write sets.

Normally the update operations of a transaction are sequentially tested for compliance with both the rules. The transaction is considered illegal if any of its update operations does not conform to at least one of the rules. However, for the purposes of analysis we assume that initially all of the update operations of a transaction are tested for compliance with the first rule. We call this procedure static detection. Then, the update operations are tested for compliance with the second rule, which we call dynamic detection. The transaction is considered illegal if any of its update operations fails to pass either the static or the dynamic detection procedure. It is obvious that our approach produces the identical results with the original detection method.

Each transaction that goes through the static analyzer causes the creation of the data dependency which the transaction conforms to. In other words, transactions that were used to generate the data dependencies are always identified as legal by the static detection. Since, by definition, the static analyzer takes into consideration all transactions that can be generated by the database application, we can assume that the false positive rate of the static detection equals to zero. Actually, this assumption seems to hold in most real-life situations, as the transactions which were omitted during the static analysis are likely to be revealed later, in course of generating the log file for dynamic analysis. The experiments conducted by [12] provide the empirical evidence of this assumption, since, according to their results, the false positive rate always equals to zero, when the detection is based only on the results of static semantic analysis.

Notice that the transactions identified as malicious by the static detection will be classified as illegal regardless of the result of the dynamic detection. Therefore, if we denote $P_{STP}$ as the true positive rate of the static detection, then the aggregate detection rate of both static and dynamic detection can be calculated as follows:

$$P_{TP} = P_{STP} + (1 - P_{STP})P_{DTP}$$

where $P_{DTP}$ is the rate at which the dynamic detection correctly identifies intrusions misclassified by the static detection.

The static detection cannot be made more or less strict, so that $P_{STP}$ is a constant value unless the database application is modified. $P_{STP}$ can be expressed as the proportion of the transactions identified as illegal by the static detection to all illegal transactions. To obtain this proportion, we need to construct a set of sample intrusions and test the static detection procedure against this set. If the selected samples are representative, we will get an indicative $P_{STP}$ value.

Unlike the static detection, the dynamic detection can be made more or less strict by regulating the threshold value. This means that there exists a Receiver Operating Characteristic (ROC) curve which plots $P_{DTP}$ against false positive rate $P_{FP}$. Although the ROC curve can also be derived empirically, we will need to conduct a large amount of tests. Therefore, it is better to use an analytical method for finding economically optimal configuration of dynamic detection.

Normally, the dynamic analyzer computes the total use probability for each data item set and marks it as infrequently used if the value is less than the threshold $\tau$. In order to comply with data dependencies, a transaction needs to access all data items of at least one data item set of each read, pre-write and post-write sets that represent these data dependencies. The dynamic detection generates an alarm signal when the transaction includes at least one data item set which is marked as infrequently used. Let us make some alterations of this method. Suppose, the data dependencies created by the static analyzer altogether contain $N$ data item sets, $D_1$, $D_2$,…,$D_N$. Instead of marking data item sets as infrequently used, the dynamic analyzer associates each data item set with its total use probability $M_{D_i}$, $1 \le i \le N$. Now, suppose that transaction $T$ accesses all members of $K$ different data item sets, $K \le N$. During the analysis of $T$, the dynamic detection finds the minimum $\mu$ of the values associated with each of $K$ data item sets,

$$\mu = \min\{M_{D_1}, M_{D_2}, ..., M_{D_K}\}.$$

Then, this value is compared against the threshold $\tau$. If $\mu$ is greater than or equal to $\tau$, then transaction $T$ does not include any infrequently used data item, i.e., all $K$ data items have the total use probability that are greater than the threshold and the transaction is normal. If $\mu$ is less than $\tau$, then the transaction includes at least one infrequently used data item, so that an alarm signal is generated. Even though the alterations we made do not change the outcome of the dynamic detection, they rearrange this procedure to the form to which we can apply an analytical method. The only exception is that the result is inverted with regard to the threshold value, so that we need to change the limits of integration. As a result, using the base formulas from [17], we express $P_{DTP}$ and $P_{FP}$ as follows:

$$P_{DTP} = \int_{-\infty}^{\frac{\tau-\mu_F}{\sigma_F}} U(x)dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\tau-\mu_F}{\sigma_F}} e^{-\frac{x^2}{2}} dx,$$

$$P_{FP} = \int_{-\infty}^{\frac{\tau-\mu_L}{\sigma_L}} U(x)dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\tau-\mu_L}{\sigma_L}} e^{-\frac{x^2}{2}} dx;$$

where $\tau$ is the detection threshold, $\mu_L$ is the mean value of $\mu$ for legal transactions, $\mu_F$ is the mean value of $\mu$ for intrusions, $\sigma_L$ is the variance of $\mu$ for legal transactions, $\sigma_F$ is the variance of $\mu$ for intrusions, and *U(x)* is the probability density function for both normal and malicious database transactions. Same variance for normal and malicious database transaction is assumed. So essentially, $P_{DTP}$ or $P_{FP}$ represents the integration of the normal density function. Figure 1 illustrates a sample computation of $P_{DTP}$ and $P_{FP}$.

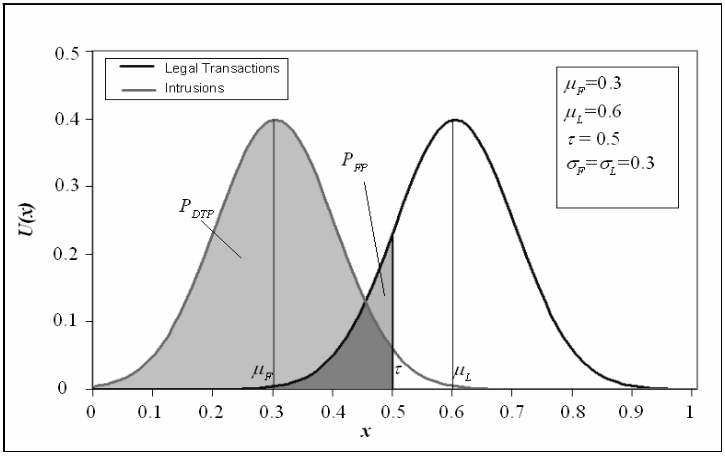

**Fig. 1.** Computation of $P_{DTP}$ and $P_{FP}$

Before performing analytical analysis we need to estimate the values of $\mu_L$, and $\sigma_L$, $\mu_F$ and $\sigma_F$. The former two values can easily be derived from the log file. In order to obtain the latter two values we need to find illegal transactions that are classified as legal by the static detection and build a set of sample intrusions. The samples must be representative; otherwise, the computed value will not indicate the real false positive rate.

### 3.4 Optimal Response Strategy

An organization may or may not have to respond to the alarm signal generated by the intrusion detection system. The response usually consists of manual investigation of the event which caused the alarm. If the event is indeed malicious in nature, the manual investigation allows recovering a part of the damage caused by the intrusion. However, the investigations are costly, since they engage resources, both people and equipment, and often interfere with the ongoing work. Therefore, an organization must decide to investigate an alarm signal, only if there is a great likelihood that the alarm was caused by an intrusion.

In order to make a reasoned decision, an organization needs to find the probability of intrusion given occurrence of alarm, which is expressed, in case of data dependency based intrusion detection, by the following formula:

$$p_A = P(I \mid A) = \frac{P(A \mid I)P(I)}{P(A \mid I)P(I) + P(A \mid \neg I)P(\neg I)} = \frac{P_{TP}\lambda}{P_{TP}\lambda + P_{FP}(1-\lambda)}$$

where λ is the intrusion rate. In fact, an organization may decide to launch an investigation even if there is no alarm signal, assuming that the intrusion detection system produced a false negative outcome. In this case, the organization will need to evaluate reliability of its intrusion detection system by computing the posterior probability of intrusion given the absence of an alarm signal, as follows:

$$p_N = P(I \mid \neg A) = \frac{P(\neg A \mid I)P(I)}{P(\neg A \mid I)P(I) + P(\neg A \mid \neg I)P(\neg I)} = \frac{(1-P_{TP})\lambda}{(1-P_{TP})\lambda + (1-P_{FP})(1-\lambda)}$$

If an investigation finds that a suspicious transaction in fact is illegal, the organization starts recovery procedure to restore the data damaged by this transaction.

Since investigations are costly, an organization may skip the investigation procedure and immediately start the database damage assessment and recovery procedures to cancel out the effects of the transaction which caused an alarm signal. That is, the organization fully relies on the decision made by the intrusion detection system. However, in case of detection error, the organization will incur losses by rolling back a legal transaction. In contrast, a manual investigation always clarifies the true type of a transaction, so that the recovery that follows a manual investigation involves only illegal transactions.

Let $c_i$ denote the cost of manual investigation, $c_r$ represent the cost of a rollback operation, $c_d$ represent the damage caused by a successful intrusion, and $c_e$ denote the loss caused by a rollback operation over a legal transaction. Then, the expected cost of a transaction when an alarm is generated is determined by the following equation:

$$F_A(i_A, r_A) = c_i i_A + p_A c_r i_A + c_r r_A + p_A c_d (1 - i_A - r_A) + (1 - p_A) c_e r_A =$$
$$= (c_i + p_A c_r - p_A c_d) i_A + (c_r + c_e - p_A c_d - p_A c_e) r_A + p_A c_d;$$

where $i_A$ is the fraction of manually investigated alarm signals, and $r_A$ is the rate at which the transactions identified as illegal are automatically rolled back by employing a database recovery procedure. This function, however, does not reflect the entire expected cost, as it does not take into consideration the losses incurred in the absence of the alarm, which are computed as follows:

$$F_N(i_N, r_N) = c_i i_N + p_N c_r i_N + c_r r_N + p_N c_d (1 - i_N - r_N) + (1 - p_N) c_e r_N =$$
$$= (c_i + p_N c_r - p_N c_d) i_N + (c_r + c_e - p_N c_d - p_N c_e) r_N + p_N c_d;$$

where $i_N$ is the rate at which transactions classified as legal are investigated, and $r_N$ is the rate at which transactions classified as legal are automatically rolled back. Figure 2 illustrates the possible responses in both alarm and no alarm cases.

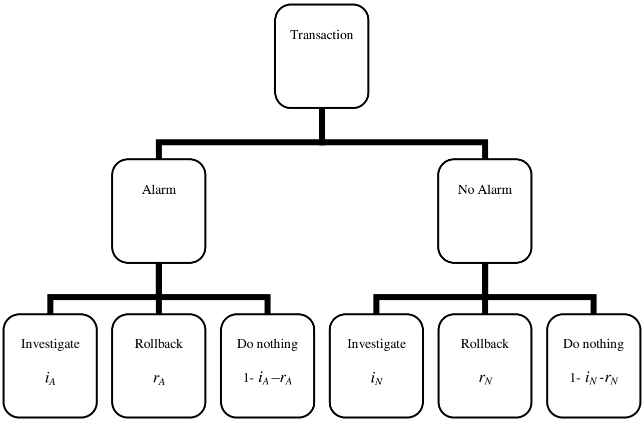

**Fig. 2.** Possible Responses

From the expected costs of a transaction in both alarm and no alarm cases, we can determine the total cost as the arithmetic mean of these values:

$$F_T = P(A) \times F_A + P(\neg A) \times F_N =$$
$$= (P_{TP}\lambda + P_{FP}(1 - \lambda)) \times F_A + ((1 - P_{TP})\lambda + (1 - P_{FP})(1 - \lambda)) \times F_N.$$

The variables $i_A$, $r_A$, $i_N$, $r_N$ define the optimal response strategy and must be chosen to provide minimal values of $F_A$ and $F_N$. The optimization conditions with regard to these functions are identical, since both of them are expressed by the same formula; the only exception is the different probability parameters. We examine the optimization conditions by the example of $F_A$.

The task of finding the minimal cost can be represented as the problem of linear programming [16] with an objective function $F_A(i_A , r_A)$ that is a subject to the following inequality constraints:

$$0 \le r_A \le 1,\ 0 \le i_A \le 1, \text{ and } 0 \le i_A + r_A \le 1.$$

These inequalities follow from the fact that $i_A$ and $r_A$ represent the proportions of responses to the received alarm signals. As shown in Figure 3, they produce the region of possible solutions limited by both coordinate axes and the line expressed by function $i_A(r_A) = 1 - r_A$.

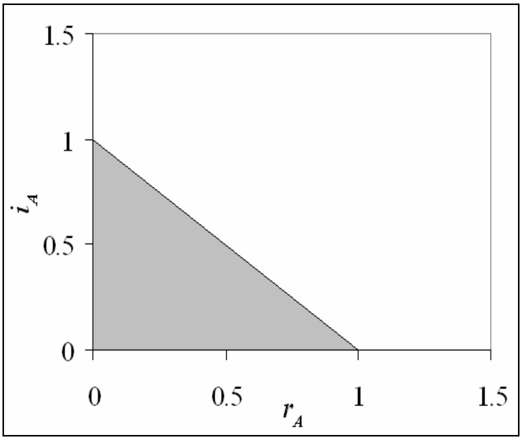

**Fig. 3.** Solution Region for $F_A$

It is known that at least one of the vertices of feasible region represents the optimal solution [16]. In our case, there exist three possible solutions that correspond to the vertices of the triangle-shaped region:

1. $i_A$=0, $r_A$=0; 2. $i_A$=1, $r_A$=0; 3. $i_A$=0, $r_A$=1.

Each of these solutions becomes optimal under certain conditions. These conditions are defined by the coefficients of the variables $i_A$ and $r_A$. For simplicity, we represent these coefficients as follows:

$$a = c_i + p_A c_r - p_A c_d,\ b = c_r + c_e - p_A c_d - p_A c_e.$$

In this case, we can represent $F_A$ as

$$F_A(i_A, r_A) = a i_A + b r_A + p_A c_d.$$

It is obvious that $i_A$=0 and $r_A$=0 is the optimal solution if $a \ge 0, b \ge 0$.

We can prove it by contradiction. First, let us find the minimal value of the expected cost:

$$F_A(0,0) = a \times 0 + b \times 0 + p_A c_d = p_A c_d .$$

Now, let us assume that there exist such $i_A$ and $r_A$ that $F_A(i_A ,r_A)$ is less than $p_A c_d$, i.e.

$$ai_A + br_A + p_A c_d < p_A c_d .$$

In this case, the following condition is necessary to hold:

$$ai_A + br_A < 0 .$$

Since the inequality constraints prevent $i_A$ and $r_A$ from accepting negative values, this condition contradicts to the initial condition that both $a$ and $b$ are greater than or equal to zero.

$i_A$=1 and $r_A$=0 is the optimal solution if $a < 0$, $a \le b$.

In this case the minimal value for $F_A(i_A ,r_A)$ is

$$F_A(1,0) = a \times 1 + b \times 0 + p_A c_d = a + p_A c_d .$$

Let us assume there exist such $i_A$ and $r_A$, $r_A > 0$, $i_A \le 1 - r_A$ ,that $F_A(i_A ,r_A)$ accepts a lesser value, i.e.,

$$ai_A + br_A + p_A c_d < a + p_A c_d .$$

Knowing that

$$ai_A \le a(1 - r_A) ,$$

we can represent the previous inequality in the form of

$$a - ar_A + br_A + p_A c_d < a + p_A c_d .$$

Simplifying it, we will get $a > b$, which contradicts to $a \le b$.

$i_A$=0 and $r_A$=1 is the optimal solution if $b < 0$, $b < a$ .

We can prove this condition in the same way as the previous one.

Now we can calculate what ratio of the real values each optimal solution corresponds to. From $a \ge 0$, follows that

$$c_i + p_A c_r - p_A c_d \ge 0 ,$$

i.e.,

$$\frac{c_i}{c_d - c_r} \ge p_A .$$

From $b \ge 0$, follows that

$$c_r + c_e - p_A c_d - p_A c_e \ge 0 ,$$

i.e.,

$$\frac{c_r + c_e}{c_d + c_e} \geq p_A .$$

From $a \leq b$, follows that

$$c_i + p_A c_r - p_A c_d \leq c_r + c_e - p_A c_d - p_A c_e ,$$

i.e.,

$$\frac{c_r + c_e - c_i}{c_r + c_e} \geq p_A .$$

Thus, the optimization conditions for $F_A$ can be formulated as shown in Table 3.

**Table 3.**

| Optimization conditions | $i_A$ | $r_A$ |
|---|---|---|
| $\frac{c_i}{c_d - c_r} \geq p_A$ and $\frac{c_r + c_e}{c_d + c_e} \geq p_A$ | 0 | 0 |
| $\frac{c_i}{c_d - c_r} < p_A$ and $\frac{c_r + c_e - c_i}{c_r + c_e} \geq p_A$ | 1 | 0 |
| $\frac{c_r + c_e}{c_d + c_e} < p_A$ and $\frac{c_r + c_e - c_i}{c_r + c_e} < p_A$ | 0 | 1 |

The same conditions for $F_N$ are achieved by replacing $P_A$ with $P_N$.

To sum up, the organization's actions to optimize the transactions should consist of the following steps:

1. Select a representative set of known illegal transactions and test the static detection against the set to estimate $P_{STP}$ as the proportion of the transaction identified as intrusions to all transactions in the set;
2. Find $\mu$ for each of the illegal transactions identified as legal by the static detection, compute mean $\mu_F$ and variance $\sigma_F$;
3. Select the representative set of known legal transactions and find $\mu$ for each of these transactions; compute mean $\mu_L$ and variance $\sigma_L$.
4. Determine $\alpha$, $\psi$ and $\varphi$ and calculate the rate of intrusions $\lambda$;
5. Determine the organization's cost metrics expressed by $c_i$, $c_r$, $c_e$, $c_d$ ;
6. Find the optimal configuration, i.e. the values of $F_T$, $\tau$, $i_A$, $r_A$ , $i_N$, $r_N$;
7. Set the threshold to $\tau$;
8. Follow the response strategy defined by $i_A$, $r_A$ , $i_N$, $r_N$ .

Since many of the initial variables change their values in course of time, the organization must periodically repeat these operations to update the optimal configuration.

## 4 Experimental Results

As mentioned in Section 3.2, there are three factors affecting the cost of the intrusion detection. Changing of any of these factors leads to the alteration of the final outcome. However, as Table 3 indicates, the absolute values of the cost metrics expressed by $c_i$, $c_{e,}$ $c_r$ and $c_d$ are less important than the proportions of these four variables. The proportions of the first two parameters are unlikely to change with time; moreover, the values of $c_i$ and $c_e$ are expected to be of the same magnitude, as both the investigation of an alarm signal and the recovery from an erroneous rollback require human interaction. In contrast, the value of $c_d$ may significantly vary, as the damage caused by an attack depends on the nature of the data stored in the database. Given that the rollback is one of the key features of database management systems, $c_r$ is expected to be much smaller than $c_i$ and $c_e$. From these considerations, the following variants of the cost metrics were selected for testing:

1. $c_r = 1, c_e = c_i = 100, c_d = 10000;$
2. $c_r = 1, c_e = c_i = 100, c_d = 50000;$
3. $c_r = 5, c_e = c_i = 100, c_d = 50000;$

In the first variant, we assume that the damage caused by a successful intrusion is 100 times greater than the cost of investigation and the cost of recovery from a falsely conducted rollback, which, in turn, are 100 times greater than the cost of a rollback operation. In the second variant we increase the potential damage, while other parameters remain the same. In the third variant we additionally increase the cost of a rollback operation.

The quality of intrusion detection is expressed by $P_{STP}$, $\mu_L$, $\mu_F$ ,$\sigma_L$ and $\sigma_F$. These parameters should not significantly change with time, unless the intrusion detection system has been deployed without a proper training. We tested the program against different sets of the parameters, and selected those whose test results are more illustrative: $P_{STP} = 0.4$, $\mu_L$=0.6, $\mu_F$ =0.4, $\sigma_L$=0.2, $\sigma_F$=0.2. In order to show how the probability of intrusion influences the optimal configuration, we chose the value of $\lambda$ to vary from $10^{-5}$ to $10^{-1}$. Figure 4 illustrates the total expected cost for all three variants of cost metrics. The x-axis of the graph is a logarithmic scale representing the values of $\lambda$. It must be noted that the data provided in this figure is a small subset of that obtained through the experiment; due to page limitation, we could not produce the entire result here.

As Figure 4 indicates, in spite of the fact that in the second variant, the value of potential damage $c_d$ is five times greater than that in the first one, the maximum value of the cost function does not increase significantly. However, a larger value of $c_d$ increases the growth rate of the cost function. In contrast, the cost of recovery from an erroneous rollback $c_r$ affects the maximum value of the cost function. Furthermore, as Figure 4 depicts, the costs start to decline at some point. Although it seems atypical, the explanation for this phenomenon is that the growth of intrusion rate decreases

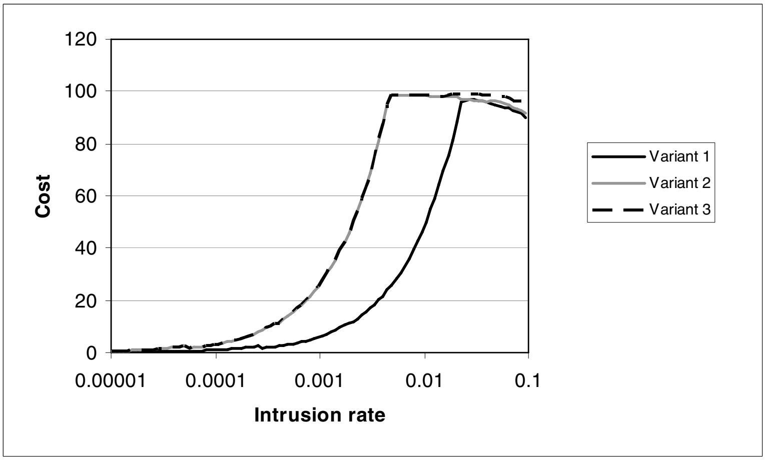

**Fig. 4.** Total Cost for Different Variants of Cost Metrics

the uncertainty of intrusion detection process. In other words, high intrusion rate permits more accurate decisions, thus, effectively responding to the attacks.

## 5 Conclusions

In this paper, we have presented a model that can be used to determine the economically optimal configuration of the data dependency based intrusion detection system. Three main parameters are taken into consideration in our model: the intrusion rate, the quality of the intrusion detection, and the cost metrics of an organization. We presented a step by step methodology that helps in finding the optimal configuration, expressed by the response strategy and the threshold value. Our experimental results suggest that the value of potential damage does not proportionately affect the total cost function. However, the recovery cost associated with an erroneous rollback significantly affects the maximum value of the cost function. Furthermore, the experiment illustrated that the dynamic detection is useful only at high intrusion rates. Therefore, while the expected rate of intrusion remains reasonably low, an organization may simply rely on the static detection and still maintain a low total expected cost of the intrusion detection. As part of our future work, we plan to conduct further experiments to identify optimal response strategy under various circumstances.

## Acknowledgement

This research has been supported in part by US AFOSR under grant FA 9550-04-1-0429 and the KY NSF EPSCoR program. We are thankful to Dr. Robert. L. Herklotz and Jeffrey Mossey for their support, which made this work possible.

## References

1. Richardson, R.: 2007 CSI Computer Crime and Security Survey, Computer Security Institute (2007), `http://gocsi.com`
2. Axelsson, S.: Intrusion Detection Systems: A Survey and Taxonomy, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden (2000), `http://www.cs.chalmers.se/~sax/pub/`
3. Axelsson, S.: The Base-rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security 3(3), 186–205 (2000)
4. Cavusoglu, H., Misra, B., Raghunathan, S.: Optimal Configuration of Intrusion Detection Systems. In: Proc. Second Secure Knowledge Management Workshop, pp. 1–7 (2006)
5. Ulvila, J.W., Gaffney, J.E.: Evaluation of Intrusion Detection Systems. Journal of Research of the National Institute of Standards and Technology 108(6), 453–473 (2003)
6. Liu, P.: Architectures for Intrusion Tolerant Database Systems. In: Proc. 18th Annual Computer Security Applications Conference, pp. 311–320 (2002)
7. Srivastava, A., Sural, S., Majumdar, A.K.: Database Intrusion Detection using Weighted Sequence Mining. Journal of Computers 1(4), 8–17 (2006)
8. Bertino, E., Kamra, A., Terzi, E., Vakali, A.: Intrusion Detection in RBAC-administered Databases. In: Proc. 21st Annual Computer Security Applications Conference, pp. 170–182 (2005)
9. Lee, V., Stankovic, J., Son, S.: Intrusion Detection in Real-time Databases via Time Signatures. In: Proc. Sixth IEEE Real-Time Technology and Applications Symposium, pp. 124–133 (2000)
10. Chung, C., Gertz, M., Levitt, K.: DEMIDS: A Misuse Detection System for Database Systems. In: Proc. Integrity and Internal Control in Information Systems: Strategic Views on the Need for Control, IFIP TC11 WG11.5, Third Working Conference, pp. 159–178 (2000)
11. Hu, Y., Panda, B.: A Data Mining Approach for Database Intrusion Detection. In: Proceedings of the 19th ACM Symposium on Applied Computing, Nicosia, Cyprus (2004)
12. Hu, Y., Panda, B.: Design and Analysis of Techniques for Detection of Malicious Activities in Database Systems. Journal of Network and Systems Management 13(3), 269–291 (2005)
13. Lee, W., Fan, W., Miller, M., Stolfo, S.J., Zadok, E.: Toward Cost-Sensitive Modeling for Intrusion Detection and Response. Journal of Computer Security 10(1-2), 5–22 (2002)
14. Debar, H., Dacier, M., Wespi, A.: Towards a Taxonomy of Intrusion-Detection Systems. Computer Networks 31(8), 805–822 (1999)
15. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In: Proc. 2000 DARPA Information Survivability Conference and Exposition, vol. 2, pp. 12–26 (2000)
16. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms, 2nd edn., pp. 770–821. MIT Press, Cambridge (2001)
17. Grinstead, C., Snell, L.: Introduction to Probability, 2nd edn., pp. 325–360. American Mathematical Society, Providence (1997)